# Rethinking Packet Forwarding Hardware

Martin Casado[*§] Teemu Koponen[†] Daekyeong Moon[‡] Scott Shenker[‡§]

## 1 Introduction

For routers and switches to handle ever-increasing bandwidth requirements, the packet "fast-path" must be handled with specialized hardware. There have been two approaches to building such packet forwarding hardware. The first is to embed particular algorithms in hardware; this is what most commodity forwarding chips do (*e.g.*, those from Broadcom, Marvell, and Fulcrum). These chips have led to amazing increases in performance and reductions in cost; for instance, one can now get 24 ports of gigabit ethernet for under $1000.

Unfortunately, this approach offers only very rigid functionality; one can't change protocols or add new features that require hardware acceleration without redoing the chip. This forces network forwarding enhancements to evolve on hardware design timescales, which are glacially slow compared to the rate at which network applications and requirements are changing.

To counter this inflexibility, several vendors have taken a different approach by introducing more flexible "network processors". These have not been as successful as anticipated, for at least two reasons. First, designers were never able to find a sweet-spot in the tradeoff between hardware simplicity and flexible functionality, so the performance/price ratios have lagged well behind commodity networking chips. For another, the interface provided to software has proven hard to use, requiring protocol implementors to painstakingly contort their code to the idiosyncrasies of the particular underlying hardware to achieve reasonable performance. These two problems (among others) have prevented general-purpose network processors from dislodging the more narrowly targeted commodity packet forwarding hardware that dominates the market today.

In this paper, we step back from these two well-known approaches and ask more fundamentally: what would we want from packet forwarding hardware, how might we achieve it, and what burden does that place on networking software?

Ideally, hardware implementations of the forwarding paths should have the following properties:

- *Clean interface between hardware and software:* The hardware-software interface should allow each to evolve independently. Protocol implementations should not be tied to particular networking hardware, and networking hardware should not be restricted to particular protocols. General-purpose network processors have largely failed the first requirement, while the current generation of commodity networking chips fail the second requirement.
- *Hardware simplicity:* In order to scale to increasing speeds, the hardware functionality should be of very limited complexity. Again, both current approaches to hardware acceleration fail to satisfy this requirement.
- *Flexible and efficient functionality:* The resulting software-hardware combination should be capable of realizing a wide variety of networking functions at high-speed and low-cost, while having short development cycles. Commodity chips work at high-speed and low-cost, but have long development cycles for new functionality. General-purpose network processors are not yet competitive on speed or cost.

In this paper we propose a different approach to hardware packet forwarding that has the potential to realize all of these goals. Our approach assigns a completely different role to hardware. Traditionally, hardware implementations have embodied the logic required for packet forwarding. That is, the hardware had to capture all the complexity inherent in a packet forwarding decision.

In contrast, in our approach all forwarding decisions are done first in software, and then the hardware merely *mimics* these decisions for subsequent packets to which that decision applies (*e.g.*, all packets destined for the same prefix). Thus, the hardware does not need to understand the logic of packet forwarding, it merely caches the results of previous forwarding decisions (taken by software) and applies them to packets with the same headers. The key task is to match incoming packets to previous decisions, so the required hardware is nothing more than a glorified TCAM, which is simple to implement (compared to other networking hardware) and simple to reason about. We describe this approach in more detail in Section 2.

One key challenge in this approach is scaling to high speeds. Achieving high forwarding rates is not a matter of the number of clock cycles needed to make a forwarding decision, but instead is one of cache hit rates; what fraction of packet forwarding can be done based on the forwarding of previous packets? This question is central to the viability of this approach, and

[*]Nicira Networks Inc.
[†]Helsinki Institute for Information Technology (HIIT)
[‡]UC Berkeley, Computer Science Division
[§]International Computer Science Institute (ICSI)

we address it in Section 3. Another key challenge is deciding to which packets a particular forwarding decision applies (*i.e.*, which fields in the packet header must match in order to apply a decision), and when this decision has been rendered obsolete by changing network state (*e.g.*, changes in forwarding tables); these system design issues are addressed in Section 4.

This approach has been motivated by a long line of flow-oriented approaches to networking hardware, from [5] to [2] (and many others). What makes our approach different from traditional connection-oriented architectures (*e.g.*, ATM) is that our approach does not require any change to networking protocols. Our approach (similar to the references [2, 5]) only changes the interface between hardware and software, but does not affect protocols or the Internet architecture. Like [3] we seek to provide a flexible software model with hardware forwarding speeds, however in our approach we propose to fully decouple the software from the hardware, while [3] provides direct access to hardware resources such as the TCAM and special purpose ASICs. We view this as a necessary and complimentary step for integrating with existing hardware.

## 2 Our Approach

### 2.1 General Description

Packet forwarding decisions deterministically depend on the header of the arriving packet and on some local state in the router/switch (such as a routing table).[1] For example, the header/state combination for ethernet forwarding is the destination MAC address and the L2 learning table, whereas for IP forwarding it is the destination IP address and the FIB.

The forwarding decision includes both the output port(s) on the network device as well as possible modifications to the packet header, such as the label swapping in MPLS or packet encapsulation/decapsulation. As long as the relevant pieces of local state have not changed, then all packets with the same packet header should receive the same forwarding behavior. This statement applies to *any* forwarding protocol that depends only on packet headers and local state.

To leverage this fact, we treat packet forwarding as a *matching process*, with all packets matching a previous decision handled by the hardware, and all non-matching packets handled by the software. We assume that when a packet is handled by software, we can infer a *flow entry* for that packet which contains four pieces of information:

**Ingress port(s):** This specifies to which ingress ports a flow entry applies. Many rules will only apply to the ingress port of the packet that generated the flow entry, but other rules (such as "packets with TTL=1 should be dropped") can be applied to all ingress ports.

**Matching rule:** The software specifies the fields in the packet header that must match for the flow entry to apply. This could be the entire header (*i.e.*, requiring a perfect match), or only some fields (*e.g.*, perhaps only the destination address to match).

**Forwarding action:** The forwarding action is the set of output ports to which the packet should be sent, along with (optionally) the rewritten packet headers for each port. Dropping the packet is represented by a special null output port.

**State dependence:** This represents the local state on which the forwarding decision is based (such as entries in a routing table or ACL database).

When a packet arrives, it is compared to the table of flow entries corresponding to its ingress port. If its packet header does not match any of the entries, then it is sent to software for processing. If it matches a flow entry, the packet is handled according to the specification of the forwarding action in the flow entry. Whenever a piece of local state changes, such as an update to an ACL or routing table, then all flow entries that depended on that state are invalidated and removed from the flow table.

Before proceeding to examples, we mention a few more detailed points. First, the matching rules must be defined so a packet matches no more than one rule. Since we assume that the software deterministically forwards each packet based on its header fields and particular system state, it should be possible to generate non-conflicting entries at runtime.

Second, we view stateful tunneling (*e.g.*, IPsec in which a sequence number is updated for every packet) as virtual output ports rather than part of the forwarding logic. This is similar to how they are handled in standard software IP forwarding engines in which they are modeled as networking interfaces. If we did not use this technique then no flow entry would be recorded for such flows, because the state on which the forwarding actions depends (which includes per-packet modifications) would be updated on every new packet arrival.

Third, there are a number of possible system designs that could determine the headers and system state used in a decision, such as a special-purpose programming language, a specialized compiler or even hardware support. We discuss these further in Section 4. However, the primary goal of this paper is not to give a specific and comprehensive system design (we are working on this and plan to discuss it in future work), but rather to explore the fundamental viability of our general approach.

### 2.2 Examples

To demonstrate how our proposal operates in practice, we discuss two examples of familiar datapath functions in

[1]We are ignoring forwarding decisions based on deep-packet inspection.

| Packet fields | Memory dependencies | Output | Modifications |
|---|---|---|---|
| eth_type = VLAN, VLAN ID | VLAN database, port configuration, L2 MAC table | access port | remove VLAN |
| eth_type ! = VLAN | VLAN database, port configuration, L2 MAC table | access port | None |
| eth_type = VLAN, VLAN ID | VLAN database, port configuration, L2 MAC table | trunk port | None |
| eth_type ! = VLAN | VLAN database, port configuration, L2 MAC table | trunk port | add VLAN |

Table 1: Packet fields, memory dependencies and potential packet modifications for VLAN tagging example.

```
if eth.type ≠ IPv4 then
    send_to(drop port); return;
end
if ip.ttl ≤ 1 then
    generate_icmp(); return;
end
if not valid ip.chksum then
    send_to(drop port); return;
end
decrement(ip.ttl); update(ip.chksum);
if no ip.options and not ip.fragmented then
    next_hop, dst_port ⟵ fib(ip.dst_ip);
    eth.src_mac ⟵ dst_port.mac;
    eth.dst_mac ⟵ arp_cache(next_hop);
    send_to(dst_port);
else
    // Complex header processing...
end
```

Figure 1: Pseudo code for IP routing.

heavy use today, IPv4 routing and VLAN tagging.

**IP routing.** We begin by describing how our approach operates with standard IPv4 routing (pseudo-code shown in Figure 1). From the standpoint of caching, IPv4 is non-trivial in that it includes multiple state dependencies as well as a number of header modifications (TTL decrement, checksum update, L2 address updates).

On packet receipt, the router performs basic header checks (protocol version, TTL), and verifies the checksum before doing a prefix lookup in the FIB. Hence, the forwarding decision depends on all of the protocol fields as well as the locally stored FIB. Moreover, because the checksum is dependent on every byte in the header (except itself), every header field must be included in the flow entry (including the TTL).[2] However, as we show in the next section, even with the full header included in the matching rule, our approach can achieve very high cache hit rates.

After looking up the next hop, the router consults the ARP cache for the next hop MAC address. The Ethernet header is updated with the MAC of the outgoing interface as well as the next hop MAC address before being sent out. Thus, local port MAC configuration, and ARP cache are added as state dependencies for the flow entry.

[2]This can be avoided by relying on endpoint checksum verification as is used in IPv6 and has been suggested elsewhere [7].

**VLAN tagging.** We consider a simple VLAN tagging implementation, which implements port-based VLAN encapsulation and decapsulation for packets on access ports, L2 forwarding within VLANs, and pass-through of VLAN packets on trunk ports. Table 1 summarizes the packet header and memory dependencies of this example for given inputs. Briefly, if an incoming packet was received on an access port, it is tagged with the VLAN ID associated with that port. This adds the state dependency of the VLAN database to any resulting decision. Once the associated VLAN has been identified (if any) the packet VLAN ID and destination MAC addresses are used for L2 forwarding. This adds the MAC learning tables to the state dependency of the flow entry. In the final step, the packet is sent out of one or more ports as determined by the L2 forwarding step. If the port is an access port (and the packet has a VLAN header), the VLAN header is removed. This adds the port configuration as a dependency.

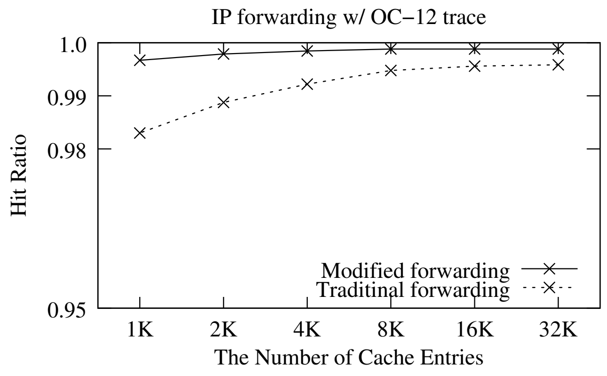
Tagging and label swapping (*e.g.*, MPLS in addition to VLAN) fit nicely with our proposal as the associated configuration state generally changes very slowly. However, for large networks, MAC learning can generate additional overhead due to learning table entry timeouts. To validate that these are manageable in practice, we explore the cache hit and miss ratios of L2 learning over enterprise traces in the following section.

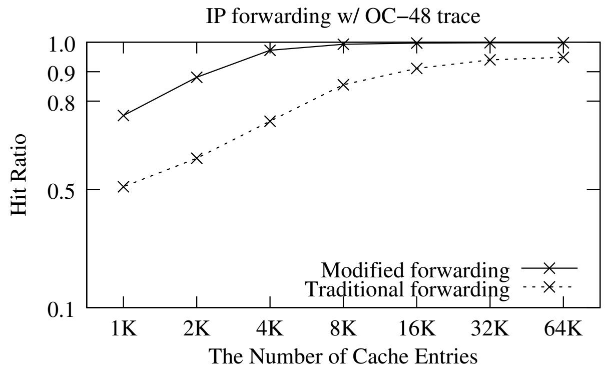
The two examples presented in this section are limited to lookup and forwarding. However, additional datapath mechanisms, such as ACLs, can be added by simply updating the software. In the case of ACLs, this would merely add the ACL table to the state dependencies. Adding forwarding for additional protocols is similarly straightforward. Since the protocol type is implicitly included in the matching rule, it will automatically be de-multiplexed by the hardware.

## 3 Scaling

In order for the software not to become a bottleneck, the cache hit rates must correspond to the speed differential between the software and hardware forwarding rates. For example, a commodity 48x1Gigabit networking chip has roughly two orders of magnitude greater forwarding capacity than a software solution using a standard PC architecture (*e.g.*, [11] or [8]). Therefore, to maintain hardware-only speeds for an integrated architecture with similar performance characteristics, the hit rates must exceed 99%.

In this section, we analyze cache hit rates using standard L2 and L3 forwarding algorithms on network traces. We explore the performance of our approach as described here, and also with minor modifications that help achieve vastly better hit rates.





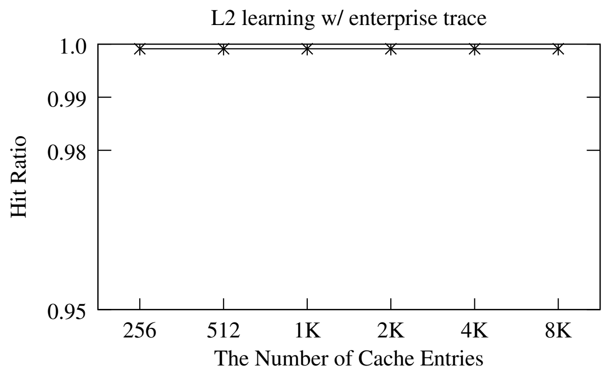


Figure 2: Results of cache hit-rate simulations for L2 and L3 forwarding algorithms.

**IP forwarding.** We first explore the cache behaviour of standard IPv4 forwarding. For this analysis we use two publicly available traces [9, 10], one collected from an OC-48 link consisting of 6,795,675 packets covering a 5 minute time window, and a second collected from an OC-12 link consisting of 6,872,338 packets covering one hour.

We assume the naive hardware implementation of a managed TCAM in which each matching rule fills up an entry. The system uses LRU when replacing cache entries and we assume that the FIB does not change during the period of analysis.

We test two variants of IPv4 fowarding and show the results in Figure 2. The first algorithm (*traditional forwarding* in the graph) strictly obeys the IPv4 forwarding specification (including full header checksum), and thus, requires a full packet header match to be cached. Reaching a 99% hit rate requires a TCAM of 4,096 entries in the OC-12 case. For the OC-48 trace, the results are significantly worse; the hit rate only reaches 95% even with an essentially infinite number of entries.

In the second variant (*modified forwarding* in the graph), we make two modifications which greatly improve the results. First, we perform an incremental update of the checksum which is limited to the TTL decrement change. As a result, no header values beyond the version, TTL, and destination are included in the cache entry.[3] Secondly, the most precise lookup is a /24 rather than full IP match (this is not an unreasonable assumption for high bandwidth links; most FIBs on such links won't contain prefixes smaller than /24s). These changes greatly improve the hit rates. In the case of the OC 48, a 99% hit rate is achieved at 8,000 entries. Further, the lowest hit rate we recorded using the OC-12 traces was 99.9889%, sufficient for speed differentials of over *three* orders of magnitude.

**L2 learning.** We also look at the performance of decision caching for standard L2 learning. Our MAC learning implementation uses 15 second timeouts of learned addresses and the hardware uses an LRU cache replacement strategy. For analysis, we use publicly available enterprise trace data [1] covering a 3 hour time period, and containing 17,472,976 packets with 1958 unique MAC addresses.

The cache hit rates of this analysis are shown in the bottom most graph of Figure 2. For all entry sizes we tested (the minimum being 256) we found the cache hit rate to be over 99.9%. We note that the use of soft state in this manner appears to have significantly lower TCAM requirements than today's commercial Ethernet switches which commonly hold 1 million Ethernet addresses (6MB of TCAM).

While much more study is warranted on the scaling properties of caching, we feel that the results are promising. Commodity TCAMs with entry widths suitable for L2 and L3 decision caching (*e.g.*, 36 bytes) and with 8k entries (sufficient for all but the OC-48 case) cost about $50 in bulk at the time of this writing. Further, we

[3]We note that IPv6 does not require per-hop IP checksums, as the design community found them redundant.

note that we make modest assumptions as to the relative forwarding speeds of the on-board CPUs. With the onset of multicore, the prospects for the speed differential between the CPU and the hardware forwarding decreasing are good. As long as the forwarding doesn't involve per-packet state changes, the software should scale as a function of the number of cores.[4]

## 4 System Design Approaches

If the basic matching approach we are advocating can achieve sufficiently high speeds, as we discussed in the last section, then the next challenge is whether we can determine the correct matching rules and state dependencies. There are two extreme approaches one can take (though we suspect an intermediate compromise will be adopted in practice): explicit definition (by the implementor), or automated inference via runtime analysis (*e.g.*, dynamic binary instrumentation). We discuss these in turn.

### 4.1 Explicit Dependency Identification

The simplest approach to determining matching rules and state dependencies is to put the onus of identifying them on the programmer. This could be done manually, forcing the programmer to explicitly annotate the forwarding decisions with the headers and system state it relies on. Note that while the programmer has to explicitly define these dependencies, they do not have to explicitly manage the flow entries themselves; once the matching rules and state dependencies are specified, a compiler would then generate the necessary code to invalidate any related flow entries upon updates to local state.

Similar methods have been used for global state management in parallel programming environments (see *e.g.*, [6]), and should be straightforward to apply to packet forwarding. An alternative approach to extending the compiler is to provide the developer with a set of library calls that provide similar functionality to the annotations.

While conceptually simple, this explicit identification approach imposes a burden on the programmer and increases the risk of error; even if the algorithm is implemented correctly, a missing state dependency could result in incorrect runtime behavior. Further, it requires that *all* state dependencies be identified, which complicates the use of non-annotated third-party libraries.

### 4.2 Transparent Dependency Identification

It would be far easier for programmers if they could focus on merely implementing the required forwarding decisions, and let the system automatically infer what the matching rules and state dependencies were. Unfortunately, while forwarding logic itself may be simple, deducing the headers and state that effect decisions is difficult without developer cooperation. For example, some of the challenges are:

- Most forwarding software will have state not directly used as a part of the forwarding decision. This includes counters, file handles, and any configuration state. In an extreme case, a global timer value is used to timestamp all incoming packets. In a naive implementation, every time-tick would invalidate all flow entries. Complex data structures further complicate the analysis by maintaining their own internal state which may change without impacting the outcome of a forwarding decision.
- Processor architecture may require the forwarding logic to access state that isn't part of the resulting decision. For example, prefixes for LPM are commonly read as a single word (beginning from the least significant bit), while only the most significant bits may be relevant to the decision.
- Pipelined and parallel execution models require careful dependency management to attribute a given state access with a particular packet.

Despite these challenges, it is worth considering whether it is possible to determine the headers and state dependencies via runtime analysis by using, for example, dynamic binary instrumentation. This could dramatically increase software overheads, so we are not yet convinced of its practicality, but we discuss it here as a promising avenue for future research.

Runtime analysis operates by tracking *all* memory references of the forwarding software while processing packets. This can be done at the granularity of bits [4], which would be optimal for our application. A simple heuristic would be to assume that any header value that was accessed by the software, or any static or heap state, is a dependency.

Clearly this approach requires disciplined programming over a limited programming model and could not effectively be applied to existing software. Further, the inference is only transparent at the syntax level. The developer must be aware of the runtime characteristics of the system and act accordingly (for example by avoiding spurious reads to global data).

### 4.3 Inferring Packet Modifications

In addition to inferring important headers and state dependencies, the system must also determine the actions to apply to the packet. While it may be possible to transparently infer the changes at runtime using the techniques discussed in the previous section, a simpler approach would have the developer specify actions explicitly in a manner similar to [2]. To be sufficiently general, the action set must be able to support the modification of arbitrary bytes as well

[4]Although, admittedly, the bus speeds between the CPUs and network hardware require special attention.

as byte insertion (for encapsulation) and deletion (for decapsulation).

## 5 Conclusions

In all but the lowest end switches and routers, packet forwarding is largely done in hardware. While not often a subject of study in the academic literature, the advances in packet forwarding hardware has been remarkable. In fact, commodity networking chips now support aggregate speeds that only a few years ago were only available on the highest-end routers.

As successful as this generation of hardware-accelerated packet forwarding has been, in the years ahead it must find a way to accommodate two trends that appear to be on a collision course:

- Speed: the demand for bandwidth continues to grow, in enterprises, datacenters, and the wide-area Internet. Backbone links have transitioned from 20Gbps to 40Gbps, core switches in datacenters have high densities of 10Gbps ports, and ever-faster switches and routers are on the horizon.
- Control: the need for better network management and security, particularly in the areas of traffic engineering and access control, has increased emphasis on measures for controlling packet forwarding.

Dealing with increasing speed in hardware calls for limiting the complexity of forwarding decisions (so they can be done efficiently in hardware) and limiting the flexibility of these decisions (so the hardware does not need to be changed). On the other hand, attaining greater control over forwarding decisions calls for greater complexity in the forwarding path, and for greater flexibility (since the nature of these control decisions will change far more frequently than the basic protocols change).

The approach described here tries to accommodate these conflicting desires by retaining full generality of function while remaining simple to implement (by hardware designers) and use (by software implementors). Any forwarding function that depends only on the packet header and local state can be implemented over the same hardware, with a very straightforward interface.

Achieving this generality and ease-of-use at high speeds requires a large enough TCAM-like cache to achieve a very low cache miss rate. Thus, the viability of our approach depends on future trends in hardware (which determines the cost of a given cache size and the speed of software processing) and network traffic (which determines the necessary cache size for a given cache hit rate). We can't make definitive projections about any of these, but our initial investigations suggest that our approach may indeed be viable. In particular, if we focus on IPv6, with its lack of per-hop checksum, then the required cache sizes are very inexpensive.

Of course, this is all very preliminary, and we hope to soon embark on a much fuller investigation. This will entail a more extensive set of traces, a more thorough analysis of the factors that determine the cache miss rate, and building a fully functioning implementation of this approach.

## 6 References

[1] LBNL/ICSI Enterprise Tracing Project. `http://www.icir.org/enterprise-tracing`.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM Computer Communications Review*, 38(2):69–74, 2008.

[3] J. C. Mogul, P. Yalagandula, J. Tourrilhes, R. McGeer, S. Banerjee, T. Connors, and P. Sharma. API Design Challenges for Open Router Platforms on Proprietary Hardware . In *Proc. 7th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-VII)*, Oct. 2008.

[4] N. Nethercote and J. Seward. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In *Proc. of ACM SIGPLAN PLDI '07*, pages 89–100, 2007.

[5] P. Newman, G. Minshall, and T. L. Lyon. IP Switching - ATM under IP. *IEEE/ACM Transactions on Networking*, 6(2):117–129, 1998.

[6] The OpenMP API Specification for Parallel Programming. `http://openmp.org/wp/`.

[7] C. Partridge, P. P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohalmi, T. Ma, J. Mcallen, T. Mendez, W. C. Milliken, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, and G. D. Troxel. A 50-Gb/s IP router. *IEEE/ACM Transactions on Networking*, 6(3):237–248, 1998.

[8] Portwell NAR-5520 Switching Appliance. `http://www.portwell.com/products/detail.asp?CUSTCHAR1=NAR-5522`.

[9] C. Shannon, E. Aben, kc claffy, and D. Andersen. The CAIDA Anonymized 2007 Internet Traces - Jan 2007. `http://www.caida.org/data/passive/passive_2007_dataset.xml`.

[10] C. Shannon, E. Aben, kc claffy, D. Andersen, and N. Brownlee. The CAIDA OC48 Traces Dataset - April 2003. `http://www.caida.org/data/passive/passive_oc48_dataset.xml`.

[11] Vyatta Dell860. `http://www.vyatta.com/products/vyatta_appliance_datasheet.pdf`.

# API Design Challenges for Open Router Platforms on Proprietary Hardware

Jeffrey C. Mogul, Praveen Yalagandula, Jean Tourrilhes, Rick McGeer,
Sujata Banerjee, Tim Connors, Puneet Sharma
HP Labs, Palo Alto
{Jeff.Mogul,Praveen.Yalagandula,Jean.Tourrilhes,Rick.McGeer,Sujata.Banerjee,Tim.Connors,Puneet.Sharma}@hp.com

## ABSTRACT

Most switch vendors have launched "open" platform designs for routers and switches, allowing code from customers or third-party vendors to run on their proprietary hardware. An open platform needs a programming interface, to provide switchlets sufficient access to platform features without exposing too much detail. We discuss the design of an abstraction layer and API designed to support portability between vendor platforms, isolation between switchlets and both the platform and other switchlets, high performance, and programming simplicity. The API would also support resource-management abstractions; for example, to allow policy-based allocation of TCAM entries among multiple switchlets.

## 1 INTRODUCTION

Traditionally, router and switch[1] platforms have either been commodity platforms running open but slow implementations, or proprietary hardware running closed but fast implementations. Most router vendors currently follow the closed-but-fast model, which gives them complete control over system quality, but has become a barrier to innovation.

Recently, major router vendors have initiated programs to provide *open router platforms* (ORPs), which allow third parties to develop software extensions for proprietary hardware. ORPs potentially support faster deployment of novel networking features; for example, one could deploy Stanford's OpenFlow [13] on an ORP.

While the typical vendor's approach to an ORP is to provide a Linux environment running on an x86 processor as part of the platform, the traditional Linux API is the wrong abstraction. These boxes are interesting precisely because they have specialized hardware features that standard Linux does not (should not) support.

We need an ORP API that offers controlled access to these hardware features. Ideally, this API would expose all of the functionality and performance of modern router hardware, while maintaining the useful properties of commodity operating systems: software portability between vendors, isolation between software components, easy management, etc. Such an API would also be the boundary between open-source upper layers, and lower layers that the router vendors insist on maintaining as proprietary trade secrets.

Previously, Handley *et al.* [7, 8] described XORP, an eXtensible Open Router Platform. XORP provides a nice abstraction for building relatively high-performance routers on top of commodity platforms. While XORP potentially could run on a proprietary-hardware open router platform (PHORP), we are not aware of such an implementation. We also believe that XORP's abstractions, such as its Forwarding Engine Abstraction (FEA), expose too little of the power of modern router hardware, and do not sufficiently address the scarcity of certain hardware resources.

In this paper we explore the design requirements for an "Open Router Proprietary-Hardware Abstraction Layer," or Orphal. Orphal's goals include support for portability of third-party components between different proprietary platforms; isolation between these components; exposing as much of the hardware's functionality as possible; and managing scarce hardware resources.

Casado *et al.* [4] argue that software-only routers are not fast enough, network processors are too complex to program, and hardware-based designs (including commodity forwarding chips) have been too inflexible. They propose a redesign of hardware-level forwarding mechanisms to provide a clean, simple, and flexible interface between hardware and software. We agree with them that the best path to flexible *and* efficient routers depends on co-evolving router hardware, extensible router software, and a clean interface between them.

Figure 1 shows how Orphal fits into a PHORP architecture. Orphal sits above the vendor-proprietary hardware and software, and also above the commodity hardware and operating system, although we see no reason to modify the standard OS APIs. (The figure shows Linux as the OS, but it could be any reasonable OS, and perhaps a virtual-machine layer as well.) In practice, Orphal would be implemented as a combination of device

[1]We use "router" and "switch" interchangeably in this paper.

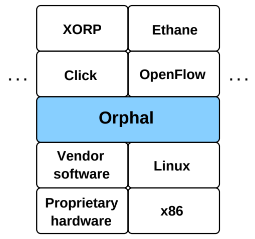


**Figure 1**: Layering in an open router platform

drivers and user-mode libraries.

One or more *switchlet* modules run above Orphal. In the figure, we show two: a Click [10]+XORP stack, and an OpenFlow [13]+Ethane [3] stack, but these are just examples. This is *not* an "active networks" approach; we expect switchlets to be installed by the router's owner.

This position paper describes some of the design challenges for Orphal. We first describe a high-level overview of a plausible design. Then, for concreteness, we focus on issues related to one particular kind of specialized hardware: Ternary Content Addressable Memories (TCAMs) used for line-rate lookups. This is motivated by our experience porting OpenFlow (see sec. 4).

## 2 Orphal API Design Overview

Orphal's goals include *resource management; controlled sharing; isolation; hardware reprogrammability; performance; portability;* and *manageability.* Orphal differs from the API of a general-purpose OS mostly because Orphal must expose interesting, router-specific hardware without sacrificing run-time efficiency.

**Resource management** A high-performance router is inherently a real-time environment, with potentially scarce resources both in the commodity computation platform, and in the proprietary hardware. Routers are often required to enforce QoS requirements, which cannot be maintained if the router itself mis-manages its resources. Orphal needs to support resource management, including allocation of resources among switchlets, consistent with the overall QoS policy and performance constraints of the system.

Which resources need to be managed? We can assume that the commodity OS will manage commodity-hardware resources (CPU, RAM, stable storage), while Orphal will manage router-specific resources such as TCAM entries, hash-table entries, buffer space, programmable ASICs, etc. We also want to manage power-related resources (powering down idle line cards, per-port rate scaling, etc.) using Orphal. One challenge is to define Orphal's resource management so that it is portable across a range of router hardware with various interesting kinds of resources; we believe that this can be done using vendor-specific switchlets that Orphal invokes via upcalls (see section 2.1).

**Controlled sharing** Orphal must provide controlled sharing of abstract resources such as forwarding-table entries, as well as the real resources (such as hash-table and TCAM entries) used to implement these abstractions.

For example, if two switchlets want to control the actions for packets for a given destination – e.g., a firewall switchlet and a QoS switchlet – how should Orphal decide which switchlets get that control? If two switchlets want to process the same packet, which one gets it first? We believe that prior work on kernel packet filters [15, 20] provides some useful models; for example, Orphal could assign precedence (priority) levels to each switchlet, and let each switchlet declare whether lower-precedence switchlets should see the packets it handles.

**Isolation** One goal of an ORP is to allow composition of switchlets from different third-party component vendors. While we need not assume that switchlets might be malicious, the potential remains for unexpected "feature interactions." (This is a problem even when all components come from the same vendor.) Two switchlets running on top of Orphal should not accidentally interfere with each other, either directly or indirectly. Thus, the system must prevent switchlets from interfering with each other's code and private state. Isolation is usually accomplished either with a process-like abstraction, or using a virtual machine abstraction. This choice is likely to be made by the router vendor, and Orphal should support either model, as transparently as possible.

**Hardware reprogrammability** We expect some router platforms to provide *programmable* hardware (not just *configurable* hardware, such as TCAM tables). For example, an ASIC in the packet-processing fast path could support programmability for deep-packet inspection (DPI) operations [9]; NetFPGA [16] is another example. Given these programmable features, should Orphal provide an API allowing switchlets to, for example, push arbitrary microcode into an ASIC, or would it be safer to simply provide access to a platform-defined library of such functions?

**Performance** Orphal must deal with many performance-related issues, such as support for multi-core parallelism in switchlet execution; prioritizing CPU sharing among switchlets; rate-limiting features of the platform; etc. We have neither the experience nor the space to discuss these further.

**Portability** Orphal must expose the platform's hardware details enough to support high performance, but without exposing too much detail: that would compromise portability, and perhaps isolation. This is a difficult challenge, especially since we lack enough experience to know what really matters. We describe, in sec. 4, our initial experiences trying to map OpenFlow's 10-tuple flow-description model onto a TCAM that supports 5-tuples.

**Manageability** Routers must already address many management issues, such as port and routing-protocol configuration. The introduction of open router architectures creates a new problem: given a multitude of separately developed switchlets, how does the router administrator create and preserve a stable configuration?

XORP, for example, provides a "router manager process" (rtmgr) [19] to handle some of these issues. Support for proprietary hardware probably complicates this task, because the introduction of a new switchlet can create new resource conflicts (e.g., not enough TCAM entries) and new feature interactions (competing uses for a given TCAM entry).

We believe the router manager will have to check that the system can support the switchlet's minimal requirements (e.g., that there are enough available TCAM entries for the switchlet to function) and to provide rollback to a previous configuration if a new one causes trouble.

The manager will also have to monitor each switchlet's dynamic resource consumption, including specialized hardware resources, so that the router administrator can make informed decisions.

We also expect administrators will want to upgrade a switchlet to a new version without rebooting the entire router. This may require Orphal support, especially to cleanly undo the hardware-related effects of an old (or failed) switchlet. For example, when a switchlet fails or is removed, its updates to the TCAM should be reversed.

## 2.1 What is a switchlet?

A *switchlet* is simply a module that runs on top of Orphal, with its own address space and thread(s) of control.[2] Orphal will support several switchlet categories, including:

- **per-packet switchlets**: These are invoked, similarly to Click *elements* [10], to handle specific packets. Since high-performance router designs try to avoid handling most packets in software, per-packet switchlets are mostly useful for exceptional packets.
- **per-flow switchlets**: Some router functions, especially for monitoring and sometimes for firewalling, are invoked once or a few times per flow. This is less likely to cause performance problems, although given mean flow lengths in the ballpark of 12 UDP packets to 50 TCP packets [2], such switchlets might still be reserved for exceptions.
- **control-plane functions**: These functions, such as routing protocols, management protocols, etc., typically are not directly related to the packet-forwarding fast path, and so are often handled in software. XORP provides a useful framework for these functions.
- **optimizer/helper modules**: We expect that the process of matching higher-level abstractions needed by switchlets to the lower-level hardware abstractions will require the use of optimization algorithms. Orphal invokes these via upcalls to optimizer switchlets. This form of policy-mechanism separation allows third parties to develop improved versions of these modules.
  Optimizer modules can also be used, for example, to provide a backing store for space-limited hardware resources. For example, Orphal could manage the hardware TCAM as a cache for a larger table managed by an optimizer module, in much the same way that an OS kernel manages a hardware Translation Buffer as a cache for its page tables.
  Additional "helper" switchlets can be used to provide policy-mechanism separation for functions such as detecting inter-switchlet conflicts in TCAM entries.

Orphal needs to balance switchlet portability against aggressive use of hardware functions that might not be present on all platforms. Thus, a switchlet can provide an optional software implementation for a function, to be used if Orphal cannot provide the necessary hardware support (either because it isn't there, or because it is oversubscribed).

For example, consider a Click module, such as the existing NetFlow package, that is most naturally implemented in hardware if the hardware is available. The module author could supply both a hardware-based (e.g., NetFPGA) version and a (less efficient) software-based version, and Orphal could transparently instantiate the most efficient version possible. (This leaves open the question of whether Orphal could feasibly change between versions dynamically; state synchronization and QoS maintenance might make this difficult.)

## 2.2 Example of Switchlets

We describe our initial experience implementing OpenFlow, and how it might be structured as switchlet, in sec. 4. Beyond that, we lack space to give detailed examples of possible switchlets, but here is a partial list:

- **Specialized firewall** switchlets could be triggered by DPI hardware to check unusual flows against security policies.
- **Specialized monitoring** switchlets could report on suspicious patterns of flow creations.
- **NAT** switchlets might require access to programmable packet-header rewriting hardware.

[2] Others have defined "switchlet" in different ways [1, 5, 17], but we can't think of a better term.

- **Dynamic VLAN** switchlets could implement setup protocols used to establish VLAN membership.

## 3 API Design Issues

The goal of Orphal is to provide a clean interface between router-specific programmable hardware, and switchlets running on general-purpose CPUs within the router platform. Routers often have a number of interesting hardware features, such as programmable DPI engines, TCAMs for route lookups, and other route-lookup hardware such as hash tables and programmable header extractors. Future routers might have additional specialized hardware, such as programmable packet-header rewriters.

In this paper we limit our detailed discussion to TCAMs, since they are widely used for high-speed forwarding, present some interesting challenges, and are the focus of our current implementation work (see sec. 4).

### 3.1 TCAM API and Resource Management

Most high-performance router hardware includes Ternary Content Addressable Memories (TCAMs). One can think of a CAM as a table whose rows each include a tag field to match against; the CAM returns the matching row (if any). In a TCAM, tag-field entries are composed not just of binary 1s and 0s, but also "X" or "don't care" values. TCAMs thus allow more compact representations of lookup tables whose tag values can include wildcards. Routers use TCAMs for functions such as IP address lookups and firewall lookups, where these wildcards are common.

While TCAMs are often the preferred solution for lookup functions, various TCAM parameters are constrained by expense (TCAM structures take a lot of die area) and power consumption (a TCAM lookup requires all rows to be active, and TCAMs consume ca. 15W/chip [21].) Thus, TCAMs present some challenges for an open router platform, and we explore these as an example of a larger set of challenges that the API must meet:

- **Limited tag-field size**: TCAM tag widths are typically limited, often to ca. 144 bits (enough for an IP/TCP 5-tuple) [14]. A single TCAM entry might therefore be insufficient to support a firewall-entry match in a single lookup, since (especially with IPv6), too many packet-header bits must be checked. This can force the hardware to support multiple lookups per packet. The API must allow switchlets to express such multi-lookup rules.
- **Limited number of rows**: TCAMs are typically limited to a few thousand rows. Thus, the platform must treat TCAM rows as a scarce resource, to be allocated among potentially competing switchlets, and the API must allow switchlets to express resource requirements.
- **Multiple "owners" for one row**: Two different switchlets might want packets that match the same TCAM row (e.g., "all TCP packets to port 80"); the API needs to manage these conflicts. (See sec. 3.4.)
- **Multiple matching rows**: Because TCAMs support wildcards, two different rows might match the same packet. But TCAM-based designs always return the lowest-index entry that matches the packet. Two switchlets might create distinct TCAM entries that either overlap, or where one covers the other; what should the system do in this case? The API needs to manage these conflicts, too. (See sec. 3.4.)
- **TCAM optimization**: Given an abstract set of matching rules, one can generate an optimized set of TCAM entries that provide the most compact (and hence most space- and energy-efficient) representation [12, 14].
- **TCAM update (insertion) costs**: TCAM-based designs generally must trade off efficient lookups against insertion costs, which can be as high as $O(n)$ in the number of rows [6]. The API might need to manage this tradeoff; it might also need to synchronize between updates and lookups (or else lookups could yield bad results during updates) [18].

### 3.2 A typical TCAM-based hardware design

Figure 2 sketches part of an idealized TCAM-based hardware design, to make some of these design challenges concrete. Each line card would have an instance, possibly serving several ports.

An incoming packet is first processed by a **pseudo-header generator**, adding to the real packet header such fields as a VLAN tag, the ID of the port where the packet arrived, etc. Assuming that the TCAM is not wide enough to do a full lookup in one step, the **header extractor** manages a multi-stage lookup; it recognizes certain high-level patterns (e.g., "IPv4 packet" or "IPv6 packet"), extracts the header fields used in each stage (e.g., first the layer-2 headers, then the layer 3+4 headers), passes these to the TCAM, and decides whether to do the next lookup stage.

Many routers use one or more **hash tables** in addition to the TCAM. Hash tables provide a cheaper mechanism for doing exact-match lookups, such as "what's the next hop for this flow?", while TCAMs are appropriate for more complex lookups – especially those including wildcards – typical of QoS and firewall (access control list) functions. For firewall functions, the line card might also include a **port-range classifier**, since arbitrary ranges of port numbers (e.g., "1023–65535") could consume too many TCAM entries. Liu [11] described a range classi-

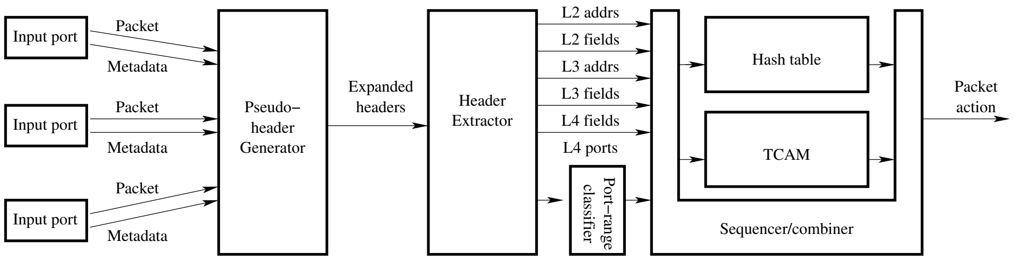


**Figure 2**: Idealized TCAM-based lookup path

fier that uses a modest-sized RAM-based lookup table.

Additional **sequencer/combiner** logic coordinates the multiple lookup stages and combines partial results to generate a final result, indicating the action to take with the packet, such as the next-hop address and the output switch port.

The TCAM, of course, is a programmable resource, but potentially so are the other functional blocks (pseudo-header generator, header extractor, port-range classifier, hash table, sequencer/combiner).

Unlike a more abstract API such as XORP, Orphal exposes all of these distinct programmable resources, since they have differing characteristics that could be exploited by sophisticated switchlets.

## 3.3 What should the TCAM API expose?

There are many ways to organize TCAMs and the associated hardware, and if switchlets are to be portable between hardware platforms, the API must either hide this variation, or expose it in a useful way. Given the challenges listed in section 3.1 (and there are others), perhaps it is implausible to create an API that provides any generality across models and vendors. However, we suspect that by choosing the right level of abstraction for exposing the TCAM hardware, Orphal can meet its goals.

For example, XORP exposes a high-level "forwarding engine abstraction" (FEA), but Orphal must expose a lower-level abstraction if the switchlets are to exploit specialized hardware features. There are things that cannot be expressed explicitly at the FEA level – for example, that certain rules should be stored in the hash table instead of the TCAM.

There is a useful API abstraction intermediate between a raw-hardware "TCAM row" and a high-level "forwarding table entry." Although a TCAM optimizer module will need access to the raw row-level version ("put these bits here"), most switchlets will use a paravirtualized view of the TCAM (PV-TCAM), which will enable Orphal to provide the controlled sharing, isolation, and resource management properties described in section 2. PV-TCAM rows look almost like real TCAM rows, but with some additional meta-information, and without a fixed mapping to actual hardware rows.

The TCAM-user API will need to provide certain functions, including (among many others):

- **tcamAddRow(tag, action, ordering)**: Used to add a row with a given tag value and action, and an intra-switchlet value to control how rules are ordered. Returns either an opaque handle for the row, or a failure indication.
- **tcamDeleteRow(handle)**: does the obvious thing.
- **tcamGetRow(handle)**: returns the corresponding TCAM entry, including statistics.
- **tcamRegisterInterest(handle, callbackFunction)**: specifies a switchlet function to be called with each packet that matches the row; the default is no callback. This is the way that switchlets can receive packets and/or discover flows.
- **tcamConflictCallback(handle, callbackFunction)**: If another, higher-priority switchlet creates a TCAM row that conflicts with the one associated with the handle, this callback informs the current switchlet that the row has been reassigned to the other switchlet's purposes. Section 3.4 discusses conflicts in more detail.

The TCAM-optimizer API will need to provide certain functions, including (among many others):

- **Loading a set of TCAM rows**: The optimizer's output needs to be loaded into the TCAM; possibly this will require some synchronization so that packets are not processed when the TCAM is in an inconsistent state.
- **Obtaining the abstract state of the TCAM database**: The optimizer's input from Orphal will consist primarily of the union of the TCAM-user requests, plus some policy settings provided by a management layer.
- **TCAM usage statistics**: Typically, TCAMs support hit counters for each row.

## 3.4 TCAM row conflcts

Multiple switchlets might try to create conflicting TCAM rows. Orphal's approach is to detect these conflicts and resolve them using an inter-switchlet priority ranking. (This seems like the simplest approach, but we are exploring others.) When a low-ranking switchlet tries to create a new row that conflicts, Orphal simply rejects the attempt. However, a high-ranking switchlet can create a row that conflicts with an existing lower-ranking row, in which case Orphal removes the low-ranking row, inserts the new one, and informs (via **tcamConflictCallback**) the low-ranking switchlet that it has lost the row. Orphal lets the switchlets figure out what to do in that case.

It is not easy to define what a "conflict" is, and conflict-checking is an expensive (NP-complete) process [12], so checking should not be embedded in Orphal *per se*. Instead, Orphal supports plug-in conflict-checking implementations using "helper" switchlets.

## 4 Our experience with OpenFlow

OpenFlow [13] is a centrally-managed flow-based network where switches are simple forwarding engines that classify packets into flows and act on them according to a policy supplied by a central controller. We are porting OpenFlow to a commercial switch, the HP ProCurve model 5406zl, and here report some of the challenges.

OpenFlow could run entirely in the switch's software, but that would not support line-rate forwarding, so we need to use the TCAM hardware. The controller expects a flexible flow classifier, so the tricky part is to match OpenFlow's flow descriptions (a 10-tuple of physical ingress port and VLAN IDs; Ethernet source, destination and type; and the standard IP/TCP 5-tuple) with what the hardware supports. The challenges include:

- **Limited number of TCAM rows:** means not all flows can be classified in hardware. So, we insert a final wild card entry in the TCAM to divert packets from other flows to the software stack. We try to minimize such slow-path packets by keeping busy flows in the TCAM.
- **Limited tag-field size:** TCAM widths (e.g., 144 bits) are typically chosen to support lookup on the IP/TCP 5-tuple ($32 + 32 + 16 + 16 + 8 = 104$ bits). OpenFlow's 10-tuple, which includes 48-bit MAC addresses, is too big for such TCAMs. However, our switch supports multiple TCAM lookups/packet at line rates, so we support the OpenFlow tuple with a multi-stage lookup.

When a packet arrives for an unknown flow, the OpenFlow forwards it to the central controller, which updates that switch (and perhaps others) with new flow-specific forwarding rules. Using Orphal, we could implement OpenFlow as a switchlet that forwards no-match packets to the controller, and installs controller-supplied responses into the forwarding table. The controller deals in 10-tuples; we intend to use a helper switchlet to convert these into patterns that the switch's TCAM can handle. This helper could also be used by other switchlets, such as firewalls.

## 5 Summary

Open router platforms offer tremendous flexibility, but exploiting the rich variety of router hardware creates complexity. Our goal for Orphal is to tame that complexity; we hope to demonstrate working systems in the near future.

## References

[1] D. S. Alexander and J. M. Smith. The Architecture of ALIEN. In *Proc. Intl. Working Conf. on Active Networks*, pages 1–12, 1999.

[2] M. Arlitt. Personal communication, 2008.

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *Proc. SIGCOMM*, pages 1–12, Aug. 2007.

[4] M. Casado, T. Koponen, D. Moon, and S. Shenker. Rethinking Packet Forwarding Hardware. In *Proc. HotNets*, Oct. 2008.

[5] N. da Fonseca, J. Castro, A.P., and A. Rios. A procedure for resource allocation in switchlet networks. In *Proc. GLOBECOM*, volume 2, pages 1885–1888, Nov. 2002.

[6] B. Gamache, Z. Pfeffer, and S. P. Khatri. A fast ternary CAM design for IP networking applications. In *Proc. ICCCN*, pages 434–439, Oct. 2003.

[7] M. Handley, O. Hodson, and E. Kohler. XORP: an open platform for network research. *SIGCOMM CCR*, 33(1):53–57, 2003.

[8] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov. Designing extensible IP router software. In *Proc. NSDI*, pages 189–202, Boston, MA, 2005.

[9] HP ProCurve. ProVision$^{TM}$ ASIC: Built for the future. `http://www.hp.com/rnd/itmgrnews/built_for_future.htm`.

[10] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *TOCS*, 18(3):263–297, 2000.

[11] H. Liu. Efficient Mapping of Range Classifier into Ternary-CAM. In *Proc. Hot Interconnects*, pages 95–100, Aug. 2002.

[12] R. McGeer and P. Yalagandula. Minimizing Rulesets for TCAM Implementation. Tech. Rep. HPL-2008-106, HP Labs, 2008.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM CCR*, 38(2):69–74, 2008.

[14] C. R. Meiners, A. X., and L. E. Torng. Algorithmic Approaches to Redesigning TCAM-Based Systems. In *Proc. SIGMETRICS*, June 2008.

[15] J. Mogul, R. Rashid, and M. Accetta. The packer filter: an efficient mechanism for user-level network code. In *Proc. SOSP*, pages 39–51, 1987.

[16] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. A Programming Model for Reusable Hardware in NetFPGA. In *Proc. PRESTO*, Aug. 2008.

[17] J. E. van der Merwe and I. M. Leslie. Switchlets and Dynamic Virtual ATM Networks. In *Proc. 5th IFIP/IEEE Intl. Symp. on Integrated Network Management*, pages 355–368, 1997.

[18] Z. Wang, H. Che, and S. K. Das. CoPTUA: Consistent Policy Table Update Algorithm for TCAM without Locking. *IEEE Trans. Comput.*, 53(12):1602–1614, 2004.

[19] XORP Project. XORP Router Manager Process (rtrmgr) Version 1.4. `http://www.xorp.org/releases/1.4/docs/rtrmgr/rtrmgr.pdf`, 2007.

[20] M. Yuhara, B. N. Bershad, C. Maeda, and J. E. B. Moss. Efficient packet demultiplexing for multiple endpoints and large messages. In *Proc. USENIX Winter Tech. Conf.*, pages 153–165, 1994.

[21] F. Zane, G. Narlikar, and A. Basu. Coolcams: power-efficient TCAMs for forwarding engines. In *Proc. INFOCOM*, volume 1, pages 42–52, 2003.

For additional related work, please see an expanded version at `http://www.hpl.hp.com/techreports/2008/HPL-2008-108.html`

# Interference Avoidance and Control [*]

Ramakrishna Gummadi[*] Rabin Patra[†] Hari Balakrishnan[*] Eric Brewer[†]
[*]MIT CSAIL [†] UCB

## Abstract

The throughput of a wireless network is often limited by interference caused by multiple concurrently active nodes. The conventional approach of using a "one-transmission-at-a-time" MAC protocol to combat interference leads to a significant loss of achievable throughput compared to schemes such as interference cancellation that keep all transmitters active simultaneously. Unfortunately, interference cancellation incurs significant computational complexity, and cannot be implemented with commodity hardware.

In this paper, we propose a practical approach for improving the throughput of interfering nodes using variable-width frequency allocation. We show that variable-width channels provide significant theoretical capacity improvements, comparable to interference cancellation for infrastructure networks. We design an algorithm that reduces interference by assigning orthogonal variable-width channels to transmitters. We evaluate a prototype implementation of this algorithm on an outdoor wireless network with ten long-distance links configured into point-to-point and point-to-multipoint topologies. We observe a throughput improvement of between 30% and 110% compared to the existing fixed-width channel allocation.

## 1 Introduction

Our goal is to build wireless networks with high aggregate throughput. So, extracting transmission concurrency is essential. But there is a trade-off: higher concurrency generally means higher interference. Previous work has been in one of two areas: MAC protocols that attempt to extract concurrency, and interference cancellation and its variants [8, 11].

For 802.11 networks, existing MAC protocols such as CSMA, Time-Based Fairness (TBF) [19] and CMAP [21] regulate concurrent transmissions carefully to ensure that collisions resulting from interference remain low. They allow only one transmitter to be active within a given channel at any time there is a risk of interference (i.e., whenever concurrent transmissions result in either packet being lost). However, serializing interfering transmissions imposes a fixed upper bound on the aggregate throughput, regardless of the number of interfering transmitters. Therefore, the average throughput per transmitter decreases with the number of interfering transmitters.

In contrast, interference cancellation (IC) deals with distinguishing between concurrently transmitted signals by demodulating and decoding all the interfering signals simultaneously. The theoretical concepts behind IC were developed in the 1960s [7, 18], especially in the context of spread-spectrum systems. Recently, researchers have investigated IC and related alternatives such as interference alignment and ZigZag decoding to mitigate the problems caused by interference [4, 8, 11]. Unfortunately, such receivers involve significant complexity because separating overlapping signals requires considerable signal processing. Moreover, the running time of such algorithms grows at least linearly with the number of concurrent transmissions a receiver overhears, none of which might ultimately be intended for the receiver.

In this paper, we ask the following question: is it possible approximate the optimal throughput provided by IC using simpler techniques, and, if so, under what conditions? We demonstrate a spectrum allocation algorithm that assigns variable-width channels to transmitters and keeps all transmitters active concurrently, thereby achieving a higher capacity than any fixed-width channel assignment scheme such as CSMA or TBF. Our result suggests that we should control interference while maintaining high concurrency.

The allure of using variable-width channels to control interference is that commodity wireless chipsets, such as Atheros and PRISM, support variable-width channels ranging from at least 5 MHz to 40 MHz [12, 13]. Recently, Moscibroda et al. [13] have studied variable-width channels to improve network throughput by allocating spectrum to APs (Access Points) based on their load. Similarly, Chandra et al. [5] have studied variable-width channels to improve a single link's throughput and energy efficiency. Here, we study variable-width channels for their ability to improve throughput among multiple interfering transmitters with backlogged flows (i.e., flows which always have some data to send).

We show that, for infrastructure networks, using orthogonal variable-width channels on the uplink from the clients to the AP not only achieves the optimum sum-capacity of $n$ concurrent transmitters predicted by Shannon's theorem, but also improves the aggregate throughput over any fixed-width TDMA scheme such as CSMA or TBF by an additional $\Theta(\log_2(n))$ bits/s/Hz. The intuition is that maintaining the transmitters on non-overlapping channels theoretically eliminates interference, while narrowing their channel widths allows the total transmitted power to be the sum of all transmitters. Thus, the aggregate transmit and received powers are increased, without adding interference. We believe that this approach also exhibit good gains for mesh networks, though we do not discuss that setting in this paper.

[*]This work was supported by the National Science Foundation under awards CNS-0721702 and CNS-0520032, and by Foxconn.

We use this intuition to develop a spectrum allocation scheme called VWID (Variable WIDth channels). When there are $n$ concurrent transmissions on a given channel, VWID attempts to split the spectrum into $n$ non-overlapping variable-width channels. The width of each channel is allocated to maximize each transmitter's throughput, subject to the constraint that no interfering transmitter receives lower throughput than it would have with fixed-width allocations. Thus, VWID provisions spectrum to reduce interference specifically, and complements MAC protocols such as CSMA that provide additional functionality such as ACKs and retransmissions to deal with both noise and interference.

We have implemented a VWID prototype and conducted a preliminary evaluation on an outdoor wireless testbed consisting of ten medium to long-distance links deployed in an urban area. We configured the testbed into point-to-point and point-to-multipoint topologies, thus representing networks typically encountered in rural point-to-point [15] and point-to-multipoint [16] settings. Even though our implementation is unoptimized, we find that VWID provides per-node throughput improvements ranging from 30%–110% by provisioning orthogonal variable-width channels to reduce interference.

## 2 Variable-width Channels Improve Throughput

We analyze the throughput improvement produced by encouraging multiple concurrent transmissions using orthogonal variable-width channels compared to TDMA schemes such as CSMA and Time-Based Fairness (TBF) that use fixed-width channels. Using variable-width channels and enabling concurrent transmissions on these channels always increases the aggregate throughput compared to using fixed-width channels, because the total transmitted and received powers are increased, while interference is still kept in check. We can also address inter-node fairness by using sufficient channel widths that guarantee that every transmitter obtains at least the throughput it obtains under the original fixed-width allocation.

We consider a single cell with $n$ clients and an AP. The AP has a single radio and antenna. Our primary result is that providing $n$ concurrent transmissions between the AP and the $n$ clients using orthogonal variable-width channels, whose width is proportional to received SINRs (signal to interference plus noise), can achieve higher aggregate throughput (by an additional $\theta(\log_2(n))$ bits/s/Hz) beyond the status quo.

Assume that the transmissions between the the clients and the AP are in the uplink, and that there is demand on all $n$ links. Consider two backlogged transmitters 1 and 2 whose signals are received with powers $P_1$ and $P_2$. The receiver noise power is $N$ per Hz. If transmitter 1 alone is active, the capacity $C_1$ of 1, assuming a Gaussian channel, is given by the Shannon-Hartley theorem: $C_1 = \log_2(1+\frac{P_1}{N})$

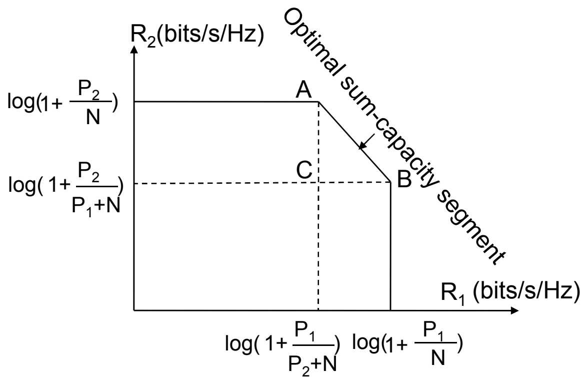


**Figure 1**: Achievable throughputs and the optimal capacity pentagon.

bits/s/Hz [7, 20]. Transmitter 1 can achieve any throughput rate $R_1$ that is less than $C_1$ [7, 20].

If both 1 and 2 are concurrently active, the theoretical capacity achievable by the two users simultaneously is called the sum-capacity. It consists of all throughput rate pairs $(R_1,R_2)$ such that:

$$\begin{aligned} R_1 &< \log_2(1+\frac{P_1}{N}) \text{ bits/s/Hz,} \\ R_1 &< \log_2(1+\frac{P_1}{N}) \text{ bits/s/Hz,} \\ R_1+R_2 &< \log_2(1+\frac{P_1+P_2}{N}) \text{ bits/s/Hz.} \end{aligned} \tag{1}$$

The achievable rates boundary, called the Cover-Wyner pentagon [1], is shown in Figure 1. The line segment A–B with slope $-1$ represents the optimal sum-capacity and is given by $R_1+R_2 = \log_2(1+\frac{P_1+P_2}{N})$. The reason is that, no matter how the two users code their transmissions, independently or cooperatively, it is not possible for them to exceed the capacity limit that occurs when there is a single user with total received power $P_1+P_2$. If both transmitters send on same frequencies, the rate pair at point A on the optimal sum-capacity segment in Figure 1 can be achieved by successive interference cancellation, in which the receiver first treats 2's signal as noise, recovers 1's signal, subtracts 1's signal from the total signal, and finally decodes 2's signal. Point B is vice-versa.

If we use variable width channels for 1 and 2 such that the total width is equal to the spectrum available to the receiver, we achieve non-interfering throughput rates for 1 and 2 that are given by:

$$\begin{aligned} R_1 &< \alpha\log_2(1+\frac{P_1}{\alpha N}) \text{ bits/s/Hz,} \\ R_2 &< (1-\alpha)\log_2(1+\frac{P_2}{(1-\alpha)N}) \text{ bits/s/Hz.} \end{aligned} \tag{2}$$

where $\alpha$ is the fraction of the spectrum allocated to 1 ($0 \leq \alpha \leq 1$). The noise term for $R_1$ in Equation 2 is reduced by a factor $\alpha$ because the signal is now confined to a narrower band, while noise still occupies the entire band with power $N$ per Hz.

**Theorem 2.1.** *If transmitters 1 and 2 are continuously backlogged, then the aggregate throughput achieved with variable width channels is strictly higher than that with any TDMA scheme such as CSMA or TBF.*

*Proof.* The total rate $R$ is given by:

$$\begin{aligned} R &= R_1 + R_2 \\ &= \alpha \log_2(1+\frac{P_1}{\alpha N}) + (1-\alpha)\log_2(1+\frac{P_2}{(1-\alpha)N}) \text{ bits/s/Hz.} \end{aligned}$$

Maximizing $R$ by setting $\frac{d}{d\alpha}R = 0$ gives $\alpha = \frac{P_1}{P_1+P_2}$, at which value $R = \log_2(1+\frac{P_1+P_2}{N})$ bits/s/Hz, which is optimal. Thus, we achieve the optimal throughput when transmitters are assigned channel widths proportional to their received power at the AP.

But no TDMA scheme, such as CSMA or TBF, is optimal (i.e., its throughput does not lie on the A–B segment) because TDMA only keeps one transmitter active at a time, thereby reducing the total transmitted and received powers. In particular, we calculate the CSMA and TBF throughputs below.

**CSMA throughput.** To a first order, CSMA allows equal number of channel accesses to nodes. The total achievable capacity under CSMA is as follows: Transmitter 1 takes $\frac{1}{R_1}$ time to send a bit, while 2 takes $\frac{1}{R_2}$ time to send its bit. Thus, CSMA sends 2 bits in time $\frac{1}{\frac{1}{R_1}+\frac{1}{R_2}}$. Thus, CSMA rate is $\frac{2}{\frac{1}{\frac{1}{R_1}+\frac{1}{R_2}}} = \frac{2R_1R_2}{R_1+R_2} \leq \sqrt{R_1R_2}$ because the arithmetic mean of two positive numbers is not smaller than the geometric mean. Substituting $R_1 = \log_2(1+\frac{P_1}{N}), R_2 = \log_2(1+\frac{P_2}{N})$, we find that this rate is less than the optimal rate $R = \log_2(1+\frac{P_1+P_2}{N})$. Moreover, the relative performance of CSMA to optimal can be seen to be arbitrarily bad if, say, $P_1 << P_2$, because transmitter 1 ends up monopolizing the channel.

**TBF Throughput.** TBF allows equal channel access and fares better than CSMA, but its capacity is also lower than the optimal power-proportional variable-width allocation. This is because, in one second, transmitter 1 sends $R_1$ bits and transmitter 2 sends $R_2$ bits. So, the achieved rate=$\frac{R_1+R_2}{2} < R = \log_2(1+\frac{P_1+P_2}{2})$, again using basic algebra. In the worst case (i.e., when $P_1 << P_2$), TDMA's rate is half the optimal power-proportional allocation with two transmitters. $\square$

To obtain more insight into how concurrent transmissions improve throughput, consider an example with two transmitters $t_1$, $t_2$ whose SINRs at the receiver are 1 each. So, $t_1$ and $t_2$ achieve a throughput of $\log_2(1+1) = 1$ bit/s/Hz individually. When they transmit concurrently, a MAC such as CSMA shares the channel by time-division multiplexing it, so that each transmitter achieves a rate of 0.5 bit/s/Hz. On the other hand, dividing the channel into two and making $t_1$ and $t_2$ transmit concurrently allows each transmitter to achieve a throughput of $\frac{1}{2}\log_2(1+2) = 0.79$ bit/s/Hz, as shown in Figure 2. Each transmitter thus improves its throughput by about 30%, while the aggregate throughput increases by about 60%. Moreover, this throughput gain of variable-width channels relative to CSMA increases both with the imbalance between the received signal strengths of the concurrent transmitters, and with the number of concurrent transmitters. For example, if $t_1$ is 8 times (or 9 dB) stronger than $t_2$ (which can happen frequently with 802.11), the throughput improvement with variable-width channels increases to more than $2\times$.

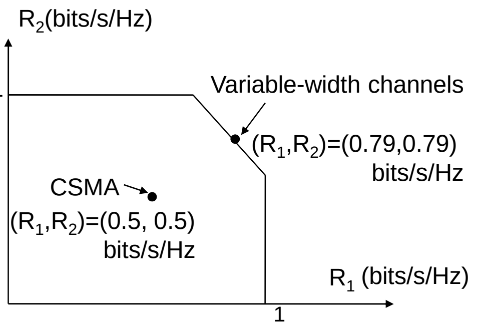


**Figure 2**: Throughputs of two transmitters when SINR=1.

Variable-width channels not only achieve higher throughput than any TDMA scheme with two concurrent transmitters, but we can also show that, $n$ concurrent transmitters using variable-width channels can improve their aggregate throughput by an additional $\theta(\log_2(n))$ bits/s/Hz over TDMA. The reason is that the aggregate capacity of $n$ transmitters using variable-width channels is $\log_2(1+\frac{nP}{N})$ bits/s/Hz, assuming that the received powers $P$ of all transmitters are equal. But TDMA schemes can only achieve a capacity of $\log_2(1+\frac{P}{N})$ in this case. So, for large $n$, variable-width channels provide an additional $\theta(\log_2(n))$ bits/s/Hz increase in aggregate capacity.

## 3 VWID Design and Implementation

Using the insight that variable-width channels improve throughput, we develop an initial version of a variable-width channel assignment algorithm called VWID.

Our platform consists of high power Atheros 802.11a Ubiquiti XR5 radios (600 mW) that work in the 5 GHz spectrum. We the Ubiquiti radio driver that allows variable channel widths (5, 10 and 20 MHz). While the normal 20 MHz-wide channel supports a maximum bit-rate of 54 Mbps according to the 802.11a standard, the half-width (i.e., 10 MHz) channel supports up to 27 Mbps, while the quarter-width (i.e., 5 MHz) channel supports up to 13.5 Mbps. In practice, we find that the achievable UDP throughput for outdoor links in our testbed is about 10 Mbps, due to interference [6] and multipath [2]. We found that both 10 MHz and 20 MHz channels attain this throughput, and that the 5 MHz channel obtains more than 8 Mbps.

Given a chunk of spectrum and a set of mutually interfering links, VWID assigns non-overlapping variable-width

**Algorithm 3.1:** ALLOCCHANNELS(InterferingLinks $n$)

```
Step 1 : Measure throughput $t[i]$ of each link $i$ with $n$ interferers
Step 2 : for $j \leftarrow 0$ to $n-1$
    do { Step 3 : Measure $t'[i,c]$ of link $i$, channel $c$ in $\binom{n}{j}$ choices
         Step 4 : if $t'[i,c] < t[i]$ eliminate channel $c$
Step 5 : Return the channel $c$ with highest $t'[i,c]$ for each link $i$
```

**Figure 3**: VWID channel selection algorithm.

(i.e., 5, 10 or 20 MHz) channels to these links so as to control interference. VWID only decreases the channel width for a link if doing so increases its throughput, thereby maintaining fairness. However, even with the fairness constraint, because VWID keeps every transmitter active while controlling interference, we find that per-node throughputs are higher for all links in many ($> 90\%$) scenarios, although these throughputs may be lower than those without the fairness constraint.

Our current implementation of VWID assigns variable-width channels to links within a single 20 MHz channel. The current best practice is to operate outdoor long-distance point-to-point or point-to-multipoint networks on a single channel because of spectrum scarcity, hardware limitations, and ease of management [3, 14, 15, 16]. So, while VWID can in principle handle any amount of spectrum, we have only instantiated and evaluated VWID for selecting 5, 10 or 20 MHz channel widths for a link.

In addition to selecting the channel width, VWID must also select the channel positions for 5 and 10 MHz channels within the 20 MHz channel. For example, assigning 5 MHz channels at 5.185 GHz and 5.195 GHz for two interfering links might be better in practice than assigning channels at 5.185 GHz and 5.19 GHz, because the former provides more channel separation even if neither provides perfect orthogonality. So, for every link, VWID considers 4 choices for placing 5 MHz channels and 2 choices for placing 10 MHz channels, in addition to retaining the 20 MHz channel option. So, we have seven channel choices for each link.

VWID measures the throughput of each link for the seven channel-width choices under interference from other interfering links, and picks the channel and channel width that provides the highest throughput for that link. While we have shown that a power-proportional channel width allocation is optimal (§2), since commodity cards do not report SINR measurements accurately, we use throughput measurements. While the worst-case complexity of VWID is $7^n$ for $n$ interfering links, we can prune the search space because we can reject channel widths and link combinations that violate the fairness constraint. For example, if a 5 MHz-wide channel tested under no interference is unable to provide more throughput than with a 20 MHz channel under interference, we can reject it immediately from all possible combinations with other links. Thus, in practice, VWID is efficient (for example, it only considers 16 combinations for four interfering links used in §4). The pseudocode for VWID is shown in Figure 3. We defer the study of a more efficient channel-width assignment algorithm and its run-time dynamics such as measurement overhead, channel allocation effectiveness and stability for future work.

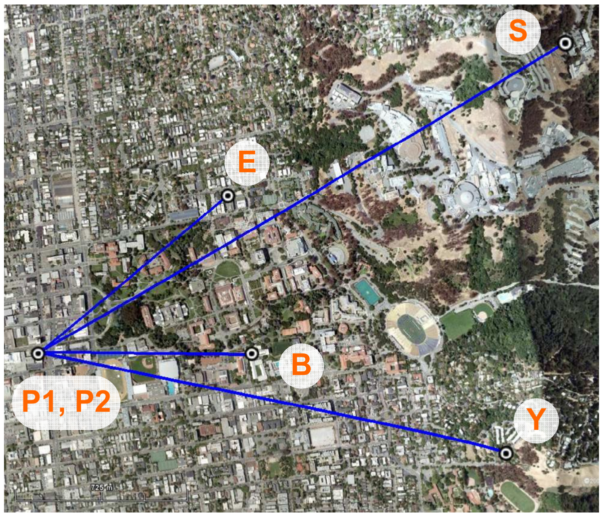


**Figure 4**: A point-to-multipoint topology configured on the outdoor testbed.

## 4 EVALUATION

We ran our experiments on our campus testbed, which consists of 6 wireless nodes and 10 links, 8 of which ranged from 1 km to 4 km 4, and 2 of which are co-located between different radios at P (Figure 4). Subsets of these links interfere with one another at either end-point, and each link interferes with at least one other link. The wireless nodes are based on 266 MHz x86 Geode single board computers running Linux kernel 2.6.19.2. The node at P has three wireless radios, the one at B has two radios and all the other nodes (S, B, E and Y) have one radio each. The nodes have directional antennas of 25 dBi gain. However, because of the relatively short distances involved, we were able to configure the links into various topologies such as point-to-point and point-to-multipoint by assigning the right transmit powers to the links. We selected a fixed bit-rate for each radio based on the maximum sustainable throughput (i.e., without getting disconnected after a while) across all its links.

We chose this outdoor setup because researchers have observed that interference imposes significant limits on achieved throughput, regardless of the supported bit-rates [6]. We modified the base driver to give us more control over MAC layer parameters such as disabling of ACKs, and changing the retransmission and Clear Channel Assessment (CCA) behavior. We experimented with various CCA settings that regulate the backoff behavior based on detected energy from concurrent transmissions. We disabled the CCA altogether and also varied the CCA energy-detection threshold between the card's minimum and maximum values. We measured unidirectional UDP and (bidirectional) TCP throughput under various CCA, ACK, and

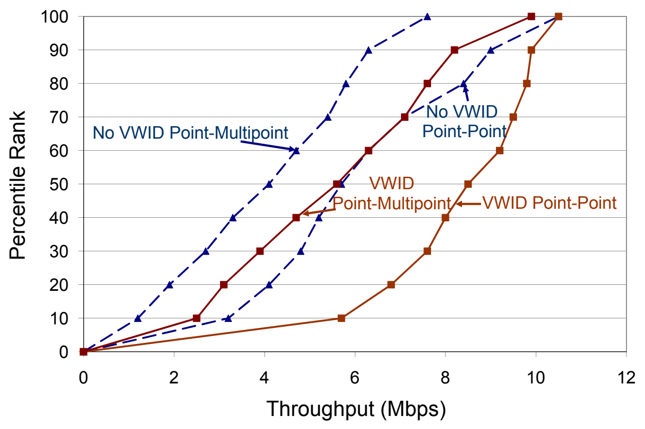


**Figure 5**: VWID increases CSMA throughput by 30%–110%.

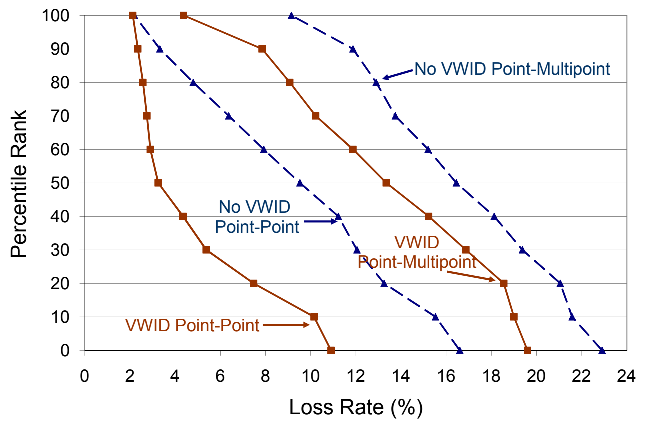


**Figure 6**: Loss rates correlate inversely with throughputs.

retransmissions configurations. We present results for UDP throughput with and without VWID under CSMA with the default ACKs, CCA and retransmission settings, because it achieved the highest throughput while maintaining fairness. While MAC configurations with CCA or ACKs switched off provide higher throughput for stronger links, they are either unstable or highly unfair to weaker links, so we do not consider them here.

Figure 5 shows the bi-directionally averaged percentile throughputs of one-way UDP flows across ten links as a CDF, and Figure 6 shows the corresponding loss rates as a CDF. Our main result is that, except for a high-throughput local link between two radios at node B, VWID improves throughput between 30%–110%. The highest improvements are for low-throughput links in low-throughput point-multipoint networks, because they suffer most from interference effects. So even modest interference relief is significant, which is a desirable outcome. While CCA mitigates some collisions, seven of the links have hidden terminals, leading to collision losses. The loss rates results exhibit good inverse correlation with the throughput plots and confirm that VWID controls interference and reduces losses, even if it means assigning narrower-width but orthogonal channels. Links in the point-multipoint topology show the biggest improvement, followed by point-point links. The reason is that point-multipoint links are forced to share bandwidth if there are not enough radios at the APs (which is the case with APs at location P in Figure 4).

We also observed that, although TCP obtains lower throughput than UDP, its relative gains with VWID are higher. The reason is that the impact of reduced interference is more significant, as in the case with point-multipoint links in Figure 5. While we have used VWID with a CSMA MAC because it provided the highest throughput in our setup, our results are also applicable to new MAC protocols based on TDMA [15, 17] that have been proposed to avoid interference and provide concurrency with links tens of kilometers long. The motivation behind these these TDMA protocols is that nodes with multiple wireless radios operating on the same wireless channel are constrained from transmitting on one radio while simultaneously receiving on another radio. While this scheduling eliminates collisions, it forces the wireless nodes to synchronize their transmissions on all their outgoing links (and, similarly, receptions), thus making the scheduling of links and flow allocation more difficult [14]. We found that VWID creates more non-overlapping variable-width channels that relieves this scheduling pressure.

## 5 Related Work

Current networks use either interference suppression on a packet-by-packet basis using MAC protocols [10, 14, 15, 16, 19, 21], or cope with interference using interference cancellation and related techniques such as interference subtraction, interference alignment and ZigZag decoding [4, 7, 8, 11, 18, 20, 22]. We have introduced the idea of interference control as a potential practical alternative, in which multiple transmitters operate concurrently, but take precautions to ensure they do not interfere with one another significantly. We use variable-width channels to achieve interference control, and ensure their orthogonality to avoid interference further.

While commodity hardware has supported variable-width channels out of necessity of narrow-width operation outside the unlicensed bands, this potential seems to have been recognized only recently. Moscibroda et al. [13] have used them for adjusting an AP's channel width based on load, while Chandra et al. [5] have examined their properties in detail for the single-link case. We complement them by examining interference control using variable-width channels. As newer standards such as 802.11-2007 mandate narrow-width channels even in unlicensed bands, we can expect more commodity hardware to offer variable-width channel support.

## 6 Conclusions and Future Work

We have shown that maintaining high concurrency by keeping multiple transmitters active concurrently, while controlling interference, increases the total system power without increasing interference, and hence increases aggregate throughput. We have examined the theoretical and practical potential of controlling interference using variable-width channels. We have developed and implemented a preliminary channel allocation algorithm called VWID based on insights from a theoretical analysis of infrastructure networks. We evaluated VWID on a small campus testbed of outdoor links configured into point-to-point and point-to-multipoint topologies, and observed up to 2x throughput improvements with narrower-width but orthogonal channels that reduce interference, and, consequently, packet losses.

Our analysis and evaluation of variable-width channels are by no means complete, and point to several pieces of future work. First, while we believe our analysis extends to mesh networks, we would like to characterize the capacity region of mesh networks with variable-width channels. Further, our current VWID algorithm is simplistic, in that it uses a brute-force algorithm that has exponential worst-case complexity. We are currently enlarging the campus testbed and deploying VWID to carry real Internet traffic to residential users. We also plan to deploy VWID in wireless networks used in developing regions that we have access to, and learn more about the strengths and weaknesses of interference control. Also, in an accompanying paper [9], we study throughput improvement strategies for bursty traffic using spread-spectrum codes, and we plan to extend them to variable-width frequencies.

**Acknowledgments.** We thank Mythili Vutukuru and the anonymous reviewers for their valuable comments.

## REFERENCES

[1] R. Ahlswede. Multi-way communication channels. In *ISIT'71*.

[2] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom'05*.

[3] S. Biswas and R. Morris. Opportunistic Routing in Multi-Hop Wireless Networks. In *HotNets'03*.

[4] V. Cadambe, S. A. Jafar, and S. Shamai. Interference alignment on the deterministic channel and application to fully connected AWGN interference networks. Nov 2007. URL `http://arxiv.org/abs/0711.2547v1`.

[5] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. In *SIGCOMM'08*.

[6] D. Gokhale, S. Sen, K. Chebrolu, and B. Raman. On the feasibility of the link abstraction in (rural) mesh networks. In *INFOCOM'08*.

[7] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[8] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *SIGCOMM'08*.

[9] R. Gummadi and H. Balakrishnan. Wireless networks should spread spectrum based on demands. In *HotNets'08*.

[10] R. Gummadi, R. Patra, S. Nedevschi, S. Surana, and E. Brewer. A radio multiplexing architecture for high throughput point to multipoint wireless networks. In *WiNS-DR'08*.

[11] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall. Interference Cancellation: Better receivers for a new Wireless MAC. In *HotNets'07*.

[12] HFA3863 Baseband Processor Data Sheet, `http://pdos.csail.mit.edu/decouto/papers/802-11-docs/hfa3863.ps`.

[13] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, and P. Bahl. Load-aware spectrum distribution in wireless LANs. In *ICNP'08*.

[14] S. Nedevschi, R. Patra, S. Surana, S. Ratnasamy, L. Subramanian and E. Brewer. An adaptive, high performance MAC for long-distance multihop wireless networks. In *MobiCom'08*.

[15] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian and E. Brewer. WiLDNet: Design and implementation of high performance WiFi based long distance networks. In *NSDI*, 2007.

[16] R. Patra, S. Surana, S. Nedevschi, and E. Brewer. Optimal scheduling and power control for TDMA based point to multipoint wireless networks. In *NSDR*, 2008.

[17] B. Raman and K. Chebrolu. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *MOBICOM'05*.

[18] P. Stavroulakis. Interference suppression techniques: A twenty-year survey. *International Journal of Satellite Communications and Networking, Vol. 21, No. 1, 2003*.

[19] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate WLANs. In *USENIX'04*.

[20] D. Tse and P. Viswanath. *Fundamentals of wireless communication*. Cambridge University Press, 2005.

[21] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing Exposed Terminals in Wireless Networks. In *NSDI'08*.

[22] L.-L. Xie and P. R. Kumar. A network information theory for wireless communication: Scaling laws and optimal operation. In *IEEE Transactions on Information Theory, Vol 50, No. 5, 2004*.

# Wireless ACK Collisions Not Considered Harmful

Prabal Dutta[†] Răzvan Musăloiu-E.[‡] Ion Stoica[†] Andreas Terzis[‡]

[†]*Computer Science Division*
*University of California, Berkeley*
*Berkeley, California 94720*

[‡]*Computer Science Department*
*Johns Hopkins University*
*Baltimore, Maryland 21218*

## ABSTRACT

We present an acknowledged anycast primitive that allows a node to wirelessly transmit a packet and efficiently determine that at least one neighbor successfully received it. The initiator transmits a single packet to a unicast, multicast, or broadcast address and all nodes that match the destination respond with identical acknowledgment packets automatically generated by the hardware. Although these acknowledgments interfere, they usually do so non-destructively, so the initiator can decode their superposition. We call such an exchange a *Backcast* and show that this operation is feasible using a commodity radio, general because it enables multiple network services, efficient because it is independent of the neighborhood size and runs in constant time, and scalable because it works with no fewer than a dozen interfering acknowledgments.

## 1 INTRODUCTION

Anycast is a fundamental and widely used communications primitive that allows a node to send data to any one of several potential recipients. One challenge with providing an *acknowledged anycast* service efficiently is that the initiator may not know *a priori* which neighbors, if any, would acknowledge a transmitted packet. The initiator generally has two options in this case. One option is to contact neighbors sequentially, assuming that they are even known in advance. Unfortunately, this approach is inefficient since it scales poorly with node density. The other option is to contact all neighbors at once, perhaps using a link layer multicast or broadcast address. This approach is confronted with the well-known ACK implosion problem in which a potentially arbitrary number of neighbors can result in an arbitrary number of replies. Wireless networks further exacerbate this problem because hidden terminals can lead to collisions that corrupt packets, reduce bandwidth, and waste energy.

Imagine, however, if acknowledged anycast could be implemented efficiently: an initiator would transmit a single packet to a multicast or broadcast address, all nodes that match the destination address would acknowledge the packet concurrently, and the initiator would correctly decode the superposition of multiple acknowledgments to learn that at least one node received the packet despite the obvious ACK collisions. We term such an exchange a *backcast* and suggest that it could offer a wireless Boolean OR service abstraction: a node could pose a *true* or *false* question to its neighbors and each neighbor would vote *false* by ignoring the packet or *true* by acknowledging it. Section 2 hypothesizes how such a service could work.

Furthermore, a reliable, efficient, and scalable acknowledged anycast service would enable or improve multiple applications. For example, a low-power, network wakeup service would be possible [7]. A low-power, receiver-initiated unicast service that eliminates the long preambles common in today's low-power listening protocols would also be feasible [8]. Finally, single-hop collaborative feedback [3] would benefit from the OR semantics of acknowledged anycast. Section 3 discusses these and other backcast applications.

Section 4 explores the veracity of our thesis – that an acknowledged anycast service can be implemented efficiently – via a range of experiments based on the IEEE 802.15.4-compliant CC2420 radio [11]. The results show that a commodity radio can decode the superposition of at least a dozen identical acknowledgments with greater than 97% probability. These results suggest that an efficient and robust one-hop anycast service that does not suffer from ACK implosion is possible with at least the O-QPSK modulation scheme used in 802.15.4.

Our results suggest some important relationships between the signal strength and quality of acknowledgments, number of responders, and delay variation. In a controlled experiment with equal path loss and round trip times between the initiator and responders, we find that the two-responder case exhibits slightly worst signal quality and reception rates than all other cases. Section 5 discusses these results in greater details and argues that the well-known capture effect does not explain the surprisingly robust performance of backcast.

## 2 BACKCAST

A backcast is a link-layer frame exchange in which a single radio frame transmission triggers zero or more acknowledgment frames that interfere non-destructively at the initiator. Figure 1 illustrates a backcast exchange involving three nodes. The two responders have their radios configured to automatically acknowledge any received frames. The backcast exchange begins with the initiator transmitting a probe frame to the hardware broadcast address. Both responders receive the probe and they both transmit *identical* acknowledgments. Although these two acknowledgments collide at the initiator, as long as certain conditions are met, this collision is non-destructive, allowing the initiator to correctly decode the acknowledgment frame and conclude that at least one of its neighbors responded.

In addition to the broadcast address, a backcast probe can be sent to a multicast or unicast address, to which only a subset of the initiator's neighbors might respond. The choice of the destination address of a backcast probe depends on the radio's capabilities as well as the needs of the communications service using backcast. For example, the hardware broadcast address might be appropriate when waking up an sleeping network while a unicast address would be appropriate for communications with a single node.

The key to a successful backcast is that ACK collisions are non-destructive. This condition can hold due to power capture if one ACK frame has a higher power than the sum of the remaining ACK frames [1], or delay capture if one ACK frame arrives some period of time before the rest [2], or message retraining capture – a "message in message" model – where the radio attempts to resynchronize mid-packet if it detects a suddenly elevated energy level [6], or trivially if the radio uses an on-off keying (OOK) modulation scheme [10].

The central hypothesis of this paper is that backcast is possible under a much wider range of conditions than what capture would predict. In particular, we hypothesize that backcast is possible using minimum shift keying (MSK) and orthogonal quadrature phase shift keying (O-QPSK) modulation schemes for certain radio designs provided that: (i) inter-symbol interference resulting from different path lengths is limited, (ii) concurrent ACK frames do not cancel each other at the physical layer, (iii) the radio can automatically generate an ACK frame with an accurate and precise turnaround time, and (iv) the superposition of multiple ACKs is semantically meaningful (e.g., the ACKs are identical). Despite this list of constraints, Section 4 shows that backcast works in practice under a range of both controlled and realistic conditions using a commodity radio.

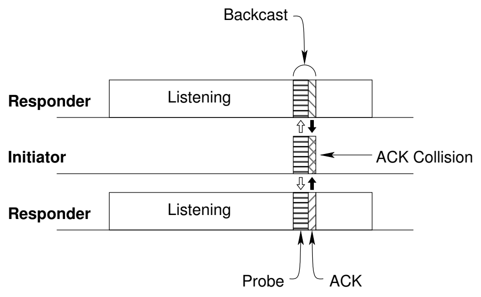


**Figure 1**: A backcast exchange involving three nodes. The backcast initiator transmits a probe frame that two responders acknowledge. Although their acknowledgments collide, they do so non-destructively, so the initiator can decode the resulting frame.

## 3 BACKCAST APPLICATIONS

In this section, we demonstrate the generality of backcast by applying it to some important network services.

### 3.1 Low-Power Asynchronous Wakeup

Waking up a multi-hop network of duty-cycled nodes is a fundamental problem in sensor networks. Applications as diverse as interactive data collection, exceptional event detection, and target tracking require nodes to wake up neighbors or even the entire network.

Dutta et al. proposed one approach to this problem [4]. In their scheme, every node periodically transmits a beacon and then briefly listens for channel activity (either a packet or increased energy). If any channel activity is detected, the node remains awake, but if no activity is detected, the node goes back to sleep. To wake up the network, the initiator listens for a time equal to the beacon period to identify all one-hop nodes. Then, during the next such period, the initiator contacts each of its one-hop neighbors in turn. These neighbors then repeat this process for the two-hop neighbors, and so on. If two or more nodes attempt to contact the same node in a lower tier, the paper conjectured that the concurrent transmissions may collide, but that the receiver would detect channel energy, remain awake, and give the transmitters a chance to enter backoff and compete.

Musăloiu-E. et al. proposed *low power probing* (LPP) as another solution to the wakeup problem [7]. According to the LPP protocol, nodes periodically broadcast short probes requesting acknowledgments. If such an acknowledgment arrives, the node wakes up and starts acknowledging other nodes' probes; otherwise it goes back to sleep. The key difference between the two approaches is that the responses in the first approach are software-generated, while LPP uses hardware acknowledgments (HACKs). What is surprising is that LPP works even if a node has many neighbors, a case in which multiple acknowledgments would collide. In fact,

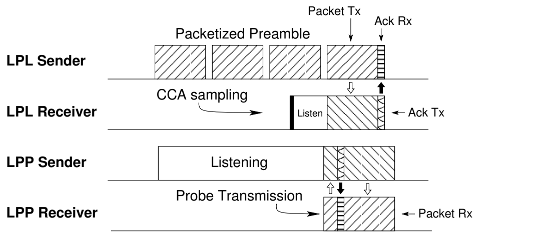


**Figure 2**: A side-by-side comparison of LPL and LPP operations. LPP replaces LPL's long preamble with listening and LPL's clear channel assessment with a backcast exchange.

LPP implicitly uses backcast to sidestep ACK implosions but the paper does not recognize this fact – something that this paper identifies.

LPP, which uses backcast, is more efficient than Dutta's proposal. The reason is that LPP does not suffer from (destructive) collisions and thus does not enter a contention phase. Furthermore, since distinguishing between collisions and other sources of noise or interference is difficult, such an approach could exhibit high false positives in practice. This observation suggests that Dutta's approach might perform poorly in dense networks deployed in interference-rich environments.

## 3.2 Low-Power Unicast

Polastre et al. proposed *low power listening* (LPL), a widely-adopted technique for low duty-cycle communications. An LPL receiver periodically checks for channel energy and stays awake upon detecting activity, while a transmitter prepends a packet with a preamble that is at least as long as the receiver's check period [8]. While LPL was designed to wake up individual nodes, LPP was designed to wake up the whole network [7]. We now come full circle by describing how LPP can be modified to wake up individual nodes and thus provide the same unicast service abstraction as LPL, while using a receiver-initiated protocol.

Directly replacing LPL with LPP, as Figure 2 illustrates, is possible yet inefficient. In the LPP protocol, a receiver transmits a probe packet to the hardware *broadcast* address and the sender responds with a HACK, causing the receiver to stay awake to receive a data packet. The problem with this approach is the sender's radio will acknowledge every probe it receives since they are sent to the broadcast address. In turn, this causes all but one of the sender's neighbors to wake up unnecessarily. Let us call this the *overreacting problem.*

LPP can be modified to avoid the overreacting problem as follows. When a sender $X$ has pending traffic for a receiver $Y$, $X$ enables its radio's hardware address recognition and sets its radio's hardware address to $Y + k$ (where $k$ is 0x8000 or 0x800000000000).

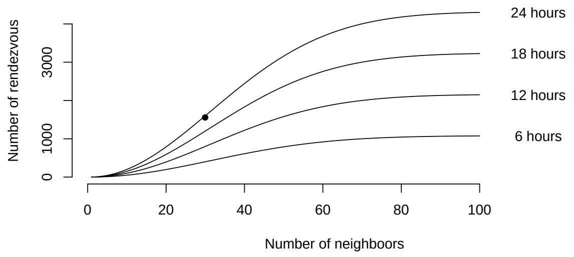


**Figure 3**: Total number of successful rendezvous, predicted by the birthday paradox, over different time intervals as a function of neighborhood size. The black dot represents the daily average number of rendezvous recorded on a 30-node testbed. In all cases the probing interval is 20 seconds and a single backcast lasts $\sim$20 msec.

Now, instead of broadcasting a probe, receiver $Y$ sends a probe to destination address $Y + k$, requesting a HACK. Sender $X$ (as well as any other nodes with pending traffic to $Y$) respond to the probe (multiple HACKs interfere non-destructively). If its probe is acknowledged, $Y$ remains awake to receive a packet while sender $X$ does not succumb to the overreacting problem.

## 3.3 Opportunistic Rendezvous

In the services outlined so far, backcasts are used as purely control traffic: signals to wake up nodes or alerts for inbound traffic. In this respect, backcast messages carry no application-level information. This observation raises the following question: *are there advantages for the probes to carry an application payload?* Note that acknowledgments cannot carry node-specific payloads as this would violate the requirement posited in Section 2 that acknowledgments be identical . We attempt to answer this question in two steps. First, we show that carrying a payload does not compromise backcast's feasibility or performance. We then sketch one service enabled by this extension.

To explore the first question we varied the probe's payload, from one byte up to its maximum size of 116 bytes for the CC2420 radio we use [11]. As expected the time necessary for a complete backcast operation increases linearly with the size of the probe. More importantly, a backcast carrying the maximum payload requires only $\sim$50% more time (31.27 msec) than one with a one byte payload (20.77 msec). The reason is that actual probe transmission corresponds to only a subset of the total time the radio is active, the rest devoted to turning the radio on and waiting for acknowledgments.

Since including application payloads generates only moderate overhead, we explore the original question through an extension to the primitive described in Section 3.1. Specifically, we augment probes to include the initiators' local clock value. Then nodes that overhear these probes can use them to perform network-wide clock synchronization (e.g., through a distributed consensus algorithm). However, since nodes keep their radios mostly off to conserve energy, this mechanism will only work if many probes are actually overheard (we term such an event, an *opportunistic rendezvous*).

Fortunately, even if a node keeps its radio on for only 20 msec during a 20 second interval (i.e., a 0.1% duty cycle), the birthday paradox works to our advantage, as Figure 3 shows. Even with few neighbors, the probability of a rendezvous is non-negligible. Furthermore, because nodes send frequent backcasts, the contact probability accumulates over time, resulting in numerous rendezvous in the span of a few hours.

### 3.4 Robust Pollcast

Demirbas et al. recently proposed *pollcast*, a two-phase primitive in which a node broadcasts a poll about the existence of a node-level predicate $P$ and then all nodes for which $P$ holds reply simultaneously [3]. The poller detects one or more positive answers by reading its radio's Clear Channel Assessment (CCA) signal. While pollcast offers a novel approach for quickly calculating predicates, the proposed mechanism has some drawbacks, as the paper acknowledges: simultaneous pollcasts within a two-hop neighborhood would cause false positives as would interference from other networks.

Backcast provides a more robust primitive for implementing pollcast, which in turn can be used to implement the applications outlined in [3]. To leverage the backcast primitive, pollcast might be modified to first transmit the predicate, then transmit the poll, and finally listen for an acknowledgment. The predicate would be sent to the broadcast address but it would also include an ephemeral identifier chosen by the initiator. Upon receiving the predicate, and evaluating it as true, a responder would enable acknowledgments and temporarily change its hardware address to match the ephemeral identifier in the probe packet. Then, a backcast probe sent to the ephemeral identifier would trigger a response from all the nodes for which the predicate was true. The CC2420 radio supports just two hardware addresses – a 16-bit one and a 64-bit one – allowing just one or two concurrent pollcasts. Future radios could perform address decoding in parallel over dozens of addresses, perhaps using a content addressable memory.

## 4 Evaluation

This section provides empirical evidence that backcast works with one commodity radio. These observations are based on experiments with very controlled parameters (Section 4.2), to larger, more realistic environments using a sensor network testbed (Section 4.3).

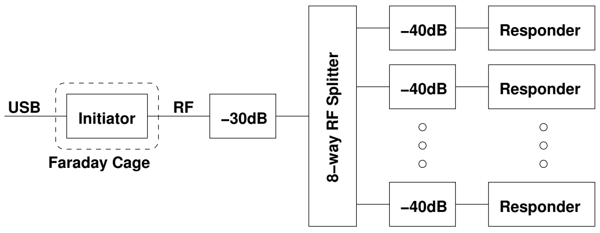


**Figure 4**: Experimental setup for the controlled tests. An initiator is connected via a 30-inch, 50 Ω RF cable and a 30 dB attenuator to the common port of an 8-way RF splitter. The other splitter ports are connected via 6-inch, 50 Ω RF cables and 40 dB attenuators to responders. A Faraday cage around the initiator limits over-the-air RF leakage.

### 4.1 Methodology

We implemented backcast in the TinyOS embedded operating system [5] and our experiments are based on the widely-used Telos mote [9] that includes the Texas Instruments CC2420, an IEEE 802.15.4 radio [11]. The 802.15.4 protocol and the CC2420 radio are ideal for demonstrating backcast because they provide the needed protocol and hardware support.

The 802.15.4 MAC defines a frame control field that includes an acknowledge request flag. If a receiver is configured for automatic acknowledgments, then an acknowledgment frame is transmitted after twelve symbol periods (192 $\mu$sec) for all incoming frames that meet three conditions: they (i) have the acknowledge request flag set, (ii) are accepted by the radio's address recognition hardware, and (iii) contain a valid CRC. Acknowledgments are transmitted without performing clear channel assessment and have the following fields: preamble, start-of-frame delimiter, length, frame control, sequence number, and frame check sequence. Notably absent from this list is a source address, ensuring that all ACKs for a given sequence number are identical.

The experiments that follow show how different responder configurations affect the acknowledgments' signal strength and quality. Signal strength is measured over the first eight symbols and reported as the received signal strength indicator (RSSI) in dBm. Signal quality (LQI) is also measured by the radio over the first eight symbols and is reported as a 7-bit unsigned value that can be viewed as the average correlation value or chip error rate.

### 4.2 Performance in a Controlled Setting

We first explore how the RSSI and LQI of acknowledgment frames are affected as the number of responders increase in a controlled setting. Figure 4 presents the setup for this experiment. Eight nodes are sequentially turned on so that the number of responders monotoni-

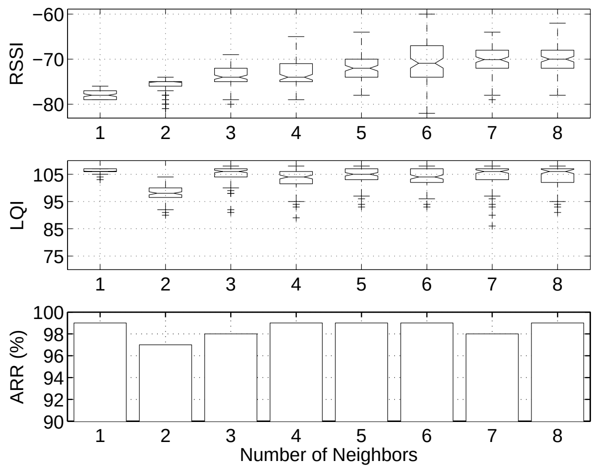


**Figure 5**: Results of the controlled experiments. The received signal strength (RSSI), link quality indicator (LQI), and acknowledgment reception rate (ARR) are shown for each trial.

cally increases from one to eight. In each of the eight trials, the initiator transmits 100 packets to the hardware broadcast address, at 125 msec intervals, and logs the RSSI and LQI values of the resulting acknowledgments. The results are shown in Figure 5 and indicate that median RSSI values increase, median LQI values stay nearly constant, both values show greater variance, and few acknowledgments are lost. Section 5 discusses these results in detail.

## 4.3 Performance in a More Realistic Setting

Next, we explore how backcast performs in a more realistic university testbed setting. The testbed is located in an office building and contains 47 Telos motes. For this experiment however, we used only 12 nodes approximately situated at the same distance from the initiator. These experiments compare the performance of hardware-generated acknowledgments (HACKs), software-generated acknowledgments (SACKs), and HACKs with randomized preamble lengths of between 3 and 16 bytes (VP-HACKs) that start at the same time but may end at different times. HACKs are automatically generated by the radio, while SACKs require the host processor to intervene, introducing considerable delay and jitter. We introduce VP-HACKs to explore how acknowledgments with smaller delay variations than SACKs interfere at the initiator. Note that while SACKs have non-uniform delays due to software processing, the VP-HACKs are all delayed by an integer multiple of the symbol time (composed of 32 chips) and the symbols themselves are orthogonal codes.

In each experiment, 500 packets are transmitted at 125 msec intervals. This procedure generates a gradual

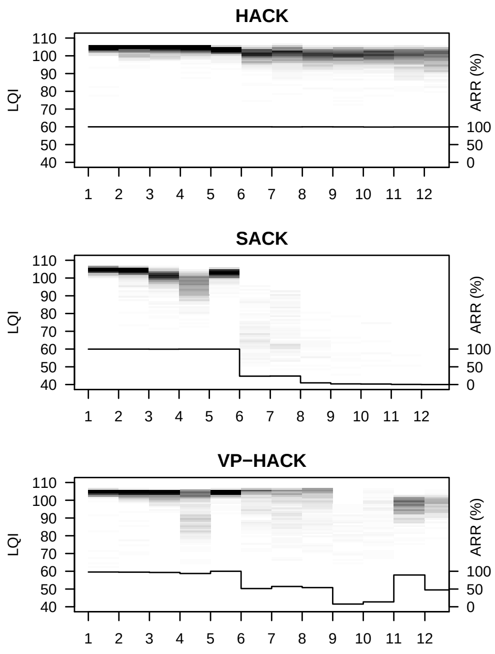


**Figure 6**: Backcast in a realistic environment, using hardware and software acknowledgments. The data are shown as a two-dimensional histogram; darker areas indicate a higher density of LQI samples. The lower line shows the average acknowledgment reception rate (ARR).

increase in the number of colliding ACK frames. The LQI and acknowledgment reception rates are shown in Figure 6. The results show that HACK and SACK LQI values exhibit higher variance and volatility as responders increase. Both HACKs with random preambles and SACKs exhibit quickly decreasing LQI and ARR values, while HACKs incur practically no loss. Section 5 discusses these results in detail.

## 5 Discussion

The results from Sections 4.2 and 4.3 suggest some important relationships between signal strength and quality of acknowledgments, number of responders, and delay variation. Further analysis, described below, supports our hypothesis that the capture effect alone cannot explain the surprisingly robust performance of backcast.

First, as the number of colliding acknowledgments increases, so does the median RSSI value. This trend is not surprising since for every doubling of nodes, an additional 3 dB of power is injected into the channel (assuming nodes transmit at nearly equal power levels and are equally distanced from the initiator). What is slightly more surprising is that RSSI variance is substantial and spans a range of 10-20 dB, which is both below and above the single node case, and that the distribution of values in the two-node case has many outliers.

These results suggest that elements of both constructive and destructive interference of the carrier signal may be at play. When three or more acknowledgments collide, both the outliers and RSSI variance *decrease*, suggesting that the statistical superposition of an increasing number of signals diminishes destructive interference, possibly due to the central limit theorem.

Second, the median LQI value is largely independent of the number of nodes in the controlled setting (except for the two node case) and it shows a slight decrease in the more realistic setting (computed, but not shown). Since LQI is inversely correlated with chip error rate, the data show that most acknowledgments are decoded with relatively few chip errors, even when a dozen acknowledgments collide. The data suggest that acknowledgment collisions are rarely destructive and in most cases not particularly corrupting either. LQI values show a lower median value for two responders than they do for either one or more than two responders, suggesting once again that elements of both constructive and destructive interference of the carrier signal may be at play. The RSSI distributions are largely symmetric with few outliers but the LQI distributions are left-tailed. This observation suggests that although collisions rarely improve the chip error rate, they can make it worse.

Finally, the data show that hardware acknowledgments exhibit negligible loss rates with no fewer than twelve concurrent packets, while software acknowledgments approach very high loss rates with just six or seven concurrent acknowledgments, as well as a substantial decline in link quality with just three or four acknowledgments. In between these two extremes are the variable-length preamble HACKs (VP-HACKs). The two distinctions between SACKs and VP-HACKs are in timing and composition. First, SACKs are delayed by multiples of the CPU clock cycle since a SACK requires software processing, but a VP-HACKs are delayed by an integer multiple of the symbol time. Since the symbols are chosen from an orthogonal set, this may explain the better performance of VP-HACKs compared with SACKs, despite the fact that VP-HACKs collide more frequently and are not even identical. Since these three types of acknowledgments differ in the delay and jitter of their transmissions, we argue the capture effect alone cannot explain the surprisingly robust performance of HACK-based backcasts.

## 6 Summary

This paper shows that a standards-based commodity radio can correctly decode the superposition of up to a dozen identical acknowledgment frames. This observation suggests that an efficient and robust acknowledged anycast service that does not suffer from ACK implosions may be feasible. The ability to transmit a multicast or broadcast packet and receive an acknowledgment in constant time independent of the number of responding nodes, an exchange we call *backcast*, enables or improves a range of useful communication services.

## 7 Acknowledgments

Special thanks to David Culler, Rodrigo Fonseca, Jonathan Hui, Mike Liang, and the anonymous reviewers for their substantial and insightful feedback. This material is based upon work partially supported by the National Science Foundation under grants #0435454 ("NeTS-NR"), #0454432 ("CNS-CRI"), and #0627611 ("CAREER"), an NSF Graduate Fellowship, a Microsoft Graduate Fellowship, and generous gifts from HP, Intel, Microsoft, and Sharp.

## References

[1] J. Arnbak and W. van Blitterswijk. Capacity of slotted aloha in rayleigh-fading channels. *IEEE Journal on Selected Areas in Communications*, 5(2):261–269, Feb 1987.

[2] D. Davis and S. Gronemeyer. Performance of slotted aloha random access with delay capture and randomized time of arrival. *IEEE Transactions on Communicationss*, 28(5):703–710, May 1980.

[3] M. Demirbas, O. Soysal, and M. Hussain. A singlehop collaborative feedback primitive for wireless sensor networks. In *Proceedings of the 27$^{th}$ IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[4] P. Dutta, D. Culler, and S. Shenker. Procrastination Might Lead to a Longer and More Useful Life. In *Proceedings of HotNets-VI*, 2007.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Proceedings of ASPLOS 2000*, 2000.

[6] R. Mud, J. Boer, A. Kamerman, H. Van Driest, W. Diepenstraten, R Kopmeiners, and H. Von Bokhorst. Wireless LAN with enchanced capture provision. US Patent No. US5987033, 19919.

[7] R. Musaloiu-E., C.-J. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7$^{th}$ International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[8] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2$^{nd}$ ACM Sensys Confence*, 2004.

[9] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proceedings of the 4$^{th}$ International Conference on Information Processing in Sensor Networks (IPSN/SPOTS)*, 2005.

[10] M. Ringwald and K. Romer. BitMAC: a deterministic, collision-free, and robust MAC protocol for sensor networks. In *Proceedings of the 2$^{nd}$ European Workshop on Wireless Sensor Networks*, 2005.

[11] Texas Instruments. 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Available at `http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf`, 2006.

# Message in Message (MIM): A Case for Reordering Transmissions in Wireless Networks

Naveen Santhapuri
Univ. of South Carolina

Justin Manweiler
Duke University

Souvik Sen
Duke University

Romit Roy Choudhury
Duke University

Srihari Nelakuduti
Univ. of South Carolina

Kamesh Munagala
Duke University

## ABSTRACT

*Message in Message* (MIM) is an exciting development at the physical layer of IEEE 802.11. Two transmissions that otherwise conflict with each other, may be made concurrent with MIM. However, the benefits from MIM are not immediate. Higher layer protocols need to be explicitly designed to enable its inherent concurrency. This paper investigates the opportunities and challenges with MIM, and demonstrates a link layer framework to harness its potential. We believe that our framework can accommodate emerging physical layer capabilities, such as successive interference cancellation (SIC).

## 1. INTRODUCTION

Physical layer research continues to develop new technologies to better cope with wireless interference. One exciting development in the recent past is *Message in Message* (MIM). Briefly, MIM allows a receiver to disengage from an ongoing signal reception, and engage onto a new, stronger signal. What could have been a collision at the receiver, may now result in a successful communication. To better understand MIM, we compare it with the traditional notion of *collision* and *physical layer capture*. We refer to Figure 1 to describe this contrast. *We assume throughout the paper that the signal of interest (SoI) is sufficiently stronger than the interference*[1].

**Collision** was widely interpreted as follows: An SoI, however strong, cannot be successfully received if the receiver is already engaged in receiving a different (interfering) signal. Most simulators adopt this approach, pronouncing both the frames corrupt [1,2]. Figure 1(c) and (d) illustrate these cases. **Physical Layer Capture** was later understood through the systematic work in [3,4]. Authors showed that capture allows a receiver to decode an SoI in the presence of interference, provided the start of both the frames are within a preamble time window. While valuable in principle, the gains from capture are limited because the 802.11 preamble persists for a short time window (20 $\mu$s in 802.11a/g). If the SoI arrived 20 $\mu$s or later, both frames will still be corrupt (Figure 1(d)).

[1]We also assume that the interference is strong enough that, in the absence of the SoI, it can be decoded by the receiver.

**Message in Message (MIM)** is empowering because it enables a receiver to decode an SoI, even if the SoI arrives after the receiver has already locked on to the interference [5]. Moreover, if the SoI arrives earlier than the interference, MIM allows reception at a lower SINR in comparison to a non-MIM receiver. Figures 1 (a), (b) and (d) identify these benefits over capture.

Two main ideas underpin the feasibility of MIM:
(i) An MIM receiver, even while locked onto the interference, simultaneously searches for a new (stronger) preamble. *If a stronger preamble is detected, the receiver unlocks from the ongoing reception, and re-locks on to this new one.* Of course, re-locking requires a higher SINR.

(ii) MIM takes advantage of the characteristics of interference signals. A strong decodable interference is better than a weak non-decodable interference, because *the ability to decode the interference enables the ability to suppress it as well*. As a result, an MIM receiver that is locked onto the interference is better equipped to suppress it, and re-lock onto the later-arriving SoI. Exploiting the same idea, if the receiver has locked onto the SoI first, and a weaker interference arrives later, the receiver is able to better "tolerate this distraction". As a result, the SoI can be received at a lower SINR.

## Link Layer Opportunity

Unless guided by link layer protocols, the intuitive benefits from MIM may not translate into throughput improvements. We argue this using the example in Figure 2. When using MIM receivers, observe that the two links can be made concurrent only if AP1→R1 starts before AP2→R2. Briefly, since R2 satisfies a higher SINR of $20dB$, it can afford to decode its SoI even in the presence of interference from AP1. However, since R1 satisfies a lower SINR, starting earlier helps in locking onto AP1's signal in the clear. Had the order of transmission been reversed, AP1→R1 transmission would experience a collision. As a generalization of this example, MIM-aware scheduling protocols need to *initiate weaker links first, and stronger links later*. In a larger network, choosing the appropriate set of links from within a collision domain,

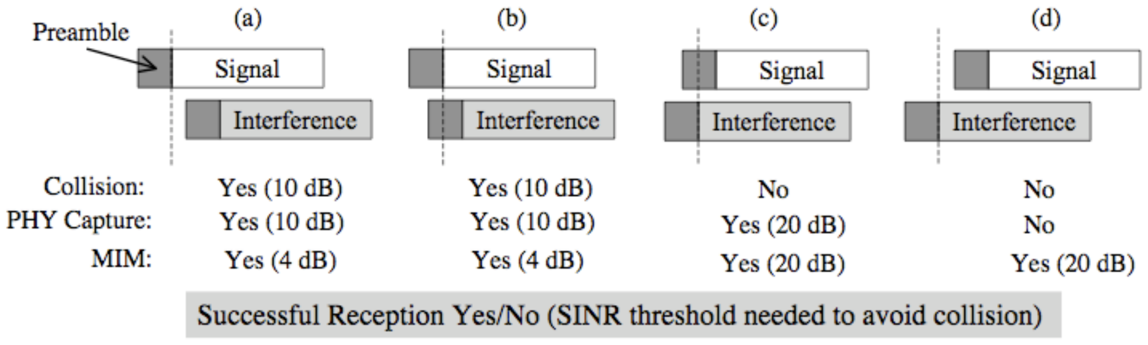

Preamble
(a)
(b)
(c)
(d)
Signal
Interference
Collision: Yes (10 dB) Yes (10 dB) No No
PHY Capture: Yes (10 dB) Yes (10 dB) Yes (20 dB) No
MIM: Yes (4 dB) Yes (4 dB) Yes (20 dB) Yes (20 dB)
Successful Reception Yes/No (SINR threshold needed to avoid collision)

**Figure 1: Evolving notion of successful reception. SINR values are approximate, and vary across hardware.**

and determining the order of optimal transmission is a non-trivial research problem. IEEE 802.11 or other MAC protocols that are unaware of MIM [6] do not ensure such orderings, failing to fully exploit MIM-capable receivers. Perhaps more importantly, *graph coloring based scheduling approaches may also be inapplicable*. This is because graph coloring approaches assume symmetric conflicts between links. Link conflicts are *asymmetric* under MIM (i.e., depend on relative order), and may not be easily expressed through simple abstractions.

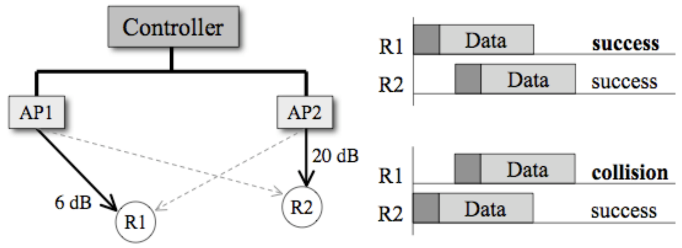


**Figure 2: AP1→R1 must start before AP2→R2 to ensure concurrency. In the reverse order, R1 cannot lock onto AP1's signal because of AP2's interference.**

In response to this rich research problem, this paper proposes an MIM-aware link layer solution that reorders transmissions to extract performance improvements. Our system is named *Shuffle* in view of its ability to shuffle the order of transmissions. Our main contributions are:

**(1) Validation of MIM through experiments on a small testbed.** We use MIM-enabled IEEE 802.11 compatible Atheros 5213 chipsets, running MadWiFi drivers on Soekris hardware. We show that order of transmission matters while decoding packets.

**(2) Analysis of optimal performance improvements with MIM**. We show that MIM-aware scheduling is NP-hard, derive upper bounds on throughput using integer programming in CPLEX, and design heuristics to attain these bounds.

**(3) Design of an MIM-aware scheduling framework, *Shuffle*, for enterprise wireless LANs.** Our approach to reordering transmissions offers consistent throughput improvements against both 802.11 and a centralized scheduling protocol. The subsequent sections expand on each of these contributions.

## 2. TESTBED MEASUREMENTS

We confirm the occurrence of MIM on a testbed of Soekris devices equipped with Atheros 5213 chipsets using the MADWiFi driver. Apart from corroborating the observations of [5] about MIM, we also show that due to MIM the order of transmission matters for successful delivery of packets. The experiment consists of two transmitters with a single receiver placed at various points in-between. This subjects the receiver to varying SINRs. To ensure continuous packet transmissions from the transmitters, we modify the MADWiFi driver to disable carrier sensing, backoff, and the inter-frame spacings (EIFS and SIFS). To time-stamp transmissions, a collocated receiver is placed at each transmitter. Using these time-stamps, we are able to merge multiple traces to determine which packets overlap in time and the relative order of overlap. We omit several implementation details, especially those related to achieving $\mu s$-granularity time synchronization among collocated receivers.

Figure 3 shows delivery ratio for different order of packet arrivals, at different positions of the receiver. For all these positions, the interference was strong, i.e., in the absence of the SoI, the interfering packets were received with high delivery ratio. Under these scenarios, observe that when the receiver is very close to the transmitter (positions 1, 2, and 3), it achieves a high delivery ratio independent of the order of reception. This is a result of achieving a large enough SINR, such that both SoI-first (SF) and SoI-last (SL) cases are successful. However, when the receiver moves away from the transmitter (positions 4 and 5), the SINR is only sufficient for the SF case, but not the SL case. Hence, only $4\%$ of the late-arriving packets get received, as opposed to $68\%$ of the early-arriving packets. This is a clear validation of MIM, and can be translated into throughput gains by deliberately regulating the start of packets.

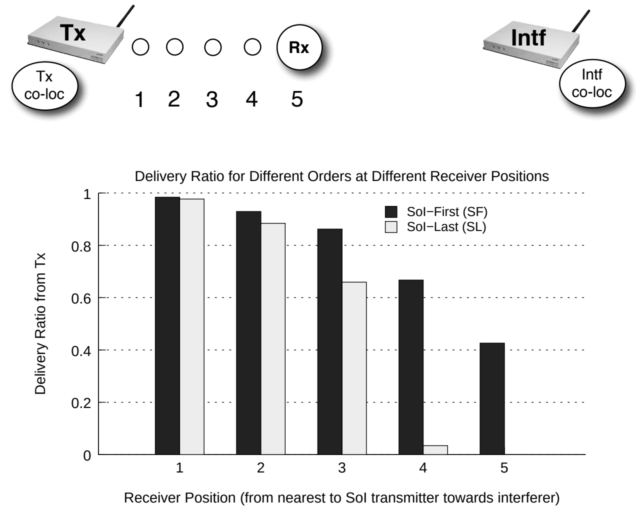


**Figure 3: Testbed validates MIM. Rx receives from Tx (at 5 positions) in presence of interferer (Intf).**

## 3. OPTIMALITY ANALYSIS

A testbed prototype validates the practicality of MIM, and indicates potential for gains in a large scale network. The natural question to ask is, *what is the maximum gain available from MIM?* Towards this, we perform an optimality analysis of MIM-capable networks. In the interest of space, we present only the main results.

Given a network, $G = (V, E)$, an MIM-enabled link scheduling algorithm needs to choose a suitable subset of links, $l \in E$, and specify their order of activation, $O_l$. The links and their order must be chosen such that the network concurrency (or network throughput) is maximized. We have shown that *Optimal MIM-aware scheduling is NP-hard*. The proof is derived through a reduction of the *Independent Set* problem, known to be NP-complete.

We developed an Integer Program (IP) to characterize the upper bounds on throughput for practical MIM-capable network topologies. The bounds are compared with an optimal MIM-incapable scheduling scheme (also NP complete). We omit the IP formulation, and only include the key results here. Figure 4 presents graphs generated using the CPLEX optimizer. Evident from the graphs, the ideal benefits from MIM can be high, especially for networks of realistic size.

## 4. SHUFFLE: AN MIM-AWARE LINK LAYER

We now propose Shuffle, a link layer solution that exploits MIM capabilities by carefully reordering transmissions. Shuffle targets enterprise WLAN (EWLAN) environments, such as universities, airports, and corporate campuses [7, 8]. In EWLANs, multiple access points (APs) are connected to a central controller through a high speed wired backbone (Figures 2 and 6). The controller, besides acting as the gateway for Internet traffic, coordinates the operations of APs. The (thin) APs follow the controller's instructions for control and data packet communication.

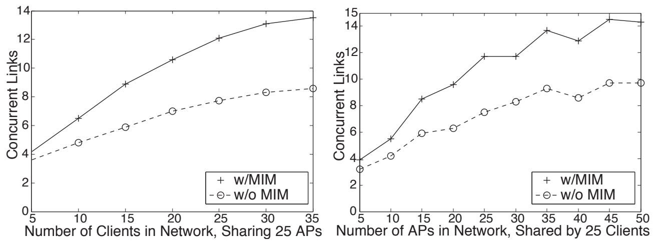


**Figure 4: MIM performance bounds using CPLEX.**

### 4.1 Protocol Architecture

Shuffle executes three main operations as follows: (1) Pair-wise interference relationships between links are characterized through a measurement-based procedure that we call *rehearsal*. (2) Utilizing the interference map, an MIM-aware scheduler (hosted at the EWLAN controller) computes a set of concurrent links, and their relative order of transmission. (3) A transmission manager coordinates the APs to execute the ordered schedule, and handles failures due to time-varying channel conditions. The rest of this section presents the details of these individual operations.

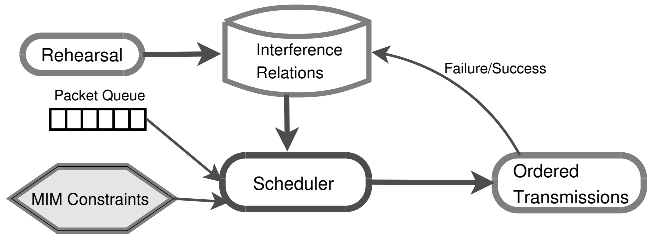


**Figure 5: Architecture of Shuffle**

#### 4.1.1 *Rehearsal*

Scheduling algorithms require the knowledge of interference relationships between links. We propose a measurement-based approach to obtain this relationship. Specifically, the EWLAN controller coordinates the APs and clients to transmit probe packets in a systematic manner. Other clients and APs sniff the RSSI values for each transmission, and feed them back to the controller. The timing of transmissions are carefully planned to ensure clear measurements. Once all the link RSSI values are accumulated, the controller merges them into an interference map. For the merging operation, we assume that interference is linearly additive [9, 10]. Hence, once the rehearsal is over, the approximate SINR of any transmission, in the presence of other interfering transmissions, can be computed. Of course, these values change due to the time-varying nature of the wireless channel and client mobility. We address this later in Section 4.1.4.

### 4.1.2 Packet Scheduling

Given the interference map of the network, the job of the MIM scheduler is to select an appropriate batch of packets from the queue, and prescribe their optimal order of transmission. Unfortunately, graph coloring algorithms on conflict graphs are not applicable in the case of MIM. This is because graph coloring assumes that conflicts between links are symmetric, whereas, due to the ordering constraints in MIM, link conflicts are actually *asymmetric*. This warrants new MIM-aware scheduling algorithms, that unsurprisingly, prove to be NP-hard. Therefore Shuffle employs a heuristic called **Least-Conflict Greedy**. In Least-Conflict Greedy, each packet in the queue is checked to identify its asymmetric conflicts with all other packets in the queue. Each packet is assigned a score based on such conflicts (a higher score if the packet *must* start first). Then, the scheduler runs a greedy ordering algorithm based on the scores of packets. The intuition is that packets with fewer conflicts (smaller score) will be included early in the batch, potentially accommodating more concurrent links. Fairness and starvation issues are certainly important challenges.

### 4.1.3 Staggered Transmissions

The scheduler outputs a batch of packets, and their staggered order of transmission. Each packet from this batch is forwarded to the corresponding AP, along with its *duration of stagger*. The APs begin transmission based on their prescribed stagger, illustrated in Figure 6. Notice that the transmissions are staggered in the order AP1→C13 before AP3→C32 before AP2→C21. Acknowledgments to these transmissions may or may not be MIM-aware. To support MIM-aware ACKs, the "stagger duration" of ACKs can be piggybacked in the data packets. To reduce clock synchronization issues, the stagger duration can be specified as the time between the receipt of the data and the start of the ACK transmission. Clients transmit the ACKs after this specified wait. Transmission failures are handled through scheduled retransmissions from the controller. The failed packet's priority is increased to ensure quicker scheduling.

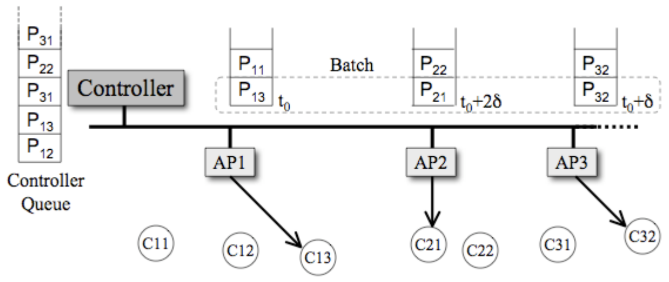


**Figure 6: Batch transmission with suitable stagger.**

### 4.1.4 Handling Channel Fluctuations

A rehearsal produces a snapshot of the interference relationships in the network. However, the interference relations change over time, and scheduling algorithms must remain abreast of such changes. For this, Shuffle employs continuous *opportunistic rehearsals*. The basic idea with opportunistic rehearsal is that clients and APs continuously record RSSI values of ongoing transmissions, and time-stamp them. These $< RSSI, \ time >$ tuples from the recent past are piggybacked in ACKs or other packets that clients send to APs. The APs forward the clients' (and their own) tuples to the controller, which in turn correlates them over time to refresh the interference map. Scheduling decisions are based on this frequently refreshed interference map, allowing Shuffle to cope with fading and fluctuations. Our measurements have shown that the conflict graph of the network remains stable over hundreds of packets, giving us reason to believe that opportunistic rehearsal will be fast enough to cope with channel changes. Of course, this needs to be confirmed with full scale implementation.

## 4.2 Limitations, Issues, and Extensions

While time synchronization is necessary to stagger packets, we believe that it need not be too tight. This is because we require packets to be staggered *more* than the preamble of an earlier packet. Conservative staggering (by adding a factor-of-safety more to the preamble duration) can accommodate clock skews. The impact on performance may only be marginal.

Shuffle needs to account for upload traffic. For this, clients could express their intent by setting a flag on ACK packets. On getting a flagged ACK, the controller could schedule the client in the next batch of transmissions. For clients that do not have an ACK to send, a short periodic time window can be allocated for contention-based (CSMA) transmissions.

Interferences in the 2.4GHz band, such as from microwaves, cordless phones, or even from other devices near the periphery of the EWLAN, can affect the scheduled transmissions. Shuffle can account for them through rehearsal if they persist for long durations. If they are transient, Shuffle will rely on retransmissions to cope with them.

The discussion of Shuffle thus far assumes that all transmissions use a fixed rate and power. If APs are allowed to transmit at varying rates and power levels, the controller may be able to extract higher spatial reuse. The impact of rate and power on MIM-aware schedules is part of our future work.

## 5. EVALUATION

We evaluate Shuffle using the Qualnet simulator. MIM capabilities were carefully modeled into the PHY and MAC layer of the simulator. The EWLAN controller was assumed to have a processing latency of 50$\mu$s, and the wired backbone was assigned 1 Gbps data rate. We used 802.11a with transmission power 19dBm, two ray prop-

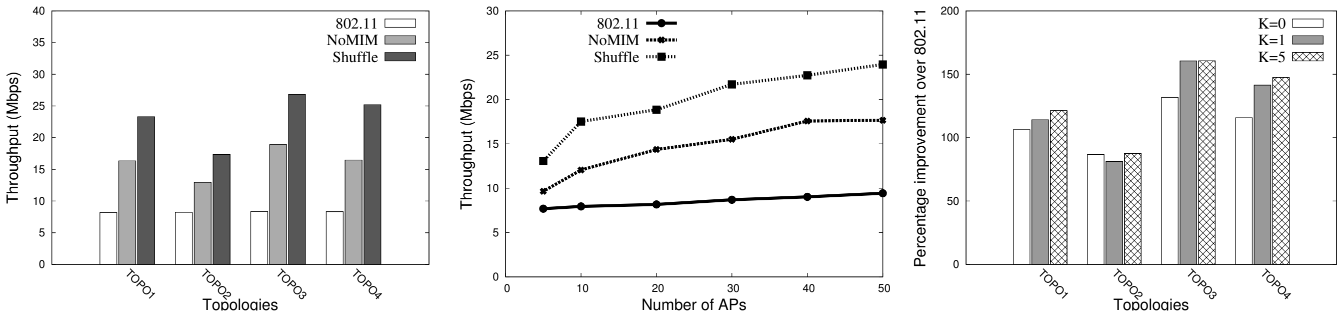


**Figure 7: (a) Throughput for university topologies. Shuffle outperforms NoMIM and 802.11, characterizing the gains from MIM. (b) Higher AP density creates more opportunities for concurrency with MIM. (c) Percentage throughput improvement with channel fading – Shuffle performs well under Rayleigh and Ricean fading.**

agation model, transmission rate 12Mbps, and a PHY layer preamble duration of 20$\mu$s. While we evaluated latency, fairness, and some scheduling variants, we report results for the primary metric of throughput.
We compare Shuffle with 802.11 and an MIM-incapable scheme, called *NoMIM*. The gain from MIM alone is reflected in the difference between *Shuffle* and *NoMIM*. The difference between *NoMIM* and *802.11* characterizes the gains from centralized scheduling. Fig. 7(a) presents throughput comparisons for topologies taken from university buildings with different number of APs on the same channel; each AP was associated to around 6 clients. As a special case, the second topology has APs associated to 20 clients, resembling a classroom setting. Shuffle consistently outperforms NoMIM and 802.11, confirming the potential of MIM-aware reordering. Evident from the difference between NoMIM and 802.11, a fair amount of benefit is also available through centralized scheduling in EWLANs.

## Impact of AP density

Next generation EWLANs may be envisioned to have very high density of access points (perhaps each Ethernet-capable desktop will act as an AP ). To understand Shuffle's scalability in high density environments, we randomly generated topologies in an area of 100x150 $m^2$. We placed an increasing number of APs (ranging from 5 to 50) at uniformly random locations in this region. Each AP is associated with 4 clients and the controller transmits CBR traffic at 1000 pkts/sec to each of the clients. Figure 7(b) illustrates that the throughput of shuffle increases as the density of APs increase. This is because the number of short (and hence high SINR) links increases with greater number of APs. This enables more links satisfying the SoI-Last threshold.

## Impact of Fading

The above simulation results were obtained without channel fading, however, the impact of channel fading can be severe, and the protocol needs to adapt to it over time. To evaluate our opportunistic rehearsal mechanisms, we simulate Ricean fading with varying K factors, and log-normal shadowing. Figure 7(c) shows the percentage improvement of Shuffle over 802.11 for different values of K. For $K = 0$ (Rayleigh Fading), the fading is severe and the improvements are less than at higher values of $K$. Still, the improvements are considerable, indicating Shuffle's ability to cope with time-varying channels. The improvements were verified to be a consequence of opportunistic rehearsals; when opportunistic rehearsal was disabled, the performance degraded.

## 6. SHUFFLE WITH MIM VS SIC

*Successive interference cancellation* (SIC) is a physical layer capability that can be exploited at the MAC layer to extract a weaker signal of interest (SoI) from a stronger interference [11]. This development calls into question the utility of Shuffle based on MIM. An obvious argument in favor of MIM is that it is feasible with current Atheros chipset based receivers whereas SIC capable receivers may be available sometime in future. Nevertheless, we address this issue in this section by first contrasting the abilities of MIM with SIC, and then arguing how Shuffle augments SIC towards even higher performance.

Suppose two frames S and I overlap at a receiver and assume S is the frame of interest. Conventionally S can be decoded only if it is stronger than I and arrives before I at the receiver. With MIM, S can be extracted regardless of whether S arrives before or after I. However, even MIM cannot help if S is weaker than I. Importantly, SIC empowers a receiver to decode I, subtract it from the combined signal (S + I + Noise), and then decode S from the residue (S + Noise). Of course, this description is an over-simplification for the purpose of brevity. Now, we observe that ordering of transmissions helps even with SIC. Consider the two cases when I is moderately stronger, or much stronger, than S. (1) If I is moderately stronger than S, then initiate I before S. This helps in decoding I first with a lower SINR threshold because its preamble is sent in the clear (recall SoI-first case from MIM). Once I is decoded well, it can be subtracted better, allowing better decoding of S. (2) However, if I is much stronger than S, then initiate I after S (recall SoI-last case from MIM). Since the receiver re-locks into I anyway, this ordering helps decode S's preamble in the clear, which is later beneficial for separating S from the residue. Thus, Shuffle's approach of reordering and staggering of transmissions facilitates concurrency in SIC as well. Now, in the case where the receiver is interested in both the frames, similar MIM-aware reordering can be helpful. One needs to view both S and I as signals of interest, say S1 and S2. Thereafter, the two signals need to be initiated in the appropriate order, depending on their relative strengths. Figure 8 shows this interplay of SIC and MIM through a logical representation.

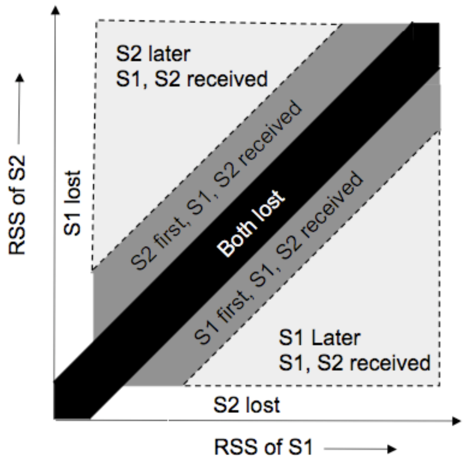


**Figure 8: Ideal ordering of transmissions and corresponding reception outcomes (assuming S1 and S2 are not too weak). (1) when S1 and S2 of comparable strength, then both frames lost (black) (2) when S1 moderately stronger than S2, and S1 started before S2, both received (dark gray) (3) when S1 much stronger than S2, and S1 started after S2, both received (light gray). The upper square is symmetric.**

SIC, when coupled with power control, opens up interesting possibilities for scheduling transmissions. Consider a scenario where S1 and S2 have packets to transmit to R1 and R2, respectively. Suppose the received signal strengths of S1 and S2 at R2 are comparable, i.e., fall within the middle black band in Figure 8. If they transmit concurrently, both will fail even with SIC because neither of the signals can be decoded for subsequent subtraction. On the contrary, if the S2 to R2 signal is made weaker by S2 transmitting at a lower power, i.e., moving from the middle black band of Figure 8 vertically towards x-axis, both transmissions can be successful. This is because both R1 and R2 can capture S1's frame which is relatively stronger, and R2 can then cancel S1's frame to extract S2's frame. The benefit of SIC is of course contingent on the signal strength values and in some cases may result in a lower rate for S2→R2.

An SIC-aware Shuffle system will also employ rehearsals and stagger transmissions as in Fig. 5. The only change is the input to the scheduler. Instead of MIM constraints, SIC will have to be satisfied while scheduling the batch of queued packets. SIC constraints permit higher spatial reuse than MIM constraints as they allow a weaker reception to be concurrent with a stronger interfering transmission. We plan to build and evaluate this order-sensitive link layer to exploit both MIM and SIC.

## 7. CONCLUSION

Physical layer capabilities like MIM and SIC are capable of better coping with interference. Although some benefits may be automatically available, significant improvements can be achieved if the link layer explicitly exploits these capabilities. This paper investigates link layer opportunities and challenges towards harnessing these capabilities. We present an MIM-aware framework named *Shuffle* that reorders transmissions to enable concurrent communications. Theoretical and simulation studies show that building MIM-aware networks are worthwhile; our small-scale prototype validates its practicality. Based on these results, we are implementing *Shuffle* with the aim of providing a fully functional system solution.

## 8. REFERENCES

[1] NS2, "http://www.isi.edu/nsnam/ns/," .
[2] Scalable Network Technologies, "Qualnet v2.6.1," www.scalable-networks.com.
[3] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing out the correct physical layer capture model in 802.11b," in *ICNP*, 2004.
[4] T. Nadeem and L. Ji, "Location-Aware IEEE 802.11 for Spatial Reuse Enhancement," *IEEE Trans. Mobile Computing*, vol. 6, no. 10, Oct. 2007.
[5] J. Lee et al, "An Experimental Study on the Capture Effect in 802.11a Networks," in *WinTECH*, 2007.
[6] Mythili Vutukuru, Kyle Jamieson, and Hari Balakrishnan, "Harnessing Exposed Terminals inWireless Networks," in *NSDI*, 2008.
[7] P. Bahl et al., "DAIR: A Framework for Managing Enterprise Wireless Networks Using Desktop Infrastructure," in *HotNets*, 2005.
[8] N. Ahmed et al., "Interference mitigation in enterprise WLANs through speculative scheduling," in *Mobicom*, 2007.
[9] R. Maheshwari, S. Jain, and S. R. Das, "On Estimating Joint Interference for Concurrent Packet Transmissions in Low Power Wireless Networks," in *WinTECH*, 2008.
[10] H. Chang, V. Misra, and D. Rubentein, "A General Model and Analysis of Physical Layer Capture in 802.11 Networks," *INFOCOM*, 2007.
[11] D. Halperin et al., "Taking the Sting out of Carrier Sense: Interference Cancellation for Wireless LANs," in *MOBICOM*, 2008.

# Can You Fool Me?
# Towards Automatically Checking Protocol Gullibility

Milan Stanojevic *UCLA* Ratul Mahajan *Microsoft Research* Todd Millstein *UCLA* Madanlal Musuvathi *Microsoft Research*

***Abstract –*** We consider the task of automatically evaluating protocol gullibility, that is, the ability of some of the participants to subvert the protocol without the knowledge of the others. We explain how this problem can be formalized as a game between honest and manipulative participants. We identify the challenges underlying this problem and outline several techniques to address them. Finally, we describe the design of a preliminary prototype for checking protocol gullibility and show that it can uncover vulnerabilities in the ECN protocol.

## 1. INTRODUCTION

Modern communication protocols are regularly used among entities that should not trust each other to faithfully follow the protocol. Sometimes trust is unwarranted because a participant may not know the identity of the other entities. Even with known identities, there is still the possibility that others have been hijacked by malicious agents. Further, an entity can inadvertently violate the protocol due to bugs in the implementation.

Whatever the cause, protocol violations can break important protocol invariants, leading to a variety of attacks. For example, Savage et al. show that a TCP receiver can fool the sender into sending faster than the intended congestion controlled rate [19]. In fact, they point out three different ways to accomplish this task. Spring et al. show that an ECN (Explicit Congestion Notification) receiver can fool the sender into ignoring congestion by simply flipping a bit [21], undermining the central purpose of the protocol. Researchers have reported similar kinds of vulnerabilities in peer-to-peer protocols, multi-hop wireless networks, and routing protocols as well [4, 14, 1].

Thus, it is strongly desirable that communication protocols be robust against accidental or intentional manipulation by participants. We say that a protocol is *gullible* if a subset of the participants can violate desirable properties of the protocol by disobeying the protocol in some fashion. Today, protocol gullibility is determined through manual inspection; for example, all vulnerabilities above were discovered manually. While significant progress has been made recently in identifying bugs in protocol implementations [7, 17, 11], bug-free honest participants may still be fooled by manipulative participants, as the examples above illustrate.

In this paper, we pose the problem of automatically evaluating a protocol's gullibility. To our knowledge, prior work has either detected gullibility manually or has automatically checked for specific vulnerabilities in security protocol specifications [13, 15, 20, 18, 5]. We define the general problem of protocol gullibility detection, discuss the challenges involved, and propose techniques that can help to address these challenges. Finally, we report on the design of a preliminary prototype gullibility checker along with some initial results on a very simple protocol.

We formalize the problem of automatic gullibility detection in Section 3 as a game between an angelic component that consists of honest players and a demonic component that consists of manipulative players. The angelic component follows the protocol, while the demonic component can take any action, including sending or not sending any particular packet. A protocol is considered gullible if there exists a strategy for the demonic component such that a desirable property of the protocol is violated.

Automatically determining protocol gullibility is challenging. The key challenge is the enormity of demonic strategy space. At any step, any particular bit pattern can be sent, and complicated strategies may involve a long sequence of particular packets. A second difficulty is the need to search the space of network conditions, because some strategies succeed only under certain network conditions. Finally, even determining when a strategy has succeeded may be difficult, requiring comparison with a reference run where all participants are honest. Section 4 describes these challenges in detail and proposes several techniques to address them.

To begin exploring the problem and our approach, we have developed a preliminary gullibility checker that incorporates a subset of the techniques we propose. We present initial results from using our checker to analyze an implementation of the ECN protocol. The checker can successfully discover the attack that was previously manually discovered [21] as well as some variations on this attack. Our results for ECN, a very simple protocol, are encouraging and so we are excited to continue exploring our approach on more complex protocols.

## 2. EXAMPLE MANIPULATIONS

We describe below some example vulnerabilities found in existing protocols, to provide insight into the kinds of manipulations that we are interested in uncovering.

**ACK division in TCP [19]:** TCP increases its congestion window in units of entire packets whenever a packet that acknowledges previously unacknowledged bytes arrives. A manipulative receiver can speed transfers by acknowledging individual bytes rather than entire packets (as an honest receiver would).

**Duplicate ACKing in TCP [19]:** A TCP sender sends more data in response to any acknowledgment, since the acknowledgment signals that some data left the network. A manipulative receiver can speed transfers by generating multiple, duplicate acknowledgments for the last received sequence number.

**Optimistic ACKing in TCP [19]:** TCP assumes that the time between a data segment being sent and an acknowledgment being received is a round trip time, and its congestion window increases as a function of that time. A manipulative receiver can speed transfers by optimistically sending acknowledgments for packet sequence numbers that have not arrived yet.

**Hiding congestion in ECN [21]:** In the ECN protocol, routers set a bit in the header to signal congestion. The receivers then reflect this bit when sending packets to the sender, at which point the senders reduce their transfer rate. A manipulative receiver can speed transfers by setting the congestion bit to zero instead of reflecting it.

**Dropping packets in multi-hop wireless networks [14]:** Multi-hop wireless networks enable connectivity by having nodes relay for one another. Manipulative nodes can simply drop all packets that they are asked to relay while reaping the benefits by having others relay for them.

**Lying about connectivity in routing protocols [1]:** Routing protocols such as OSPF and BGP rely on nodes accurately reporting their connectivity, i.e., who they connect to and the connection cost. Manipulative nodes can significantly distort routing paths by lying.

**Attacks in DHTs [4]:** Castro *et al.* present a variety of ways in which a manipulative node can hurt the overlay. The set of possible manipulations is large. It includes how the node identifiers are generated, how routing messages are propagated, and how messages are forwarded.

All the vulnerabilities above were discovered manually. We want to automate the search for these kinds of vulnerabilities. Researchers have proposed fixes to these vulnerabilities. Automated tools can also help determine if the fixes are themselves robust.

## 3. PROBLEM STATEMENT

Consider a communication protocol that two or more parties use in order to achieve a common goal. The honest participants execute the protocol correctly. The manipulators have complete freedom in what they choose to send (or not send) and when. Multiple manipulators may also collude. We seek to determine if the protocol is gullible, that is, the manipulators can prevent the honest participants from achieving the goals of the protocol.

We formalize this problem as a two-player game between $A$ and $D$, the *angelic* and *demonic* components. $A$ consists of honest players. These players communicate with each other and with $D$ by exchanging messages. $D$ consists of manipulative players, possibly colluding through out-of-band mechanisms. Finally, assume that there is a property $P$ over the state of $A$ and $D$ that represents an invariant that the protocol should never violate.

The game proceeds by alternating moves of the two components. On its turn, $A$ chooses one of the possible actions allowed by the protocol, such as a packet receive or a timer event. The choices available to $A$ represent the nondeterminism that is outside $D$'s control. In particular, actions of the network, e.g., packet losses, are considered moves of $A$. $D$ responds by consulting its attack strategy. Its move involves either sending an arbitrary packet to $A$ or choosing not to respond. The protocol under study is gullible if there exists a strategy for $D$ with which $D$ can eventually drive the system to a state that violates $P$.

Our problem formulation has two notable features. First, we distinguish between angelic and demonic nondeterminism, because different methods are required to systematically search over them. Given a state, the angelic component usually has a handful of moves that obey the rules of the protocol. The demonic component can, however, send an arbitrary packet, representing an astronomically huge search space at each step. Moreover, the demonic component might use a stateful strategy, whereby the choices made at one step depend on the preceding choices. In contrast to our formulation, the current work on model checking system implementations [7, 17, 11] has either no demonic component or a very restricted one.

Second, the property $P$ that identifies when manipulation has happened depends on the protocol under test. As a simple example, $P$ can state that the connection queue should not be able to remain full forever, which represents a denial-of-service attack. A more complicated example property would limit the throughput that can be obtained by the demonic component. Checking for violations of this kind of property is important, since many discovered manipulations concern resource allocation attacks [19, 21, 14, 4]. For such cases, we propose that $P$ be specified in terms of a comparison to a "reference" behavior that occurs if the demonic component were to honestly follow the protocol. For example, congestion control protocols could require that no receiver gets more data than what it would get by following the protocol; DHTs and wireless relaying protocols could require that the ratio of packets relayed and generated not change; and routing protocols could require that forwarding table pointers not change. More generally, we could specify that important state variables for a protocol not change.

## 4. CHALLENGES AND TECHNIQUES

We now describe the challenges in building a practical tool to test protocol gullibility and propose techniques to address these challenges.

## 4.1 Challenge: Practical Strategy Search

The primary challenge that we face is that the space of possible demonic strategies is huge. There are $2^{12000}$ possibilities for a 1500-byte packet that the manipulator can send. Further, that is just the size of the search space for a single message; complicated attacks may depend on sending a particular sequence of packets.

We propose a variety of techniques to reduce the search space size. These techniques leverage the structure inherent in network protocols as well as some properties that are common across large classes of protocols. While these search-space reduction techniques can potentially cause our tool to miss some attacks, we believe that many vulnerabilities of interest, including most of the ones described in Section 2, can still be found. Systematically discovering such vulnerabilities would be a significant improvement in the current state of art and is a first step towards automatically discovering other kinds of attacks.

• *Consider only the header part of the packet* The control flow of most protocols is based on the contents of the header and not on the payload. Hence, instead of considering the entire packet, we focus on headers. Within a packet, what is header versus payload depends on the protocol being tested. For instance, for IP everything other than the IP header constitutes the payload even though that payload may contain TCP headers. This technique enormously reduces the search space. If the header size is 40 bytes, for 1500-bytes packets, the single-step search space goes from $2^{12000}$ to $2^{320}$.

• *Consider only syntactically correct packets* Protocol packets are not random bit strings but have specific formats. For instance, the checksum field occurs in a certain location and is computed over specific bits. Honest participants typically discard incoming packets that are not in the requisite format. Accordingly, rather than searching the space of all bit patterns, we focus on packets that comply with that format. The packet format can be provided by the user or automatically inferred [6] from traces. While there may be some attacks that involve non-compliant packets, these are very likely to be due to implementation bugs, such as buffer overflows. We are not interested in finding such bugs; other methods, such as fuzzing [16, 8], are more apt for finding them.

• *Exploit header-field independence* To further reduce the search space, we assume that most header fields are independent of one another with respect to possible vulnerabilities. That is, most vulnerabilities can be discovered through a systematic search within the possible values for groups of fields, keeping the values of other fields unchanged. For instance, for TCP one might assume that the sequence number and acknowledgment number fields are independent, allowing us to independently search the fields for possible attacks. Since each field is 32 bits, searching them together would yield a space of $2^{64}$ possible values, which searching them independently yields a much smaller space of $2 \times 2^{32} = 2^{33}$ values. We rely on the user to provide information about which fields should be considered independent of one another.

• *Consider only limited-history strategies* We expect that many interesting attacks require the manipulator to take only a small number of basic steps. The attack is then carried out by repeated application of this small sequence of steps. For instance, the ECN vulnerability mentioned earlier is a single-step strategy: the manipulator needs to send a specific bit pattern in order to get more bandwidth. Therefore, our tool can bound the length of strategies that it will consider to a small constant.

• *Leverage program analysis techniques* As mentioned above, we consider only syntactically correct packets, since other packets are typically ignored. However, many syntactically correct packets may also be ignored by an honest participant. For instance, if IPv4 is the protocol being checked, packets with anything but the value of 4 in the version field are ignored. As another example, TCP senders ignore acknowledgments for sequence numbers below the last acknowledged sequence number. We propose to identify the legal values of header fields using program analysis of protocol source code; our search can then ignore other values for these fields.

Program analysis can also help direct the strategy search itself by identifying conditions under which an honest participant's state can change in ways that are beneficial to a manipulator. For instance, program analysis of a TCP implementation can determine values of header fields in a received packet that cause the honest participant to send more bytes. We intend to allow the tool user to provide a set of variables in the honest participants' state, and program analysis will direct the search to strategies that cause these variables' values to change.

For both types of analyses above, we hope to leverage recent work on *directed random testing* [9], which combines *symbolic execution* [12] with program testing.

## 4.2 Challenge: Variable network conditions

Some protocol vulnerabilities are exploitable only under certain network conditions. For instance, the ECN bit-flipping vulnerability comes to light only if the network is congested and thus marks congestion bits in some packets. If we simulate only uncongested paths, we will not be able to uncover this ECN vulnerability.

Therefore, for each strategy considered during strategy search, we must search the space of possible network path behaviors. We make the simplifying assumption that the paths between each pair of participants are independent of one another. We can then separately characterize each such path, for instance, by its loss rate and latency. The behavior of each path defines a space of network conditions, which our tool searches for each demonic strategy.

| Type | Description | Default strategies |
|---|---|---|
| Fixed | Fields that should not be modified | None |
| Checksum | Fields that represent a checksum over certain bits in the header | None |
| Enum | Fields that take on specific bit patterns. E.g., Protocol field in IP header | Pick a value deterministically; pick one at random |
| SeqNum | Fields that represent sequence numbers | Subtract or add a constant value; multiply or divide by a constant value |
| Range | Fields that take on a range of value. E.g., Addresses and port numbers | Pick a value deterministically; pick one at random |
| Id | Fields that contain identifiers. E.g., IP identifier in IP header, or node identifier in peer-to-peer protocols | Pick one at random |
| Other | All other fields | User-specified |

**Table 1: Fields types and default strategies in our system.**

### 4.3 Challenge: Determining when a strategy has been successful

As mentioned earlier, for complex properties, particularly those related to resource allocation, a strategy's success cannot be determined simply by running the strategy; instead we must compare against reference behavior when all participants are honest. However, two individual runs of the protocol cannot be meaningfully compared directly, due to the nondeterminism in the angelic component (e.g., network conditions). For example, under a given network condition of 20% loss rate, the behavior of TCP is not unique but depends on which specific packets are lost.

Therefore, to compare a manipulated protocol against reference behavior, we run each version of the protocol multiple times under a given network condition, obtaining a distribution for each over some set of metrics (e.g., number of messages sent in a given amount of time). We can then use statistical tests (e.g., the Kolmogorov-Smirnov test or Student's *t*-Tests) to determine if the two distributions are statistically different.

## 5. PROTOTYPE CHECKER

To study the feasibility of automatically checking protocols for gullibility, we are implementing a prototype checker. Our current implementation uses all of the techniques described above except for program analysis. Further, rather than asking the user for information about field independence, we simply assume that all fields are independent of one another. Finally, we use network simulation to explore angelic nondeterminism.

Our system works directly with protocol implementations written in the Mace language for distributed systems [10]. We create the manipulators as modifications of an "honest" implementation of the protocol. At a high level, the manipulator has three types of actions that cover the space of possible strategies: $i)$ drop packets that honest players would have sent; $ii)$ modify the contents of packets that honest players would have sent; and $iii)$ send packets when honest players would not have sent a packet.

We have created a simple API for dropping and modifying packets, along with a default implementation of the API (a C++ class). Wherever the protocol source code calls the function to send a packet, we insert a call into our API's `modify_or_drop` function, passing the packet to be sent. Our function returns the modified packet and indicates whether it should be dropped.

We have not yet implemented packet insertions at arbitrary times, the third action above. Our plan is to add a timer to the protocol implementation. An arbitrary packet can be sent when this timer fires.

The protocol tester provides three required inputs and one optional input to our system:

**1. Network configuration** This includes the number of participants, and the fraction of participants that are honest. If the protocol is asymmetric, such as TCP sender vs. receiver, the tester also specifies which aspect to test. This setup information is distinct from network path conditions (e.g., loss rate) for which we automatically explore the various possibilities. In the future we plan to automatically generate and test a range of possible setups as well.

**2. Variables of interest** The tester specifies the property to be tested indirectly by providing a set of variables of interest within the protocol implementation. The intent is that the tester is interested in ways in which a manipulator can cause deviations in the values of these variables. If the implementation does not already have the necessary variables, we expect the testers to implement them. For instance, while the current congestion window is an existing state variable in a TCP implementation, total traffic sent may not be, but it can be easily implemented.

**3. Protocol header format** The format is specified in terms of the header fields and their types. The types that we currently use are listed in Table 1. This list is based on an informal survey of several common protocols and it may grow as we experiment with more protocols. For some of the types, e.g., *Enum* and *Range*, the format must specify the possible values of their fields.

**4. A packet modifier class** The packet modification operation produces a modified version of the input packet. Each header field is modified independently, depending on its type. The default modification strategies are shown in Table 1. All default modification strategies are memoryless: they do not depend on what was sent before. Users can optionally provide their own packet modifier class implementing our API, in order to specify their own strategies for these and other field types.

Given these inputs, we use the simulation engine provided by the MACE framework [11] to evaluate protocols.

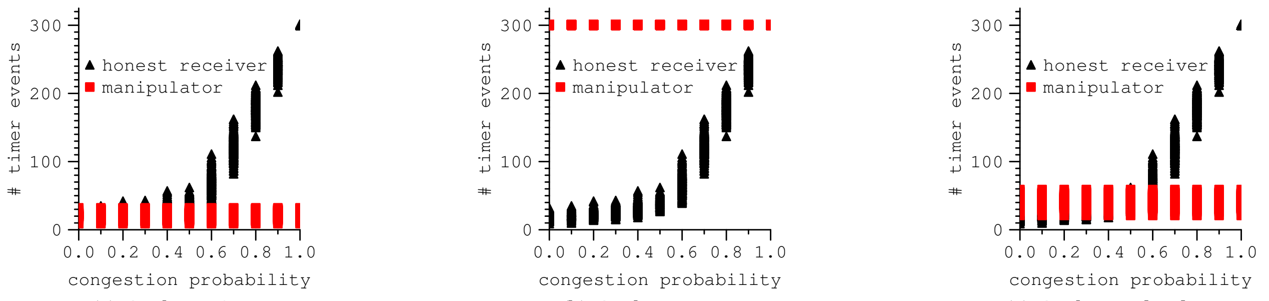


**Figure 1: Results of using our prototype to check the gullibility of the ECN protocol. Each graph corresponds to a different cheating strategy by the receiver. The $x$-axis indicates the fraction of packets on which the congestion bit was set. The $y$-axis is the number of timer events to deliver 300 packets to the receiver.**

In each run of the simulator, we fix a particular set of network path conditions as well as a particular modification strategy for each header field. The cross product of network path conditions and possible modification strategies thereby forms the state space that our engine explores.

For each pair of a set of network conditions and a modification strategy, we simulate the protocol multiple times. We also simulate multiple runs of a completely honest version of the protocol for each possible set of network conditions. If there are statistically significant differences in the values of variables of interest between the honest and manipulated runs for a given strategy and set of network conditions, we deem the protocol to be gullible.

## 6. CASE STUDY: ECN

We now present results from using the preliminary version of our tool to test the ECN protocol. We chose to start with ECN because of its simplicity. Nonetheless, a study of even this simple protocol reveals many relevant insights and provides initial evidence for the feasibility of automatic gullibility checking.

We implemented a version of the ECN protocol in the Mace framework. Its header has only one field, of type Enum with 0 and 1 as the possible values. The field value is initially 0, and it is set to 1 by the network to indicate congestion. The sender starts by sending one packet to the receiver. The receiver acknowledges each received packet. The honest receiver's acknowledgment reflects the value of the congestion bit in the packet it received. Dishonest receivers are free to do anything. In response to receiving an acknowledgment that does not indicate congestion, the receiver sends two new packets. If congestion is indicated, the sender does not send any new data. In addition, the sender has a timer that fires periodically. If no packet has been sent since the last firing, the sender sends a new packet, to keep the information flow going.

We specify the network setup as having two nodes, one sender and one receiver, and we tell our tool to investigate manipulation by receivers. We also specify that protocol behavior should be measured in terms of the number of timer events. This measure indirectly captures the total time needed to send a particular number of packets. Because we do not explicitly simulate time, we cannot directly measure throughput. Finally, we emulate different network conditions by configuring different probabilities of setting the congestion bit.

The graphs in Figure 1 show the protocol behavior as a function of the network conditions. They show this behavior with both the honest receiver as well as with different cheating strategies. The set of cheating strategies in this case involve setting the congestion bit to 0, 1, or randomly. We conduct 200 trials for each combination of receiver strategy and network conditions, with each trial simulating the sending of 300 packets. We show all data points thus obtained, to demonstrate the variance with network conditions.

The figure shows how our tool can automatically determine which strategies work and quantify their impact. It shows that a misbehaving receiver can speed transfers by always setting the bit to zero and it can slow transfers by always setting the bit to one. Further, the misbehaving receiver can speed transfers even by setting the congestion bit randomly.

The graphs show that the impact of a cheating strategy is visible only under certain network conditions. For instance, the impact of setting the congestion bit to zero does not show until 40% of the packets have the congestion bit set and that of setting the bit randomly does not show until 60% of the packets have the congestion bit set. This behavior points to the importance of simulating different network conditions along with cheating strategies. Without simulating different network conditions, a gullibility checker might incorrectly infer that certain strategies are unsuccessful.

## 7. RELATED WORK

We are directly inspired by the recent success [7, 17, 22, 11] of systematic search techniques, as implemented by a model checker, in finding safety and liveness errors in system implementations. This class of research focuses on scaling to large systems and on nondeterminism arising from the timing of various events in the system. The demonic nondeterminism considered is fairly restricted and is limited to the network dropping packets or a system reboot occurring at an arbitrary instant. A straightforward extension of these techniques to include actions of an all-powerful malicious attacker does not scale.

Another related class of prior research applies formal verification techniques to validate abstract models of security protocols against malicious attacks [13, 15, 20, 18, 5]. Our work is most closely related to the use of model checking [20, 5] to systematically enumerate the behavior of the protocol in the presence of an attacker aiming to acquire a secret only known to honest participants. The attacker models considered are powerful and can generate arbitrary packets by combining parts of previously exchanged messages and publicly known encryption keys. However, due to the inherent state-space explosion problem, the problem instances considered are small hand-abstracted models of security protocols with very few message exchanges. In contrast, our work targets larger general-purpose network protocol implementations. We also check more complex properties, in particular those that compare the attacker against an honest version under the same network conditions. The price for these generalizations is that our approach is appropriate for finding attacks but not for producing a formal proof of correctness.

Automated fuzzing techniques [16, 2], which subject a system to random sequences of inputs, can find many errors in systems, some of which can lead to malicious attacks. As an extension, directed random testing [9, 3, 8] generates inputs based on symbolic execution of the program. These techniques have been used to find implementation errors, particularly errors in validating inputs. Our focus is on finding vulnerabilities in the protocol design itself. Therefore we restrict ourselves to well-formed input packets. However, as mentioned earlier we are interested in adapting fuzzing and testing techniques to our setting, in order to reduce the search space.

## 8. CONCLUSIONS

We have proposed and defined the problem of automatically checking protocols for gullibility, i.e., their vulnerability to manipulation by some of the participants. We identified the challenges in building a practical tool for this task and proposed several techniques to address them. We are currently developing a prototype checker using these techniques. Early results from using our tool to check the ECN protocol are promising, automatically identifying vulnerabilities and the network conditions under which they are exploitable. We are excited to improve our tool and apply it to new classes of protocols.

**Acknowledgments** We thank Stefan Savage, Amin Vahdat, and David Wetherall for feedback on this paper. This work was supported in part by NSF Grants CCF-0427202, CCF-0545850, and CNS-0725354.

## 9. REFERENCES

[1] A. Barbir, S. Murphy, and Y. Yang. Generic threats to routing protocols. Technical Report RFC-4593, IETF, Oct. 2006.

[2] Browser Fuzzing for Fun and Profit. http://blog.metasploit.com/2006/03/browser-fuzzing-for-fun-and-profit.html.

[3] C. Cadar, V. Ganesh, P. M. Pawlowski, D. L. Dill, and D. R. Engler. Exe: automatically generating inputs of death. In *ACM Conference on Computer and Communications Security*, pages 322–335, 2006.

[4] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI*, Dec. 2002.

[5] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000.

[6] W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irun-Briz. Tupni: Automatic reverse engineering of input formats. In *The 15th ACM Conference on Computer and Communications Security (CCS)*, 2008.

[7] P. Godefroid. Model checking for programming languages using verisoft. In *Principles of Programming Languages (POPL)*, Jan. 1997.

[8] P. Godefroid, A. Kiezun, and M. Y. Levin. Grammar-based whitebox fuzzing. In *PLDI*, pages 206–215, 2008.

[9] P. Godefroid, N. Klarlund, and K. Sen. Dart: Directed automated random testing. In *PLDI '05: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 213–223, New York, NY, USA, 2005. ACM.

[10] C. Killian, J. W. Anderson, R. Baud, R. Jhala, and A. Vahdat. Mace: Language support for building distributed systems. In *PLDI*, June 2007.

[11] C. Killian, J. W. Anderson, R. Jhala, and A. Vahdat. Life, death, and the critical transition: Finding liveness bugs in system code. In *NSDI*, Apr. 2007.

[12] J. C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7), 1976.

[13] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.

[14] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Encouraging cooperation in multi-hop wireless networks. In *NSDI*, May 2005.

[15] C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[16] B. P. Miller, L. Fredriksen, and B. So. An empirical study of the reliability of unix utilities. *Commun. ACM*, 33(12):32–44, 1990.

[17] M. Musuvathi, D. Park, A. Chou, D. Engler, and D. Dill. CMC: A pragmatic approach to model checking real code. In *OSDI*, Dec. 2002.

[18] L. C. Paulson. Proving properties of security protocols by induction. In *CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations*, page 70, Washington, DC, USA, 1997. IEEE Computer Society.

[19] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *Computer Communication Review*, 29(5), 1999.

[20] V. Shmatikov and U. Stern. Efficient finite-state analysis for large security protocols. In *CSFW*, pages 106–115, 1998.

[21] D. Wetherall, D. Ely, N. Spring, S. Savage, and T. Anderson. Robust congestion signaling. In *ICNP*, Nov. 2001.

[22] J. Yang, P. Twohey, D. R. Engler, and M. Musuvathi. Using model checking to find serious file system errors. In *OSDI*, pages 273–288, 2004.

# Revisiting Smart Dust with RFID Sensor Networks

Michael Buettner
University of Washington

Ben Greenstein
Intel Research Seattle

Alanson Sample
University of Washington

Joshua R. Smith
Intel Research Seattle

David Wetherall
University of Washington, Intel Research Seattle

## ABSTRACT

We argue that sensing and computation platforms that leverage RFID technology can realize "smart-dust" applications that have eluded the sensor network community. RFID sensor networks (RSNs), which consist of RFID readers and RFID sensor nodes (WISPs), extend RFID to include sensing and bring the advantages of small, inexpensive and long-lived RFID tags to wireless sensor networks. We describe sample applications suited to the space between existing sensor networks and RFID. We highlight the research challenges in realizing RSNs such as the use of intermittent power and RFID protocols suited to sensor queries.

## 1 INTRODUCTION

In the late 1990s, the vision of "smart-dust" was articulated by the research community. This vision was predicated on advances in microelectronics, wireless communications, and microfabricated (MEMS) sensing that were enabling computing platforms of rapidly diminishing size. The early proponents imagined devices one cubic millimeter in size with capabilities sufficient to power themselves, sense the environment, perform computation, and communicate wirelessly [7]. Large-scale deployments of such devices would enable a wide range of applications such as dense environmental monitoring, sensor rich home automation and smart environments, and self-identification and context awareness for everyday objects.

The past decade has seen significant effort and progress towards the original motivating applications. In particular, wireless sensor networks (WSNs) based on "mote" sensing platforms have been applied to many real-world problems. Remote monitoring applications have sensed animal behavior and habitat, structural integrity of bridges, volcanic activity, and forest fire danger [5], to name only a few successes. These networks leveraged the relatively small form-factor (approximately 1" x 2") of motes and their multihop wireless communication to provide dense sensing in difficult environments. Due to their low power design and careful networking protocols these sensor networks had lifetimes measured in weeks or months, which was generally sufficient for the applications.

Despite this success, WSNs have fallen short of the original vision of smart-dust. They have not led to an approximation of sensing embedded in the fabric of everyday life, where walls, clothes, products, and personal items are all equipped with networked sensors. For this manner of deployment, truly unobtrusive sensing devices are necessary. The size and finite lifetime of motes make them unsuitable for these applications.

We argue in this paper that Radio Frequency Identification (RFID) technology has a number of key attributes that make it attractive for smart-dust applications. Passive UHF RFID already allows inexpensive tags to be remotely powered and interrogated for identifiers and other information at a range of more than 30 feet. The tags can be small as they are powered by the RF signal transmitted from a reader rather than an onboard battery; aside from their paper thin antennas, RFID tags are approximately one cubic millimeter in size. Moreover, their lifetime can be measured in decades as they are reliable and have no power source which can be exhausted. These advantages have resulted in the widespread deployment of RFID for industrial supply-chain applications such as tracking pallets and individual items. However, RFID technology is limited to only identifying and inventorying items in a given space.

The RFID Sensor Networks (RSNs) we advocate in this paper extend RFID beyond simple identification to in-depth sensing. This combines the advantages of RFID technology with those of wireless sensor networks. In our previous work, we have demonstrated the technical feasibility of building small, battery-free devices that use the RFID PHY and MAC layer to power themselves, sense, compute, and communicate; we refer to these devices as Wireless Identification and Sensing Platforms (WISPs)[12, 13]. While other research efforts such as [3] have combined RFID with sensing, to the best of our knowledge, the Intel WISP is the only RFID sensor node with computational capabilities and that operates in the long range UHF band.

While the feasibility of WISPs has been established by this earlier work, how to harness many such devices to create RSNs is an open question. An RFID sensor net-

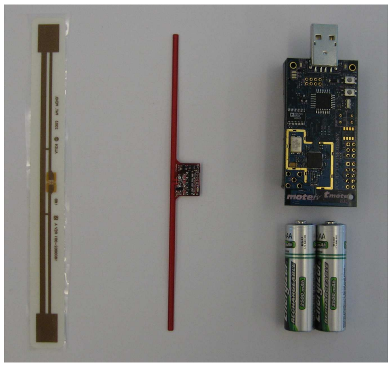


**Figure 1**: Commercial UHF RFID tag, Accelerometer WISP, Telos mote with batteries

work consists of multiple WISPs and one or more readers. Consequently, realizing full-scale RSNs will require development at both the WISP and the reader, as new protocols and techniques must be developed unlike those of either RFID or WSNs.

The focus of this paper is the applications that RSNs enable and the systems challenges that must be overcome for these to be realized. As the traditional RFID usage model is very different from that of WSNs, RSNs face substantial challenges when trying to integrate the two technologies. For example, unlike WSNs, RSNs must cope with intermittent power and unlike RFID must support sensor queries rather than simply identification.

## 2 From Motes and RFID to RSNs

Two technologies have been widely used to realize real-world monitoring applications: wireless sensor networks via motes, and RFID via standard tags and readers. We describe and contrast each technology and then present their combination (Table 1) as RFID sensor networks (RSNs). We use prior work on the WISP [12, 13] to demonstrate the technical feasibility of this combination. Representative devices for the three technologies are show in Figure 1.

### 2.1 Wireless Sensor Networks (Motes)

Currently, most WSN research is based on the Telos mote [10], which is a battery powered computing platform that uses an integrated 802.15.4 radio for communication. These motes are typically programmed to organize into ad-hoc networks [15] and transmit sensor data across multiple hops to a collection point. To extend network lifetime, motes duty cycle their CPU and radio (e.g., with low-power listening [9]), waking up intermittently to sense and communicate. With a duty cycle of 1%, motes can have a lifetime of up to three years before the batteries are exhausted.

Using multihop communication, WSNs can sense over great distances, which has made them idea for a wide range of applications. However, the large size of the mote and its finite lifetime makes it unsuitable for applications where sensing must be embedded in small objects, or in inaccessible locations where batteries cannot be replaced.

### 2.2 RFID

While there are a number of different RFID specifications, that of greatest interest for sensing applications is the EPCglobal Class-1 Generation-2 (C1G2) protocol [4], as it is designed for long-range operation. The C1G2 standard defines communication between RFID readers and passive tags in the 900 MHz Ultra-High Frequency (UHF) band, and has a maximum range of approximately 30 feet. A reader transmits information to a tag by modulating an RF signal, and the tag receives both down-link information and the entirety of its operating energy from this RF signal. For up-link communication, the reader transmits a continuous RF wave (CW) and the tag modulates the reflection coefficient of its antenna. By detecting the variation in the reflected CW, the reader is able to decode the tag response. This is referred to as "backscattering," and requires that a tag be within range of a powered reader.

The MAC protocol for C1G2 systems is based on Framed Slotted Aloha [11], where each frame has a number of slots and each tag will reply in one randomly selected slot per frame. Before beginning a frame, a reader can transmit a *Select* command to reduce the number of active tags; only tags with ID's (or memory locations) that match an included bit mask will respond in the subsequent round. After a tag replies, the reader can choose to *singulate* the tag, or communicate with it directly, and read and write values to tag memory. These mechanisms enable rapid tag identification and unicast read and write.

RFID tags are fixed function devices that typically use a minimal, non-programmable state machine to report a hard-coded ID when energized by a reader. As they are powered by the reader, the device itself can be very small, though the antenna requires additional area. As the antenna is flexible and paper thin, their small size means they can be affixed to virtually any object to be identified However, RFID tags provide no general purpose computing or sensing capabilities.

### 2.3 RFID sensor networks (WISPs + readers)

We define RFID sensor networks (RSNs) to consist of small, RFID-based sensing and computing devices (WISPs), and RFID readers that are part of the infrastructure and provide operating power. RSNs bring the advantages of RFID technology to wireless sensor networks. While we do not expect them to replace WSNs for all applications, they do open up new application spaces where small form-factor, long-lived, or inaccessible devices are paramount. Our hope is that they will

| | CPU | Sensing | Communication | Range | Power | Lifetime | Size (inches) |
|---|---|---|---|---|---|---|---|
| WSN (Mote) | Yes | Yes | peer-to-peer | Any | battery | $< 3$ yrs | 3.0 x 1.3 x .82 (2.16 $in^3$) |
| RFID tag | No | No | asymmetric | 30 ft | harvested | indefinite | 6.1 x 0.7 x .02 (.08 $in^3$) |
| RSN (WISP) | Yes | Yes | asymmetric | 10 ft | harvested | indefinite | 5.5 x 0.5 x .10 (.60 $in^3$) |

**Table 1**: Comparison of Technologies

elegantly solve many sensor network applications, e.g., home sensing and factory automation where installing or carrying readers is feasible.

Prior work at Intel Research demonstrates that WISPs can be built today. The most recent Intel WISP is a wireless, battery-free platform for sensing and computation that is powered and read by a standards-compliant UHF RFID reader at a range of up to 10 feet. It features a wireless power supply, bidirectional UHF communication with backscatter uplink, and a fully programmable ultra-low-power 16-bit flash microcontroller with analog to digital converter. This WISP includes 32K of flash program space, an accelerometer, temperature sensor, and 8K serial flash. Small header pins expose microcontroller ports for expansion daughter boards, external sensors and peripherals.

The Intel WISP has been used to implement a variety of demonstration applications that read data from a single sensor unit. These include the first accelerometer to be powered and read wirelessly in the UHF band, and also the first UHF powered-and-read strain gage [17]. Even without its sensing capabilities, the Intel WISP can be used as an open and programmable RFID tag: the RC5 encryption algorithm was implemented on the Intel WISP [2]. We believe this is the first implementation of a strong cryptographic algorithm on a UHF tag.

## 3 Example Applications

RFID sensor networks have broad applicability wherever sensing, small form factor, embeddability, longevity, and low maintenance are desired and fixed or mobile readers are feasible. This section highlights applications within this space and some of the key design considerations.

### 3.1 Blood

Blood transfusions save lives, replacing blood lost during surgery, illness, or trauma. After donation, blood is bagged and refrigerated between 1° and 6° C and has a shelf life of about 35 to 42 days. Refrigerators used to store blood are monitored for outages and temperature fluctuations, and collection dates are recorded on blood bags. However, the temperature of the bag itself is rarely monitored with any regularity. This makes it difficult to determine if a given bag was warmed to unsafe levels, such as if it is near the front of the refrigerator and the door is often opened. Additionally, it is difficult or impossible to gauge exposure during transport from a donor to a bank, between banks, and ultimately to a patient.

WISPs with temperature sensors could be attached directly to individual blood bags and queried for their measurements. Such sensors must be small (one could imagine affixing sensors with something like a price tag gun), and inexpensive to the point of being disposable.

To understand the challenges in building such an application, an Intel WISP was attached to a container of milk (a suitable and widely available approximation of a bag of blood), and its temperature was monitored over the course of 24 hours [16]. For this study, a storage capacitor (roughly the size of a pea) was attached to the WISP to log sensor data for up to a day when out of range of a reader.

### 3.2 Brains

Research in neuroscience has explored using neural sensors for controlling prosthetic limbs [14]. Sensors placed outside the skull can capture neural activity, but the signals are too coarse-grained and noisy to be effective. With surgery, sensors can be placed directly on the brain, resulting in much higher resolution and finer control of the limb. Using conventional technologies (e.g., motes) presents difficulties with respect to power because batteries need to be replaced via invasive surgical procedures, as is the case with pacemakers.

An RFID sensor network is well suited to this application. A patient would have WISPs equipped with neural probes placed inside the skull. These could then draw power from and communicate with a device outside the body, e.g., an RFID reader worn as a cap, bracelet, or belt. We have completed initial studies that show the feasibility of integrating neural sensors with the WISP [6].

### 3.3 The Elderly

Providing care for the elderly is one of the largest healthcare costs facing us today, particularly as the "baby boomer" generation ages. Keeping people in their homes for as long as possible significantly reduces these costs and increases quality of life. The difficulty with this is in detecting and reacting to emergencies, such as the patient falling or forgetting to take critical medication. Currently, families have no choice but to hire costly support personnel to regularly check-in on their loved ones.

Traditional RFID has been explored to help monitor the behavior of the elderly. For example, by having the patient wear a short range RFID reader bracelet and placing RFID tags on a toothbrush, toothpaste, and faucet, software can infer that an elderly person is brushing her teeth when these tags are read in succession [8]. Such fine-grained sensing requires very small devices, and is simpler and more respecting of privacy than competing approaches using computer vision, where video of the person is continuously recorded and analyzed.

Adding sensing (e.g., an accelerometer) to long range RFID tags would have several key advantages. Rather than requiring a person to wear a short-range reader, which can be taken off, a few long-range readers could be placed in the home and behavior could be determined via direct communication with the objects that are being interacted with. This explicit information would be more accurate in detecting behavior than inference based only on object identifiers.

RSNs are an appropriate solution for the above applications and those like them. Our initial studies using the WISP show the potential of existing RFID sensing devices for use in such applications. However, these studies involved only a single WISP. Combining many such devices into a full RSN will require further research.

## 4 Challenges

RSNs combine the technology of RFID and sensing with the usage models of sensor networks. At the device level, the WISP shows that it is feasible to combine sensing with RFID. However, at the systems level, challenges arise due to the mismatch between the RFID usage model and that of wireless sensor networks. We detail several challenges in this section.

### 4.1 Intermittent Power

RFID readers provide an unpredictable and intermittent source of power. This makes it difficult for WISPs to assure that RSN tasks will be run to completion. WISPs are powered only when in range of a transmitting RFID reader and, for regulatory and other reasons, readers do not transmit a signal continuously. Instead, they transmit power for a brief period before changing channels or entirely powering down. For standard RFID tags where the task is simply to transmit the identifier, this style of communication is sufficient. However, it is a poor fit for RSN tasks that span many RFID commands.

The WISP harvests energy from a reader and stores this energy in a capacitor. When enough energy is harvested, the WISP powers up and can begin sensing and communicating. However, sensing and communication drain power from the WISP. This can result in the WISP losing power in the middle of an operation depending on the task and the reader behavior. A further complication is that receiving, transmitting, performing computation, and reading/writing to memory all consume different amounts of energy.

To run tasks to completion, WISPs will require support for intermittently powered operation. They must be able to estimate the energy required to complete a task, perhaps based on task profiling or energy budgets, and compare it with estimated reserves. To work well in this regime, RSN devices may also need to cooperate with RFID readers for power management. This would involve signaling by either the reader, of its intended transmission time, or by the WISP, of its needs. Even with signaling, it will be difficult to predict power expectations because the rate at which energy is harvested depends on the frequency of the reader and the proximity of the device to the reader, both of which will change over time. Thus, to increase the kinds of tasks that could be supported, large tasks might need to be split into smaller, restartable stages; intermediate results between the stages could be maintained in device storage (flash or RAM) or be offloaded to the reader.

To extend functionality when away from a reader, one approach would be to provide a small amount of energy storage on the device, e.g., a capacitor, and store excess energy when close to an active reader. This storage capacitor would be small relative to a battery, because it would be intended only for short term usage and is wirelessly recharged over time. The Data Logger WISP used for the milk carton study takes this approach, using a super-capacitor that, when fully charged, sustains low duty-cycle operation for more than a day. The type of tasks that this WISP enables would be limited, due to energy requirements, and the period of functionality would be limited due to leakage. Additionally, unpowered operation would likely stress tradeoffs between stages. For example, writing to flash is significantly more energy intensive than computing with RAM but preserves valuable data for later use.

### 4.2 Asymmetric Sensing Protocols

The communication paradigm of RFID results in systems that are limited by up-link bandwidth. When the data of interest is simply each tag's identity, this constraint is not a problem. However, it makes it difficult to develop efficient protocols for gathering sensor data that changes over time. In WSNs, nodes are peers in terms of the physical and link layers of their communication, e.g., each mote has an 802.15.4 radio capable of sending and receiving transmissions with other nodes that are in range. In contrast, because they draw on RFID, RSN nodes are highly asymmetric in terms of their communication abilities. With RFID, readers are able to transmit messages to all tags and tags can transmit messages to the reader. However, tags can do so only when the reader initiates communication, and tags cannot communicate directly with each other even when powered by the reader.

These differences complicate the design of protocols for gathering sensor data. Currently, WISPs with new sensor data must wait until they are interrogated by a reader. This increases the likelihood of many devices wanting to use the bandwidth limited channel at the same time. Techniques to perform data pre-processing within the network (on each RSN device) could help to some extent. However, the standard RFID strategy of identifying and then communicating with each device is wasteful as only some devices would have relevant data – a more dynamic strategy based on the value of the sensor data would be more effective.

Consider the eldercare application. A reader might have hundreds of accelerometer WISPs in its field of view. Because all the WISPs share a single reader channel, the update rate per tag would be very low if every tag were simply queried for sensor data sequentially. However, at any given moment, only a few objects would typically be in motion (and therefore producing non-trivial accelerometer sensor values). Furthermore, the set of objects that are moving would change dynamically, as objects are put down and picked up. One might want a protocol which gives priority to the most active objects, politely "yielding" to new objects when they start to move.

Existing RFID solutions do not support anything like this functionality. As a first step, one could have WISPs with sensor activity below a threshold not respond to the reader. But an appropriate threshold level might depend on what is occurring in the room, and such a simple scheme would not support the "polite yielding" described above.

For another example of what RSN protocols might be asked to do, consider the blood application. When many blood bags are read simultaneously, one might want to interrogate the bags with the largest temperature excursions first. But since the distribution of temperature excursions would not be known a priori by the reader, the protocol would need to (implicitly) estimate this information. It might for example ask if any WISP has a larger temperature excursion than $E$. If no device responds, the $E$ response threshold could be repeatedly halved until the appropriate scale is found. The key requirement would be to estimate an aggregate property of the data without exhaustively collecting that data. Finally, RSN protocols might be power aware as well. A WISP that was about to lose power might be given priority over those with ample power.

As the WISP has limited program space, it may not be possible to program a WISP such that it will be well matched to all possible application scenarios. For example, WISPs may need different protocols for different deployments, and these needs may change over time. However, the communication model of passive RFID means that the reader can have a large degree of control over what code is loaded and executed on the WISPs at any point in time.

In contrast to mote message reception, which consumes energy from small onboard batteries, WISP message reception is powered entirely by the reader. Thus, frequent and large code transfers are feasible, which would allow for the complete retasking of WISPs with costs in terms of latency only. Moreover, since downlink communication is cheap when in range of the reader, WISPs might not need to be as "smart" as motes. Rather than requiring WISPs to interpret queries, readers could tell WISPs exactly what to do, down to the instruction level.

To fully exploit the potential of RSNs, new tools must be developed. As a first step, we are developing an RFID reader platform based on the Universal Software Radio Peripheral (USRP). This platform, when used in conjunction with the WISP, would allow for the development of new protocols at both the MAC and PHY layers. Thus far we have used it for monitoring RFID systems [1].

### 4.3 Repurposing C1G2

There would be substantial practical benefit to realizing RSN protocols using the primitives of the C1G2 standard: Commercial off-the-shelf readers could be used for RSN research and deployment, and WISPs would interoperate with ordinary (non-sensing) tags. However, the extent to which RSN protocols could be implemented within the C1G2 standard is an open research question. Additionally, there is the practical consideration of commercial readers not exposing low-level functionality and not implementing the complete C1G2 specification. Because of this, even RSN protocols built on top of the C1G2 specification might not be implementable using standard readers.

Our experience with the Intel WISP suggests that basic RSN applications could be approximated using standard C1G2 readers. To read sensor data from a C1G2 WISP, the device would first be *singulated*, at which point a temporary *handle* would be requested from the tag. A reader could then use this handle to address the device and read sensor data from pre-defined memory locations. However, the handle would persist only until the reader singulates another tag or the tag loses power. Thus, reading from more than one WISP would incur substantial protocol overhead due to singulation and handle management. Consequently, simple use of the existing C1G2 protocol could provide some level of sensing functionality, but at a significant cost in terms of efficiency.

Along with reading sensor data, the C1G2 protocol could support basic sensor queries using the *Select* command. If the reader knows that a sensor value is written to a particular memory location, it could issue a *Select* command with a mask which matches that location for sensor values over a given threshold. Consequently, only WISPs with sensor values over that threshold would reply during the next frame. More generally, the *Select* command could be used as a general purpose broadcast channel. The bit mask in the command could be repurposed and interpreted, in the most general case, as opcodes and data. As multiple *Selects* could be sent before each frame, complex tasking and querying could be achieved in this manner.

The above mechanisms show that there is potential for using the C1G2 standard to implement RSN protocols. This would have the advantage of being implementable using current reader technology, given a reader that is sufficiently programmable. However, these mechanisms may prove too inefficient or may simply be poorly matched to many applications. Further experimentation is needed.

## 5 Conclusion

By exploiting RFID technology, we believe that we can expand the application space of wireless sensor networks to ubiquitous, embedded sensing tasks. We have sketched sample sensor network applications in the space between traditional mote networks and RFID for supply-chain monitoring. We have described key systems and networking challenges related to intermittent power and RSN protocols for sensor queries. We expect RSNs to be a fruitful new space for networking and systems research, as there is significant work that must be done to translate the capabilities of the WISP into full-fledged RSNs.

## References

[1] M. Buettner and D. Wetherall. An empirical study of uhf rfid performance. In *Proc. Mobicom (to appear)*, 2008.

[2] H. J. Chae, D. J. Yeager, J. R. Smith, and K. Fu. Maximalist cryptography and computation on the wisp uhf rfid tag. In *Proc. Conference on RFID Security*, 2007.

[3] N. Cho, S. Song, S. Kim, S. Kim, and H. Yoo. A 5.1-uw uhf rfid tag chip integrated with sensors for wireless environmental monitoring. In *IEEE Biological Circuits and Systems (BioCAS) 2008, submitted*, 2008.

[4] EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 mhz-960 mhz version 1.0.9. 2005.

[5] C. Hartung, R. Han, C. Seielstad, and S. Holbrook. Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proc. MobiSys*, 2006.

[6] J. Holleman, D. Yeager, R. Prasad, J. Smith, and B. Otis. Neural wisp: An energy harvesting wireless brain interface with 2m range and 3uvrms input referred noise. In *Proceedings of the 31th European Solid-State Circuits Conference, Grenoble*, 2005.

[7] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for ‘smart dust. *Journal of Communication Networks*, pages 188–196, 2000.

[8] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, and D. Haehnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004.

[9] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys*, 2004.

[10] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proc. IPSN/SPOTS*, 2005.

[11] L. G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, 1975.

[12] A. P. Sample, D. J. Yeager, P. S. Powledge, and J. R. Smith. Design of an rfid-based battery-free programmable sensing platform. In *IEEE Transactions on Instrumentation and Measurement (accepted)*, 2008.

[13] J. R. Smith, A. P. Sample, P. Powledge, A. Mamishev, and S. Roy. A wirelessly powered platform for sensing and computation. In *Proc. Ubicomp*, 2006.

[14] D. Taylor, S. H. Tillery, and A. Schwartz. Direct cortical control of 3d neuroprosthetic devices. *Science*, 296:1829–1832, 2002.

[15] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. SenSys*, 2003.

[16] D. Yeager, R. Prasad, D. Wetherall, P. Powledge, and J. Smith. Wirelessly-charged uhf tags for sensor data collection. In *Proc. IEEE RFID*, 2008.

[17] D. J. Yeager, A. P. Sample, and J. R. Smith. Wisp: A passively powered uhf rfid tag with sensing and computation. In M. I. Syed A. Ahson, editor, *RFID Handbook: Applications, Technology, Security, and Privacy*. CRC Press, 2008.

# Eat All You Can in an All-You-Can-Eat Buffet: A Case for Aggressive Resource usage

Ratul Mahajan Jitendra Padhye Ramya Raghavendra Brian Zill
*Microsoft Research*

***Abstract —*** In contrast to a focus on efficiency, we advocate aggressive usage of available resources. This view is embodied in what we call the Buffet principle: continue using more resources as long as the marginal cost can be driven lower than the marginal benefit. We illustrate through several examples how this seemingly obvious principle is not adhered to by many common designs and how its application produces better designs. We also discuss broadly the considerations in applying the Buffet principle in practice.

## 1. INTRODUCTION

Alice walks into a restaurant with an all-you-can-eat buffet. She wants to eat enough to avoid hunger until the next meal. Should she eat based on the expected time until her next meal, or should she eat as much as she can?

The second strategy is clearly superior. It provides the best possible protection against hunger, limited only by the capacity of Alice's stomach. With the first strategy, misestimation of the time of the next meal or of the activity level lead to hunger. And note that both strategies cost the same.

Surprisingly, system design often follows the first strategy today. For instance, consider the task of adding forward error correction (FEC) to transmissions over a wireless channel. In current designs, the number of added FEC bits tends to be a function of the anticipated bit error rate [2, 4, 23, 8], independent of the available spectrum resources. This method protects against packet loss as long as the errors are fewer than anticipated but fails with higher or bursty errors. This failure is unfortunate if there are available resources that would otherwise go to waste.

Underlying the use of the first strategy today is a desire for efficient use of available resources. In the FEC example, adding the number of bits that is a function of the common-case error rate is an efficient way to use the spectrum. More bits might be considered wasteful usage. Yet if that spectrum would otherwise go unused, the real waste is in not taking advantage of it to improve performance. As demonstrated by the examples above, a singular focus on efficiency can lead to poor performance.

Based on these observations, we put forth the *Buffet* principle: continue using more resources as long as the marginal cost can be driven lower than the marginal benefit. Stated differently, efficiency of resource usage should not be a driving concern if more resources can be used at a lower cost than the benefit from the additional use.

Through several case studies, we show that applying the Buffet principle produces designs that are qualitatively different and arguably perform better. Our cases span a range of systems and resource types, including erasure coding over lossy channels, replication for reliability, managing control traffic, and speculative execution. The diversity of these examples points to the broad applicability of the principle.

The key challenge in applying the Buffet principle is that the default way to greedily use resources tends to be costly. For example, in the FEC scenario, if the network is CSMA-based and a transmitter greedily pads its data transmissions, other transmitters will suffer and total network goodput will drop. Unless this challenge can be overcome, efficiency-oriented designs are likely prudent.

Our case studies suggest that this challenge can be met in many settings. In the FEC scenario, for instance, it can be met by having transmitters send additional FEC bits in separate, short transmissions that occur with a lower priority, so that they are less likely to hinder other data transmissions. In addition to prioritization, we identify opportunistic usage when the resource is vacant, utility-driven usage, and background usage as useful methods in building Buffet-based systems.

We also discuss broadly the other challenges in applying the principle, its limitations, and scenarios where it can be naturally applied. These scenarios are where the opportunity cost of greedily using resources can be effectively controlled; where the resource in question goes to waste if not used; and where greedy usage by one user does not hurt others. The potential limitations of Buffet-based designs are that performance can become a function of the amount of spare resources and greedy usage of one resource can increase the latency of certain tasks and bottleneck other resources.

We do not claim that the Buffet principle has never been used before. For example, one recent work appears to use it [6], and there are undoubtedly others as well. In contrast to these works, the contribution of this paper lies in an explicit and general specification of the principle and in provoking a broader discussion of its value. In this respect, we are inspired by the end-to-end argument [16], which articulates a broadly useful principle across the design of many systems.

We also do not claim that the principle can be universally applied, only that it offers a useful perspective on system design. The most attractive aspect is that the per-

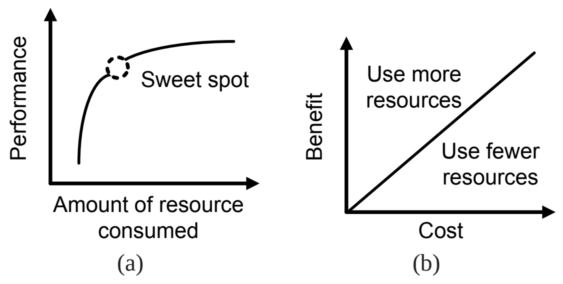


**Figure 1: (a): The thinking underlying many efficiency-centric designs. (b): A simplistic illustration of the Buffet principle.**

formance of Buffet designs would be limited primarily by the amount of available resources, rather than likely artificial limitations driven by efficiency concerns. However, its full potential can only be understood in the context of concrete, practical designs. We are currently building two different systems based on the Buffet principle.

## 2. THE (SOMETIMES MISPLACED) FOCUS ON EFFICIENCY

In this section, we describe how the focus on efficiency manifests in system design today and when it may be unwarranted. In many systems, the amount of resources used depends on design choices, rather than it being a simple function of workload. Examples include systems that: $i)$ add FEC to data transmitted over a communication channel, where the amount of resources consumed depends not only on the payload but also on the extent of error correction added; $ii)$ replicate data over multiple storage devices, where the amount of resources consumed depends on the degree of replication; $iii)$ prefetch libraries into memory before user programs ask for it (to speed execution), where the amount of resources used depends on the aggressiveness of prefetching.

To those familiar with the design of such systems, Figure 1(a) may appear familiar. The $x$-axis represents the amount of resources consumed and the $y$-axis represents performance. In the FEC case, these can be the number of added bits and the fraction of packets correctly received.

System designers often use such a graph as a guide. They try to find the “sweet spot” such that: $i)$ before it, consuming more resources brings great additional benefit; and $ii)$ beyond it, there are diminishing returns. The sweet spot is an attractive operating point when efficiency, which may be characterized as performance per unit of resource consumed, is a central goal.

However, an exclusive focus on efficiency can be misplaced. We outline specific examples in §4, but the general characteristics of such situations are the following.

• Extra resources can be used such that the marginal cost is low and the resource itself is of “use it or lose it” variety, that is, not using it leads to unnecessary wastage. Such resources include disk space, channel capacity, etc. While the returns from using resources beyond the sweet spot are low, they nevertheless represent additional benefit, which should be had when the cost is low.

• The amount of resource usage needed to hit the sweet spot is hard to determine accurately because the system is dynamic. This occurs, for instance, when the failures or packet losses are bursty; here, even if the view of average failure or loss rate is accurate, burstiness implies that at any given instance the system may be operating far from the sweet spot. The system would perform better and the design may be simpler as well if the focus was on using as much resource as possible, rather than trying to operate at constantly a moving target.

We argue that instead of focusing exclusively on efficiency, the designers must take a holistic look at the resources at their disposal and use them aggressively. Towards this end, we propose the Buffet principle.

## 3. THE BUFFET PRINCIPLE

The Buffet principle is easily stated: *continue using more resources as long as the marginal cost can be driven lower than the marginal benefit.* Figure 1(b) illustrates it somewhat simplistically, without capturing the dynamics of marginal cost and benefit and thus the fact that the designs may get less efficient as more resources are used.

The simplicity of the Buffet principle is deceptive, to the extent that it might seem obvious and in wide usage. But system design today is often not approached from the perspective advocated by it. This point will be clarified below and in the case studies outlined in the next section.

For a quick illustration, however, consider TCP, the dominant transport protocol for reliable communication. At first glance, it may appear that TCP uses the Buffet principle because it tries to estimate and consume all available bandwidth. However, TCP consumes all available bandwidth only if there is sufficient amount of new data, for instance, during a large file transfer. It will not use the spare bandwidth to proactively protect existing data from loss.

For example, consider the case where TCP’s congestion window is 8 packets and it receives only 4 packets from the application. TCP will send only 4 packets even though the path can support more, assuming that congestion window reflects available bandwidth. It will send more only after a packet is determined to be lost, which takes at least a round trip time.

A Buffet-based transport protocol might preemptively send each packet twice, thus using the spare bandwidth to provide faster loss recovery. Of course, whether such a protocol is practical depends on whether other data can be protected from the aggressive bandwidth usage by duplicate packets.

As suggested by the example above, the key to successfully applying the Buffet principle is that the aggressive resource usage advocated by it must be enabled in a way that does not hurt overall performance. Otherwise, the marginal cost would be high and an efficiency-focused design would in fact be prudent. The default way to aggressively use resources often has a high cost; for instance, the duplicate packets above may lead to higher overall loss rate. This reason is perhaps why many system designs tend to focus on efficiency, almost by default; it is not the case that designers are leaving obvious gains on the table.

Approaching the design from the perspective of the Buffet principle challenges designers to devise methods to lower the impact of aggressive resource usage. The examples below highlight that this is likely achievable in many cases. The resulting designs can be qualitatively different, sometimes simpler, and perform better.

Applying the Buffet principle also requires us to quantify or at least compare the cost and benefit of using more resources. This exercise is system-specific and must account for all relevant economic and performance-related factors. We discuss this challenge in §5.1.

## 4. CASE STUDIES

We now describe several settings that can benefit from Buffet-based designs. We classify them based on the nature of the resource of interest. Our designs are not complete but are meant to highlight the diversity of settings where the principle can be applied. The next section has a more general discussion of considerations surrounding the application of the principle.

### 4.1 Wireless spectrum or bandwidth

#### *4.1.1 Forward error correction (FEC)*

Wireless media tends to be error-prone and the bits inferred by the receiver may be corrupted in transmission. Adding FEC bits can help recover from some of the bit errors and improve performance by reducing packet loss. The trade-off here is that each additional bit can lower packet loss but also steal transmission capacity.

FEC designs that we are aware of either add a fixed number of bits to each transmission or a number that adapts based on estimated bit error rate (BER) [2, 4, 23, 8]. Current designs use efficiency arguments similar to those in §2 and add bits corresponding to the sweet spot where additional bits present a diminishing reduction in loss rate. However, by not explicitly considering available resources, they either unnecessarily lose packets even when there are spare resources or create unnecessarily high FEC overhead under heavy load. Either way, throughput suffers.

A Buffet-based FEC design can enable the maximal protection against bit errors that the amount of available spectrum resources can provide. Such a design will add some minimum number of FEC bits to all transmissions, perhaps based on the expected common case BER. On top of that, it will greedily add more FEC bits as long as there are spare resources.

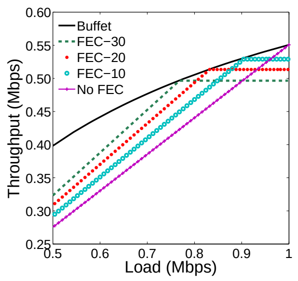


**Figure 2: Throughput with different FEC mechanisms as a function of offered load.**

We illustrate the benefits of such a design using a simple simulation experiment in which 8000-bit packets are sent over a channel with 1 Mbps capacity and a BER of $10^{-6}$. We assume an optimal FEC code: when $k$ bits of FEC are added, the packet is successfully delivered if any 8000 out of 8000+$k$ bits are received without error. Figure 2 shows the throughput in this setting without FEC, with different levels of added FEC, and with a Buffet design where the minimum number of FEC bits is zero. FEC-$K$ refers to adding FEC bits equivalent to K% of the packet size – current FEC designs would sit on one such curves. We see that the Buffet-based FEC performs the best across the board. For any given load level, the Buffet-based design matches the best other design. Individual other designs suffer significantly either under low load or under high load.

The example above also suggests how Buffet designs can be simpler. Current FEC designs need to carefully decide how many bits to add based on estimated BER or packet losses [2, 4, 23]. This task is complicated by the bursty and dynamic nature of the error process, and misestimations hurt throughput. Buffet designs skirt this complexity altogether. By simply adding as many bits as the currently available resources allow, they can get the best performance at all load and BER levels.

A challenge, however, in the design of a Buffet-based FEC system is to ensure that greedy addition of FEC bits does not lead to fewer data bits being transmitted (e.g., due to carrier sensing). This property is easy to achieve in systems where transmitters have a short- or long-term dedicated share of the medium, as may be the case for satellite links or long-distance point-to-point links [8, 14].

It can also be achieved in CSMA-based systems. Instead of embedding all FEC bits in the data packet itself, we can embed the minimum number of required bits in the packet. The additional bits are transmitted separately with lower priority, which makes it more likely for data transmissions of other senders to acquire the medium. Such priority mechanisms can be implemented today using recent WiFi hardware that supports quality of service (QoS) enhancements (802.11e) [1]. We can further reduce the impact of greedy FEC bits by making FEC-only packets small, so that even when they do acquire the medium, they delay data transmissions by only a short amount of time.

We are currently designing such an FEC system. Our focus is on VoIP and multimedia streaming. For these applications, the aggressive addition of FEC bits would lead to more timely data delivery, compared to retransmissions based on exponential backoffs.

#### *4.1.2 Erasure coding for lossy paths*

Rationale similar to the one above also applies to protection against packet losses. For this setting as well, current designs can lead to avoidable data loss. As an example, consider a recent system, called Maelstrom [5], that uses erasure coding to combat losses in dirty fiber. It adds a fixed amount of redundancy to the data stream, based on the observation that loss rates in fiber are generally low and adding more redundancy would use more resources in the common case. With Maelstrom, data would be lost whenever loss rate is higher than the level of protection. A Buffet-based system can provide greater protection from losses by utilizing all remaining path capacity for erasure coded packets.

The key challenge here is to send coded packets such that they do not steal bandwidth from normal data traffic. This is easily accomplished in a system like Maelstrom if sits on the two ends of the fiber. It can also be accomplished by marking redundant information as lower priority, so that routers drop them first during periods of congestion. A way to accomplish it without router support is to send erasure coded data opportunistically, only when the queues are estimated to be empty.

We are building a system that uses the third method above. It targets paths provided by cellular providers from moving vehicles; such paths tend to be lossy with unpredictable loss rates [15]. Their roughly stable capacity lets us estimate when the queues are empty and erasure coded packets can be sent. This system is meant for users that subscribe to an unlimited data plan, and thus the marginal cost of sending erasure coded data is only performance-related, not economic. Our early experiments show a negligible drop in throughput due to aggressive coding, even under high offered load. They also show an appreciable reduction in packet losses.

#### *4.1.3 Mobility updates*

The performance of systems that exhibit a high-degree of mobility, such as a mobile ad hoc network (MANET), depends on the frequency of mobility updates. A higher frequency yields better performance as nodes will have a more up-to-date view of the topology, but it can also swamp data traffic. Existing systems get around this trade-off by setting the frequency of updates to a tolerable level that is based on an analysis similar to the sweet spot reasoning presented in the previous section [10, 3]. Such systems may perform poorly in situations with higher than anticipated mobility levels even when there is spare capacity to support a high update frequency.

A Buffet-based mobility update mechanism will provide better performance whenever spare capacity is available. The practical difficulty here again is ensuring that the additional updates do not hurt data traffic. This can be accomplished using a priority mechanism similar to the one suggested above for FEC transmissions.

#### *4.1.4 Routing in delay tolerant networks (DTNs)*

As further evidence of the value of the Buffet principle, we note that system design in the domain of DTN routing has evolved from not using the principle to using it. Many DTN routing protocols replicate messages along multiple paths to improve their delivery probability. Older protocols limit the amount of replication to prevent a few messages from dominating network resources [17, 12, 18]. Because this limit is not guided by the amount of available storage or bandwidth between replication end points, these designs can perform poorly even when plenty of resources are available. A recent protocol, called RAPID [6], implicitly uses the Buffet principle. It replicates as much as available resources allow. To prevent network domination by a few messages, it takes a utility-driven approach in which messages are replicated based on their expected utility. Messages that have been replicated more have lower utility. The authors demonstrate that RAPID significantly outperforms older designs.

### 4.2 Storage

#### *4.2.1 Long-term storage*

Replication protects data against node failures and latent sector errors in disks. The amount of replication, however, is often pre-determined today, based on anticipated failure rate. This unnecessarily limits the protection level even when there may be spare resources. A replication system based on the Buffet principle will provide maximal protection given available resources.

Consider two scenarios. The first is replication across one or more disks on a single computer. Today's mechanisms such as various RAID configurations are based on a preset amount of replication that provides protection against a certain number of failures. This can lead to data loss when more failures occur even though ample working storage may still be available. A Buffet-based design will replicate aggressively to fill all available storage, thus providing maximal possible protection. The key challenge is to not hurt read and write performance in the process, which we believe can be accomplished by relegating the task of additional replication to the background and conducting it only when the disk is idle.

The second scenario is replication across computers in a data center or in a wide-area peer-to-peer system. Here too, the system will be more reliable with replication that uses all available resources rather than a fixed replication level. The key challenge is to manage the bandwidth impact of aggressive replication, which is a particularly relevant concern for the wide-area setting. We believe that this concern can be handled through background transfer protocols such as TCP Nice [21].

#### 4.2.2 *Short-term storage*

Program execution can be slowed by the time it takes to load the program and the libraries it uses into memory. One can speed this up by preemptively loading in memory commonly used binaries when there is space (and time) available. Indeed, this strategy has already been proposed or implemented for modern operating systems [9, 19]. A Buffet-based strategy will maximize performance by aggressively filling available memory, instead of being limited to the most promising candidates.

Similar ideas have been explored in the context of prefetching web pages that users are likely to view in the future [11, 22, 13]. If the bandwidth impact of such prefetching can be controlled, for instance, using TCP Nice, such systems should aggressively fill available cache capacity to maximize user-perceived performance.

### 4.3 Computational resources

Speculative execution is a commonly used technique in modern processors. In it, parts of code are executed even though the results may eventually be discarded, depending on the outcome of the (if) conditions that occur prior to these parts. The execution of the program is non-sequential to parallelize processing. When the results prove useful, speculative execution boosts performance. The performance benefit of speculative execution depends on the accuracy of branch prediction. Conventionally, only one branch is speculatively executed even though additional resources may be available for executing more branches. More recent designs attempt to execute multiple paths [20]. For maximal performance, a Buffet design would speculatively follow as many paths as current resources levels allow. As the number of cores inside processors increase, such a design would increasingly outperform strategies that limit speculative execution to more likely paths.

## 5. APPLICABILITY CONSIDERATIONS

In this section, we discuss broadly the issues related to applying the Buffet principle in practice. These are based on our early experiences and will be refined over time.

### 5.1 Challenges in applying the principle

There are two key challenges. The first challenge of course is ensuring that greedy resource usage does not detract from other productive work. The last section mentions several techniques to address this challenge in the context of specific examples. We summarize them here. One technique is prioritization, so that greedy tasks get lower priority. Prioritization can be explicit, e.g., embedding priority in packet headers for routers. It can also be implicitly implemented by sources, by them deferring to other tasks, e.g., background transfers of TCP Nice [21] and the use of higher inter-frame spacings in 802.11e [1]. Prioritization may not suffice in settings where aggressive usage of multiple nodes need to be traded-off with one another based on their relative benefit. Utility-driven resource consumption, which is a generalization of prioritization, can help here. In it, tasks are executed in order of their estimated utility, as in RAPID [6]. Yet another technique is opportunistic usage, as in our erasure coding system (§4.1.2) in which greedy usage occurs only when the resource is idle. We believe that one of these techniques or a combination can be applied in many situations.

The second challenge is quantifying or at least being able to compare the marginal benefit and cost of using more resources. For cost, the primary difficulty is taking into account the opportunity cost of greedily using resources, that is, for what else could those resources be used. This is not a concern where the greedily allocated resource can be easily reclaimed when needed or would otherwise remain unused. But it could be problematic otherwise. Additionally, if precise accounting is desired, we need to quantify the cost of the side-effects produced by greedy usage as well (§5.3).

We can avoid the task of quantifying marginal cost by driving it to zero or negligible levels. The techniques above for managing greedy usage help here. If done successfully, we can continue to use more more resources until the marginal benefit becomes negative.

Quantifying marginal benefit can also be tricky, e.g., in the face of correlated failures [7]. But because the marginal benefit of using more resources is usually positive, more resources can be used whenever the marginal cost is negligible.

### 5.2 Applicable resources

Two categories of resources are well-suited for applying the Buffet principle. The first is non-conservable resources, i.e., those that would go to waste if not used. Storage, bandwidth, and computational resources are typically non-conservable. An example of a conservable resource is battery power.

We do not claim that the Buffet principle does not apply to conservable resources, only that it is easily applied to non-conservable resources. Applying it to conservable resources requires a more involved evaluation of marginal benefit that includes predictions of future behavior.

The second category is where the resource is not shared with non-Buffet users who may not be able to differentiate normal usage from greedy usage with lower value. Such users might reduce their own consumption on observing aggressive usage, which would reduce overall system throughput. In some cases, Buffet users can co-exist with non-Buffet users. For instance, our wireless FEC design coexists by implementing greedy usage at lower priority and deferring to non-Buffet users.

### 5.3 Side effects of greedily using resources

We have encountered three side-effects. First, the system performance becomes a function of the workload itself. For example, our FEC design loses fewer packets at lower loads and more at higher loads; in current designs, the loss rate for transmitted packets is independent of load. One might argue that such load-dependent performance abstraction is hard for applications. But observe that performance is already often load-dependent. For instance, wireless loss rate increases with load because the collision frequency increases. Even along wired paths, loss rate observed by applications can depend on load. Sending at 0.9 Mbps along a 1 Mbps capacity path leads to no loss but sending at 1.1 Mbps leads to 10% loss.

The second side-effect is that greedy usage can strain the system. It can increase task-completion latency. For instance, a read request for a disk block will have to wait longer if it arrives during greedy replication. The level of latency increase depends on how fast the greedy task can be completed or preempted. It can be controlled by keeping individual greedy tasks short or preemptable. Another strain is that aspects of the system that were originally not the bottleneck can become bottlenecks with greedy usage. For instance, disk I/O bandwidth may become a bottleneck with aggressive replication, even if it was not previously. Careful design is needed to alleviate this problem.

A final side-effect is that with greedy usage, the resources will frequently appear fully utilized. This behavior will typically not matter but it may in some cases, for instance, if administrators use utilization levels to make provisioning decisions. This issue can be dealt with by separately counting normal and greedy usage.

### 5.4 Benefit of the principle in practice

It depends on the workload and the amount of available resources. So it might vary from none to a lot. For example, in our erasure coding system, a Buffet-based design leads to zero loss under low load and a a loss rate that is equal to the underlying path loss rate under heavy load. The appropriate view of a Buffet design is that its performance is limited by the amount of spare resources instead of specific design parameters, and thus it provides the best performance for a given level of resource investment.

## 6. CONCLUSIONS

We articulated the Buffet principle, which advocates a different perspective on system design than a singular focus on efficiency. Through several examples, we explain how Buffet designs differ from efficiency-centric designs and how they are likely to perform much better. We also discussed broadly the considerations surrounding the application of the principle in practice. This discussion points to both strengths as well as limitations. Overall, we find the principle promising and offering a useful perspective on system design. Instead of being limited by artificial design choices, Buffet designs have the potential to provide the best performance for the level of available resources. Its eventual worth can be understood only by studying the performance of many concrete designs, which is an active area of research for us.

**Acknowledgments** We thank John Douceur for feedback on this paper.

## 7. REFERENCES

[1] Medium access control (MAC) quality of service enhancements. IEEE Standard, 2005.
[2] J.-S. Ahn, S.-W. Hong, and J. Heidemann. An adaptive FEC code control algorithm for mobile wireless sensor networks. *Journal of Communications and Networks*, 7(4), 2005.
[3] S. Ahn and A. Shankar. Adapting to route-demand and mobility in ad hoc network routing. *Computer Networks*, 38(6), 2002.
[4] A.Levisianou, C.Assimakopoulos, N-F.Pavlidou, and A.Polydoros. A recursive IR protocol for multi-carrier communications. In *Int. OFDM Workshop*, Sept. 2001.
[5] M. Balakrishnan, T. Marian, K. Birman, H. Weatherspoon, and E. Vollset. Maelstrom: Transparent error correction for lambda networks. In *NSDI*, Apr. 2008.
[6] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *SIGCOMM*, Aug. 2007.
[7] J. R. Douceur and R. P. Wattenhofer. Large-scale simulation of replica placement algorithms for a serverless distributed file system. In *MASCOTS*, Aug. 2001.
[8] M. Emmelmann and H. Bischl. An adaptive MAC layer protocol for ATM-based LEO satellite networks. In *VTC*, Oct. 2003.
[9] B. Esfabd. *Preload: An adaptive prefetching daemon*. PhD thesis, University of Toronto, 2006.
[10] R. K. Guha, Y. Ling, and W. Chen. A light-weight location-aware position update scheme for high mobility networks. In *MILCOM*, Oct. 2007.
[11] Z. Jiang and L. Kleinrock. An adaptive network prefetch scheme. *IEEE JSAC*, 16(3), 1998.
[12] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *SAPIR*, Aug. 2004.
[13] V. Padmanabhan and J. Mogul. Using predictive prefetching to improve World-Wide Web latency. In *SIGCOMM*, Aug. 1996.
[14] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: Design and implementation of high-performance WiFi-based long distance networks. In *NSDI*, Apr. 2007.
[15] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MAR: A commuter router infrastructure for the mobile Internet. In *MobiSys*, June 2004.
[16] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM ToCS*, 2(4), 1984.
[17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN*, Aug. 2005.
[18] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Performance analysis of mobility-assisted routing. In *MobiHoc*, May 2006.
[19] Windows Vista SuperFetch. http://www.microsoft.com/windows/products/windowsvista/features/details/superfetch.mspx.
[20] A. K. Uht. *Speculative Execution in High Performance Computer Architectures*, chapter Multipath Execution. CRC Press, 2005.
[21] A. Venkataramani, R. Kokku, and M. Dahlin. TCP-Nice: A mechanism for background transfers. In *OSDI*, Dec. 2002.
[22] Q. Yang and H. H. Zhang. Integrated web prefetching and caching using prediction models. *WWW*, 4(4), 2001.
[23] L. Zhao, J. W. Mark, and Y. Yoon. A combined link adaptation and incremental redundancy protocol forenhanced data transmission. In *GLOBECOM*, Nov. 2001.

# Plugging Into Energy Market Diversity

Asfandyar Qureshi (MIT CSAIL)

## Abstract

*In North America, electricity prices exhibit both temporal and geographic variation—the later exists due to regional demand differences, transmission inefficiencies and generation diversity. Using historical market data, we characterize the variation and argue that existing distributed systems should be able to exploit it for significant economic gains. We consider pricing in cloud computing systems, and also use simulation to estimate the advantage of dynamically shuffling computation between different energy markets.*

## 1 Introduction

Electricity is becoming increasingly expensive, and now accounts for a large fraction of the cost of ownership for data centers [1]. It is expected that by 2012, in the US, 3-year energy costs for data centers will be at least twice as much as the server investment [2].

At the same time, deregulation, regional demand variations and energy source diversity have resulted in an uneven and occasionally volatile cost landscape. In the US, electricity prices at two different places can have very different annual averages (figure 1), and prices at a location can vary day to day by a factor of five (figure 4).

Like cost, the *utility* gained by a distributed system's clients may also depend on location. Generally, a client receives less utility if their request is served far away from them. Many existing systems typically maintain multiple replicas, routing clients to nearest replicas, attempting to maximize client utility, while ignoring the geographic variation of cost.

In such replicated systems, it is possible to trade-off between computing in a high cost market versus computing in a lower cost market but with reduced client utility. Shifting clients away from their best replicas, to ones situated in cheaper energy markets, may reduce quality-of-service but yields significant monetary savings.

To some extent, this trade-off is implicit in the placement of large data centers in low-cost energy markets (Google in Oregon and Microsoft in Illinois) rather than in high-demand locations (e.g. New York City). We argue that, due to existing price volatility, this trade-off should be a dynamic choice rather than a static one.

This paper investigates the implications of electricity price volatility and locational variation to Internet scale systems. We argue that there is something to be gained, by building price-sensitive distributed systems, that automatically integrate up-to-date market information, and make cost/performance trade-offs.

We sketch the connection between computing cost and energy cost and establish the significance of locational variation. Using historical electricity market data, we show that the day-to-day, monthly, and yearly variation is substantial. We note that daily prices, at locations near Internet peering points, exhibit exploitable volatility.

We briefly cover how cloud computing providers could increase their margins by being sensitive to geographic variation in energy prices—either with price differentiation or by using cost-optimized routing.

Finally, we use simulation and a 2006-2008 history of US market prices to explore cost/performance trade-offs within Internet-scale replicated systems. We simulate *selective blackouts*, where one or more replicas are deactivated in response to market signals. We quantify possible energy cost savings and discuss practical implications.

To the best of our knowledge, this paper contains the first proposal for distributed systems to use online optimization to algorithmically exploit information from electricity futures and/or spot markets.

## 2 Background

### 2.1 Concerns about Electricity Cost

Data center energy costs are becoming an increasingly dominant component of overall operating costs. The cost of electricity is poised to overtake the cost of equipment [3]. In the US: in 2000 three-year energy costs were one-tenth the server equipment expenditures; by 2009 the cost of electricity is expected to equal server expenditure; and by 2012, energy is expected to cost at least twice the equipment investment [2]. These expectations take into account recent advances in data center energy efficiency. For a denser non-traditional data center (e.g., Sun's S20 [4]), 2-year energy costs could already exceed the equipment cost, depending on configuration and location.

Additionally, in absolute terms, servers consume a substantial amount of electricity. Servers and their support infrastructure (e.g., cooling) accounted for about 1.2% of US electricity consumption in 2005, about 45 million MWh, or 2.7 billion dollars [5]. By 2010, this is projected to grow to 3% of total US consumption [5].

Consequently, for companies with large computing facilities, even a fractional reduction in electricity costs can translate into a large overall savings. For example, it was estimated that Google owned 450,000 servers worldwide in 2006 [6] and that each server consumed upwards of 200 watts [7]. Each watt used by a computer results in at least two watts drawn from the electric grid [1, 3]. We can, conservatively, estimate that Google servers used around 1.6 million MWh in a year, or 95 million dollars

| Location | 2004 | 2005 | 2006 | 2007 |
|---|---|---|---|---|
| New York (NYC) | 63.1 | 93.5 | 70.9 | 77.1 |
| New England (MA) | 53.7 | 78.6 | 60.9 | 67.9 |
| Southwest (Palo Verde) | 50.1 | 67.4 | 57.6 | 61.7 |
| Southeast (SERC/FRCC) | 48.6 | 70.8 | 55.5 | 59.1 |
| PJM Interconnect (West) | 41.7 | 60.6 | 50.1 | 56.9 |
| Northwest (MID-C) | 44.5 | 63.0 | 50.2 | 56.6 |
| California (NP-15) | 38.4 | 54.4 | 43.4 | 54.6 |
| Texas (ERCOT-North) | 42.3 | 66.5 | 51.4 | 52.0 |
| Midwest (Cinergy) | | 38.4 | 40.5 | 46.1 |

**Figure 1**: Annual average prices [9], in \$/MWh, sorted by 2007 prices.

worth of electricity, at US rates[1]. Therefore, every 1% savings in energy cost could save a large company like Google, a million dollars a year. Google is not alone. Microsoft expects to deploy 800,000 servers by 2011 [6], and the five leading search companies may have already deployed more than 2 million servers [8].

New cooling technologies, architectural redesigns, DC power, multi-core servers, virtualization and energy aware load balancing algorithms, have all been proposed as ways to reduce the energy consumed by a single data center. That work is complementary to ours. However, this paper is concerned with reducing *cost*—our approach can achieve this, even if it causes consumption to rise.

## 2.2 Electricity Markets

Although market details differ regionally, this section provides a high-level view of deregulated electricity markets, providing a context for the rest of the paper. The discussion is based on markets in North America, but the ideas generalize to other regions with diversified markets.

Electricity is produced by government utilities and independent power producers using a variety of sources. In the United States, this includes nuclear (about 10%), coal (around 30%), natural gas (nearly 40%) and hydroelectric (roughly 8%) [10].

Producers and consumers are connected to an electric *grid* of transmission lines. Electricity cannot be stored easily, so supply and demand must continuously be balanced. In addition to connecting nearby nodes, the grid can be used to import and export electricity from/to distant locations. The United States is divided into ten markets [9], with varying degrees of inter-connectivity. Congestion on the grid, transmission line losses, and market seams issues either limit how electricity can flow, or influence the price at a given location [11].

The existence of rapid price fluctuations reflects the fact that short term demand for electricity is far more elastic than short term supply. Electricity cannot always be efficiently moved from low demand areas to high demand areas, and power plants cannot always ramp up easily. In contrast, we have long used high performance networks and load balancing techniques to relocate computation. We can move our demand closer to a low-cost supply.

[1] $450,000 \times 200W \times 2 \times 24h \times 365 = 1.5678 \times 10^{12} Wh$ @ $6¢/kWh$

| Location | Nearby City | Hub | Market |
|---|---|---|---|
| A | San Jose, CA | NP15 | California |
| B | San Diego, CA | SP15 | California |
| C | Portland, OR | MID-C | Northwest |
| D | Chicago, IL | Illinois | Midwest |
| E | Ashburn, VA | PJM-West | PJM |
| F | Houston, TX | ERCOT-H | ERCOT |
| G | Miami, FL | Florida | Florida |

**Figure 2**: Seven locations, near different Internet exchange points, and in different electricity markets.

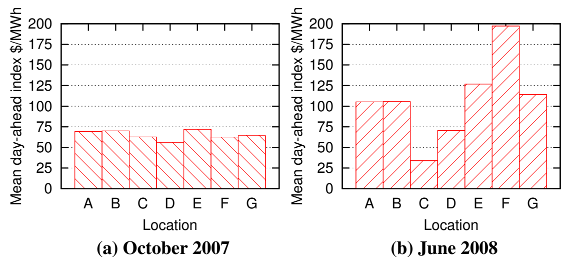


**Figure 3**: Monthly variation in wholesale market prices can be substantial and geographically dissimilar. For example, comparing (b) with (a): prices at F trebled, while those at C halved.

While short-term and long-term contracts may account for most of what is consumed, electricity can also be bought in *wholesale* markets. In most regions, day-ahead, hour-ahead, and spot markets exist. In this paper we focus on day-ahead markets. Such markets allow consumers to determine the price of electricity the day before it is delivered. Day-ahead prices are forward signals, that can be used to decide how much to consume.

A caveat: companies running data centers may have contracts with electricity providers, do not buy directly from the wholesale market, and so may be buffered from the price volatility we are looking to exploit. Contractual details are hard to come by; this paper ignores contracts.

In reality, there is a great deal more complexity, but our market model is simple: a futures market exists; day-ahead prices are accessible and variable; and different locations see prices that are not perfectly correlated.

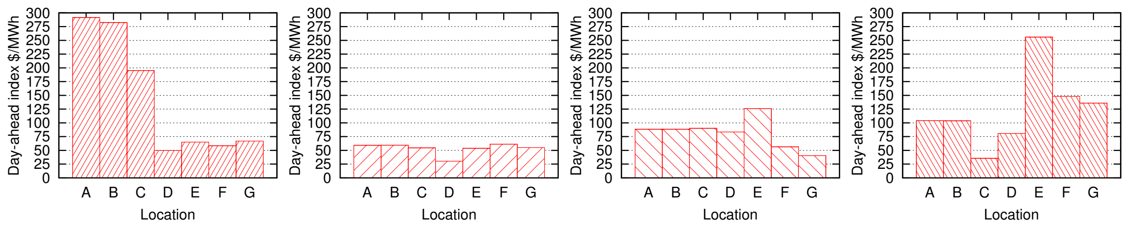
## 2.3 Computation Cost

A service provider accepts requests, performs some computations, and produces responses. The provider incurs some cost in fulfilling this demand.

We model the total computation cost ($C$) incurred by a service provider at a given location as follows: a large fixed component, the infrastructure cost ($I$), and a significant variable component ($V$), which is a monotonically increasing function of demand.

In this formulation, $I$ includes the amortized infrastructure investment, staff salaries, etc. $V$ includes both network and energy costs, but we ignore network costs. Studies have shown that electricity consumption closely follows CPU utilization [12]. Using techniques like multicore CPUs and virtualization, resources can be allocated on-demand, causing electricity use to step up.

The *marginal* computation cost is the incremental cost



**(a) July 05, 2007** **(b) Sept. 14, 2007** **(c) Jan. 22, 2008** **(d) June 10, 2008**

**Figure 4**: Day-ahead wholesale market prices exhibit significant volatility. For example, prices at E were $54/MWh in (b) but $256/MWh in (d).

of handling one more request, or the derivative of $C$ w.r.t. demand. The *average* cost is the mean cost per request, or $C$ divided by demand.

Differences in electricity prices will always show up in the marginal computation cost for different locations, assuming constant server energy efficiency. If electricity costs are a large enough fraction of overall cost, price volatility will begin to palpably affect average cost.

## 3 Electricity Price Variation

This paper posits that electricity prices vary dynamically, that prices at different locations are not perfectly correlated, and that differences can be exploited for economic gain. Rather than presenting a theoretical discussion, we take the experimental approach, grounding ourselves in historical market data, from multiple sources [9, 10, 13].

We begin with average annual electricity prices, tabulated in figure 1 for several locations. In 2007, Northeast prices were over 1.5 times Midwest prices, contributing to the impracticality of large data centers in the Northeast.

The remainder of the paper focuses on the smaller set of seven locations from figure 2, all of which are near major Internet exchange points (IXPs) and cover a number of diverse electricity markets.

Apart from annual variation, prices also exhibit seasonal and monthly variation. Figure 3 shows average prices for two different months. In the South, in June '08 the energy needed to handle a million requests would have cost twice as much in Houston (location F) compared to Miami (G). In October '07, the cost difference would have been relatively insignificant. Similarly, on the West coast, in June, electricity in California (A) was thrice as expensive as electricity in Oregon (C), but in October prices were roughly the same. Furthermore, the relative ordering of prices was very different in the two months. Houston (F), for example, moved from the second cheapest market to the most expensive.

Part of the market diversity arises because different regions produce electricity in different ways. For example, in 2006: in Oregon, natural gas accounted for 8% and hydroelectric for 68% of the summer generation capacity; whereas in Texas, natural gas accounted for 71% and coal for 20% of the summer capacity [10]. Consequently, record high natural gas prices in 2008 have had much larger impact on Texas than on Oregon.

Prices in wholesale markets also exhibit significant day-to-day volatility, for a variety of reasons. For example, a localized event such as a heat wave in California could drive up local demand, elevating West-coast prices. Figure 4 shows day-ahead prices for four different days. Price spikes such as those shown in figure 4a and figure 4d occasionally occur. Price volatility has many hard-to-predict causes (e.g., accidents, equipment malfunctions, weather, fuel costs, demand volatility, market manipulation, etc.). Figure 5 shows a more detailed picture for some locations, plotting the evolution of day-ahead market prices from January 2006 through June 2008. Some notable features: seasonal effects, short-term spikes, and only partially correlated behaviour. A detailed discussion is beyond the scope of this paper.

In this paper we restrict ourselves to day-ahead market prices. However, significantly more price volatility exists in hour-ahead and spot markets [11]. Traditional consumers cannot respond quickly enough, but distributed systems can re-route computation at millisecond scale, to modulate their consumption. Beyond our findings in this paper, there may be opportunities within spot and hour-ahead markets, that traditional electricity consumers cannot exploit, but distributed systems can exploit.

## 4 Pricing in Clouds

With the rise of web-based computing and the computing-as-a-utility model, many companies are renting out their infrastructure to third-party applications. Examples include Amazon's EC2, Google's AppEngine and Sun's Grid. Applications are billed by the resources they consume: computation cycles, network I/O and storage.

How much does it cost a provider to perform one unit of work on behalf of a hosted application? How much does it cost Amazon to handle a single client request on behalf of a hosted web application?

Cost depends on where the request is routed. We have already established that marginal computation costs can differ radically with location and in time. Furthermore, refer back to the cost model from section 2.3. Large cloud providers (Amazon and Google) will already need to absorb their fixed costs. They need to build multiple data centers, and keep machines up and running, to support their own primary services. The cost to them of performing some incremental work on behalf of a hosted applica-

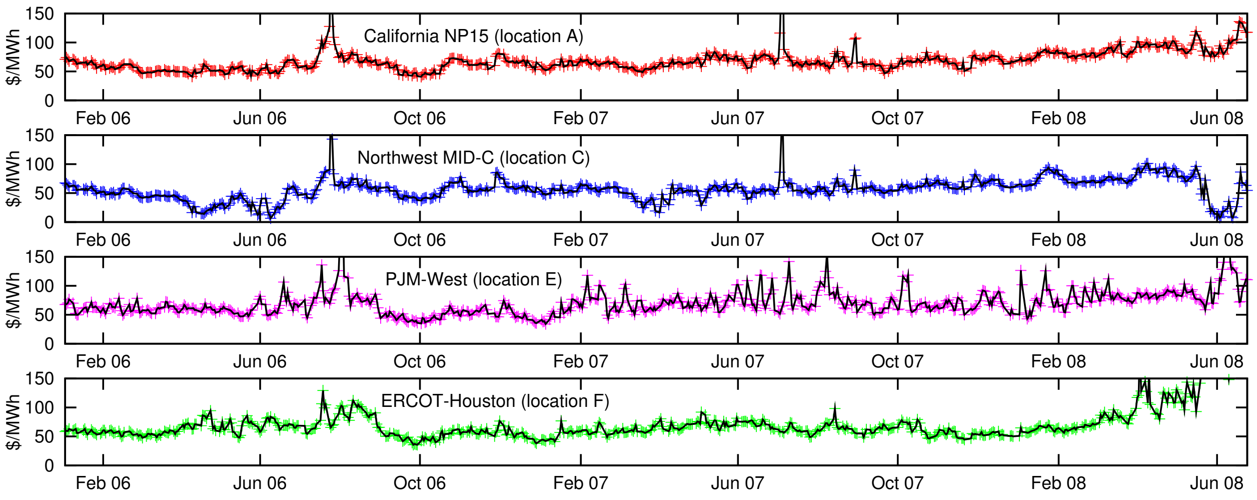


**Figure 5**: Day ahead index prices at different hubs, from January 2006 through June 2008 [13]. Note the seasonal dips in the hydro dominated Northwest, and the 2008 upward trend in California and Texas, both of which are heavily dependent on natural gas. Price spikes reached $350.

tion will be dominated by the marginal cost, mainly the cost of the additional watts they expend.

By charging a fixed compute-price, while being able to decide where to buy electricity, cloud providers are missing an opportunity. With a price structure that embraces energy cost diversity, and by using a cost-conscious replication strategy, cloud providers can increase their margins or lower their prices.

Buyers care about how much they are charged and what performance their users receive. Providers can build energy cost differences into some pricing plans, allowing buyers to make trade-offs. For example, free applications should always be hosted in the lowest cost locations, capacity permitting. Additionally, some buyers may be willing to pay premiums for regionally optimized performance. The *Dallas Morning News* website, having regionally concentrated demand, values proximity, and can therefore be billed to compensate for elevated prices.

These ideas can be mapped to content distribution networks. For instance, a CDN provider could charge a premium for hosting content in high energy cost markets.

## 5 Selective Blackouts

Internet-scale systems composed of replicas in different electricity markets can exploit price disparities to substantially reduce their total energy costs, by using Information from energy futures markets, and dynamically shifting consumption away from high-cost regions. Through simulation, we show that an approach based on this idea could yield considerable monetary savings.

### 5.1 System Model

In the systems we focus on, storage and computing infrastructure can be decomposed into a number of blocks, where each block is a complete replica of the system[2].

[2]Less flexible but acceptable: strict subsets are complete replicas.

The blocks may be:

- For large companies, the blocks are large data centers, owned and operated by the company. Each block can have many thousands of physical machines, and easily consume 4500 kW [1].
- The blocks can be much smaller data centers. In the extreme, blocks may be one or more of Sun's data-center-in-a-container [4], each with fewer than 300 machines and 500 kW of peak consumption.
- For small providers, the different blocks can be leased floor-space in data centers owned by other parties[3]. The main difference between this case and the above cases is control over infrastructure: in the earlier cases if the provider decided to turn off the machines, they can also shut off cooling etc.

An incoming client request to such a system can be served by any of the replicas. In existing systems, replicas tend to be placed near IXPs, such as the locations in figure 2. Conventionally such systems attempt to keep all replica locations active. In order to maximize performance, client requests are routed to their closest replicas.

In the discussion that follows, we assume that the system is over-provisioned: some subset of the replicas has enough capacity to handle the peak load.

We also make a number of simplifying assumptions. We model demand as constant and uniformly distributed. When some blocks are deactivated, we assume client requests will be spread evenly over the remaining blocks and that total energy use is therefore constant. Furthermore, we assume that a deactivated block consumes zero energy, and that the startup/shutdown process also consumes no energy[4]. Finally, we assume that shutting down one replica does not affect prices at any other replicas.

[3] Our work is only relevant when electricity charges are metered.

[4] This ignores the cost of synchronization during replica reactivation.

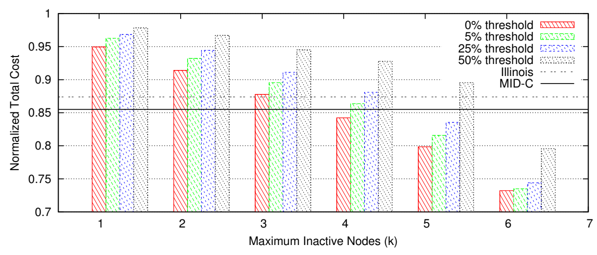


**Figure 6**: Total electricity costs for seven replica simulations, using 2006-2008 market data. A cost of 1.0 represents running all seven.

We use the number of active replicas as a first-order approximation for performance. We defer a proper analysis of the performance impact of our proposal.

## 5.2 Selective Blackouts

With enough excess capacity, one or more replicas can be turned off. This will result in suboptimal system performance and reduce reliability, but can also significantly reduce energy costs.

Deciding which replicas should be active on any given day can be modeled as an optimization problem. Each day, day-ahead market prices can be fed into an automated mechanism that determines which replicas should be deactivated the next day. The set of active replicas changes infrequently, at most once per day, making this compatible with existing routing techniques (e.g., DNS).

Given $n$ replicas, we constrain that no more than $k$ replicas can be deactivated on any given day. Thus the $(n-k)$ lowest cost replicas are always active, regardless of absolute prices. This provides a consistent performance baseline. Replicas remain active as long as their prices are close to the highest price we must pay for baseline performance. We only force deactivation when a significant price disparity exists.

More formally, given day-ahead prices, we derive the set of active replicas $A$ as follows:

$$\begin{aligned} L &= \{(n-k) \text{ lowest cost replicas}\} \\ \phi &= max(\{price_r \text{ for } r \in L\}) \\ A &= \{\text{replica } r \text{ iff } price_r \leq (1+\tau)\cdot\phi\} \end{aligned}$$

$\tau$ is a threshold parameter, expressing our sensitivity to price disparity, as a percentage of the baseline price $\phi$.

## 5.3 Simulation Results

We simulated the above selective blackout mechanism using historical prices, wholesale market data from 2006 through 2008 [13], and found that significant cost savings were possible. Demand was modeled as being constant in time and uniformly distributed in space.

**North America Seven.** We first simulated a seven-node system, one node at each location from figure 2. Figures 6 and 7 summarize the results.

Simulations imply that adding a single redundant node can reduce total electricity costs by 5% (see figure

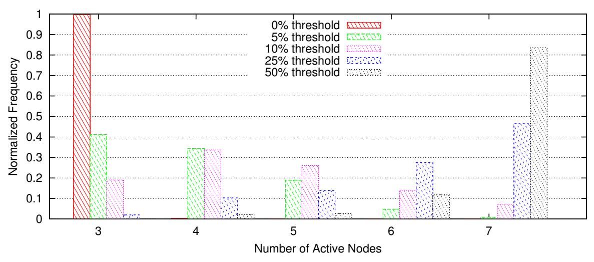


**Figure 7**: The distribution of the number of active replicas, from simulations using 2006-2008 market data, with $n$ = 7 and $k$ = 4.

6, 0% threshold and $k$ = 1). These savings are the result of being able to dynamically deactivate nodes during periods of locally elevated prices. Statically picking the best six locations is not enough. Section 3 already illustrated that an always optimal set of six may not exist, and our simulations reinforced this: all nodes were active some of the time. The most active 6-subset accounted for 27% and the next most active 23%, so no subset dominated.

For $k \geq 4$, blackouts can result in total energy costs lower than the average cost of the least expensive market (MID-C line in figure 6). With only one baseline node and six redundant nodes ($\tau$ = 0, $k$ = 6), energy cost is 85% that of the cheapest node. This is a savings of 27%, compared to running all seven.

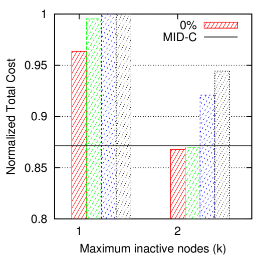
The threshold parameter $\tau$ can be used to trade off between cost and performance. Figure 7 shows how the number of active replicas depends on $\tau$ for $k$ = 4. With a threshold of 5%, the median number of active replicas is 4 ($\mu$ = 3.9) and the total cost roughly matches the cheapest market (see figure 6). With a threshold of 25%, the median number of active replicas is 6 ($\mu$ = 6.1) and the cost is close to the second-cheapest market. At the same time, in contrast with building a large data center in a cheap market, computation resources are now more likely to be near an IXP that provides a fast path to a random client. This can dramatically improve performance.

**West Coast Three.** With all seven nodes, we can take advantage of regional diversity (e.g., a heat-wave in California does not put pressure on the Illinois hub). Even though, nearby locations in the same market tend to have correlated prices, selective blackouts can still be useful.

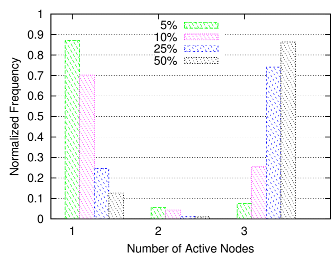
To demonstrate this, we simulated a three node west-coast system (NP15, SP15 and MID-C). With blackouts (50% threshold, $k$ = 2) the resulting total cost is 6% lower than the cost of continuously running all three, and 8% higher than the cost of computing everything in Oregon. The median number of active replicas is 3 ($\mu$ = 2.7). For this to work, Oregon must retain maximum capacity—on some days it is the only active replica. See figure 8.

## 5.4 Some Lessons

Building extra, occasionally deactivated, replicas will incur some additional infrastructure cost. However, we have shown extra replicas can reduce total energy costs.



(a) Electricity Costs



(b) Active replica distribution, k=2

**Figure 8**: West coast simulation results.

If expected energy savings exceed the up-front infrastructure investment and added maintenance costs, it makes economic sense to use some variant of this blackout mechanism. If data center energy costs in the US double in the next four years [2], or if the replicas are modular data centers [4] with low fixed costs, dynamic blackouts could make a tangible impact on operating costs.

Additionally, apart from yielding savings, blackouts can reduce risk, by dampening the impact of unanticipated price fluctuations. The mechanism described here, for example, would automatically integrate market information and route around multi-day weather problems.

The choice of where to build a data center is typically seen as a *static* optimization problem. If energy costs continue to rise relative to equipment, it may be better modeled as a *dynamic* problem. Despite the economies of scale inherent to large data centers and the possibility of local tax incentives, a company looking to build a monolith should consider building many smaller blocks (e.g., [4]) spread over different energy markets. Redundant capacity is already built into these systems. It may be better to spread these resources, rather than concentrating them.

## 6 Conclusion

We set out to show that the diversity and day-to-day volatility of today's electricity markets can be exploited in some meaningful way by existing distributed systems. Using data from wholesale electricity markets and simulation, we were able to show that replicated systems can make meaningful cost/performance trade-offs and may be able to achieve substantial energy cost reductions. Many possibilities for future work exist within this area.

In order to understand the trade-offs, a good performance model is necessary. We use the number of active replicas as a coarse performance metric. A better approach would have been to analyze the network latencies between clients and active replicas, assuming a uniform client distribution, using census data, or using server logs. The impact on reliability should also be considered.

Another convenient simplification was to assume constant demand. In reality, demand varies regionally and temporally [14, 15]. Depending on the situation, there may be ways to exploit features within demand signals.

We presented selective blackouts as an illustration of a price-conscious optimization mechanism, rather than as a proposed design. A mature mechanism should synthesize information from both supply (cost) and demand (performance/utility) and derive the best way to use available resources. In addition, hour-ahead and spot-prices are more volatile than day-ahead prices, so more frequent optimization should yield higher savings.

Further, our idea of relating energy costs to computation costs implies that auctions within computing grids can be used to match buyers and sellers, to increase the total economic surplus in the computing utility market.

Finally, contracts complicate the picture, making it unclear who would reap the savings we calculated. Power providers may be willing to index charges to market prices, since this transfers some risk to consumers. If, on the other hand, contracts fix the cost of electricity, a deactivated data center would allow the producer to sell the surplus electricity on the wholesale market. While this would not impact the data center's bottom line, the provider would benefit, and—if resource scarcity has caused the price elevation—the public would benefit.

## Acknowledgements

This paper was improved by helpful comments from John Guttag, Hari Balakrishnan, Srikanth Kandula, Vladimir Bychkovsky, and the anonymous reviewers. This work was supported in part by Nokia.

## References

[1] J. G. Koomey, K. G. Brill, P. Turner, J. Stanley, and B. Taylor, "A Simple Model for Determining True Total Cost of Ownership for Data Centers," white paper, Uptime Institute, 2007.

[2] K. G. Brill, "The Invisible Crisis in the Data Center: The Economic Meltdown of Moore's Law," white paper, Uptime Institute, 2007.

[3] L. Marchant, "Data Center Growing Pains," in *USENIX Large Installation System Admin. Conference*, 2007. Presentation.

[4] Sun Microsystems, Modular Datacenter `http://www.sun.com/products/sunmd`.

[5] J. G. Koomey, "Estimating Total Power Consumption by Servers in the U.S. and the World," tech. rep., LBNL, 2007.

[6] J. Markoff and S. Hansell, "Hiding in Plain Sight, Google Seeks an Expansion of Power," *the New York Times*, June 2006.

[7] L. A. Barroso, "The price of performance," *ACM Queue*, 2005.

[8] G. Gilder, "The Information Factories," *Wired mag.*, Oct. 2006.

[9] United States Federal Energy Regulatory Commission, Market Oversight. `http://www.ferc.gov`.

[10] United States Department of Energy, Official Statistics. `http://www.eia.doe.gov/fuelelectric.html`.

[11] P. L. Joskow, "Markets for Power in the United States: an Interim Assessment," tech. rep., MIT CEEPR, Aug. 2005.

[12] X. Fan, W.-D. Weber, and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," in *ACM International Symposium on Computer Architecture*, 2007.

[13] Platts, "Day-Ahead Market Prices," in *Megawatt Daily*, McGraw-Hill. 2006-2008.

[14] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-Scale Video-on-Demand Systems," in *European Conference in Computer Systems*, 2006.

[15] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *IEEE Network*, vol. 14, no. 3, 2000.

# On Delivering Embarrassingly Distributed Cloud Services

Ken Church
Microsoft
One Microsoft Way
Redmond, WA, USA
church@microsoft.com

Albert Greenberg
Microsoft
One Microsoft Way
Redmond, WA, USA
albert@microsoft.com

James Hamilton
Microsoft
One Microsoft Way
Redmond, WA, USA
jamesrh@microsoft.com

## ABSTRACT

Very large data centers are very expensive (servers, power/cooling, networking, physical plant.) Newer, geo-diverse, distributed or containerized designs offer a more economical alternative. We argue that a significant portion of cloud services are embarrassingly distributed – meaning there are high performance realizations that do not require massive internal communication among large server pools. We argue further that these embarrassingly distributed applications are a good match for realization in small distributed data center designs. We consider email delivery as an illustrative example. Geo-diversity in the design not only improves costs, scale and reliability, but also realizes advantages stemming from edge processing; in applications such as spam filtering, unwanted traffic can be blocked near the source to reduce transport costs.

## Categories and Subject Descriptors

K.6.4 [System Management]: Centralization/decentralization.

## General Terms

Management

## Keywords

Embarrassingly Distributed, Economies of Scale, Spam, POPs (Points of Presence).

## 1. Introduction

Large data centers are being built today with order 10,000 servers [1], to support "cloud services" – where computational resources are consolidated in the data centers. Very large (mega) data centers are emerging with order 150,000 multi-core servers, realized, for example, as 150 containers with 1000 servers per container.[1] In total, cloud service providers are on a path to supporting up to a million servers (some providers are rumored to have already crossed this point), in tens to hundreds of locations.

Imagine a family of solutions with more or less distribution, ranging from a single POP (point of presence) to a million. This paper will explore trade-offs associated with size and geo-diversity. The trade-offs vary by application. For *embarrassingly distributed* applications, i.e. applications with relatively little need for massive server to server communications, there are substantial opportunities for geo-diversification to improve cost, scale, reliability, and performance. Many applications fall somewhere in the middle with ideal performance at more than one POP, but less than a million. We will refer to mega-datacenters as the mega model, and alternatives as the micro model.

[1]http://perspectives.mvdirona.com/2008/04/02/FirstContainerizedDataCenterAnnouncement.aspx

**Table 1: Options for distributing a million cores across more or less locations (POPs = Points of Presence).**

| POPs | Cores/POP | Hardware/POP | Co-located With/Near |
|---|---|---|---|
| 1 | 1,000,000 | 1000 containers | Mega-Data Center |
| 10 | 100,000 | 100 containers | |
| 100 | 10,000 | 10 containers | Fiber Hotel |
| 1,000 | 1,000 | 1 container | |
| 10,000 | 100 | 1 rack | Central Office |
| 100,000 | 10 | 1 mini-tower | P2P |
| 1,000,000 | 1 | embedded | |

Large cloud service providers (Amazon, Microsoft, Yahoo, Google, etc.) enjoy economies of scale. For example, large providers enjoy a wide set of buy/build options for the wide area network to support internal and external data transport to their data centers, and can create and manage dedicated networks, or buy network connectivity arguably at costs near those incurred by large network service providers. In the regional or metro area (e.g., pipes from data centers to the wide area network) and in peering (e.g., to large broadband service providers), these large cloud service providers have less choice and may incur higher costs. Nevertheless, by buying numerous and/or large pipes and delivering large volumes of traffic, the cloud service providers can obtain significant discounts for data transport. Savings in computational and networking resources can in turn be passed on to creators of cloud service applications, owned and operated by the cloud service provider or by third parties.

One might conclude that economies of scale favor mega-data centers, but it is not that simple. By analogy, large firms such as Walmart, can expect favorable terms primarily because they are large. Walmart can expect the same favorable term no matter how they configure their POPs (stores). Economies of scale depend on total sales, not sales per POP. In general, economies of scale depend on the size of the market, not mega vs. micro.

Large data centers are analogous to large conferences. A small (low budget) workshop can be held in a spare room in many universities, but costs escalate rapidly for larger meetings that require hotels and convention centers. There are thousands of places where the current infrastructure can accommodate a workshop or two, but there is no place where the current infrastructure could handle the Olympics without a significant capital investment. Meetings encounter diseconomies of scale when they outgrow the capabilities of off-the-shelf venues.

So too, costs escalate for large mega-data centers. For example, if a mega-data center consumes 20MW of power at peak from a given power grid, that grid may be unable or unwilling to sell another 20MW to the same data center operator. In general, the infrastructure for a new mega data center (building, power, and networking) calls for building/lighting up significant new components. Yet, if the data centers are smaller (under the micro moel), there is increased opportunity to exploit the overbuild in what is already there in the current power grid and networking fabric. There are thousands of places where the current infrastructure could handle the load for a container sized data center, but there is no place where the current infrastructure can handle a thousand containers without a significant capital investment. Data centers encounter various diseconomies of scale when they become so large that they require significant investment in infrastructure.

It is risky and expensive to put all our eggs in one basket. Paraphrasing Mark Twain (*The Tragedy of Pudd'nhead Wilson*), if all our eggs are in one basket, then we must watch that basket carfully. In the mega-data center this means a very high degree of redundancy at many levels – for example in power delivery and provisioning [1]. For example, as we cannot afford to lose the entire site owing to power failure or network access failure, the mega data center may incur large costs in batteries, generators, diesel fuel, and in protected networking designs (e.g., over provisioned multiple 10 GE uplinks and/or SONET ring connectivity to the WAN).

Many embarrassingly distributed applications could be designed at the application layer to survive an outage in a single location. Geo-diversity can be cheaper and more reliable than batteries and generators. The more geo-diversity the better (at least up to a point); *N*+1 redundancy becomes more attractive for large *N*. Geo-diversity not only protects against short term risks (such as blackouts), but also longer term risks such as a supplier cornering a local market in some critical resource (network, power, etc.). Unfortunately, in practice, inefficiencies of monopoly pricing can dominate other considerations. With small containerized data centers, it is more feasible to adapt and provision around such problems (or leverage the capability to do so in negotiations), if the need should arise.

On the other hand, there are limits to geo-diversification. In particular, it is much easier to manage a small set of reliable sites. There is little point to provisioning equipment in so many places that supply chain management and auditing become overwhelming problems. It may be hard to run a distributed system without on site workforce with timely physical access to the machine rooms. (Yet, new containerized designs have promise to dramatically mitigate the need for timely physical access[1].)

Though there has been some degree of reporting[1-14] on the nature of large and small data centers, much remains proprietary, and there has been little discussion or questioning of basic assumptions and design choices. In this paper, we take up this inquiry. In Section 2, to understand the magnitude of the costs entailed in mega-data center physical infrastructure, we compare their purpose built design with a gedanken alternative where the servers are distributed among order 1000 condominiums. The results suggest smaller footprint data centers are well worth pursuing. In Section 3, we consider networking issues and designs for mega and micro data centers, where the micro data centers are order 1K to 10K servers. In Section 4, we ask whether there are large cloud service applications that are well suited to micro data center footprints, specifically examining solutions that can be realized in an "embarrassingly distributed" fashion, and look at email in some depth. In Section 5, we contrast mega and micro data centers taking a more tempered view than in Section 2. We conclude in Section 6.

## 2. Power and Diseconomies of Scale

How do machine room costs scale with size? In a recent blog,[2] we compared infrastructure costs for a large data center with a farm of 1125 condominiums and found the condos to be cheaper. Condos might be pushing the limits of credulity a bit but whenever we see a crazy idea even within a factor of two of current practice, something is interesting, warranting further investigation.

A new 13.5 mega-watt data center costs over $200M before the upwards of 50,000 servers that fill the data center are purchased. Even if the servers are built out of commodity parts, the data centers themselves are not. The community is considering therefore moving to modular data centers. Indeed, Microsoft is deploying a modular design in Chicago[3]. Modular designs take some of the power and mechanical system design from an upfront investment with 15 year life to a design that comes with each module and is on a three year or less amortization cycle and this helps increase the speed of innovation.

Modular data centers help but they still require central power, mechanical systems, and networking systems. These systems remain expensive, non-commodity components. How can we move the entire datacenter to commodity components? Consider a radical alternative: rather than design and develop massive data centers with 15 year lifetimes, let's incrementally purchase condos (just-in-time) and place a small number of systems in each. Radical to be sure, but condos are a commodity and, if this mechanism really was cheaper, it would be a wake-up call to reexamine current industry-wide costs and what's driving them.

See Table 2 for the back of the envelope comparison showing that the condos are cheaper in both capital and expense. Both configurations are designed for 54K servers and 13.5MWs. The data center costs over $200M, considerably more than 1125 condos at $100K each. As for expense, the data center can expect favorable terms for power (66% discount over standard power rates). Deals this good are getting harder to negotiate but they still do exist. The condos don't get the discount, and so they pay more for power: $10.6M/year >> $3.5M/year. Even with the deal, the data center is behind because it isn't worth $100M in capital to save $7M/year in expense. But to avoid comparing capital with expense, we simplified the discussion by renting the condos for $8.1M/year, more than the power discount. Thus, condos are not only cheaper in terms of capital, but also in terms of expense.

In addition to saving capital and expense, condos offer the option to buy/sell just-in-time. The power bill depends more on average usage than worst-case peak forecast. These options are valuable under a number of not-implausible scenarios:

[2]http://perspectives.mvdirona.com/2008/04/06/DiseconomiesOfScale.aspx

**Table 2: Condos are cheaper than data center in both capital and expense.**

| | | Large Tier II+ Data Center | Condo Farm (1125 Condos) |
|---|---|---|---|
| **Specs** | Servers | 54k | 54k (= 48 servers/condo × 1125 Condos) |
| | Power (Peak) | 13.5 MW (= 250 Watts/server × 54k servers) | 13.5MW (= 250 Watts/server × 54k servers = 12 KW/condo × 1125 Condos) |
| **Capital** | Building | over $200M | $112.5M (= $100k/condo × 1125 Condos) |
| **Annual Expense** | Power | $3.5M/year (= $0.03 per kw/h ×24×365 hours/year ×13.5MW) | $10.6M/year (= $0.09 per kw/h×24×365 hours/year × 13.5MW) |
| **Annual Income** | Rental Income | None | $8.1M/year (= $1000/condo/month × 12 months/year × 1125 Condos − $200/condo/month condo fees. We conservatively assume 80% occupancy) |

1. Long-Term demand is far from flat and certain; demand will probably increase, but anything could happen over 15 years.
2. Short-Term demand is far from flat and certain; power usage depends on many factors including time of day, day of week, seasonality, economic booms and busts. In all data centers we've looked into, average power consumption is well below worst-case peak forecast.

How could condos compete or even approach the cost of a purpose built facility built where land is cheap and power is cheaper? One factor is that condos are built in large numbers and are effectively "commodity parts." Another factor is that most data centers are over-engineered. They include redundancy such as uninterruptable power supplies that the condo solution doesn't include. The condo solution gets it's redundancy via many micro-data centers and being able to endure failures across the fabric. When some of the non-redundantly powered micro-centers are down, the others carry the load. N+1 redundancy is particularly attractive for embarrassingly distributed apps (Section 4).

It is interesting to compare wholesale power with retail power. When we buy power in bulk for a data center, it is delivered by the utility in high voltage form. These high voltage sources (usually in the 10 to 20 thousand volt range) need to be stepped down to lower working voltages which brings efficiency losses, distributed throughout the data center which again brings energy losses, and eventually delivered to the critical load at the working voltage (240VAC is common in North America with some devices using 120VAC). The power distribution system represents approximately 40% of total cost of the data center. Included in that number are the backup generators, step-down transformers, power distribution units, and Uninterruptable Power Supplies (UPS's). Ignore the UPS and generators since we're comparing non-redundant power, and two interesting factors jump out:

1. Cost of the power distribution system ignoring power redundancy is 10 to 20% of the cost of the data center.
2. Power losses through distribution run 10 to 12% of the power brought into the center.

It is somewhat ironic in that a single family dwelling gets two-phase 120VAC (240VAC between the phases or 120VAC between either phase and ground) delivered directly to the home. All the power losses experienced through step down transformers (usually in the 92 to 96% efficiency range) and all the power lost through distribution (dependent on the size and length of the conductor) is paid for by the power company. But when we buy power in quantity, the power company delivers high voltage lines to the property and we need to pay for expensive step down transformers as well as power distribution losses. Ironically, if we buy less power, then the infrastructure comes for free, but if we buy more then we pay more.

The explanation for these discrepancies may come down to market segmentation. Just as businesses pay more for telephone service and travel, they also pay more for power. An alternative explanation involves a scarce resource, capital budgets for new projects. Small requests for additional loads from the grid can often be granted without tapping into the scarce resource. Large requests would be easier to grant if they could be unbundled into smaller requests, and so the loads could be distributed to wherever there happens to be spare capacity. Unbundling requires flexibility in many places including the applications layer (embarrassingly distributed apps), as well as networking.

## 3. Networking

In addition to power, networking issues also need to be considered when choosing between mega data centers (DCs) and an alternative which we have been calling the micro model:

- **Mega model**: large DCs (e.g., 100,000 – 1,000,000 servers).
- **Micro model**: small DCs (e.g., 1000 – 10,000 servers).

The mega model is typical of the networks of some of today's large cloud service providers, and is assumed to be engineered to have the potential to support a plethora of services and business models (internal as well as hosted computations and services, cross service communication, remote storage, search, instant messaging, etc.) These applications need not be geo-diverse, and in practice many of today's applications still are not. Thus, the reliability of the application depends on the reliability of the mega-data center. In the micro model, we consider applications engineered for N+1 redundancy at micro data center level, which then (if large server pools are required) must be geo-diverse. In both models, we must support *on-net* traffic between data centers, and *off-net* traffic to the Internet. We focus the discussion here on off-net traffic; considerations of on-net traffic lead to similar conclusions. While geo-diversity can be difficult to achieve – especially for legacy applications – geo-diversity has advantages, and the trend is increasingly for geo-diverse services. For embarrassingly distributed services (Section 4), geo-diversity is relatively straightforward.

Under the mega model, a natural and economical design has the cloud service provider creating or leasing facilities for a dedicated global backbone or Wide Area Network (WAN). Off-net flows traverse: (1) mega data center to WAN via metro (or regional) links, (2) WAN to peering sites near the end points of flows, (3) ISPs and enterprises peered to the WAN. The rationale is as follows. Large cloud service providers enjoy a wide set of buy/build options across networking layers 1, 2, 3 in creating wide area and metro networks.[3] Via a global WAN, the cloud service provider can "cold potato" route to a very large set of peers, and thereby reap several benefits: (1) settlement free peering with a very large number of tier 2 ISPs (ISPs who must typically buy transit from larger ISPs), (2) lower cost settlement with tier 1 ISPs as high traffic volumes are delivered near the destination, and (3) (importantly) a high degree of unilateral control of performance and reliability of transport. In the metro segment, there will be typically some form of overbuild (SONET ring or, more likely, multiple diverse 10GE links from the data center) of capacity to protect against site loss. A strong SLA can be supported for different services, as the cloud service provider has control end to end, supporting, for example, performance assurances for database sync, and for virtualized network and computational services sold to customers who write third party applications against the platform.

In the micro model, a vastly simpler and less expensive design is natural and economical, and is typical of many content distribution networks today. First, as the computational resources are smaller with the micro data center, the networking resources are accordingly smaller and simpler, with commodity realizations possible. To provide a few 10 GE uplinks for the support of 1K to 10K servers commodity switches and routers can be used, with costs now in the \$10K range[18]. In the mega data center, these network elements are needed, as well as much larger routers in the tree of traffic aggregation, with costs closer to \$1M. In the micro model, the cloud service provider buys links from micro data center to network services providers, and the Internet is used for transit. Off-net traffic traverses metro links from data center to the network service providers, which deliver the traffic to the end users on the Internet, typically across multiple autonomous systems. As we assume N+1 redundancy at micro data center level, there is little or no need for network access redundancy, which (coupled with volume discounts that come from buying many tail circuits, and with the huge array of options for site selection for micro data centers) in practice should easily compensate for the increase in fiber miles needed to reach a larger number of data centers. In buying transit from network providers, all the costs of the mega model (metro, wide area, peering) are bundled into the access link costs. Though wide area networking margins are considered thin and are becoming thinner, the cost of creating dedicated capacity (mega model) rather than using already created shared capacity is still higher. That said, in the micro model, the cloud service provider has ceded control of quality to its Internet access providers, and so cannot support (or even fully monitor) SLAs on flows that cross out multiple provider networks, as the bulk of the traffic will do. However, by artfully exploiting the diversity in choice of network providers and using performance sensitive global load balancing techniques, performance may not appreciably suffer. Moreover, by exploiting geo-diversity in design, there may be attendant gains in reducing latency.

[3] While less true in the metro area, a user of large wide area networking resources can fold in metro resources into the solution.

## 4. Applications

By "embarrassingly distributed" applications, we mean applications whose implementations do not require intense communications within large server pools. Examples include applications:

- Currently deployed with a distributed implementation: voice mail, telephony (Skype), P2P file sharing (Napster), multicast, eBay, online games (Xbox Live),[4] grid computing;
- Obvious candidates for a distributed implementation: spam filtering & email (Hotmail), backup, grep (simple but common forms of searching through a large corpus)
- Less obvious candidates: map reduce computations (in the most general case), sort (in the most general case), social networking (Facebook).

For some applications, geo-diversity not only improves cost, scale, reliability, but also effectiveness. Consider spam filtering, which is analogous to call gapping in telephony[17]. Blocking unwanted/unsuccessful traffic near the source saves transport costs. When telephone switching systems are confronted with more calls than they can complete (because of a natural disaster such as an earthquake at the destination or for some other reason such as "American Idol" or a denial of service attack), call gapping blocks the traffic in central offices, points of presence for relatively small groups of customers (approximately 10,000), which are likely to be near the sources of the unsuccessful traffic. Spam filtering should be similar. Blocking spam and other unwanted traffic mechanisms near the source is technically feasible and efficient[14] and saves transport. Accordingly, many cleansing applications, such as spam assassin[15], can operate on both mail servers and on end user email applications.

Email is also analogous to voice mail. Voice mail has been deployed both in the core and at the edge. Customers can buy an answering machine from (for example) Staples and run the service in their home at the edge, or they can sign up with (for example) Verizon for voice mail and the telephone company will run the service for them in the core. Edge solutions tend to be cheaper. Phone companies charge a monthly recurring charge for the service that is comparable to the one-time charge for the hardware to run the service at home. Moving the voice mail application to the edge typically pays for itself in a couple of months. Similar comments hold for many embarrassingly distributed applications. Data center machine rooms are expensive, as seen in Section 2. Monthly rents are comparable to hardware replacement costs.

Let us now consider the email application in more depth.

### 4.1 Email on the Edge

Microsoft's Windows Live Hotmail has a large and geo-diverse user base, and provides an illustrative example. Traffic volumes are large and volatile (8x more traffic on some days than others), largely because of spam. Hotmail blocks 3.4B spam messages per day. Spam (unwanted) to ham (wanted) ratios rarely fall below 70% and can spike over 94%, especially after a virus outbreak.

[4] Online games actually use a hybrid solution. During the game, most of the computation is performed at the edge on the players' computers, but there is a physical cloud for some tasks such as match making and out-of-bandwidth signaling.

Adversaries use viruses to acquire zombies (bot farms). A few days after an outbreak, zombies are sold to spammers, and email traffic peaks soon thereafter.

Hotmail can be generally decomposed into four activities, all of which are embarrassingly distributed:

1) Incoming Email Anti-malware and Routing
2) Email Storage Management
3) Users and Administrator Service
4) Outgoing Email Service

**Incoming Email Anti-malware and Routing:** Mail is delivered to the service via SMTP. Load balancers distribute incoming connections to available servers. Edge blocks are applied to reject unwanted connections via IP black lists and anti-spam/virus mechanisms. Additional filters are applied after a connection is established to address Directory Harvest Attacks (en.wikipedia.org/wiki/E-mail_address_harvesting) and open relays (en.wikipedia.org/wiki/Open_mail_relay).

**Email Storage Management:** The store has to meet requirements on reliability, availability, throughput and latency. It is common practice to build the store on top of a file system (although propriety blob storage solutions are also popular). Header information and other metadata are maintained in a structured store for speed.

**Users and Administrator Service:** Requests come into the service from users in a variety of protocols including POP, IMAP, DAV, Deltasync, HTTP (web front ends). These requests are typically sent to pools of protocol servers. The protocol servers make authentication requests for each user to a separate authentication service: looking up the user's email address and finding the appropriate email storage server for that user, making internal transfer requests from the storage server, and returning the results in the appropriate format. Administrative requests are handled in the same way although with different permission and scope from normal users.

**Outgoing Email Service:** The outgoing email service accepts e-mail send requests from authenticated users. These messages are typically run through anti-malware facilities to avoid damaging the overall service reputation by distributing malware. And then the messages are routed as appropriate internally or externally.

### 4.2 Implementing Email near the Edge

Although Windows Live Hotmail and other email services are currently implemented as central in-the-core services with relatively few (10) data centers, more POPs could improve response time and service quality by distributing work geographically. Some mail services (such as Yahoo) migrate mailboxes as users move (or travel). Reliability can be achieved by trickling data from a primary server to a secondary server in another location, with small impact on overall cost. Order 100 POPs are sufficient to address latencies due to the speed of light, though more POPs enhance features such as blocking unwanted traffic near the source.

Microsoft Exchange Hosted Services[13] provides an example in the marketplace of hosted email anti-malware services.

## 5. Mega vs. Micro

Applications in the data center fall roughly into two classes: large analysis and service. Many large analysis applications are best run centrally in mega data centers. Mega data centers may also offer advantages in tax savings, site location and workforce centralization. Interactive applications are best run near users. Interactive and embarrassingly distributed applications can be delivered with better QoS (e.g., smaller TCP round trip times, and greater independence of physical failure modes) via micro data centers. It can also be cheaper to deliver such services via micro data centers.

With capital investment for a mega data center that run \$200M to \$500M before adding servers, the last point is important. Major components of the mega data center infrastructure are not commodity parts; e.g., 115 KVA to 13.2 KVA and 13.2 KVA to 408 VA transformers. Moreover, mega data centers are constructed with high levels of redundancy within and across layers[1]. In particular, power redundancy (UPS, resilient generator designs, seas of batteries, backup cooling facilities, and storage for 100K gallons of diesel) consumes at least 20% of the total infrastructure spend. In contrast, micro data center designs use commodity parts. With resilience in the network of micro data centers, there is little or no spend on generators, diesel, redundant cooling; the cost of many levels of redundancy disappears. As a result, the unit capital cost of resources in the mega data center exceeds that of the micro data center. To capture this in a simple model, we assume that resources have unit cost in the micro data center, but the same resources cost $U$ in the mega data center, where $U \geq 1$.

While varying by application, networking and power consumption needs scale with the workload. If we split workload from a single large center into $K$ smaller centers, then some efficiency may be lost. A compensatory measure then is to use load balancing (e.g., via DNS or HTTP level resolution and redirection). For example, an overloaded micro data center might redirect load to another micro data center (chosen in a random, or load and proximity sensitive manner). This can reclaim most of the efficiencies lost. New traffic is introduced between micro data centers can be mitigated by measures discussed earlier: edge filtering, application or network layer DDoS scrubbing (see e.g. [22]), time shifting of traffic needed to assure resilience and optimization of transport costs between fixed sites (e.g., locating near fiber hotels in metro areas). To first order, as capital costs of the data center dominate operational costs of networking and power[1], and taking into account available measures, we do not see the uplift in networking costs from internal transfers as appreciable.

To get some understanding of the worst case for networking and power capital costs, let's consider a simple model for the case of no cross data center load balancing. A parsimonious model of Internet workload[19][20], ideally suited to scenarios such as data centers that multiplex large numbers of flows, models workload as $m_t + \sqrt{a\, m_t}\, w_t$ where $m_t$ is the time varying mean traffic rate, $w_t$ is a stationary stochastic process with zero mean and unit variance (e.g., Fractional Gaussian Noise), and the single parameter $a$ captures the "peakedness" or bustiness of the load. For example, this model can capture the phenomenon seen for an email provider in Figure 1. (Peakedness is sensitive to large workload spikes that are not filtered out[19] – though well run services must ultimately manage these by graceful degradation and admission control[21], with some workload turned away (spikes crossing the capacity line in Figure 1.)) If the workload is decomposed into $K$ individual streams, with constant parameters $m_i, a_i$, and with independent realizations of a common Gaussian process, the model continues to hold with $m = \sum_1^K m_i$, and peakedness $a = 1/m \sum_1^K m_i\, a_i$, the weighted sum.

A service provider needs to design networking and power to accommodate most peaks. Assuming uniformity, independent

Gaussian behavior, and focusing on loads $m$ during busy hours, the resource required for the mega center can be estimated as $m + n\sqrt{am}$, where the new parameter $n$ captures the SLA. (Setting $n = 2$ corresponds to planning enough capacity to accommodate workload up to the 97.5th percentile.) As the unit cost of resources in the mega data centers is $U$, the total resource cost is then $U[m + n\sqrt{am}]$. Similarly, the total resource cost for the micro data center is $K$[m/K+$n\sqrt{am/K}$]. Thus, the spend to support a mega data center beyond that needed to support $K$ micro data centers without load balancing comes to $m(U-1) - n\sqrt{ma}(\sqrt{K} - U)$. For large resource demands $m$, the result hinges on the unit cost penalty $U$ for the mega data center. If $U$ is even slightly larger than 1, then for large $m$ the first term dominates and mega data center costs more. If unit costs are identical $(U = 1)$, then in the case of no load balancing, the micro data centers cost more -- though the increment grows with $\sqrt{m}$ and so is a vanishing fraction of the total cost, which grows with $m$. Specifically, the increment grows with a workload peakedness term $\sqrt{a}$, a fragmentation term $\sqrt{K} - 1$, and a term $n$ reflecting the strength of the SLA.

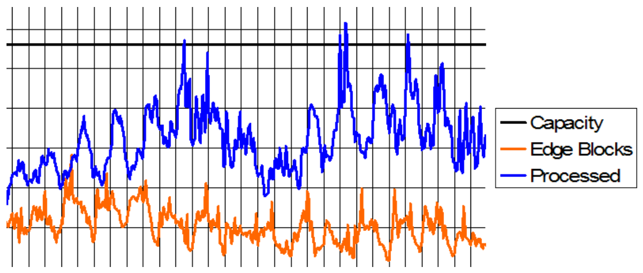


**Figure 1. Processed (upper curve) and blocked (lower curve) traffic to an email provider (under spam attack).**

## 6. Conclusions

Cloud service providers are on a path to supporting up to a million servers. Should we build a few mega datacenters under the mega model, or lots of smaller datacenters under the micro model? When applicable, the micro model is simpler and less expensive, both in terms of power (section 2) and networking (section 3); geo-diversity and N+1 redundancy eliminate complicated and expensive protection mechanisms: batteries, generators, and redundant access and transit networks. The micro model is not appropriate for all applications, but it is especially attractive for embarrassingly distributed applications, as well as applications that use small pools of servers (less than 10,000). Section 3 mentioned a number of examples, and described email in some detail. For spam filtering, geo-diversity not only simplifies the design, but the extra points of presence can block unwanted traffic near the source, a feature that would not have been possible under the mega model. Putting it all together, the micro model offers a design point with attractive performance, reliability, scale and cost. Given how much the industry is currently investing in the mega model, the industry would do well to consider the micro alternative.

## REFERENCES

[1] Hamilton, J. "An Architecture for Modular Datacenters," *3rd Biennial Conference on Innovative Data Systems Research (CIDR)*, 2007.

[2] Weiss, A., "Computing in the Clouds," *ACM Networker*, Vol. 11, Issue 4, pp 18-25, December 2007.

[3] Manos, M., Data Center World, 2008, (see Mike Rath: http://datacenterlinks.blogspot.com/2008/04/miichael-manos-keynote-at-data-center.html)

[4] Arnold, S., "The Google Legacy: How Google's Internet Search is Transforming Application Software," *Infonetics*, 2005.

[5] *Caterpillar Rental Power Spec Sheets*, http://www.cat.com/cda/layout?m=43125&x=7, 2006.

[6] Cringley, R, *Google-Mart: Sam Walton Taught Google More About How to Dominate the Internet than Microsoft Ever Did*, http://www.pbs.org/cringely/pulpit/2005/pulpit_20051117_000873.html, Nov. 2005.

[7] Cummins. *Cummins Rental Power Generation*, http://www.cumminspower.com/na/services/rental/, 2006.

[8] Hoelzle, U., *The Google Linux Cluster*, http://www.uwtv.org/programs/displayevent.asp?rid=1680, Sept., 2002.

[9] IDC, *Worldwide and U.S. Software as a Service 2004-2008 Forecast and Analysis*, IDC #31016, Mar., 2004.

[10] Interport Maintenance Co, Inc., *Container Sales Pricing*, http://www.iport.com/sales_pricing.html, 2006.

[11] Kahle, B., *Large Scale Data Repository: Petabox*, http://www.archive.org/web/petabox.php, Jun., 2004.

[12] Menon, J., *IBM Storage Systems*, http://www.almaden.ibm.com/almaden/talks/cerntalksumm.pdf, IBM, 2001.

[13] Microsoft, *Microsoft Exchange Hosted Services*, http://www.microsoft.com/exchange/services/default.mspx, 2006.

[14] Microsoft, *Microsoft Compute Cluster Server 2003*, http://www.microsoft.com/windowsserver2003/ccs/default.mspx, 2006.

[15] Spam Assassin, http://spamassassin.apache.org/.

[16] Nortel Networks, *ISO 668 Series 1 Steel Freight Containers*, http://www.nortel.com, 2006.

[17] *Engineering and Operations in the Bell System*, AT&T Bell Laboratories, 1977.

[18] Greenberg, A.; Maltz, D.; Patel, P.; Sengupta, S.; Lahiri, P., "Towards a Next Generation Data Center Architecture: Scalability and Commodization," Workshop on Programmable Routers for Extensible Services of Tomorrow (*Presto*), 2008.

[19] M. Roughan and A. Greenberg and C. Kalmanek and M. Rumsewicz and J. Yates and Y. Zhang, "Experience in measuring Internet backbone traffic variability: Models, metrics, measurements and meaning," Proc. International Teletraffic Congress (*ITC-18*), 2003.

[20] I. Norros, "A storage model with self-similar input," Queueing Systems, 6:387--396, 1994.

[21] Hamilton, J., "On Designing and Deploying Internet-Scale Services," USENIX Large Installation Systems Administration Conference (*LISA*), 2007.

[22] Cisco Guard, http://www.cisco.com.

# Dr. Multicast: *Rx* for Data Center Communication Scalability[*]

Ymir Vigfusson, Cornell University
Hussam Abu-Libdeh, Cornell University
Mahesh Balakrishnan, Cornell University
Ken Birman, Cornell University
Yoav Tock, IBM Haifa Research Lab

## Abstract

Data centers avoid IP Multicast because of a series of problems with the technology. We propose *Dr. Multicast* (the MCMD), a system that maps traditional IPMC operations to either use a new point-to-point UDP multisend operation, or to a traditional IPMC address. The MCMD is designed to optimize resource allocations, while simultaneously respecting an administrator-specified acceptable-use policy. We argue that with the resulting range of options, IPMC no longer represents a threat and could therefore be used much more widely.

## 1 Introduction

As data centers scale up, IP multicast (IPMC) [8] has an obvious appeal. Publish-subscribe and data distribution layers [6, 7] generate multicast distribution patterns; IPMC permits each message to be sent using a single I/O operation, reducing latency both for senders and receivers (especially, for the last receiver in a large group). Clustered application servers [1, 4, 3] need to replicate state updates and heartbeats between server instances. Distributed caching infrastructures [2, 5] need to update cached information. For these and other uses, IPMC seems like a natural match.

Unfortunately, IPMC has earned a reputation as a poor citizen. Routers must maintain routing state and perform a costly per-group translation[11, 9]. Many NICs can only handle a few IPMC addresses; costs soar if too many are used. Multicast flow control is also a black art. When things go awry, a multicast storm can occur, disrupting the whole data center. Perhaps most importantly, management of multicast use is practically unsupported.

Our paper introduces *Dr. Multicast* (the MCMD), a technology that permits data center operators to enable IPMC while maintaining tight control on its use. Applications are coded against the standard IPMC socket interface, but IPMC system calls are intercepted and mapped into one of two cases:

- A true IPMC address is allocated to the group.
- Communication to the group is performed using point-to-point UDP messages to individual receivers, using a new *multi-send* system call.

The MCMD tracks group membership, using a gossip protocol. It translates each send operation on a multicast group into one or more send operations, optimized for system objectives. Finally, to implement this optimization policy, it instantiates in a fault-tolerant fashion a service that computes the best allocation of IPMC addresses to groups (or to overlapping sets of groups), adapting as use changes over time.

Users benefit in several ways:

- Policy: Administrators can centrally impose traffic policies within the data center, such as limiting the use of IPMC to certain machines, placing a cap on the number of IPMC groups in the system or eliminating IPMC entirely.
- Performance: The MCMD approximates the performance of IPMC, using it directly where possible. When a multicast request must be translated into UDP sends, the multi-send system call reduces overheads.
- Transparency and Ease-of-Use: Applications express their intended communication pattern using standard IPMC interfaces, rather than using hand-coded implementations of what is really an administrative policy.
- Robustness: The MCMD is implemented as a distributed, fault-tolerant service.

We provide and evaluate effective heuristics for the optimization problem of allocating the limited number of IPMC addresses, although brevity limits us to a very terse review of the framework, the underlying ($NP$-complete) optimization question, and the models used in the evaluation.

[*]This work was supported by grants from AFRL, AFOSR, NSF, Cisco and Intel.

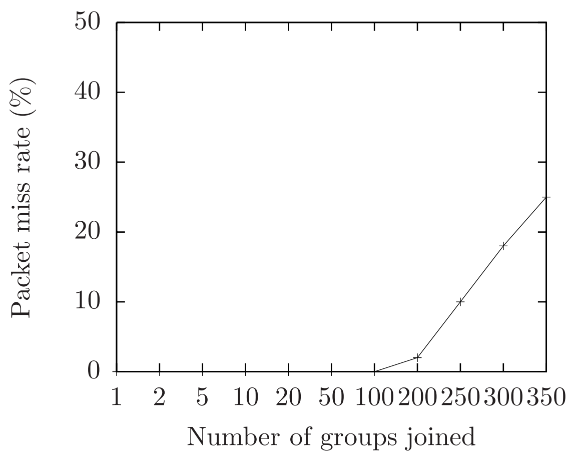


**Figure 1**: Receiver packet miss rate vs. number of IPMC groups joined

## 2 IPMC in the Data Center

Modern data centers often have policies legislating against the use of IPMC, despite the fact that multicast is a natural expression of a common data communication pattern seen in a wide range of applications. This reflects a number of pragmatic considerations. First, IPMC is perceived as a potentially costly technology in terms of performance impact on the routing and NIC hardware. Second, applications using IPMC are famously unstable, running smoothly in some settings and yet, as scale is increased, potentially collapsing into chaotic multicast storms that disrupt even non-IPMC users.

The hardware issue relates to imperfect filtering. A common scheme used to map IP group addresses to Ethernet group addresses involves placing the low-order 23 bits of the IP address into the low-order 23 bits of the Ethernet address [8]. Since there are 28 significant bits in the IP address, more than one IP address can map to an Ethernet address. The NIC maintains the set of Ethernet mappings for joined groups and forwards packets to the kernel only if the destination group maps to one of those Ethernet addresses. As a result, with large numbers of groups, the NIC may accept undesired packets, which the kernel must discard.

Figure 1 illustrates the issue. In this experiment, a multicast transmitter transmits on $2k$ multicast groups, whereas the receiver listens to $k$ multicast groups. We varied the number of multicast groups $k$ and measured the CPU consumption as well as the packet loss at the receiver. The transmitter transmits at a constant rate of 15,000 packets/sec, with a packet size of 8,000 bytes spread across all the groups. The receiver thus expects to receive half of that, i.e. 7,500 packets/sec. The receiver and transmitter have 1Gbps NICs and are connected by a switch with IP routing capabilities. The experiments were conducted on a pair of single core Intel® Xeon™ 2.6GHz machines. Figure 1 shows that the critical threshold that the particular NIC can handle is roughly 100 IPMC groups, after which throughput begins to fall off.

The issue isn't confined to the NIC. Performance of modern 10Gbps switches was evaluated in a recent review [10] which found that their IGMPv3 group capacity ranged between as little as 70 and 1,500. Less than half of the switches tested were able to support 500 multicast groups under stress without flooding receivers with all multicast traffic.

The MCMD addresses these problems in two ways. First, by letting the operator limit the number of IPMC addresses in use, the system ensures that whatever the limits in the data center may be, they will not be exceeded. Second, by optimizing to use IPMC addresses as efficiently as possible, the MCMD arranges that the IPMC addresses actually used will be valuable ones – large groups that receive high traffic. As seen below, this is done not just by optimizing across the groups as given, but also by discovering ways to aggregate overlapping groups into structures within which IPMC addresses are shared by multiple groups, permitting even greater efficiencies.

The perception that IPMC is an unstable technology is harder to demonstrate in simple experiments: as noted earlier, many applications are perfectly stable under most patterns of load and scale, yet capable of being extraordinarily disruptive. The story often runs something like this. An application uses IPMC to send to large numbers of receivers at a substantial data rate. Some phenomenon now triggers loss. The receivers detect the loss and solicit retransmissions, but this provokes a further load surge, exacerbating the original problem. A multicast storm ensues, saturating the network with redundant retransmission requests and duplicative multicasts. With MCMD the operator can safely deploy such an application: if it works well, it will be permitted to use IPMC; if it becomes problematic, it can be mapped to UDP merely by changing the acceptable use policy. More broadly, the MCMD encourages developers to express intent in a higher-level form, rather than hand-coding what is essentially an administrative policy.

## 3 Design

The basic operation of MCMD is simple. It translates an application-level multicast address used by

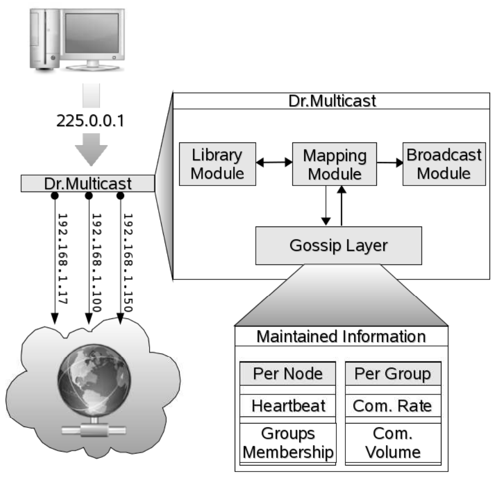


**Figure 2**: Overview of the MCMD architecture

an application to a set of unicast addresses and network-level multicast addresses. MCMD has two components (see figure 2):

- A *library* module responsible for the *mechanism* of translation. It intercepts outgoing multicast messages and instead sends them to a set of unicast and multicast destinations.

- A *mapping* module responsible for the *policy* of translation. It determines the mapping from each application-level address and a set of unicast and network-level multicast addresses.

## 3.1 Library Module

The MCMD library module exports a `<sockets.h>` library to applications, with interfaces identical to the standard POSIX version. By overloading the relevant socket operations, MCMD can intercept join, leave and send operations. For example:

- `setsockopt()` is overloaded so that an invocation with the IP_ADD_MEMBERSHIP or IP_DROP_MEMBERSHIP option as a parameter results in a 'join' message being sent to the mapping module. In this case, the standard behavior of `setsockopt` – generating an IGMP message – is suppressed.

- `sendto()` is overloaded so that a send to a class D group address is intercepted and converted to multiple sends to a set of addresses from the kernel.

The library module interacts with the mapping module via a UNIX socket. It pulls the translations for each application-level group from the mapping module. Simultaneously, it pushes information and statistics about grouping and traffic patterns used by the application to the local mapping module.

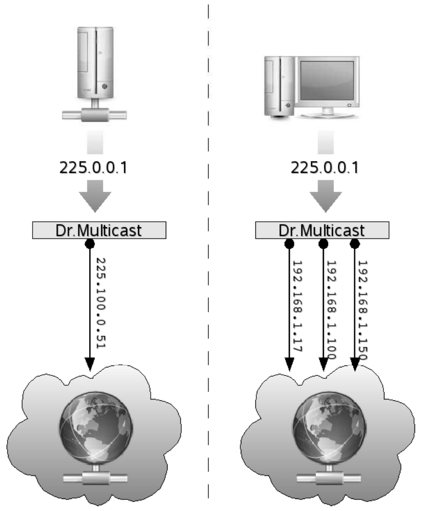


**Figure 3**: Two under-the-hood mappings in MCMD, a direct IPMC mapping on the left and point-to-point mapping on the right.

## 3.2 Mapping Module

The mapping module plays two important roles:

- It acts as a Group Membership Service (GMS), maintaining the membership set of each application-level group in the system.

- It allocates a limited set of IPMC addresses to different sets of machines in the data center and orchestrates the IGMP joins and leaves required to maintain these IPMC groups within the network.

The mapping module uses a gossip-based control plane using techniques described in [13]. The gossip control plane is extremely resilient to failures and includes a decentralized failure detector that can be used to locate and eject faulty, i.e. irresponsive, machines. It imposes a stable and constant overhead on the system and has no central bottleneck, irrespective of the number of nodes.

The gossip-based control plane essentially replicates mapping and grouping information slowly and continuously throughout the system. As a result, the mapping module on any single node has a global view of the system and can immediately resolve an application-level address to a set of unicast and multicast addresses without any extra communication. The size of this global view is not prohibitive; for example, we can store membership and mapping information for a 1000-node data center within a few

MB of memory. For now, we're targetting systems with a low enough rate of joins, leaves and failures per second to allow for global replication of control information. In the future, we'll replace the global replication scheme with a more focused one to eliminate this restriction.

Knowledge of global group membership is sufficient for the mapping module at each node to translate application-level group addresses into network-level unicast addresses. To fulfill the second function of allocating IPMC addresses in the system, an instance of the specific mapping module running on a particular node in the system acts as a leader. It aggregates information from other mapping modules (via the gossip control plane) and calculates appropriate allocations of IPMC addresses to mandate within the data center. The leader can be chosen according to different strategies – one simple expedient is to query the gossip layer for the oldest node in the system. The failure of the leader is automatically detected by the gossip layer's inbuilt failure detector, which also naturally updates the pointer to the oldest node.

### *3.2.1 Gossip Control Plane*

We first describe an implementation of the mapping module using only the gossip-based control plane. However, the Achilles heel of gossip at large system sizes is *latency* – the time it takes for an update to propagate to every node in the system. Consequently, we then describe approaches to add extra control traffic for certain kinds of critical updates – in particular, IPMC mappings and group joins – that need to be distributed through the system at low latencies.

**Gossip-based Failure Detector:** The MCMD control plane is a simple and powerful gossip-based failure detector identical to the one described by van Renesse [13]. Each node maintains its own version of a global table, mapping every node in the data center to a timestamp or heartbeat value. Every $T$ milliseconds, a node updates its own heartbeat in the map to its current local time, randomly selects another node and reconciles maps with it. The reconciliation function is extremely simple – for each entry, the new map contains the highest timestamp from the entries in the two old maps. As a result, the heartbeat timestamps inserted by nodes into their own local maps propagate through the system via gossip exchanges between pairs of nodes.

When a node notices that the timestamp value for some other node in its map is older than $T_1$ seconds, it flags that node as 'dead'. It does not immediately delete the entry, but instead maintains it in a dead state for $T_2$ more seconds. This is to prevent the case where a deleted entry is reintroduced into its map by some other node. After $T_2$ seconds have elapsed, the entry is truly deleted.

The comparison of maps between two gossiping nodes is highly optimized. The initiating node sends the other node a set of hash values for different portions of the map, where portions are themselves determined by hashing entries into different buckets. If the receiving node notices that the hash for a portion differs, it sends back its own version of that portion. This simple interchange is sufficient to ensure that all maps across the system are kept loosely consistent with each other. An optional step to the exchange involves the initiating node transmitting its own version back to the receiving node, if it has entries in its map that are more recent than the latter's.

**Gossip-based Communication:** Thus far, we have described a decentralized gossip-based failure detector. Significantly, such a failure detector can be used as a general purpose gossip communication layer. Nodes can insert arbitrary state into their entries to gossip about, not just heartbeat timestamps. For example, a node could insert the average CPU load or the amount of disk space available; eventually this information propagates to all other nodes in the system. The reconciliation of entries during gossip exchanges is still done based on which entry has the highest heartbeat, since that determines the staleness of all the other information included in that entry.

Using a gossip-based failure detector as a control communication layer has many benefits. It provides extreme resilience and robustness for control traffic, eliminating any single points of failure. It provides extremely clean semantics for data consistency – a node can write only to its own entry, eliminating any chance of concurrent conflicting writes. In addition, a node's entry is deleted throughout the system if the node fails, allowing for fate sharing between a node and the information it inserts into the system.

**Group Membership Service:** The mapping module uses the gossip layer to maintain group membership information for different application-level groups in the system. Each node maintains in its gossip entry – along with its heartbeat timestamp – the set of groups it belongs to, updating this whenever the library module intercepts a join or a leave. A simple scan of the map is sufficient to generate an alternative representation of the membership information, mapping each group in the system to all the nodes that belong to it. If a node fails, its entry is removed from the gossip map; as a result, a subsequent scan of the map generates a groups-to-nodes table that excludes the node from all the groups it belonged to.

**Mapping Module Leader:** As mentioned previously, the gossip layer informs the mapping module of the identity of the oldest node in the system, which is then elected as a leader and allocates IPMC addresses. To distribute these allocations back into the system, the leader can just update its own entry in the gossip map with the extra IPMC information. When a receiver is informed of a relevant new mapping, it issues the appropriate IGMP messages required to join or leave the IPMC group as mandated by the mapping module.

A "pure" gossip protocol can have large propagation delays, resulting in undesirable effects such as senders transmitting to IPMC groups before receivers can join them. To mitigate these latency effects, the leader periodically broadcasts mappings at a fixed, low rate to the entire data center. The rate of these broadcasts is tunable; we expect typical values to be a few packets every second. The broadcast acts purely as a latency optimization over the gossip layer; if a broadcast message is lost at a node, the mapping is eventually delivered to it via gossip.

**Latency Optimization of Joins:** We are also interested in minimizing the latency of a join to an application-level multicast group; i.e., after a node issues a join request to a group, how much time elapses before it receives data from all the senders to that group? While the gossip layer will eventually update senders of the new membership of the group, its latency may be too high to support applications that need fast membership operations. The latency of leave operations is less critical, since a receiver that has left a group can filter out messages arriving in that group from senders who have stale membership information until the gossip layer propagates the change.

In MCMD, we explore two options to speed up joins. The first method is to have receivers broadcast joins to the entire data center. For most data center settings, this is a viable option since the rate of joins in the system is typically quite low. This approach is drawn on figure 2. The second method is meant for handling higher churn; it involves explicitly tracking the set of senders for each group via the gossip layer. Since each node in the system knows the set of senders for every group, a receiver joining a group can directly send the join using multiple unicasts to the senders of that group. The second option incurs more space and communication overhead in the gossip layer but is more scalable in terms of churn and system size.

Switching between these two options can be done by a human administrator or automatically by a designated node, such as the mapping module, simply by observing the rate of membership updates in the system via the gossip layer. Once again, the broadcasts or direct unicasts do not have to be reliable, since the gossip layer will eventually propagate joins throughout the system.

### 3.3 Kernel Multi-send System Call

Sending a single packet to a physical IPMC group is cheap since the one-to-many multiplexing is done on a lower level by routing or switching hardware in the network. However, when IPMC resources are exhausted, the group-address mapping in MCMD will map a logical IPMC group to a set of unicast addresses corresponding to its members. Thus a single `sendto()`-call at the interface would produce a series of sends at the library and kernel level of identical packets to a number of physical addresses. We modified the kernel to help alleviate the overhead caused by context-switching during the list of sends. We implemented a *multi-send* system call on the Linux 2.6.24 kernel with a `sendto()`-like interface that sends a message to multiple destinations.

## 4 Optimizing Resource Use

Beyond making IPMC controllable and hence safe, the MCMD incorporates a further innovation. We noted that our goal is to optimize the limited use of IPMC addresses. Such optimization problems are often hard, and indeed the optimization problem that arises here we have proven to be $NP$-complete (details omitted for brevity). Particularly difficult is the problem of mapping multiple application-level groups to the same IPMC address: doing so shares the address across a potentially large set of groups, which is a good thing, but finding the optimal pattern for sharing the addresses is hard.

A *topic* is a logical multicast group. Our algorithm can be summarized as follows.

- Find and merge all identically overlapping topics into *groups*, aggregating the traffic reports.
- Sort groups in descending order by the product of the reported traffic rate and topic size.
- For each group $G$, assign an IPMC address to topic $G$, unless the global or user address quota for $\geq 3$ members have been exceeded.

- Enlist all remaining users in $G$ for point-to-point communication over unicast.

Thus a large topic with high traffic is more likely to be allocated a dedicated IPMC address. Other groups might communicate over both IPMC and point-to-point unicast for members that have exceeded their NIC IPMC capacity, and yet others might perform multicast over point-to-point unicast entirely.

## 5 Related Work

Brevity prevents a detailed comparison of our work with previous work of [14, 15]; key differences stem from our narrow focus on data center settings. Our mathematical framework extends that of [12], but instead of inexact channelization we investigate zero filtering.

## 6 Conclusion

Many major data center operators legislate against the use of IP multicast: the technology is perceived as disruptive and insecure. Yet IPMC offers very attractive performance and scalability benefits. Our paper proposes Dr. Multicast (the MCMD), a remedy to this conundrum. By permitting operators to define an acceptable use policy (and to modify it at runtime if needed), the MCMD permits active management of multicast use. Moreover, by introducing a novel scheme for sharing scarce IPMC addresses among logical groups, the MCMD reduces the number of IPMC addresses needed sharply, and ensures that the technology is only used in situations where it offers significant benefits.

## References

[1] BEA Weblogic. `http://www.bea.com/framework.jsp?CNT=features.htm&FP=/content/products/weblogic/server/`, 2008.

[2] GEMSTONE GemFire. `http://www.gemstone.com/products/gemfire/enterprise.php`, 2008.

[3] IBM WebSphere. `http://www-01.ibm.com/software/webservers/appserv/was/`, 2008.

[4] JBoss Application Server. `http://www.jboss.org/`, 2008.

[5] Oracle Coherence. `http://www.oracle.com/technology/products/coherence/index.html`, 2008.

[6] Real Time Innovations Data Distribution Service. `http://www.rti.com/products/data_distribution/`, 2008.

[7] TIBCO Rendezvous. `http://www.tibco.com/software/messaging/rendezvous/default.jsp`, 2008.

[8] Deering, S. Host Extensions for IP Multicasting. *Network Working Request for Comments 1112 (August 1989)* (1989).

[9] Fei, A., Cui, J., Gerla, M., and Faloutsos, M. Aggregated multicast: an approach to reduce multicast state. *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE 3* (2001).

[10] Newman, D. Multicast performance differentiates across switches. `http://www.networkworld.com/reviews/2008/032408-switch-test-performance.html`, 2008.

[11] Rosenzweig, P., Kadansky, M., and Hanna, S. The Java Reliable Multicast Service: A Reliable Multicast Library. *Sun Labs* (1997).

[12] Tock, Y., Naaman, N., Harpaz, A., and Gershinsky, G. Hierarchical clustering of message flows in a multicast data dissemination system. In *IASTED PDCS* (2005), S. Q. Zheng, Ed., IASTED/ACTA Press, pp. 320–326.

[13] van Renesse, R., Minsky, Y., and Hayden, M. A gossip-based failure detection service. In *Middleware'98, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing* (England, September 1998), pp. 55–70.

[14] Wong, T., and Katz, R. An analysis of multicast forwarding state scalability. In *ICNP '00: Proceedings of the 2000 International Conference on Network Protocols* (Washington, DC, USA, 2000), IEEE Computer Society, p. 105.

[15] Wong, T., Katz, R. H., and McCanne, S. An evaluation on using preference clustering in large-scale multicast applications. In *INFOCOM (2)* (2000), pp. 451–460.

# How to evaluate exotic wireless routing protocols?*

Dimitrios Koutsonikolas[1], Y. Charlie Hu[1], Konstantina Papagiannaki[2]
[1]Purdue University and [2]Intel Research, Pittsburgh

## ABSTRACT

The advent of static wireless mesh networks (WMNs) has largely shifted the design goals of wireless routing protocols from maintaining connectivity among routers to providing high throughput. The change of design goals has led to the creation of many "exotic" optimization techniques such as opportunistic routing and network coding, that promise a dramatic increase in overall network throughput. These "exotic" techniques have also moved many mechanisms such as reliability and rate control, that used to be below or above the routing layer in traditional protocols, to the routing layer.

In this paper, we first review the above evolution of routing protocol design and show that the consolidation of mechanisms from multiple layers into the routing layer poses new challenges to the methodology for evaluating and comparing this new generation of routing protocols. We then discuss the diverse set of current practices in evaluating recently proposed protocols and their strengths and weaknesses. Our discussion suggests that there is an urgent need to carefully rethink the implications of the new merged-layer routing protocol design and develop effective methodologies for meaningful and fair comparison of these protocols.

Finally, we make several concrete suggestions on the desired evaluation methodology. In particular, we show that the traffic sending rate plays a fundamental role and should be carefully controlled.

## 1. RENAISSANCE OF WIRELESS ROUTING PROTOCOL DESIGN

The recent evolution of wireless networking from the ad hoc networking era to the mesh networking era has ignited a new Renaissance of routing protocol design for multihop wireless networks.

In the ad hoc networking era, the primary challenge faced by routing protocols (e.g., DSR [11], AODV [17]) was to deal with frequent route breaks due to host mobility in a dynamic mobile environment. Accordingly, most research efforts were focused on designing efficient route discovery/repair schemes to discover or repair routes with minimum overhead. The routing process itself was simple; once a route from the source to a destination was known, each hop along the route simply transmitted the packet to the next hop via 802.11 unicast. These protocols relied on 802.11 unicast (with its built-in ACK-based local recovery scheme and exponential backoff) to deal with packet loss due to channel errors or collisions.

The design goals of the ad hoc routing protocols also drove their evaluation methodology. The comparison between different protocols was usually in terms of Packet Delivery Ratio (PDR) and control overhead (e.g. [2, 5]). The offered load, typically of some constant rate, was low so that the resulting data traffic and control overhead do not exceed the network capacity. The main parameter varied in the evaluations was the *pause time* of the random waypoint mobility model, which characterized how dynamic the environment was. The focus of such a methodology was to offer a direct comparison of various protocols' ability to transfer data to the destination under host mobility, while incurring low control overhead. Interestingly, often times the protocol comparisons boiled down to tradeoffs between PDR and control overhead [2, 5].

Transition to WMNs changed these rules. In a WMN, routers are static and hence route changes due to mobility are not a concern anymore. The main performance metric is now *throughput*, often times even at the cost of increased control overhead.

The first major effort towards the new design goal was on designing link-quality path metrics (e.g., ETX [4], ETT [6]) that replaced the commonly used shortest-path metric. The protocols using these link-quality metrics still followed the layering principle: the routing layer finds a good route, and 802.11 unicast is used to deliver packets hop by hop.

**Opportunistic Routing.** Seeking further throughput improvement, researchers looked into new, "exotic" techniques, which largely abandoned the layering principle. The first such technique was opportunistic routing as demonstrated in the ExOR protocol [1]. Instead of having a decoupled MAC and routing layer, ExOR explored an inherent property of the wireless medium, its broadcast nature. Instead of first determining the next hop and then sending the packet to it, it broadcasts the packet so that all neighbors have the chance to hear it; among those that received the packet, the node closest to the destination forwards the packet. This also implies that some coordination is required, so that the neighboring nodes can agree on who should rebroadcast the packet next. To reduce the coordination overhead, ExOR proposed sending packets in batches.

**Intra-flow network coding.** The second "exotic" technique applied network coding to multihop wireless networks. With network coding, each mesh router randomly mixes packets it has received before forwarding them. The random mixing ensures with high probability that nodes will not forward the same packet, and hence coordination overhead is min-

*We thank the reviewers for their insightful comments. This work was supported in part by NSF grant CNS-0626703.

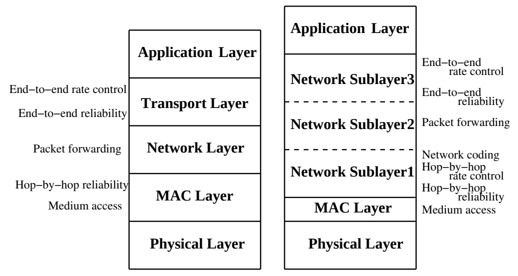
imized. Network coding has one more positive effect. It resembles traditional Forward Error Correction (FEC) techniques, which offer reliability through redundancy, with the extra advantage that it is applied at every hop, and not end-to-end [16, 8]. Together, network coding eliminates the need for reliability on a per-hop or per-packet basis. Since each coded packet contains information about many packets, the destination can reconstruct the original data if it receives sufficiently many packets. MORE [3] was the first protocol to combine opportunistic routing with network coding.

Both techniques use unreliable 802.11 broadcast as the hop-by-hop forwarding technique, which is a significant departure from traditional routing protocols. The use of broadcast is a necessity for opportunistic routing as well as effective network coding. Since the MAC now does not have to deal with retransmissions and exponential backoffs, it can send at much higher packet rates than in the unicast mode; it is essentially limited only by carrier sensing. Sending at higher rates potentially implies higher goodput. Since the design goal is focused on high throughput, this observation has an immediate implication for the evaluation methodology of these new protocols: instead of using a constant rate (CBR) of $X$ packets per second, the source node should send *as fast as the MAC allows.*

However, making the sources send as fast as the MAC allows has a serious side effect. It can cause congestion in the network if the aggregate transmission rate of the nodes exceeds the network capacity. As [14] showed, in contrast to the wired Internet, where congestion is the result of a complex interaction among many flows, in a wireless network, congestion can happen even with a single flow, in a simple topology and even with 802.11 unicast. The use of broadcast in this new generation of routing schemes simply worsens the situation, since the lack of exponential backoff in the 802.11 broadcast mode means nodes never really slow down.

**Rate control.** With congestion, the queues of the nodes become full, causing significant packet loss. We thus need to reintroduce the mechanism for preventing the network from reaching this state: rate control. SOAR [18] is a new opportunistic routing protocol that has a built-in rate control mechanism, both at the source (using a sliding window) and at intermediate routers (using small queues to avoid unpredictable queuing delays). Other protocols (e.g., [19, 9]) propose hop-by-hop, backpressure-based mechanisms to limit the amount of traffic injected in the network. Hence, *rate control, which used to be the responsibility of higher layer protocols (transport or application), is now brought down to the routing layer.*

**Inter-flow network coding.** The final frontier is that of *increasing the network capacity* itself! The basic idea is again simple: a router can XOR packets from different flows (hence inter-flow network coding as opposed to intra-flow network coding discussed previously) and broadcast them. If the next hop of each flow has already overheard all the mixed packets except for the one destined for it, it can XOR them again with the XORed packet to obtain its own packet.



(a) Traditional Network Stack (b) New Network Stack

**Figure 1**: The evolution of the protocol stack.

COPE [12] was the first protocol that brought this idea from theory into practice. By mixing packets belonging to different flows and transmitting them as one, one reduces the total number of transmissions required, and hence increases the "effective" capacity of the network.

Since the technique stretches the capacity of the network, the most natural way to show its improvement, i.e., the implied evaluation methodology, is to subject the network to a traffic load (not too much) above the physical capacity, i.e., the network should already be congested before network coding is turned on, which will then increase the effective capacity just enough to eliminate the congestion.

**Reliability.** Since 802.11 broadcast is unreliable, with the exception of intra-flow network coding, which embraces FEC, all other techniques, which rely on MAC-layer broadcast, require some ARQ-based recovery mechanism. ExOR uses end-to-end retransmissions by going through the same batch of packets until 90% of them are received by the destination; SOAR and COPE use asynchronous cumulative hop-by-hop acknowledgments; COPE also relies partly on 802.11 unicast (known as pseudobroadcast [12]). Hence, in addition to rate control, one more mechanism, *reliability, which used to be the responsibility of either upper (end-to-end) or lower (hop-by-hop) layers, is now brought to the routing layer.*

In summary, the "exotic" techniques used in new routing protocols for WMNs have largely abandoned the layering principle and adopted a merged-layer approach, as shown in Figure 1. Mechanisms that used to be at lower or higher layers are now blended into the routing layer. *This consolidation of mechanisms and techniques into the routing layer has made the evaluation of routing protocol performance a much subtler task than before.* For example, some mechanisms and techniques may be conflicting: inter-flow network coding desires traffic load to be above the network capacity while rate control targets the exact opposite.

In the next section, we discuss the resulting diverse set of current practices in evaluating this new generation of routing protocols. We show that, in contrast to traditional routing protocols, there have been no clear guidelines that drive the evaluation of these protocols; often times each new protocol is evaluated with a different methodology.

## 2. STATE OF AFFAIRS

There have been many high-throughput routing protocols for WMNs proposed over the last few years. Due to the page

**Table 1**: Methodologies used in evaluating recent high-throughput WMN routing protocols.

| | Evaluation Methodology | Example |
|---|---|---|
| **Unreliable protocols** | Make both protocols reliable but in different ways | ExOR [1] |
| | Evaluate for a wide range of sending rates, with deteriorating PDR | COPE [12] |
| | Compare a protocol with rate control against a protocol without rate control | SOAR [18] |
| | Old ad hoc methodology: keep the sending rate fixed below capacity, measure PDR | ROMER [20] |
| **Reliable protocols** | Compare a reliable protocol against an unreliable protocol | MORE [3] |
| | Compare a reliable protocol against an unreliable protocol under TCP | noCoCo [19] |
| | Modify an unreliable protocol to incorporate the same reliability mechanism of a new protocol | noCoCo [19] |

limit, we review here the evaluation methodologies used in a subset of them, as summarized in Table 1.

## 2.1 Evaluation of Unreliable Protocols

In the case of unreliable protocols (e.g., for multimedia applications that do not require 100% PDR), the main objective is high throughput perceived by the destinations, i.e., high goodput. The new trend in the evaluation methodology is to saturate the network, letting the sources send as fast as possible so that the traffic load in the network exceeds the available capacity; then measure the maximum amount of traffic the protocol can deliver to the destination.

However, such a methodology is flawed in that it completely deemphasizes the PDR metric. The fact that certain applications do not require 100% PDR does not mean that reliability is a factor that can be completely neglected. Many applications have certain lower bounds for reliability; for example the quality of a video deteriorates with packet loss, and hence if the PDR drops below a threshold, the video quality becomes unacceptable.

**Practice 1: Making both protocols reliable.** ExOR guarantees reliable end-to-end delivery of 90% of each batch; every node keeps retransmitting packets belonging to a given batch until they are acknowledged by a node closer to the destination. The last 10% of the packets could incur a lot of overhead if they were sent through ExOR, and hence they are sent through traditional routing, which does not offer any guarantee for end-to-end reliability.

The authors argued that a direct comparison of ExOR with traditional routing would be unfair and they conducted the experiments in a way that guaranteed 100% PDR with both of them. In each case, the size of the file to be downloaded was 1MB. Instead of using traditional routing to carry the last 10% of the file, the evaluation of ExOR was based on the transmission of a 1.1 MB file, so as to compensate for loss. In contrast, the traditional routing protocol was only used to determine the route offline. The 1MB file was then transfered sequentially hop-by-hop, thus eliminating collisions, and also packet drops due to queue overflows.[1]

While this methodology was largely fair, it eliminated one important feature of traditional routing that does not exist in ExOR: spatial reuse. To avoid duplicate transmissions, nodes in ExOR are assigned priorities, and only one node transmits at a time – hence, coordination is achieved at the cost of reduced spatial reuse. In contrast, with traditional routing simultaneous transmissions can take place across the network as long as they do not interfere with each other. This advantage can turn into a drawback in the presence of a large number of hidden terminals. In other words, by trying to make the comparison fair by adding reliability to traditional routing, the authors also removed one feature of traditional routing. Whether this feature harmed traditional routing depends on the particular environment used for the evaluation.

**Practice 2: No rate control - varying the sending rate.** COPE in [12] was compared against a traditional routing protocol (Srcr), under UDP traffic.[2] In an 802.11a network with a nominal bitrate of 6Mbps, the experiment was repeated for gradually increased total offered load. The aggregate throughput over the total offered load for the two protocols was then presented, as shown in Figure 2 (Figure 12 in [12]).

We make several observations on Figure 2. First, the advantage of COPE is best shown when the traffic load in the network is pushed beyond the capacity. Since it is not clear what the traffic load is, the best thing is to measure throughput for varying offered load, as done by the authors. As expected, at low loads, COPE performs similarly to traditional routing. As the load increases, COPE offers on *average* 3-4$x$ throughput improvement over traditional routing. Second, like traditional routing, the goodput of COPE also peaks when the offered load is around the effective capacity of the network (now higher because of inter-flow network coding), and decreases quickly as the load further increases, and the PDR value, which can be easily calculated by dividing the y value by the x value, deteriorates sharply, possibly below the acceptable level of many applications. Third, if the protocols have rate control mechanisms, ideally the goodput should remain constant when the offered load is increased to beyond the network capacity. Since neither protocol has rate control, we witness the decline of the goodput.

**Practice 3: Comparing a protocol with rate control against a protocol without.** SOAR applies sliding window-based rate control at the sources, trying to discover the optimal sending rate online. In contrast, traditional routing has no rate control. This immediately creates a challenge for a fair comparison of the two protocols. Faced with this challenge, the authors decided to perform the evaluation in a saturated network, where each source transmits at 6Mbps, same as the nominal bitrate of the network.

[1]The packet losses due to channel errors were masked in the testbed through 802.11 retransmissions.

[2] [12] also evaluated COPE and Srcr under TCP. In that case, although the two protocols are unreliable, reliability is provided by the transport layer.

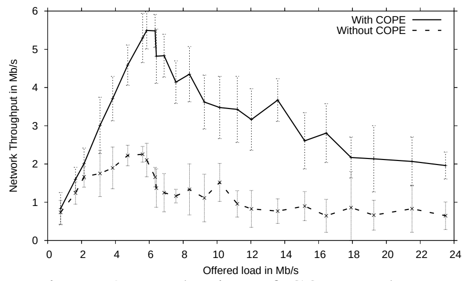


**Figure 2**: Evaluation of COPE and traditional routing in an ad hoc network for UDP flows. Reproduced Figure 12 from [12].

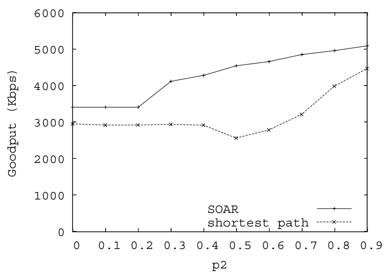


**Figure 3**: Evaluation of SOAR and traditional routing: 4 parallel flows in a grid topology. Reproduced Figure 14(b) from [18].

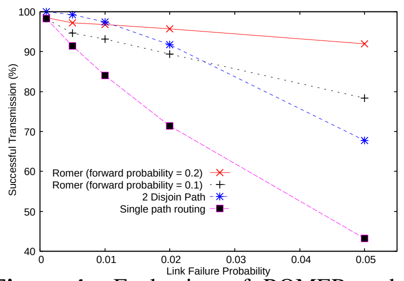


**Figure 4**: Evaluation of ROMER and traditional routing: PDR over link failure probability. Reproduced Figure 5 from [20].

Saturating the network creates an adverse situation for traditional routing, which is expected to perform poorly under these conditions and suffer significant packet loss due to queue overflows. In contrast, SOAR adapts the sending rate online, based on the network conditions. SOAR was shown to offer a large throughput improvement over traditional routing (one example is shown in Figure 3, Figure 14(b) in [18]). However, it is not clear what part of the improvement is because of the opportunistic forwarding and what part is because of the rate control.

**Practice 4: Old methodology (for evaluating ad hoc protocols).** ROMER [20] is another opportunistic routing protocol which exploits link bitrate diversity in order to maximize throughput. It uses the 802.11 unicast autorate adaptation mechanism, and tries to send traffic over high rate links, in contrast to ExOR and SOAR, which always use a fixed link bitrate. ROMER was evaluated under yet another methodology different from ExOR and SOAR. The authors compared the PDR (and throughput gain) achieved by ROMER over traditional routing, following the old methodology for ad hoc protocol evaluation. The parameter varied is the link failure probability, while the source sending rate is kept constant (and the value is unclear).

Due to the autorate adaptation, it is difficult to estimate the capacity of the network used for the evaluation. The high delivery rates achieved (at least) by ROMER (in Figure 4, Figure 5 in [20]) make us conjecture that the sending rate was not high enough to congest the network, in contrast to in [18] and [12]. However, a single sending rate does not reveal the maximum gain achieved by ROMER, in particular if this rate is far below the capacity of the network.

## 2.2 Evaluation of Reliable Protocols

Traditional routing protocols left to the transport layer the responsibility for end-to-end reliability. However, TCP, the de facto reliable transport layer protocol for the wired Internet, has been reported to perform poorly in multihop wireless networks [7, 13, 10], especially in environments with many hidden terminals and highly lossy links. The reason is that TCP performs congestion control in addition to reliability and correlates these two mechanisms. High packet loss causes TCP flows to suffer timeouts and excessive backoff, and it prevents them from increasing their window size and utilizing the wireless medium efficiently. This is the reason many new protocols ignore TCP, and incorporate mechanisms for end-to-end reliability at the network layer instead.

**Practice 5: Comparing a reliable with an unreliable protocol.** In [3], MORE is compared against traditional routing showing a median throughput gain of 95%. The authors used UDP traffic for both protocols sent at the maximum possible data rate, i.e., the source transmitted as fast as the MAC allowed. As we have already explained, in a highly congested environment, 802.11 unicast cannot help traditional routing to recover from packet drops due to queue overflows. In contrast, with MORE there is no queuing. With a batch size of $k$ packets, every MORE router only needs to keep $k$ linearly independent packets in a buffer; linearly dependent packets do not include any new information and can be safely dropped. Hence, a MORE router does not experience losses due to queue overflows, no matter how fast it receives packets from its upstream nodes. In addition, the FEC element contained in network coding masks packet losses due to collisions and channel errors through redundancy. Thus, a reliable protocol was compared against an unreliable one.

This does not necessarily mean that the comparison favored MORE over traditional routing. In the evaluation of the two protocols, a fixed size file was sent from the source to the destination with each protocol, however with traditional routing only a fraction of this file is finally delivered to the destination. Depending on the fraction of the file that is lost and the time taken for the transfer, this evaluation could favor any of the two protocols. In other words, adding an end-to-end reliability mechanism to traditional routing would increase the numerator of the throughput formula (the amount of data delivered) but it would also increase the denominator (the time taken for the total transfer); this could lead to either an increase or a decrease to the throughput achieved with traditional routing.

**Practice 6: Running an unreliable protocol under TCP.** An easy way to provide end-to-end reliability with an unreliable routing protocol is to run it under TCP; no change is required to the protocol itself. This is one of the approaches followed by [19] in the evaluation of noCoCo. noCoCo improves COPE by scheduling the transmissions at the nodes in order to maximize the gain from inter-flow network coding. Coupled with scheduling in noCoCo is a backpressure, hop-by-hop congestion control mechanism. This mechanism eliminates queue overflows and packet dropping and guarantees end-to-end reliable packet delivery. Hence, in noCoCo, sources do not transmit as fast as the MAC allows; their sending rates are limited by the congestion control mechanism.

In the evaluation, noCoCo was compared against COPE [12] and traditional routing. The main goal was to quantify the gains of coordinated network coding used in noCoCo against opportunistic network coding, used in COPE. TCP was used with COPE and traditional routing to provide reliability (and congestion control) at the transport layer. However, TCP is known to perform poorly in multihop wireless networks; in addition, it was shown to interact poorly with COPE and limit the coding opportunities and consequently the throughput gain [12]. Hence, this methodology again blurred the true gain from coordinated coding, since different congestion control and reliability mechanisms are used. The authors acknowledged this point and noted that it should be taken into account when trying to interpret the results.

**Practice 7: Modifying an unreliable protocol.** To finally isolate the gain from coordinated coding, the authors of noCoCo also modified traditional routing and COPE to use the same backpressure-based algorithm for congestion control and reliability, thus removing the negative side-effects of TCP.

### 2.3 Use (or No Use) of Autorate Adaptation

802.11 unicast allows a sender to change the bit rate automatically, based on the quality of the link to the receiver. On the other hand, the majority of the "exotic" optimization techniques are based on 802.11 broadcast, and hence most of the new routing protocols based on these techniques (with the exception of ROMER) do not use autorate adaptation. For "fair" comparison, the evaluation of these protocols often disables autorate adaptation for the traditional, unicast routing, e.g., in [1, 12, 18, 19] (one notable exception is [3]). We argue the contrary; the methodology is unfair to traditional routing if it can benefit from autorate adaptation.

## 3. RECOMMENDATIONS

We have witnessed the inconsistencies in the current evaluation methodologies of the new generation of routing protocols. In the following, we make recommendations for more consistent and meaningful evaluation methodologies.

**The importance of rate control.** Rate control is fundamental for the optimal operation of any (unreliable or reliable) protocol, as it ensures that the traffic load does not exceed the network capacity limit.

Figure 5 shows our envisioned throughput performance for well designed unreliable protocols. Traditional routing under UDP has no rate control mechanism incorporated. When the offered load exceeds the network capacity, packets start

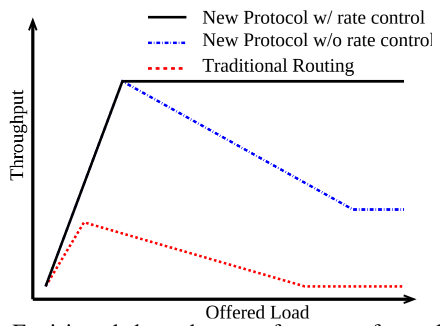


**Figure 5**: Envisioned throughput performance for well designed unreliable protocols (with built-in rate control), in contrast to traditional routing and high-throughput protocols without rate control.

getting dropped due to congestion, possibly reducing the throughput much below its possible maximum value. New protocols with "exotic" techniques are expected to offer a dramatic increase to the throughput; they can even increase the capacity bound (e.g., from inter-flow network coding). However, without rate control, congestion can build up and throughput will also start decreasing when the (new) capacity point is exceeded. By adding appropriate rate control, the goodput is expected to remain constant when the offered load is beyond the capacity. One implication of this design guideline is that there may be no need to vary the offered load beyond the capacity point any more.

For reliable protocols, PDR remains 100% but the argument for rate control is still valid. When reliability is provided through the traditional way (ARQ), some rate control is implicitly imposed, since retransmissions are given priority over new packets. However, when reliability is part of the "exotic" technique (e.g., intra-flow network coding embraces FEC), the source may never slow down, unless explicitly forced by rate control. In any case, exceeding the capacity of the network will lead to unpredictable behavior which will appear either in the form of increased delays, severe unfairness among flows, or reduced throughput. As an example, the gain of MORE over traditional routing in [3] is reduced in the presence of multiple flows. A related recommendation is that a protocol should also be evaluated with multiple flows, e.g., as in [3, 18], as the rate control for each flow becomes more challenging.

Note that the best method for applying rate control in wireless networks is still an open problem and is out of the scope of this paper. In general, online mechanisms (both end-to-end, e.g., sliding-window based [18], and hop-by-hop, e.g., backpressure based [19, 9]) or even offline computations [14] can be applied.[3]

**Isolating the benefit from new optimization techniques.** The evaluation of a new protocol that exploits a new optimization technique should try to isolate the gain from this "exotic" technique, alone. The tricky part here is that in adding a new optimization technique, a new protocol often incorporates other old techniques brought down to the rout-

[3]Interestingly, the importance of rate control has attracted significant interest in recent years in the theory community in the form of cross-layer optimizations (e.g. [15]).

ing layer from the upper layers, such as end-to-end reliability and rate control. To isolate the benefit of the new optimization, such techniques should be also incorporated in the traditional routing protocols. Similarly, comparing a reliable protocol against an unreliable one should be avoided; if the new protocol includes a mechanism for end-to-end reliability, a similar mechanism should be added to the old protocol.

**Separating rate control from end-to-end reliability.** When comparing a new reliable protocol to an unreliable one, the simplest method to add end-to-end reliability to the unreliable (traditional or not) routing protocol is to run it under TCP [19]. While this approach is simple, as no modification to the protocol itself is required, it may obscure the performance gain.

If the new protocol includes only reliability but no online congestion control (e.g., as is the case with FEC-style reliability), it is overkill to run the old protocol under TCP which includes both mechanisms which interact with each other. In this case, the throughput gap between the new and the old protocols may appear larger as a result of poor performance of TCP congestion control.

If the new protocol includes both reliability and online rate control (e.g., as is the case with ARQ-style reliability), it can be compared against the old protocol under TCP as a base-case comparison. Even so, since it is known that TCP performs poorly in wireless environments, it may still be unclear what the real gain from the new "exotic" technique is.

We advocate that in both cases, one should attempt to incorporate the reliability/rate control features of the new protocol to the old protocol, following the methodology of [19]. In this case, the comparison will be able to isolate the gain from the "exotic" technique exploited in the new protocol. We acknowledge this is not always easy to do. In some cases the reliability and congestion control mechanisms are disjoint components of the new protocol, not related to the new "exotic" technique used (e.g., in noCoCo). In this case reliability is typically provided in the traditional way (through retransmissions). This disjoint mechanism should be also incorporated to the old protocol used for comparison. In other cases, the reliability component of the new protocol may be part of the "exotic" technique itself (e.g., in MORE), and not a disjoint ARQ component. In such cases, the reliability component should be carefully added to the old protocol, for example, by adding FEC, and not by running it under TCP, so that the comparison is not affected by the negative effects of TCP's rate control mechanism.

**How to incorporate rate control to traditional routing?** Similar arguments against TCP apply here. If two unreliable protocols are compared, one with a rate control component and one without, running the second protocol under TCP is not a good solution, because the reliability mechanism is not required. What should be done is again incorporating the rate control mechanism of the new protocol to the old protocol. For example, in the evaluation of SOAR, the window-based rate control mechanism used in SOAR could be easily incorporated to traditional routing; in that case the comparison would isolate the gain of opportunistic forwarding.

**MAC autorate adaptation.** We argue that a good practice is for new "exotic" protocols to make an attempt to incorporate autorate adaptation. We acknowledge this is not an easy task and perhaps it is not always feasible. Even in those cases, we argue autorate adaptation should always be enabled for the case of traditional routing; an "exotic" protocol should be shown to outperform traditional routing both with and without autorate adaptation.

## 4. SUMMARY

In summary, we postulate that a fundamental reason for the complexity of evaluating high-throughput WMN routing protocols is that the research community still does not have a unified framework for understanding the interactions of MAC layer, congestion, interference, network coding, and reliability. WMN routing schemes are still being proposed as point solutions in a space of options; the real problem goes beyond how to evaluate them, but rather lies in how to understand the fundamental roles of their constituent parts.

## 5. REFERENCES

[1] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *Proc of ACM SIGCOMM*, 2005.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of ACM MobiCom*, 1998.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.

[4] D. S. J. D. Couto, D. Aguayo, J. C. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.

[5] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of IEEE INFOCOM*, 2000.

[6] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MobiCom*, 2004.

[7] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP throughput and loss. In *IEEE Infocom*, 2003.

[8] M. Ghaderi, D. Towsley, and J. Kurose. Reliability Gain of Network Coding in Lossy Wireless Networks . In *Proc. of IEEE INFOCOM*, 2008.

[9] C. Gkantsidis, W. Hu, P. Key, B. Radunovic, S. Gheorghiu, and P. Rodriguez. Multipath code casting for wireless mesh networks. In *Proc. of CoNEXT*, 2007.

[10] A. Gupta, Wormsbecker, and C. Williamson. Experimental evaluation of TCP performance in multi-hop wireless ad hoc networks. In *Proc. of MASCOTS*, 2004.

[11] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic, 1996.

[12] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, 2006.

[13] V. Kawadia and P. Kumar. Experimental investigations into TCP performance over wireless multihop networks. In *SIGCOMM E-WIND Workshop*, 2005.

[14] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, Z. Zhong, G. Deshpande, and E. Rozner. Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution. In *Proc. of HotNets-VI*, 2007.

[15] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks. In *Proc. of IEEE INFOCOM*, 2005.

[16] D. Lun, M. Medard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *Proc. of IWCT*, 2005.

[17] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of IEEE WMCSA*, 1999.

[18] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu. Simple opportunistic routing protocol for wireless mesh networks. In *Proc. of IEEE WiMesh*, 2006.

[19] B. Scheuermann, W. Hu, and J. Crowcroft. Near-optimal coordinated coding in wireless multihop networks. In *Proc. of CoNEXT*, 2007.

[20] Y. Yuan, H. Yang, S. H. Wong, S. Lu, and W. Arbaugh. ROMER: Resilient opportunistic mesh routing for wireless mesh networks. In *Proc. of IEEE WiMesh*, 2005.

# Wireless Networks Should Spread Spectrum Based On Demands [*]

Ramakrishna Gummadi Hari Balakrishnan
MIT CSAIL

## Abstract

Today's local-area, mesh and cellular networks assign a single narrow-band channel to a node, and this assignment remains fixed over long time scales. Using network traces, we show that the load within a network can vary significantly even over short time scales on the order of tens of seconds. Therefore, we make the case for allocating spectrum *on-demand* to nodes and regions of the network that need it. We present an architecture that shares the entire spectrum on-demand using spread-spectrum codes. If implemented, the system will achieve fine-grained spectrum allocation for bursty traffic without requiring inter-cell coordination. Preliminary experiments suggest a throughput improvement of 75% over commodity 802.11b networks. By eschewing the notion of channelization, and matching demand bursts with spectrum dynamically, better wireless networks that sustain higher throughputs may be designed.

## 1 Introduction

Wireless spectrum is a precious resource. The holy grail for the designers of wireless data networks is to maximize the aggregate network throughput within the frequency band alloted to the network. The current approach toward this goal is to first provision frequency bands to access points that form cells. Then, within each cell, a MAC protocol determines which nodes in the cell can transmit at any given time. Together, provisioning and scheduling attempt to ensure high spatial reuse (i.e., maximize the number of successful concurrent transmissions), thereby improving throughput.

In most current in-building or campus-wide wireless LANs, network administrators provision cellular resources over long time scales (weeks or months). Even in situations where access points (APs) are able to dynamically pick their operating channels from a wide-band selection, they pick a fixed-width channel in which to operate. The result is that an AP or a cellular base station uses a fixed chunk of the spectrum whenever it transmits, as does a client within a given cell. This fixed-width allocation causes significant throughput problems due to congestion on the wireless medium. Such problems have been identified at individual 802.11 hotspots [15] as well as at sites with multiple APs [11].

Fundamentally, a fixed spectrum allocation is sub-optimal because it does not track demand, which varies across different regions in the network and with time. Prior work has reported significantly varying demands at conferences [3], on campuses [12], and in enterprises [5]. For example, consider a conference hotel that runs multiple APs, each assigned a different channel to reduce interference. During the day, spectrum resources ought to be allocated to the APs in the conference rooms and away from the APs where there are few users. This strategy would achieve higher network throughput. The same argument applies to a typical office building, or to the wide-area cellular system during rush hour, or to disaster relief situations when many agencies and nodes all converge at a given location. 802.11 and cellular networks use a variety of techniques such as Dynamic Frequency Selection (DFS) [10] and cell breathing [2] to distribute load across cells. These technologies shift demand to cells that are lightly loaded, but weaken the received signal strength and limit throughput. In contrast, we advocate moving spectrum to cells that see higher demands.

We are not the first to recognize this fundamental shortcoming in existing wireless networks. A recent paper by Moscibroda et al. [14] makes the case for replacing the fixed-width channel of an AP and its client with a variable-width channel that is adjusted at ten-minute intervals in order to capture demand variations seen across APs. This proposal is sufficient if the observed demand at an AP is roughly constant over the channel-width update period. However, if the traffic is bursty, it can waste spectrum because nodes using narrow-width channels that have data to send cannot use the spectrum allocated to temporarily idle cells assigned broader-width channels. Decreasing the channel-width update period increases inter-AP coordination and can decrease stability and induce oscillations. In this paper, we focus on the problem of improving the throughput of networks with variable demands that are also highly bursty.

First, we present a trace study of wireless packet traces from the OSDI 2006 conference showing that demands can be both highly variable and bursty. Then, we argue that, for such networks, we should dispense with the notion of channelization. To demonstrate the validity of this viewpoint, we design and implement a direct-sequence spread-spectrum architecture called ODS (On-Demand Spectrum), in which every node uses the entire available spectrum. A node spreads its signals across the entire spectrum using a fixed-length pseudo-random spreading code. Such spreading decreases the received throughput compared to a non-spreading transmitter that uses the entire spectrum, even if the spectrum is large enough to accommodate the bandwidth expansion incurred by spreading. To compensate for this throughput loss, we allow a node to use more than one spreading code at

[*]This work was supported by the National Science Foundation under awards CNS-0721702 and CNS-0520032, and by Foxconn.

the same time and bond its transmissions. The exact number of codes a node uses simultaneously depends on its demand, as well as on the demands of other interfering nodes. If no interfering node has data to send, the node increases the number of spreading codes it uses to recover the throughput loss incurred by spreading. This policy decreases the effective spreading factor and simulates a non-spreading transmitter that uses the entire spectrum, not just a single fixed-width channel. If some interfering nodes have data to send, the node decreases the number of codes it uses by a corresponding amount.

ODS uses a random policy for selecting the pseudo-random spreading codes, and an adaptive receiver feedback mechanism to handle the challenging problem of fine-grained spectrum allocation without requiring excessive synchronization. Although the idea of using spread-spectrum to reduce synchronization while sharing spectrum is not new, what is new is the mechanism to regulate the number of spreading codes based on observed demand. In the process, ODS resolves the tension between the main appealing aspect of CSMA, which is that any node can send data on an entire channel without delay or coordination, and the chief benefit of channelization, which is that a network operator can limit interference across spatial regions.

We prototyped ODS using the USRP software radio platform. Our system allows us to transmit signals at 1 and 2 Mbps data rates, which are spread using a 11-chip pseudo-random spreading code similar to the Barker code used in 802.11b. Since the USRP is too slow to sustain the high data rates of the chipped spread-spectrum signals, we implement the spread-spectrum functionality within the USRP FPGA, and transmit only the low-rate data streams between the USRP and the host PC. Our implementation is preliminary and unoptimized. We compare the performance of ODS against commodity 802.11b radios that use a fixed Barker code in every device. Even when six transmissions interfere, we find that, where 802.11b achieves only 45% of the aggregate throughput without interference, ODS achieves 80% of aggregate throughput, an improvement of 75% over 802.11b. This improvement results from assigning spectrum using multiple codes based on demand, instead of using a single and fixed spreading code.

## 2 The Case for Spectrum On Demand

To demonstrate that traffic can be both highly variable and bursty, we present a small set of results obtained from over-the-air traces collected during the OSDI 2006 conference [4]. Several recent studies have examined the performance of 802.11 networks in hotspot settings extensively [11, 15], and found that such networks perform worse than expected under congestion because of contention-related losses and delays, as well as subsequent retransmissions and rate fallbacks. These studies also suggest that the best way to improve performance is to send smaller packets at faster rates.

In contrast, we posit that the primary contributor to such poor performance is the varying nature of the demand itself over both short (tens of seconds) and long (minutes to hours) time scales. From the OSDI packet traces, we were able to easily identify short time-periods (30-second intervals) in which the demand across various APs varied by more than a factor of 3.5. Further, we found that demands can change completely over long time scales of several minutes to hours when factors such as user movement and activity cause demand to be shifted to a different portion of the network.

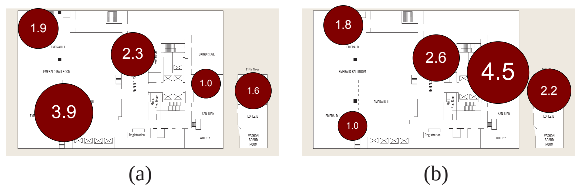


Figure 1: Relative demands across five APs during two consecutive 30-second intervals (the areas of the circles are proportional to APs' demands). Demand can thus be both widely variable and bursty.

Figure 1 shows the traffic demands at five APs over two consecutive 30-second intervals. An AP's demand is calculated as the amount of data originated by the AP within the 30-second period as determined by all sniffers that were able to observe the AP. Even though the APs were well-engineered in terms of orthogonal channel assignments and placements, Figure 1 shows that demands are highly variable and bursty. While Figure 1 shows only one data point of bursty demands, we have found that instances of such bursty patterns occur frequently in the trace. We leave a thorough quantification of demand variability to future work.

## 3 On-Demand Spectrum Architecture

ODS makes two major changes to the way spectrum is currently allocated and used:

1. ODS allocates spectrum to nodes dynamically based on their demands, and
2. ODS enables nodes to exploit concurrent transmissions by allocating multiple spreading codes to nodes.

ODS has three components. The first one is a mechanism that allows a receiver to estimate the future traffic demands of its transmitters so that it can allocate the spectrum across these transmitters. This mechanism works over short time scales of several packet transmissions. The second is a mechanism for receivers to decide how to assign spreading codes to transmitters according to the estimated demands. The third is a mechanism to ensure that concurrent transmissions using these codes can occur successfully, by allowing a transmitter to adaptively discover how many of its allocated codes can be successfully used before mutual interference due to concurrent transmissions decreases usable capacity.

We first describe the mechanisms that allow a receiver to determine the traffic demands of transmitters, and use them to estimate the transmitters' code allocation (§3.1). Because it is infeasible to coordinate receivers during code allocation, we propose and analyze the performance of a random code-selection policy that assigns a fixed-length pseudo-random number (PN) code sequences to transmitters in an uncoordinated manner. We fix the length of the PN codes at 11 chips, for 802.11b compatibility. We show that the random code-selection policy has good expected performance, while its best-case performance approaches that of the optimum centralized assignment to within a constant factor of $e$ (§3.2). Then, we describe how a transmitter uses all its codes concurrently (§3.3), and finally describe how transmitters adaptively detect when excessive concurrency turns into interference (§3.4).

## 3.1 Code Allocation

We assume that each node $n$ has some packet transmission demand of $d_n$ bits that must be transmitted as soon as possible, and can transmit at an average rate of $r_n$ bits/s. $r_n$ is the average bit-rate that the transmitter sees after rate-adaptation, which works at smaller time scales than demand scheduling. We assume that there are enough codes in the system, and that every node has access to at least one code by default. We assume that these codes can be decorrelated well at a receiver. The code availability and the decorrelation assumptions can be approximated in reality by using PN sequences that have low cross-correlation.

ODS allocates PN codes to transmitters in proportion to their demands. Demands are dictated both by the actual number of bits $d_n$ that a transmitter needs to send, and the average bit rate $r_n$ at which it can send them. Each receiver adaptively estimates these quantities on behalf of its transmitters, based on previously observed demands and rates of the transmitter. This estimation procedure is a simple moving average filter over a period of 30 seconds, which we found works well on the OSDI traces.

Once the receiver estimates its transmitters' demands, it assigns each transmitter $n$ a number of codes $c_n = c\left\lceil \frac{\frac{d_n}{r_n}}{\sum_i \frac{d_i}{r_i}} \right\rceil$, where $c$ is the codebook size, which is the total number of available codes. ODS uses a codebook size of $c = 128$ by default, which is large enough to utilize a 22 MHz-wide 802.11b spectrum fully. Further, this code assignment means that, assuming that the number of clients associated with an AP is not more than $c = 128$, every node gets at least one PN code (which is statically configured).

Interestingly, yet somewhat counter-intuitively, it follows from this formula that, given two transmitters with the same data load but different average bit rates, a receiver allocates more codes to the slower transmitter that to the faster transmitter, so as to increase concurrency, and improve the mean packet transmission time. Such a policy has fairness implications different from the status quo, and we defer a thorough study to future work.

Two potential issues in ODS are security (including various forms of Denial of Service concerns) and mobility. Since codes are ultimately allocated by the receiver, it is possible to enforce expressive policies for a transmitter's code allocation at the receiver. Further, since every receiver has access to at least one PN code, it is not possible for selfish nodes to completely deny network access, because spread-spectrum provides some jamming immunity as long as the statically assigned code is kept secret from the jammer. Small amounts of mobility do not pose serious problems to ODS because a receiver dynamically allocates codes to transmitters on a short-term basis, and because each node's statically assigned code is portable. However, continued mobility could cause problems, and we defer this problem to future work.

## 3.2 Code Selection

ODS uses an uncoordinated code assignment policy based on random selection of PN codes. Each receiver assigns a certain number of randomly chosen codes from a relatively large, but fixed, codebook of PN codes to each of its transmitters, without coordinating with other receivers. Conflicts may arise in such a receiver-driven code selection when two uncoordinated receivers allocate the same PN code to their transmitters. We assume that when two concurrent transmissions use the same code, they are both corrupted. Otherwise, both transmissions are correctly decoded. This is a pessimistic model because, depending on the received signal strengths, one or both transmissions might still be successfully decoded.

We now analyze the throughput of this code-selection policy. Let $k$ denote the number of randomly selected codes assigned to each transmitter $T$, and let $n$ denote the number of receivers around $T$. From the perspective of $T$, the expected number of conflict-free codes $\lambda$ it expects to be able to select is $\lambda = k(1-\frac{k}{c})^n$. The reason is that each of the $k$ codes has a probability of $1-\frac{k}{c}$ of not being in conflict with the code selected by any other receiver in $T$'s vicinity. Due to the independence selection property of codes and concurrent transmitters, this formula for $\lambda$ captures the expected number of conflict-free codes selected by this policy.

$\lambda$ represents the expected throughput achievable using conflict-free codes, not including the one code statically assigned to every node. In Figure 2, we plot the performance of $\lambda$ as we increase the number of codes $k$ allocated to each node. The number of available codes $c = 128$. Each curve in the plot represents the average throughput improvement seen by a transmitter using multiple codes over a transmitter using a single code, when other contending nodes also pick $k$ codes independently.

We show that random code-selection is both efficient and robust. For a given number of contending users $n$ and a given code size $c$, the per-node throughput $\lambda = k(1-\frac{k}{c})^n$ is optimized when $k_{\text{opt}} = \frac{c}{n+1}$, and is equal to $\lambda = \frac{c}{n+1}(\frac{n}{n+1})^n$. As we increase both the code size and the number of contending nodes keeping their ratio fixed, $\lambda$ asymptotically approaches $\frac{c}{ne}$. Thus, the optimum uncoordinated random code selection is within a constant factor of the optimum fully coordinated

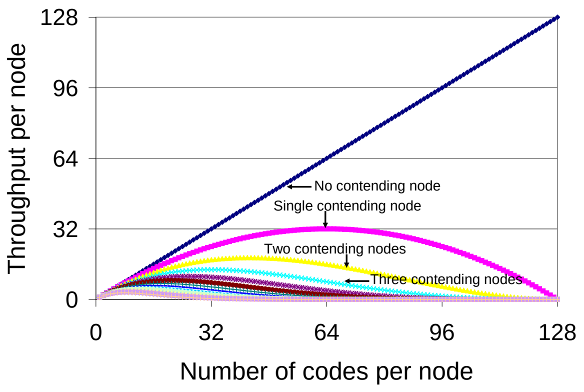


Figure 2: Per-node throughput under varying number of interferers.

strategy. Further, from the shape of the curves in Figure 2, it is apparent that the penalty for sub-optimal selection of $k$ is not severe as long as $k$ is chosen to be approximately equal to $k_{\text{opt}}$. Thus, random selection is robust to incorrect estimation of the number of contending users $n$.

### 3.3 Code Bonding

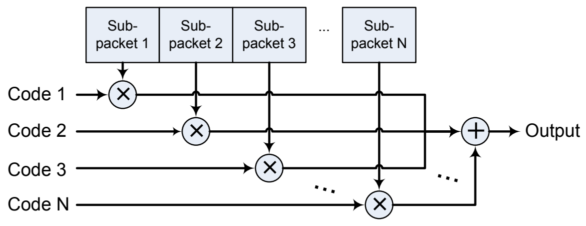


Figure 3: Illustration of bonding. Each sub-packet is spread using a different PN code, and all sub-packets are sent in parallel.

Bonding is a way of sending multiple sub-packets of a packet concurrently using separate PN codes for each of these sub-packets (Figure 3). The motivation is that, during intervals of low demand, a large portion of the entire spectrum can be allocated to a single node, which can then use it to speed up the overall packet transmission by first spreading the individual sub-packets with their own spreading codes and then multiplexing the coded signals onto the wide-band spectrum (Figure 3). Similarly, during intervals of high demand, fewer codes can be allocated per node, so that fewer sub-packets can be bonded; in the worst-case, every node uses only one code, so that there is no sub-packet bonding. The bonding limit is dictated by SINR considerations. As the number of sub-packets increase, the coding distance between two coded signals decreases, so it makes it harder for the receiver to accurately decorrelate the sub-packets in the presence of interference and noise.

### 3.4 Feedback-based Adaptation

ODS makes the entire spectrum usable by one or more nodes, so we aim to maximize concurrent transmissions, as long as they do not cause unacceptable interference to other concurrent transmissions. Statically deciding what the optimum amount of concurrency is for an arbitrary topology is an extremely challenging problem: if an active transmitter uses too few codes, spectrum is wasted, but if too many transmitters use too many codes concurrently, the achieved capacity is decreased because every code of every transmitter interferes with other transmitters.

To safely bound concurrency, ODS uses an adaptive mechanism that uses feedback from the receiver. A transmitter assumes that a coded transmission is lost due to mutual interference with some probability $p$. If it is correctly received, the transmitter has overestimated the interference from other concurrent transmitters, and so decreases $p$. If it is incorrectly received, the transmitter increases $p$. In the ideal case when the channel and other traffic are both invariant, the probability will converge to either 1 or 0, depending on the presence or absence of mutual interference. If the probability reaches 1, the transmitter decreases the code rate by dropping that code from its allocated code set and decreasing its coding rate. So, the transmitter's own performance improves because of the lowered coding rate, which also has the positive effect of simultaneously reducing the network-wide mutual interference levels. In case network conditions improve, the transmitter receives positive feedback about this fact from the receiver's decoder, which will see improved decoding performance. On the other hand, if conditions deteriorate, the transmitter will decrease the coding rate. We defer a careful study of protocol dynamics such as the adaptation rate and stability to future work.

## 4 Implementation and Evaluation

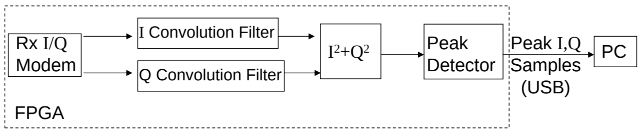


Figure 4: ODS PN-code despreader design.

We built an ODS prototype using the USRP hardware. Our main challenge was implementing high-rate coded samples. The current USRP is limited by the throughput of the USB bus to a throughput of 32 MB/s. Supporting an 802.11b-compliant spread-spectrum stream means, assuming 2 Mbps data and 11-chip codes, we must support $2 \times 11 \times 2 = 44$ Msps to satisfy Nyquist sampling. Since each sample is 16-bits, we need a throughput of 88 MB/s, which cannot be met. Instead, we implemented support for spreading and despreading the data in the FPGA on the USRP itself, so that only the actual data needs to be shipped across USB.

This design is shown in Figure 4, which shows the ODS-specific signal processing that is carried out on the FPGA for the receiver section; transmitter section is similar. The incoming $I, Q$ samples are decorrelated with the ODS-selected spreading code in the "Convolution Filter" blocks. We then sum the amplitudes of the filtered $I,Q$ samples and look for peaks in the summed signal. We output only these peak samples, which correspond to the decorrelated data values. Our implementation of random coding was based on the Barker receiver implementation provided by the Utah SPAN lab [6].

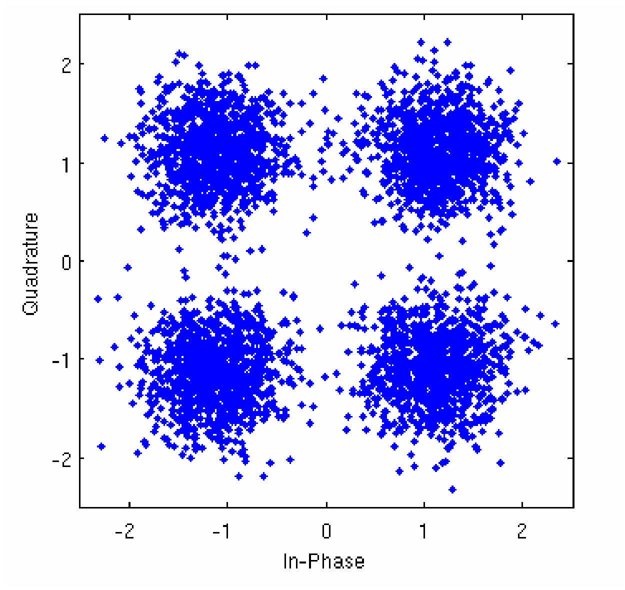


Figure 5: I,Q outputs of the PN-code despreader.

Figure 5 shows the decorrelated values of the $I,Q$ symbols as received on the PC for a 2Mbps DQPSK transmission. The symbols are clustered depending on what 2-bit data values they were modulated with. Thus, the spreading/despreading implementation provides satisfactory performance. We then do data demodulation, as well as ODS-specific processing, such as code allocation and multi-code bonding, on the host PC. Since the FPGA can only support only one PN-code despreading, we use multiple USRPs to implement bonding.

To test the end-to-end performance of ODS, we show the BER (bit-error rate) plots of the received data at varying SINR (signal-to-interference-plus noise) ratios with and without interference. We calibrate received and noise powers using a spectrum analyzer. Figure 6 shows the BER vs. SINR of a receiver with and without interference. Data can be received at relatively low SINRs because of the spread-spectrum processing gain. Further, the throughput does not degrade significantly with interference because the two concurrent transmissions use randomly selected PN codes.

To test ODS under different demands and levels of interference, we used a configuration with twelve nodes and six interfering links. We measured the throughput obtained on a link that could bond two 802.11b channels to obtain up to 4 Mbps without interference. We then increased the interference (and, hence, demands) on other links, and measured the bonded link's throughput when the number of interfering links is varied between 1 and 5. Our main finding is that, even in this high interference scenario and with a relatively unoptimized implementation, ODS could sustain up to 80% of the ideal throughput of 4 Mbps (i.e, it achieved 3.18 Mbps) across all six links, while, under similar conditions, 802.11b PRISM cards could only manage 1.8 Mbps in total. Thus, ODS improves 802.11b throughput by more than 75% by tolerating interference better under loaded conditions. We leave large-scale experiments to future work.

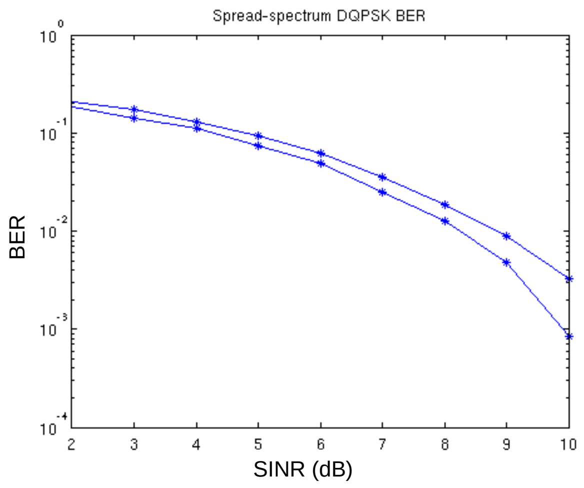


Figure 6: SINR vs. BER plots with and without interference.

## 5 Related Work

ODS uses spread-spectrum codes instead of frequencies. There are both pros and cons with this choice. While frequency-division multiplexing can provide orthogonality without causing mutual interference, it suffers from a significant drawback within our architectural context—codes can be finely divided and allocated, but fine-grained frequency division requires more sophisticated hardware than is currently available. For example, even though a current wireless chipset such as Atheros 5005GS can support variable-width channels of 5, 10 and 20 MHz [14] and bit-rates down to 0.25 Mbps, it still consumes a 5 MHz spectrum at a minimum to support the 0.25 Mbps rate. In contrast, commodity PRISM chips such as HSP3824 [9] provide a 16-chip spread-spectrum programmability. However, the advantage of using frequencies is that we can use much higher bit-rates with commodity cards (up to 54 Mbps with OFDM modulation used in 802.11a/g). In an accompanying paper [8], we exploit this high bit-rate facility along with variable-width channel support to study how much throughput improvements are obtainable with non-bursty, backlogged flows.

Spread-spectrum codes are used widely in cellular networks. For example, IS-95 voice networks and CDMA2000 data networks use spread-spectrum codes. However, these systems allocate a fixed amount of spectrum to a base station, and a heavily-loaded base station cannot borrow spectrum from its neighboring cells (which are on different channels, in order to mitigate co-channel interference). Instead, users are redirected to neighboring cells, which means the received signal is weaker than if spectrum were allocated locally based on demand. ODS can be applied to such cellular networks to handle bursty traffic. A heavily loaded ODS base station allocates more codes to its clients, while a lightly loaded base station apportions fewer codes.

CDMA has been proposed as a solution for resource multiplexing in a number of previous proposals for multi-hop wireless networks (e.g., [16]). The basic idea is that, by using spread-spectrum processing, geographically separated nodes can communicate concurrently in a much more efficient manner than CSMA would allow. Each node in the network is assigned a single code. Their main goal is not to deal with variable demands but to maintain communication links among neighbors under disruptions due to interference or mobility, by carefully reducing the transmit power of some radios in the network. In contrast, ODS does not alter the transmit power of nodes. Instead, it allocates more bonded codes to nodes that have higher demands, either because they actually have more data to send or because they are connected at lower bit rates than other nodes due to interference or noise. Our overall goal of satisfying short-term demands by allocating more spectrum to more demanding portions of the network is also different than the disruption-tolerance focus of these works.

At the link level, several frequency hopping strategies to mitigate interference have been proposed recently, e.g., SSCH [1] and MAXchop [13]. Some commercial APs also switch frequencies dynamically to minimize external interference. All these works still assume constant and equal internal demand at all nodes, while ODS allocates the entire spectrum based on fluctuating traffic demands at nodes.

## 6 Conclusions and Future Work

We made the case for handling bursty traffic better in wireless networks. We presented ODS, which achieves uncoordinated and fine-grained spectrum allocation based on observed demands. We found that ODS improves the throughput of interfering links by 75% over 802.11b under high interference conditions. Our preliminary results suggest that wireless networks can see significant throughput improvements by eschewing channelization completely, and instead by matching bursty demands with spectrum dynamically.

We plan to conduct a more thorough evaluation of ODS at higher bit-rates and larger topologies. More fundamentally, we would like to characterize the capacity and achievable rates of wireless networks with bursty traffic. Unlike the traditional notion of Shannon capacity that determines the fixed rates achievable by nodes, the instantaneous throughput with bursty traffic depends on the total received power, which in turn depends on the number of active transmitters [7]. So, we want to characterize this new notion of "bursty capacity" as a function of nodes' duty cycles and received powers.

**Acknowledgments.** We thank Kyle Jamieson for his contributions to earlier drafts of this paper, and the anonymous reviewers for their valuable feedback.

## References

[1] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *MobiCom'04*.

[2] P. Bahl, M. T. Hajiaghayi et al. Cell breathing in wireless LANs: Algorithms and evaluation. *IEEE TMC*, 6 (2), 2007.

[3] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *SIGMETRICS'02*.

[4] R. Chandra, R. Mahajan, and V. Padmanabhan. OSDI'06 traces. URL `ftp://ftp.research.microsoft.com/pub/ratul/OSDI2006-data/`.

[5] Y.-C. Cheng, J. Bellardo et al. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM'06*.

[6] Full-bandwidth 802.11b implementation, `http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.80211bReceiver`.

[7] R. Gallager. A perspective on multiaccess channels. *IEEE Transactions on Information Theory*, 31(2), Mar 1985.

[8] R. Gummadi, R. Patra, H. Balakrishnan, and E. Brewer. Interference avoidance and control. In *HotNets'08*.

[9] HSP3824 Direct-Sequence Spread-Spectrum Baseband Processor Data Sheet, `http://www.datasheetcatalog.org/datasheets2/28/283233_1.pdf`.

[10] IEEE 802.11h: Spectrum managed 802.11a (5 GHz) for European compatibility (2004), `http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&isnumber=27870&arnumber=1243739&punumber=8810`.

[11] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding link-layer behavior in highly congested IEEE 802.11b wireless networks. In *E-WIND'05*.

[12] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *MobiCom'02*.

[13] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly. Distributed channel management in uncoordinated wireless environments. In *MobiCom'06*.

[14] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, and P. Bahl. Load-aware spectrum distribution in wireless LANs. In *ICNP'08*.

[15] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *E-WIND'05*.

[16] T. J. Shepard. A channel access scheme for large dense packet radio networks. In *SIGCOMM'96*.

# An Architecture for Extensible Wireless LANs

Rohan Murty[†], Alec Wolman[‡], Jitendra Padhye[‡], and Matt Welsh[†]
[†]Harvard University, [‡]Microsoft Research

## 1 Introduction

Today's wireless LANs are a mess. The current 802.11 family of WLANs involves a jumble of competing standards, and a slew of implementations with varying degrees of interoperability and conformance to those standards. This situation has arisen in part from the need to innovate and evolve these networks over time, driven by new applications and increasing load. In the future we expect this situation to get worse, given the shift towards wireless as the dominant access network. Driving these changes are new devices such as Wi-Fi enabled VOIP handsets and audiovisual equipment, as well as new services such as Apple's Time Capsule which performs background backups over wireless. In the longer term, we anticipate WLANs will become the *default* access network and will support filesystem and server traffic as well.

Currently, it is not unusual for wireless LAN users to experience performance and reliability problems. A significant factor is the scarcity and poor utilization of the wireless spectrum, which suffers from a "tragedy of the commons". Scaling up WLANs to meet new traffic demands, especially time-critical applications involving audio, video, or sensor telemetry data, is proving to be difficult. This is in spite of underlying innovations at the PHY layer, which largely address the need for more throughput, but not how that throughput is managed. Moreover, enterprises often have an interest in imposing customized policies on WLAN traffic, for example, prioritizing time- and safety-critical traffic over large file downloads.

Existing wireless LANs make poor use of the wireless spectrum, largely due to the "intelligence" which is hard-coded into the vendor-specific software and firmware of wireless LAN clients. For example, WLAN clients control the decisions for AP associations, transmit power control, and physical data rate adaptation. The 802.11 standards specify the mechanisms, yet the policy is left entirely up to vendor-specific implementations. As a result, vendors view these areas as an opportunity to innovate and compete with each other. However, the end result of these attempts to innovate are limited, and we argue that is primarily an architectural limitation: by viewing the client as a stand-alone entity that solely uses local information about the devices it is interacting with, many important opportunities for improving the behavior of these algorithms cannot be realized.

The current approach to innovation and evolution in wireless LANs is primarily through standardization, which has resulted in an alphabet soup of protocols within the 802.11 family. Certain Wi-Fi vendors offer vendor-specific WLAN extensions such channel bonding, or non-standard data rates to support communication with weak signals. Such extensions only work when both the AP and the client are using the same brand of Wi-Fi chipset and software drivers, which prevents widespread adoption. The downside to standardization is primarily the glacial progress in deploying new protocols. The standards process takes a very long time to reach agreement, and even after standards are ratified it takes a long time to replace and/or upgrade the wide variety of client equipment utilizing the infrastructure.

We argue that to move away from the current mess, we need to rethink the basic architecture of wireless LANs. Our focus is not on changing the fundamental building blocks such as PHY-layer coding schemes or the CSMA nature of the MAC. Rather, we are interested in a developing an architecture that allows for extensibility, to ensure WLANs can adapt to meet future needs. We are guided by two key design principles:

- *Whatever we design today will be wrong in five years.* If history is any guide, we cannot anticipate all future uses for wireless LANs. Furthermore, the best way to evaluate innovations is through actual deployments with real users. With current WLANs, deploying new hardware and upgrading NICs and drivers for all of the affected clients is an expensive proposition, not to mention the management and personnel costs involved. We argue that an extensible WLAN can adapt to new uses, and can allow rapid deployment and evaluation of experimental designs.
- *Infrastructure should manage the wireless spectrum.* Networks can make the best use of resources by shifting much of the responsibility for managing the wireless spectrum (such as associations, power control, channel assignment, and physical layer rates) to the infrastructure, away from the individual clients. This has the additional benefit of making it easier to evolve the system because clients take much less of the responsibility for spectrum management. This approach also allows administrators to customize the network's policies for handling different traffic demands.

This paper describes *Trantor*[1], a new architecture for wireless LANs. Trantor's architecture is based on global management of channel resources, taking this responsibility explicitly away from clients and moving it into the infrastructure. To provide extensibility, the interface between the infrastructure and clients is simple and relatively low-level. Clients implement a small set of relatively simple commands which allows the complicated logic of the algorithms to exist primarily within the infrastructure. The commands fall into two categories: *measurement commands* allow the infrastructure to instruct clients to gather local information on channel conditions, such as RSSI from visible APs, and to report this information periodically; and *control commands* allow the infrastructure to control the behavior of clients, such as setting the transmit power or instructing a client to associate with a specific AP. Each client still implements a basic CSMA MAC for individual packet transmissions, but is otherwise not responsi-

[1]Named after the ruling planet of the first Galactic Empire as described by Isaac Asimov in the *Foundation series*.

| Proposed Standard | Year Proposed | Year Incorporated |
|---|---|---|
| 802.11m | 1999 | 2007 |
| 802.11d | 2001 | 2007 |
| 802.11h | 2003 | 2007 |
| 802.11k | 2003 | 2008 |
| 802.11r | 2004 | 2008 |
| 802.11T | 2004 | No |
| 802.11v | 2005 | No |

**Table 1**: **Proposed amendments to 802.11, dates of first task group meeting and dates of incorporation into the standard**

ble for most aspects of wireless management.

Trantor takes its cue from recent work on centralized management of resources in DenseAP [12], yet it pushes this approach much further by adding support for controlling physical data rates, transmission power, and clear-channel assessment. This approach can yield tremendous gains in efficiency, and also can provide a control point for the infrastructure to impose policies for shaping certain classes of traffic, prioritizing individual users, and so forth. Trantor's architecture is inherently evolvable to support new classes of applications, and supports global policy decisions to manage traffic load. Yet another key benefit of Trantor is that it allows the infrastructure to collect and use historical information (e.g. observations of client behavior over long time scales) to customize the behavior of a WLAN to the characteristics of its particular environment.

## 2 Background and Motivation

There is mounting evidence that wireless LAN architectures cannot keep up with demands of new classes of applications. Wireless LANs are already the most popular access network in homes and hotspots, and are rapidly becoming dominant within the enterprise. Apart from laptops, wireless interfaces are now found in a wide range of consumer devices, including smartphones, PDAs, media servers, set-top boxes, and network-attached storage appliances. Indeed, we anticipate that wireless LANs will dominate on desktops and even some servers, leading to increased channel load and new traffic patterns (such as filesystem traffic).

Industry is scrambling to respond to this challenge by introducing ever more sophisticated upgrades to the 802.11 standard, including 802.11e (QoS support), 802.11k (association protocol), 802.11d (regulatory domains), and many others. The need for interoperability between clients and APs, as well as backwards compatibility with previously-deployed technologies, mandates standardization which is an inherently slow process. Table 1 shows some of the 802.11 amendments currently in task group, along with the year the task group started. This situation has gotten to the point where we have "metastandards" designed to manage the documentation (802.11m) and testing (802.11T) of other standards. To illustrate the difficulty in deploying new improvements, consider the time from when the first security flaws in WEP were discovered to the present. Although today WPA and WPA2 have significant use, there are still a surprisingly large number of APs using WEP.

Another factor leading to the current situation is the heavy reliance on clients to make decisions about their use of the wireless spectrum. Even if such decentralized decision-making were optimal (which it is not), differences in vendors' implementations of the standards can lead to interoperability problems and inefficient use of the spectrum. One possible solution is to add more coordination to the client-AP interaction. For example, 802.11e provides support for QoS by allowing certain classes of traffic to be prioritized over others. However, this requires changes to both the clients and APs, slowing innovation. Ideally, we should be able to achieve the same goal without having to upgrade the client logic.

A third confounding factor is the commodity nature of 802.11 hardware, intended to drive down costs for NICs as well as home access points. A laptop must work equally well in a simplified 802.11b installation at home or in a sophisticated enterprise network comprising multiple standards. It is difficult to optimize an 802.11 implementation in such a milieu, requiring more and more complexity to be pushed into the OS drivers.

Our goal is to lift wireless LANs out of this quagmire of standards and develop a simple and extensible network architecture that supports: heterogeneous traffic demands; fine-grained control over network management; customized policies for traffic shaping and prioritization; and rapid innovation. The key idea is to strip most of the complexity away from the client and push it into the infrastructure, which we argue is better suited to managing the wireless spectrum and performing network management.

Some commercial efforts are a small step in this direction, although they are hampered by the need to maintain backwards compatibility with existing 802.11 networks. Cisco, Meru Networks, and others support intelligent radio resource management for enterprise WLANs, collecting measurements on interference, channel utilization, and other metrics to optimize network capacity. Research projects such as DenseAP [12] and MDG [6] have investigated approaches for managing 802.11 resources centrally to increase capacity. Similarly DIRAC [15] explored managing APs centrally. However, these systems are still limited by client-side behavior that may interact poorly with the goal of network-wide optimization. On the other hand, in [13] the author proposes equipping APs with analog-to-digital converters such that they are oblivious to the PHY/MAC layers being used at the client. As a result, all intelligence in the network is pushed to the clients.

Of course, stripping complexity from the clients is not without its challenges and potential pitfalls. One key question is how much complexity *can* be removed from the client without losing control and efficiency. At one extreme, *all* aspects of radio channel management of the client, including the PHY layer, modulation scheme, and the MAC, could be relegated to the infrastructure. However, in this work we decided to base our system on the core pieces of the existing 802.11 standards rather than pushing to this extreme. One reason is that this makes it much easier for us to implement and experiment with our architecture.

Nevertheless, we can experiment with many other aspects of wireless management, including channel assignment, AP associations, power levels, PHY rates, and channel bandwidths,

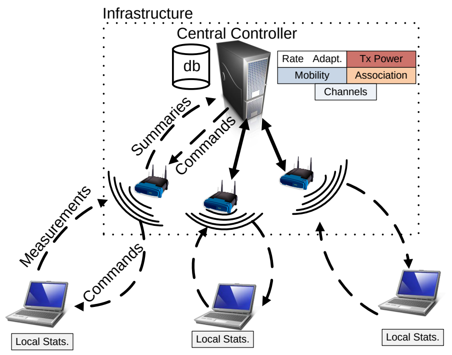


**Figure 1**: Proposed Trantor architecture. APs send summaries to the Central Controller (CC) about local information as well as responses from clients.

because these knobs can be tuned at coarser time scale. We also retain base 802.11 functionality (associations, authentication, etc.) to support legacy clients and hardware. Hence, Trantor builds on top of an underlying CSMA MAC and base 802.11 functionality.

Related to this issue is a key question: at *what point* should one stop modifying the client? Is it at the application layer, the MAC, the PHY, etc.? One approach is to realize the SDR vision by using an analog-to-digital converter at the client which downloads the entire PHY and MAC layers from the infrastructure. This is an extreme design point as it offers the highest degree of extensibility. However, moving to a complete SDR side steps the issue of what functionality should be implemented at the clients and what responsibility should be given to the infrastructure? For example, should clients continue to decide when to transmit and which modulation schemes to use? At the heart of these issues is a debate on the ideal separation of functionality between the client and the infrastructure, given that we want to be able to enable rapid innovation in the wireless network without sacrificing performance. For the rest of this paper we address this debate in the context of 802.11 only but as part of our ongoing and future work we are also actively exploring how PHY/MAC layers weigh in on these questions.

The second key challenge is determining how much information the client should collect and report to the infrastructure to assist in management decisions. Given complex and dynamic environments exhibiting interference, multipath, and node mobility, the number of variables that affect an individual client's link quality can be extremely large. Ideally, each client could report measurements on the observed channel occupancy, RSSI from multiple APs (and other clients), and variation in channel conditions over time. However, collecting this information could involve high overheads and interact poorly with power-saving measures at the client.

In the following sections, we outline the Trantor architecture and describe our approach to these challenges.

## 3 Trantor Architecture

In this section we first outline the Trantor architecture and then describe the various management aspects of the system.

In a typical WLAN architecture (such as Aruba [1] or Meru [2]) deployed in the enterprise, the infrastructure consists of APs managed by a central controller (CC). In this context, management entails channel assignment and transmit power control. A controller manages the channel and transmit power for each AP. Clients on the other hand make many decisions independent of the infrastructure. In some small networks, there is no CC and each AP acts independently [3].

The Trantor architecture is illustrated in Figure 1. The main difference between current architectures and Trantor is that various decisions clients currently make have been shifted to the CC. In Trantor, clients receive commands, collect measurements, and report back to the infrastructure. The CC manages all APs and clients in the system. Each AP periodically sends summaries consisting of its own local measurements as well as the local measurements collected by each associated client. Therefore, the CC receives information from every wireless node in the system, be it an AP or a client. The CC does not receive reports from a client not associated with the network.

Based on the summaries received from APs, the CC executes various algorithms and can do one of the following: (i) send a command to an AP to execute a particular action (for example, change channels, disassociate a particular client, etc.), or (ii) send a command to a client (via an AP). Later in this section we elaborate on the commands the CC can send to the client. Note that this design does not require true centralization: a "logical" central controller can be implemented in a decentralized manner across all the APs in the system. The key point however is that most of the intelligence resides in the infrastructure and not with the clients.

This architecture is extensible because policy changes can be easily introduced into the system at the CC. Since clients and APs report summaries to the CC, the infrastructure also has global knowledge of the system. The CC also utilizes a database to store received summaries to main historical knowledge of the system.

Trantor is intended for an enterprise environment where there is one administrative domain that manages all APs. Academia and industry have examined centralizing the data plane and certain decisions such as channel assignments. DenseAP [12] examined centralizing associations. Trantor pushes this further by building an extensible architecture where those decisions that can benefit from global and historical knowledge have been moved from clients to the infrastructure. This approach distinguishes Trantor from prior work. In the future, as wireless networks evolve and potentially newer decision-making aspects of clients are introduced, we envision moving more such decisions to the infrastructure. However, note that decisions such as going into power saving mode do not necessarily benefit from global knowledge and hence are retained at clients. Also, in this paper, we only focus on infrastructure mode in clients and not on ad-hoc mode since the former remains the dominant use of wireless networks in the enterprise.

We now describe the design of clients and the infrastructure in the system.

### 3.1 Client Design

Clients (and APs) are dumb agents controlled by the infrastructure. Table 2 outlines the various commands the CC sends to APs and clients in the system. Most commands are common to

| | |
|---|---|
| $ListNodes()$ | Report list of $< mac, rssi >$ heard |
| $ReportLoss(n)$ | Report loss-rate when sending to $n$ |
| $ReportReTrans(n)$ | # of retransmissions when sending to $n$ |
| $ReportAirTime(n)$ | Report air-time utilization |
| $TxPackets(x, s, n)$ | send $x$ packets, each of $s$ bytes to $n$ |
| $Associate(ap)$ | Associate with AP $ap$ |
| $SetTxLevel(p)$ | Set transmit power to $p$ |
| $SetCCA(t)$ | Set CCA threshold to $t$ |
| $SetRate(r)$ | Transmit all future packets at rate $r$ |
| **AcceptClient** $(c)$ | AP lets client $c$ associate with it |
| **Handoff** $(ap, c)$ | Handoff client c to AP $ap$ |
| **EjectClient** $(c)$ | Disassociate $c$ from the network |

Table 2: **Sample set of commands the CC sends to clients and APs. $n$ is a wireless node. Commands in bold are applicable to APs only.**

both APs and clients. We first focus on how these commands are used by the CC when dealing with clients.

**Collecting Measurements:** The infrastructure can use the commands listed in Table 2 to estimate packet losses, retransmissions, RSSI of packets from APs, other clients in the vicinity, and channel utilization, all as seen by a client. Our working hypothesis is that such information is fundamental to all macro-level decisions such as associations, handoffs, power control, and rate-adaptation [6, 12]. Clients collect measurements over a measurement window $w$, which is selected by the infrastructure. $w$ may be changed over time to permit finer- or coarser-grained measurements from each client.

A challenge for the infrastructure is to normalize measurements reported by different clients which may be using different radio chipsets. For example, raw RSSI values reported may vary across clients due to variance in receiver sensitivity.

**Active Probing:** Using the $TxPacket$ command, the CC can instruct clients to perform active measurements. Active probes can be used to directly ascertain link quality, congestion, and other conditions that can be difficult to derive from passive measurements alone. They can also assist in diagnosing performance issues in the network; we discuss this further in Section 4.

Measurements collection and active probing by clients are unique to the Trantor architecture and represent two fundamental primitives to support extensibility. By combining these mechanisms, the CC can collect detailed measurements of the network state and factors that affect client performance, such as traffic patterns, interference, and channel congestion. Such an approach can potentially reduce the need for a dedicated wireless monitoring infrastructure [8, 5].

Collecting this information from clients achieves three key goals of the Trantor architecture. (i) the infrastructure can optimize the overall performance observed by clients in the network, by tuning many aspects of individual clients' use of the radio channel. (ii) the infrastructure can impose policies to manage competing uses of the radio channel. (iii) Trantor can automatically diagnose and remedy performance problems through centralized observation and control.

We briefly present two example uses of the measurements collected by the infrastructure.

- *Conflict graph construction:* Using information collected from clients, Trantor constructs a conflict graph of the set of clients currently interfering with each other on the same channel, whether or not those clients are currently associated with the same AP [4]. This information can be used to mitigate interference by tuning channel assignments and transmission power control of individual clients.
- *Active AP selection:* Trantor can leverage active probing measurements between clients and APs to optimize client-AP associations. If the loss-rate between a client and its AP rises above a given threshold, rather than relying strictly on client RSSI measurements (as is currently done), the CC initiates active probing between the client and multiple nearby APs to determine the best association.

## 3.2 Infrastructure Design

In Trantor, the infrastructure bears the additional responsibilities of managing client-AP associations, channel assignment, power control, and rate-adaptation.

There is a mutual interdependence between these various aspects of wireless channel management. For example, managing associations affects the the number of clients on a given channel (since clients are assigned to APs fixed on a single channel) which in turn has the potential to increase interference. Reducing transmit power levels of interfering nodes can mitigate this problem, but it also affects the reception rate for a given data rate (since the probability of successfully decoding a packet for a data rate is determined by a SNR threshold). Performing a joint optimization of these decisions is a non-trivial problem. However, in Trantor since the infrastructure has global and historical knowledge of the performance of the wireless network as well as control over client behavior, it has the potential to address this problem. This is an aspect of the system we are actively exploring. Prior WLAN architecture proposals have lacked such information and hence it has been harder for them to address this problem. We briefly describe possible techniques to address these management decisions.

**Client-AP Associations:** Using $Associate(ap)$, the CC has the ability to control which AP a client can associate with. It can also use **AcceptClient** and **RejectClient** to prevent a client from associating with an AP. As prior work [6, 12] has shown, client-AP association decisions must take into account load at the AP as well as the quality of the client-AP connection. As mentioned earlier, Trantor can leverage active probing measurements for this purpose. Furthermore, historical information can also help improve association decisions. For example, prior work [7] has observed clients in certain locations in an office building in spite of receiving strong signals from an AP, experience heavy packet losses due to a poor wireless channel. Such information can be used to quickly converge on a client-AP association decision.

**Transmit Power Control:** Prior work [6] has shown how coordinated power control can lead to an increase in overall network capacity. We adopt a similar approach. The CC also has control over each client and AP's CCA threshold since it is required to set the appropriate power level at these nodes.

**Rate Adaptation:** Prior work on rate-adaptation has focused on clients adjusting rates based on local information such as packet loss or RSSI of received packets. Packet losses at a client commonly occurs due to one of the following reasons: (i) collisions caused by hidden terminals, (ii) local channel noise. The remedy to (i) is to increase or fix the current data rate. The remedy to (ii) is to lower the current rate in order to improve the SNR of the signal. Hence, it is important to distinguish between the two cases when determining the next course of action for rate adaptation. Most prior work in this space suffer from the lack of additional information that can help distinguish between these two cases.

Prior work has shown that some cooperation between clients and the infrastructure can help a client pick better rates [10]. In Trantor, the availability of global and historical knowledge can facilitate rate adaptation further. We argue data rates must be adjusted based on a longer term view of the network rather than just the recent few packets. Hence, based on reports from nearby APs and clients (global knowledge) and observing the behavior of the network over long periods of time (historical knowledge), the CC can potentially ascertain the reason behind significant packet losses in the network [8]. Based on the measurements received from APs and clients, the CC constructs a conflict graph and uses a probabilistic analysis to determine if an AP or client is experiencing loss due to a hidden terminal like problem or due to channel noise. Using this analysis it instructs each node precisely which data-rate to transmit at. A node continues to transmit at the same rate until its told to change its transmission rate by the CC.

There is a tradeoff between using global knowledge for centralized rate adaptation and the timescale over which this can be performed for each wireless node. A slow rate adaptation can result in an AP or client temporarily experiencing poor performance. Our working hypothesis is that nodes do need to change data rates but *not* as often as prior work has come to expect. In other words, we do not expect the wireless medium to be choppy on a sustained basis and therefore we prefer choosing a “correct” rate slowly than an “incorrect” rate quickly.

**Mobility:** Since the infrastructure handles associations it is must also handle mobility. While a client can be instructed to explicitly associate with a different AP (hence the actual cost of the handoff is negligible), delays might be incurred by the infrastructure in gathering measurements, analyzing them, and determining if a handoff should take place. This is where historical knowledge of the wireless network is key for improving performance. In an office deployment, clients typically move along corridors or hallways. The infrastructure can observe such patterns and predict the trajectory that a client will take. This can help reduce the time taken to determine when a client must switch APs. To enable handoffs, the CC uses $Associate$ to inform the client to switch associations and **Handoff** to inform the source AP to send the association state and buffered packets to the destination AP.

**Classifying Clients:** The ability to offer differentiated services to clients is a key gain the Trantor architecture has to offer. To achieve this the system must be able to quickly classify clients based on their traffic. Such classification is important because it impacts the association and handoff decisions. For example, VOIP clients tend to suffer when contending with bulk transfer clients for the same part of the spectrum. Therefore, in Trantor, we can associate clients to different APs on different channels based on their traffic classification. This entails clustering VOIP clients together when performing associations or handoffs. Furthermore, it is also important the system classifies a client quickly since this can impact the handoff latency. We address the classification problem using a lightweight technique whereby each AP monitors the flow of packets to/from a client. Using a technique similar to one proposed in [11], observing a few samples of packet size, port number, and inter-packet arrival time, the AP classifies the client’s traffic as (i) latency sensitive or (ii) a bulk transfer. However, there can be cases when a client simultaneously starts a Skype call (VOIP) and also begins a file download. We currently classify such clients as being bulk transfer agents.

## 4 Benefits of Trantor

In this section we discuss several tangible benefits that the Trantor architecture provides.

**Traffic differentiation:** Trantor permits the network administrator to impose local policies on the network to prioritize certain clients over others based on their traffic. For example, hospital environments may want to prioritize data from hospital instruments over standard WiFi usage, and companies may want to limit large media downloads during the day. This could entail various approaches such as grouping client associations based on their traffic type or rate limiting certain clients more than others.

**Site-specific policies:** A direct consequence of extensibility is the ability to customize the behavior and performance of the wireless network based on the context. Prior work has shown wireless traffic patterns fluctuate by time of day as well as location [14]. For example, a large auditorium or conference room might experience heavy spikes of traffic congestion during meetings, whereas dormitories may experience heavier loads at night. To deal with such situations, the infrastructure needs to dynamically provision the spectrum based on the client traffic mix. We present two example policies and briefly describe how they impact the various decisions made by the infrastructure.

- *All APs on the corridor in the 2nd floor must prioritize VoIP traffic over other kinds of traffic between 9 a.m. and 4 p.m..* Because VoIP clients are sensitive to packet losses and jitter, we want them to be able to use lower data rates and at the same time not have to contend for the medium with other clients performing bulk transfers. Hence, the infrastructure would impose an association policy that prioritizes associations and handoffs from VoIP clients, assigns higher data rates and power levels to VoIP clients, and hands off mobile clients more aggressively.
- *Based on the number of web clients on the 1st floor, the first floor wireless network must devote X% of APs to interactive traffic.* In this case the infrastructure would enforce an association policy that only permits interactive traffic clients to

use certain APs. This policy could also entail more aggressive rate-adaptation.

**Fault Diagnosis:** Based on client measurements reports, the infrastructure can detect, resolve, or at least shed more light on performance anomalies and outages in the network. Two typical examples of such diagnosis are as follows.

- *Rate Anomaly*: The rate anomaly problem arises due to the "worst client" impacting the performance of other wireless clients in the vicinity, and prior work has shown this can significantly reduce WLAN capacity [9]. In Trantor, because the infrastructure controls the transmission rate for each client, it can now detect such rate imbalance situations and either increase the data-rate for the offending client or change its association to a different AP.
- *Reasoning about client losses*: Using global and historical knowledge, the infrastructure can attempt to ascertain why a client experiences significant losses. This impacts rate adaptation as well as transmit power (at the APs). However, persistent losses (despite rate and power changes) could be used to diagnose whether a particular area of the building suffers from poor channel conditions.

## 5 Discussion and Conclusions

We present Trantor, an extensible architecture for WLANs. The fundamental tenet of the Trantor architecture is to move wireless management decisions from clients into the infrastructure when such decisions can benefit from global and/or historical knowledge. As part of this we proposed centralizing various wireless management aspects. Trantor is also able to provide better customization of a WLAN according to an environment. We now outline some key challenges we plan to investigate related to the Trantor architecture.

**Dealing with malicious clients**: It is relatively easy to detect violations where a client does not follow the infrastructure's instructions. For example, based on reports from APs and other clients, it is easy to detect such violations and disassociate the offending client from the network. However, it is a much harder problem to determine if a malicious (or faulty) client is sending spurious reports. One potential way to address this problem is to verify such reports with reports from other APs and clients in the neighborhood, but this remains an open issue.

**Scalability:** The scalability of the Trantor infrastructure depends on a host of factors including: the rate at which each client is polled for measurements; the size of the measurements reports; and the ability of the CC to make quick decisions during handoffs and change rates quickly when necessary. To be effective in enterprise settings, the central controller must be designed to handle a large network consisting of thousands of APs and clients. One strategy to scale gracefully is to use a zoned approach in which separate controllers are assigned to distinct physical zones in the network (such as different buildings, or floors of a building), with the assumption that limited sharing is required across zone controllers to make effective network management decisions.

**Presence of other interfering networks**: An argument for clients retaining their decision making abilities is for them to react to interference from other competing wireless networks in the vicinity. However, since clients are always reporting measurements to the infrastructure, such events can be dealt with effectively in our proposed infrastructure as well. The infrastructure can profile which other competing networks are operating in the vicinity and use this information when determining policies for clients.

**Security:** Trantor's extensibility can make it easier to deploy new security mechanisms. This is critical to the operation of a wireless network because in the event a security mechanism is found to be flawed (WEP, for example), without depending on new standards to be adopted, it is easy to implement and push out new security mechanisms quickly in Trantor. For example, WPA2 was proposed as a replacement for WEP via 802.11i and it did not require hardware changes. Such updates can be easily rolled in Trantor but it would require expanding the interface listed in Table 2.

**Responsiveness**: One open question is whether the clients need to adapt their behavior more rapidly than can be easily accommodated by the Trantor architecture, with its cycle of collecting data, analyzing it centrally, and then sending out commands to cause the clients to adapt.

## References

[1] Enterprise solutions from aruba networks, http://www.arubanetworks.com/solutions/enterprise.php.

[2] Meru networks, http://www.merunetworks.com.

[3] Xirrus, http://www.xirrus.com/products/.

[4] N. Ahmed and S. Keshav. SMARTA: A Self-Managing Architecture for Thin Access Points. In *CoNEXT*, 2006.

[5] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. Enhancing the Security of Corporate Wi-Fi Networks Using DAIR. In *MobiSys*, 2006.

[6] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-driven Guidelines for 802.11 WLAN Design. In *MobiCom*, 2007.

[7] R. Chandra, J. Padhye, A. Wolman, and B. Zill. A Location-based Management System for Enterprise Wireless LANs. In *NSDI*, 2007.

[8] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benko, J. Chiang, A. C. Snoeren, G. M. Voelker, and S. Savage. Automated Cross-Layer Diagnosis of Enterprise Wireless Networks. In *SIGCOMM*, 2007.

[9] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Infocom*, 2003.

[10] G. Judd, X. Wang, and P. Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *MobiSys*, Uppsala, Sweden, 2008.

[11] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS*, 2005.

[12] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing High-Performance Enterprise Wireless Networks. In *NSDI*, San Francisco, CA, April 2008.

[13] S. Singh. Challenges: Wide-Area wireless NETworks (WANETs). In *MOBICOM*, 2008.

[14] D. Tang and M. Baker. Analysis of a Local-Area Wireless Network. In *MobiCom*, Boston, MA, August 2000.

[15] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, and J. J.-R. L. DIRAC: a software-based wireless router system. In *MOBICOM*, 2003.

# Breaking Up the Transport Logjam

Bryan Ford
Massachusetts Institute of Technology[*]
baford@mit.edu

Janardhan Iyengar
Franklin & Marshall College
jiyengar@fandm.edu

## ABSTRACT

Current Internet transports conflate transport semantics with endpoint addressing and flow regulation, creating roadblocks to Internet evolution that we propose to address with a new layering model. Factoring endpoint addressing (port numbers) into a separate *Endpoint Layer* permits incremental rollout of new or improved transports at OS or application level, enables transport-oblivious firewall/NAT traversal, improves transport negotiation efficiency, and simplifies endpoint address space administration. Factoring congestion control into a separate *Flow Layer* cleanly enables in-path performance optimizations such as on satellite or wireless links, permits incremental rollout of new congestion control schemes within administrative domains, frees congestion control evolution from the yoke of "TCP-friendliness," and facilitates multihoming and multipath communication. Though this architecture is ambitious, existing protocols can act as starting points for the new layers—UDP or UDP-Lite for the Endpoint Layer, and Congestion Manager or DCCP for the Flow Layer—providing both immediate deployability and a sound basis for long-term evolution.

## 1. INTRODUCTION

Typical transport protocols combine several functions, such as identifying application endpoints via port numbers [38,49], providing end-to-end congestion control [27], utilizing alternate communication paths [33,46], and implementing reliable/ordered communication [37,46,49]. Lumping these functions into one layer has made the transport layer brittle and difficult to evolve, however, by preventing evolution of individual transport functions without affecting *the entire transport layer*. Since firewalls and NATs [45] must understand transport headers to extract port numbers, for example, new transports [28, 46] are almost undeployable because they cannot pass through existing middleboxes. Similarly, new congestion control schemes [20] and performance enhancing proxies [11] cannot be deployed on specific segments of a communication path without breaking end-to-end semantics [41] and fate-sharing properties [16].

To remove these evolutionary roadblocks, we propose splitting the Transport Layer into (at least) three separate layers, shown in Figure 1. We factor out the function of identifying logical communication endpoints—traditionally represented as 16-bit port numbers—into an *Endpoint Layer* protocol to be shared among transports. We factor out congestion control and other performance-related mechanisms into a separate *Flow Regulation Layer*, or simply *Flow Layer*. The services remaining in the Transport Layer are limited to providing the end-to-end communication semantics needed by higher-level layers, such as reliability, ordering, and error recovery.

[*]Now at the Max Planck Institute for Software Systems (MPI-SWS)

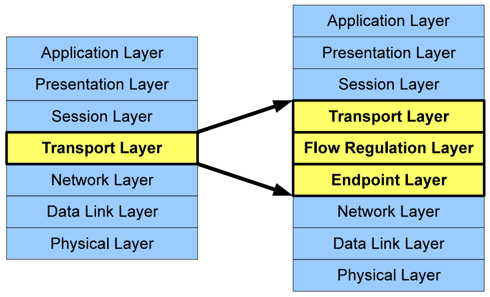


**Figure 1: Breaking up the Transport Layer**

In contrast with prior work that factored out these functions for specific technical reasons [6, 18, 28, 50], our focus is on identifying and addressing *evolutionary* impediments to Transport Layer development. Our primary contribution is a new architectural model that better facilitates evolution, and that places a variety of existing, often mutually exclusive "transport hacks" into a clean and interoperable framework.

Section 2 details the purpose, architecture, and practical implications of our Endpoint Layer, and Section 3 similarly details our Flow Regulation Layer. Section 4 outlines issues in implementing and further evolving the Endpoint, Flow, and Transport layers, and Section 5 concludes.

## 2. THE ENDPOINT LAYER

Our first modification to the classic Internet architecture is separating the function of identifying *logical endpoints* or *ports* out of transport protocols and into a common underlying *Endpoint Layer*. We view the Endpoint Layer as an extension to the Network Layer: where the Network Layer provides *inter-host addressing and routing* via IP addresses, the Endpoint Layer provides *intra-host addressing and routing* via port numbers. The Endpoint Layer does not otherwise affect the underlying best-effort delivery service: higher layers are responsible for congestion control, ordering, and reliability. All higher layers ideally reside atop a single Endpoint protocol, sharing one endpoint address space per host.

Our insight is that building new transports atop a common Endpoint Layer, instead of atop IP as in the current model, may facilitate flexibility and protocol evolution in several ways. We first address architectural foundations, followed by practical benefits of the proposed model. We leave Endpoint Layer implementation issues to Section 4, which proposes reusing UDP [38] as an already widely supported "Endpoint Layer protocol," and then suggests paths toward richer functionality.

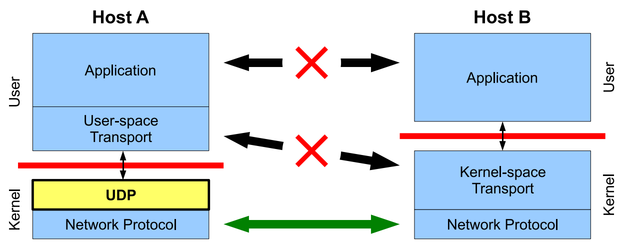


**Figure 2: A UDP-based user-space transport *cannot* interoperate with a "native" IP-based kernel-space transport.**

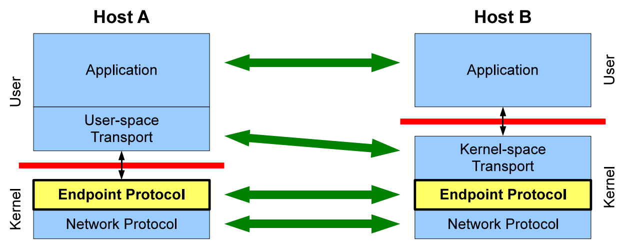


**Figure 3: A UDP-based user-space transport interoperates with a UDP-based kernel-space transport.**

## 2.1 Architectural Perspective

All standard Internet transports [28, 37, 38, 46, 49] multiplex transport sessions onto a host's few IP address(es) via 16-bit port numbers. Each transport implements this multiplexing separately and embeds port numbers in its own transport header, making port numbers a common design pattern but not a shared facility. Nevertheless, each port space tends to be functionally equivalent; the IANA historically assigns well-known ports consistently across transports although the port spaces are technically independent.

Embedding port numbers into transports is consistent with the OSI reference model [56], where each layer provides its own space of *service access points* (SAPs) for higher layers to bind to: IP addresses correspond to *Network-SAPs* (NSAPs), port numbers to *Transport-SAPs* (TSAPs), and OSI additionally has *Session-SAPs* and *Presentation-SAPs*. The full "identity" of an endpoint consists of the SAPs of all layers bundled together: IP address and port number on the Internet, all four SAPs in OSI. This *layered multiplexing* design has appeal but causes known problems: Tennenhouse argued against it due to the difficulty of real-time scheduling across layers [50], and Feldmeier elaborated on several related issues [18].

An alternative approach is to treat the intra-host addressing provided by port numbers or SAPs as an extension to the inter-host addressing already provided by the Network Layer, and implement this intra-host addressing *once* in a facility shared by all higher layers. In Sirpent [14], intra-host addresses (port numbers) are part of the source routes the network layer uses for routing. An analogous design with CIDR addressing would be to assign each physical host or network interface a whole "virtual subnet" of addresses representing the logical endpoints on that physical host. It may be too late to merge port numbers into IP addresses, but our Endpoint Layer revisits the idea of *sharing* one endpoint space among upper-level protocols instead of each transport implementing its own.

## 2.2 Practical Benefits

Independent of the concerns of Tennenhouse and Feldmeier, factoring out endpoint multiplexing brings several practical benefits that are relevant today: transport implementation flexibility, firewall/NAT traversal, and transport protocol negotiation.

### *2.2.1 Transport Implementation Flexibility*

The IP header's 8-bit Protocol field was intended to distinguish between only a few standard transport protocols, not between many application-level endpoints, so most operating systems prohibit unprivileged applications from "hooking" IP Protocol numbers in the way they can allocate and use ports. The OS thus reserves the right to implement new "first-class" transports. If an application wishes to deploy its own transport protocol that is not yet supported by the host OS, it must layer the new transport atop UDP. The resulting application-level transport not only has second-class status but is *unable to interoperate* with a first-class OS-level implementation of the same transport on another host, as shown in Figure 2. This restriction creates a barrier to the deployment of new transports, since the easiest way to deploy new protocols incrementally is often to bundle them with the applications that need them.

If new transports are built atop an Endpoint Layer, however, applications can easily ship with new transports implemented in user-space libraries requiring no special privilege. Once a transport begins migrating into OS kernels, kernel-level and user-level implementations of the same transport can remain interoperable, as shown in Figure 3.

### *2.2.2 Transport-Independent Middlebox Traversal*

For better or worse, *middleboxes* such as firewalls and network address translators (NATs) are now ubiquitous, and most of them are sensitive to the full endpoints of a given flow: not only IP addresses but port numbers as well. Since each transport traditionally implements its own port space, middleboxes must parse transport headers, and so only the few already-ubiquitous transports—TCP and UDP—can traverse most middleboxes. New transports like SCTP [46] and DCCP [28] that are designed to run directly atop IP thus cannot traverse most middleboxes. NAT proliferation has in effect shifted the Internet's "narrow waist"—the ubiquitous interface atop which new protocols may be built and reliably deployed—upward to encompass not just IP but also TCP and UDP [40].

By building new transports atop a shared Endpoint Layer, middleboxes need to understand only Endpoint Layer and not Transport Layer headers. Middleboxes can still recognize and optimize the handling of specific transport protocols if desired, but doing so is no longer a *prerequisite* for traversal. The Endpoint Layer also provides a clean space for mechanisms allowing hosts to "advertise" endpoints intended to be publicly reachable, enabling middleboxes to create persistent bindings for them as policy permits—a demand currently met via ad hoc, transport-specific mechanisms such as those in UPnP [53].

### *2.2.3 Negotiation of Alternative Transports*

Many application protocols such as RPC and SIP can use several alternative transports. Every application packet is traditionally associated with *exactly one* transport protocol, how-

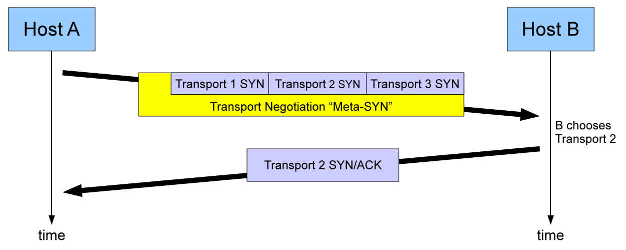


**Figure 4: Applications can negotiate among several UDP-based transports with no extra round trips.**

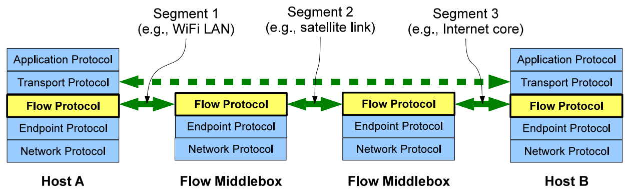


**Figure 5: An end-to-end path composed of multiple Flow Layer segments. Flow middleboxes can optimize network performance based on the properties of a specific segment, such as a satellite link.**

ever, via the IP header's Protocol field. Negotiating *which* transport to use for a communication session therefore requires the initiating application either to use a special transport exchange just for this negotiation, or to open new sessions "speculatively" for each supported transport, only to continue using the most preferred one that succeeds and shut down the rest.

Building transports atop a shared Endpoint Layer with one port space, in contrast, leaves transport identification and negotiation under the application's control. The Internet already follows this design philosophy for Session Layer and Presentation Layer functions, leaving their negotiation up to applications (e.g., HTTP's persistent streams and content encodings); our architecture extends this flexibility to the Transport Layer.

Without prescribing specific mechanisms, we suggest one way an application in our model might combine transport negotiation with the initial exchange of the selected transport, avoiding unnecessary round-trips or state setup. The application first locally requests from each supported transport a copy of the "SYN" packet the transport *would* send to initiate a new session. The application collects the SYN packets for all such transports, bundles them together into one "Meta-SYN" packet, and sends the Meta-SYN to the responding endpoint, as shown in Figure 4. The responding application breaks apart the Meta-SYN, passes the SYN for some transport it supports to its implementation of that transport, and subsequent communication proceeds normally via that transport. This design assumes that packets for different transports are distinguishable from each other and from Meta-SYN packets; the application might interpose a minimal header for this purpose if required.

A side effect of making endpoints transport-independent is to close the debate over whether to allocate well-known ports across several transports at once. IANA would need to manage only one port space, and existing applications could adopt new transports without having to register new ports for each.

## 3. THE FLOW REGULATION LAYER

Our *Flow Regulation Layer*, or simply *Flow Layer*, manages the performance of a flow between a pair of endpoints. The Flow Layer takes the underlying best-effort delivery service, which typically provides limited information about available bandwidth and other network characteristics, and builds a *flow-regulated best-effort delivery service*, which "knows" how to regulate the flow of packets for best use of the available path(s). The Flow Layer implements congestion control [27] and may encapsulate performance-related mechanisms such as performance enhancing proxies [11], end-to-end multihoming [46], multipath transmission [33], and forward error correction.

The idea of factoring congestion control into a separate protocol is embodied in the Congestion Manager (CM) [6] and Datagram Congestion Control Protocol (DCCP) [28]; these protocols offer starting points for our Flow Layer, as discussed in Section 4. Beyond merely factoring out congestion control, our insight is that the Flow Layer is a clean place to implement many performance-related mechanisms, enabling them to benefit many transports, and avoiding interference with transport reliability or end-to-end fate-sharing [16]. The following sections explore several such performance enhancement techniques: dividing communication paths into segments for performance tuning, utilizing multiple redundant communication paths, and aggregating flows to improve fairness or efficiency.

### 3.1 Path Segmentation

Our architecture permits devices in the network, called *flow middleboxes*, to interpose on Flow Layer communication by dividing a path into *segments*, as shown in Figure 5. Flow middleboxes "split" the path by terminating one segment's Flow Layer connection and initiating a new one for the next segment. Each segment may consist of several Network Layer hops; path segmentation does not imply hop-by-hop congestion control [34], although the latter may be viewed as a limit case of path segmentation.

Flow middleboxes do not touch Transport Layer headers or payloads, so they are compatible with any transport protocol. Since flow middleboxes affect only communication performance and not transport semantics, they serve in precisely the role for which the end-to-end principle [41] justifies such in-network mechanisms. In contrast with the analogous technique of TCP splitting [4], where transport state may be lost if a middlebox fails after acknowledging data received on one segment but before transmitting it on the next, Flow Layer splitting preserves end-to-end fate-sharing [16] because flow middleboxes hold only performance-related soft state.

Motivations for splitting a communication path into individually congestion-controlled segments include performance benefits from reduced RTT, specialization to network characteristics, and administrative isolation. We expore each in turn.

#### 3.1.1 Performance Benefits from Reduced RTT

A TCP flow's throughput is adversely affected by large round-trip time (RTT), especially in competition with flows of smaller RTT [19]. In addition, since information requires one RTT to propagate around the control loop, *any* end-to-end congestion control scheme's responsiveness to changing conditions is limited by RTT. Subdividing a communication path into independently congestion-controlled segments reduces each segment's RTT to a fraction of the path's total RTT, which can improve both throughput and responsiveness. This benefit has been noted in the context of hop-by-hop congestion control schemes for packet-switched [34], cell-switched [29], and wireless networks [54]. The Logistical Session Layer [48] similarly leverages this effect to improve wide-area grid performance. Our Flow Layer thus provides a semantically clean way to obtain the benefits of shorter RTTs within segmented paths.

#### 3.1.2 *Specialization to Network Characteristics*

The best congestion control scheme for a communication path often depends on the characteristics of the underlying network [8]. Classic TCP congestion control [27] performs well on wired LANs and the Internet core, but poorly on networks that are loss-prone due to transmission errors or mobility, and on long-delay connections such as satellite links or wireless wide-area networks. Since integrating diverse networks is a fundamental goal of the Internet [16], we must assume that any communication path may traverse several network types, each of which might place conflicting requirements on any single end-to-end congestion control scheme. New end-to-end schemes are available for high-bandwidth, long-delay links [20], and others for mobile ad hoc networks [31], but will any one scheme perform well on a path that includes links of both types (and others)? Path segmentation in the Flow Layer provides a clean method of specializing congestion control to the characteristics of individual path segments while avoiding the pitfalls of traditional performance enhancing proxies [11].

Other fixes are available for specific performance issues [5], but we feel that none of them solves the general network path heterogeneity problem. A "sledgehammer approach" is to open parallel TCP streams over one path, either at transport [44] or application level [2], boosting throughput at the cost of fairness by amplifying TCP's aggressiveness [21]. TCP snooping [7] enables intermediate nodes to retransmit lost packets and suppress duplicate acknowledgments without violating TCP's semantics, but this technique is transport-specific and does not allow adjacent segments to run independent congestion control schemes. Many approaches assume that only the "last hop" of a path requires specialization—an assumption violated by important scenarios such as wireless mesh networks [1]. In contrast, our architecture supports any number of segments and permits independent performance tuning of each.

#### 3.1.3 *Administrative Isolation*

Even where one end-to-end congestion control scheme may be technically adequate, the Internet's inertia makes it politically difficult to agree on, evolve, and deploy new end-to-end schemes. Any new scheme encounters resistance unless it is "TCP-friendly"—no more aggressive than TCP Reno—since the new scheme's flows will compete with Reno streams "in the wild." But since the Internet does not *enforce* TCP-friendliness [21], selfish or unaware users can and do deploy unfairly aggressive mechanisms anyway—e.g., in the form of TCP-unfair UDP flows [15] or concurrent TCP flows [30].

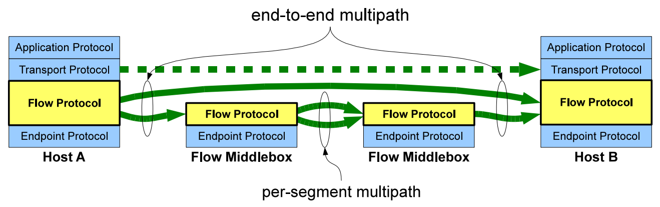


**Figure 6: Flow Layer multipath communication example. The multihomed hosts use two end-to-end paths, one passing through a pair of middleboxes implementing an in-network multipath segment.**

Path segmentation offers an incremental solution to congestion control evolution: split the Flow Layer path at administrative boundaries, and deploy the new scheme only on segments traversing domains in which the scheme has been adequately tested and approved, preserving TCP-friendliness on other segments. Path segmentation allows network administrators to roll out a new scheme *one administrative domain at a time*, and homogenize the congestion control algorithms used within their domain if desired, ensuring that the new scheme's flows compete only with each other within the domain and not with Reno flows or other arbitrary schemes deployed by end hosts.

Even for end-to-end streams not conforming to our architecture—e.g., flows with congestion control in the Transport Layer or no congestion control at all—homogeneous congestion control can still be enforced within a domain if needed, by encapsulating such streams in a Flow Layer "tunnel" while crossing that domain. Our architecture thus provides a clean framework for proposed mechanisms that use per-flow state at border routers to implement new congestion control schemes within a domain [47], or to enforce TCP-friendliness [39] or differential service agreements [24].

## 3.2 Multipath Communication

There are many ways to exploit alternative network paths to improve reliability [23], balance load [36], or enhance security [32]. To be deployable, however, a multipath scheme must be compatible with upper layer protocols designed assuming single-path routing, and must remain interoperable with single-path routing domains. Our architecture addresses these deployment issues by permitting end hosts and flow middleboxes to implement multipath communication end-to-end or in the network, as shown in Figure 6.

#### 3.2.1 *Flow Layer Multihoming*

The Flow Layer provides a clean place to implement end-to-end *multihoming*: binding several endpoints together to provide multiple paths over the existing routing infrastructure. In contrast with transport multihoming [33, 46], multihoming in the Flow Layer can benefit any transport without interfering with transport semantics. An address rewriting mechanism similar to shim6 [43] in the Flow Layer can make all of a host's endpoints appear as one to these transports.

Path segmentation in our architecture can also facilitate the incremental deployment of multipath routing. A multipath routing protocol may be deployed within an administrative domain, surrounded by flow middleboxes that can exploit avail-

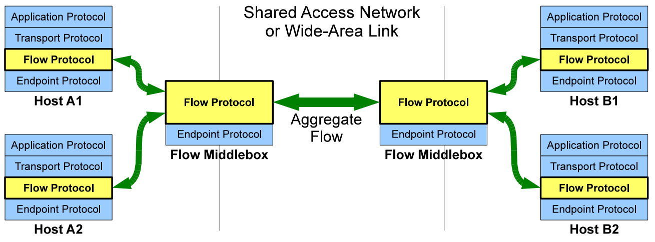


**Figure 7: Flow Layer aggregation example containing two end-to-end flows, which appear as one flow to the intermediate network.**

able paths in flow segments crossing that domain, without affecting external segments (see Figure 6). Alternatively, or simultaneously, a multi-site organization might deploy flow middleboxes at site boundaries to distribute inter-site traffic across redundant wide-area links.

### *3.2.2 Coping with Path Diversity in Upper Layers*

Naïvely distributing packets among multiple paths with varying delay, whether end-to-end or in-network, can confuse the congestion control and reliability mechanisms of existing transports [10]. In our architecture, a multihomed Flow Layer can avoid this confusion by implementing per-path congestion control, but the Transport Layer remains responsible for retransmission and thus vulnerable to similar confusion. To support arbitrary transports, therefore, a multihomed Flow Layer needs to preserve the illusion of single-path delivery, either by using only one path at once as SCTP does [46], or through order-preserving traffic dispersion [23].

Multipath-aware transports [26] and applications [3] can benefit from the ability to maintain per-path state and explicitly associate packets with paths. Through a simple path indexing mechanism inspired by *path splicing* [35], which we do not elaborate here for space reasons, a multipath Flow Layer in our architecture can expose alternative paths to upper layer protocols capable of using them, while retaining compatibility with multipath-oblivious protocols.

## 3.3 Flow Aggregation

Finally, the Flow Layer provides a clean point at which to *aggregate* related flows when desired, so that the intervening network treats the aggregate as one flow (see Figure 7). Flow aggregation can provide several benefits including reuse of congestion control state and improved fairness.

### *3.3.1 Reuse of Congestion Control State*

Since an aggregate of many transport instances is typically longer-lived and represents more traffic than any of its constituents, measurements of the aggregate's characteristics can benefit from a longer history and more samples. Transport extensions have been proposed to aggregate congestion control state across reincarnations of one transport session [12], across concurrent sessions [51], across transport protocols [6], and across hosts in an edge network [55].

Placing optimizations such as these in the Flow Layer allows arbitrary transports to benefit from them, and permits aggregation to be performed cleanly within the network as well as end-to-end. In our architecture, for example, a flow middlebox can aggregate congestion control state across the hosts in an edge network and use that information to optimize flows crossing that middlebox transparently, without requiring end host modifications as in TCP/SPAND [55].

### *3.3.2 Fairness Control*

TCP's per-stream "fairness" notion often fails to match the expectations of users and network operators [13]; Flow Layer aggregation may be useful to implement higher-level fairness policies. For example, an ISP may want each *customer* to get equal bandwidth at bottlenecks in its network, regardless of whether a customer uses few transport instances (web browsing, SSH) or many (BitTorrent). To implement such a policy, the ISP could deploy flow middleboxes at its borders that aggregate all segments crossing its network into one "macro-flow": since each macro-flow has one congestion control context, each macro-flow gets an equal share of congestion bottleneck bandwidth. Most such macro-flows will connect one customer's access router to one of a few upstream network attachment points, so this meta-flow fairness should approximate a per-customer fairness policy. Flow aggregation can thus implement policies similar to those motivating hierarchical fair queuing schemes [9], without changing interior routers.

## 4. IMPLEMENTATION AND EVOLUTION

One of the benefits of the proposed architecture is that existing protocols already provide starting points for implementing its new layers. Since these existing protocols were designed in the traditional architectural framework, however, the fit is not perfect, so further development will be needed.

- **Endpoint Layer:** UDP [38] provides a pervasive first approximation to our Endpoint Layer. Viewing UDP not as transport but as implementing a common endpoint space to be shared by all (new) transports, it becomes worthwhile to consider evolving this shared endpoint space, to support larger port numbers or service names for instance [52]. Also needed are extensions enabling NATs and firewalls to detect which endpoints within a private network are intended to be publicly reachable, and create persistent bindings for them as policy permits. Our intent is for the Endpoint Layer to provide these services, with incremental deployment facilitated by dual-stack Endpoint Layer gateways that map between UDP and the new Endpoint Protocol.
- **Flow Regulation Layer:** Both DCCP [28] and CM [6] approximately implement our Flow Layer, and each has unique features we would like to see combined in one protocol. CM offers aggregation of congestion control state across flows and a packet transmission API that facilitates application-layer framing (ALF) [17], whereas DCCP provides explicit negotiation of congestion control schemes. We also need to examine how to reposition them atop the Endpoint Layer, and to develop extensions supporting Flow Layer optimizations such as path segmentation, multipath communication, and flow aggregation.
- **Transport Layer:** Finally, new Transport Layer protocols will build upon the Flow Layer to offer communication abstractions such as reliable byte streams [49], reliable data-

grams [37], media frames [42], multi-streams [46], or structured streams [22]. We need to consider how to reposition transports atop the Flow Layer in an incremental and backward-compatible way, and how the absence of congestion control in the Transport Layer may impact transport mechanisms such as (fast) retransmit and receive window control.

## 5. CONCLUSION

Although the OSI protocol stack has been dead for years, its layering model remains the standard frame of reference for the Internet, and aspects of its layering model have created serious roadblocks to Internet evolution. By factoring endpoint addressing into a common Endpoint Layer instead of distributing it among transports as in OSI, we obtain more flexibility in transport implementation and deployment, transport-oblivious firewall/NAT traversal, and more efficient transport negotiation. Similarly, by factoring congestion control into an intermediate Flow Layer, we decouple performance-oriented flow regulation from transport semantics, enabling the clean, modular, and incremental deployment of a host of performance optimizations both end-to-end and in the network, without interfering with transport reliability or fate-sharing. The new architectural model therefore appears promising, although many protocol details remain to be worked out.

Our model may appear to make the Internet architecture more complex, but we believe this complexity has already been forced upon us via the patchwork of interposers that have proliferated across the Internet [11, 25]. Our proposal provides a framework in which to fit these interposers together cleanly, recognizing and satisfying the needs that have led to the prevalence of these middleboxes. This project is ambitious, and many unresolved issues remain, such as NAT traversal details, buffering issues in flow middleboxes, APIs and interfaces between the new layers, and cross-layer dependencies. We hope to resolve these issues as we work out mechanisms and protocols to implement the new architecture.

## 6. REFERENCES

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4), Mar. 2005.

[2] M. Allman, H. Kruse, and S. Ostermann. An application-level solution to TCP's satellite inefficiencies. In *1st WOSBIS*, Nov. 1996.

[3] J. Apostolopoulos et al. On multiple description streaming with content delivery networks. In *INFOCOM*, June 2002.

[4] A. V. Bakre and B. Badrinath. Implementation and performance evaluation of indirect TCP. *IEEE Transactions on Computers*, 46(3):260–278, Mar. 1997.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE Transactions on Networking*, 5(6), Dec. 1997.

[6] H. Balakrishnan, H. S. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *SIGCOMM*, Sept. 1999.

[7] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *1st MOBICOM*, Nov. 1995.

[8] C. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 38(1):40–46, Jan. 2000.

[9] J. C. R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *SIGCOMM*, pages 143–156, Aug. 1996.

[10] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *Computer Communications Review*, 32(1), Jan. 2002.

[11] J. Border et al. Performance enhancing proxies intended to mitigate link-related degradations, June 2001. RFC 3135.

[12] R. Braden. T/TCP – TCP extensions for transactions, July 1994. RFC 1644.

[13] B. Briscoe. Flow rate fairness: Dismantling a religion. *Computer Communications Review*, 37(2):63–74, Apr. 2007.

[14] D. R. Cheriton. Sirpent: A high-performance internetworking approach. In *SIGCOMM*, Sept. 1989.

[15] J. Chung, Y. Zhu, and M. Claypool. FairPlayer or FoulPlayer? — head to head performance of RealPlayer streaming video over UDP versus TCP. Technical Report WPI-CS-TR-02-17, Worcester Polytechnic Institute, May 2002.

[16] D. D. Clark. The design philosophy of the DARPA Internet protocols. In *SIGCOMM*, Aug. 1988.

[17] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM*, pages 200–208, 1990.

[18] D. C. Feldmeier. Multiplexing issues in communication system design. In *SIGCOMM*, Sept. 1990.

[19] S. Floyd. Connections with multiple congested gateways in packet-switched networks, part 1: One-way traffic. *ACM CCR*, 21(5):30–47, Oct. 1991.

[20] S. Floyd. HighSpeed TCP for large congestion windows, Dec. 2003. RFC 3649.

[21] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *Transactions on Networking*, 7(4):458–472, Aug. 1999.

[22] B. Ford. Structured streams: a new transport abstraction. In *SIGCOMM*, Aug. 2007.

[23] E. Gustafsson and G. Karlsson. A literature survey on traffic dispersion. *IEEE Network*, 11(2):28–36, Mar. 1997.

[24] A. Habib and B. Bhargava. Unresponsive flow detection and control using the differentiated services framework. In *PDCS*, Aug. 2001.

[25] M. Holdrege and P. Srisuresh. Protocol complications with the IP network address translator, Jan. 2001. RFC 3027.

[26] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *Transactions on Networking*, 14(5):951–964, Oct. 2006.

[27] V. Jacobson. Congestion avoidance and control. pages 314–329, Aug. 1988.

[28] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP), Mar. 2006. RFC 4340.

[29] H. T. Kung and A. Chapman. The FCVC (flow-controlled virtual channels) proposal for ATM networks: A summary. In *1st ICNP*, Oct. 1993.

[30] Y. Liu, W. Gong, and P. Shenoy. On the impact of concurrent downloads. In *WSC*, 2001.

[31] C. Lochert, B. Scheuermann, and M. Mauve. A survey on congestion control for mobile ad-hoc networks. *WCMC*, 7(5):655–676, June 2007.

[32] W. Lou and Y. Fang. A multipath routing approach for secure data delivery. In *MILCOM*, Oct. 2001.

[33] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *9th ICNP*, Nov. 2001.

[34] P. P. Mishra and H. Kanakia. A hop by hop rate-based congestion control scheme. In *SIGCOMM*, Aug. 1992.

[35] M. Motiwala et al. Path Splicing. In *SIGCOMM*, Aug. 2008.

[36] S. Murthy and J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *INFOCOM*, Mar. 1996.

[37] C. Partridge and R. Hinden. Version 2 of the reliable data protocol (RDP), Apr. 1990. RFC 1151.

[38] J. Postel. User datagram protocol, Aug. 1980. RFC 768.

[39] A. Rangarajan and A. Acharya. ERUF: Early regulation of unresponsive best-effort traffic. In *7th ICNP*, Oct. 1999.

[40] J. Rosenberg. UDP and TCP as the new waist of the Internet hourglass, Feb. 2008. Internet-Draft (Work in Progress).

[41] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *TOCS*, 2(4):277–288, Nov. 1984.

[42] H. Schulzrinne et al. RTP: A transport protocol for real-time applications, July 2003. RFC 3550.

[43] Site multihoming by IPv6 intermediation (shim6). `http://www.ietf.org/html.charters/shim6-charter.html`.

[44] H. Sivakumar, S. Bailey, and R. Grossman. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *SC2000*, Nov. 2000.

[45] P. Srisuresh and K. Egevang. Traditional IP network address translator (Traditional NAT), Jan. 2001. RFC 3022.

[46] R. Stewart, ed. Stream control transmission protocol, Sept. 2007. RFC 4960.

[47] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *SIGCOMM*, Aug. 1998.

[48] M. Swany. Improving throughput for grid applications with network logistics. In *SC2004*, Nov. 2004.

[49] Transmission control protocol, Sept. 1981. RFC 793.

[50] D. L. Tennenhouse. Layered multiplexing considered harmful. In *1st International Workshop on Protocols for High-Speed Networks*, May 1989.

[51] J. Touch. TCP control block interdependence, Apr. 1997. RFC 2140.

[52] J. Touch. A TCP option for port names, Apr. 2006. Internet-Draft (Work in Progress).

[53] UPnP Forum. Internet gateway device (IGD) standardized device control protocol, Nov. 2001. http://www.upnp.org/.

[54] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *Transactions on Networking*, 15(1):133–144, Feb. 2007.

[55] Y. Zhang, L. Qiu, and S. Keshav. Speeding up short data transfers: Theory, architectural support and simulation results. In *10th NOSSDAV*, June 2000.

[56] H. Zimmermann. OSI reference model—the ISO model of architecture for open systems interconnection. *Transactions on Communications*, 28(4):425–432, Apr. 1980.

# Reducing Transient Disconnectivity using Anomaly-Cognizant Forwarding

Andrey Ermolinskiy[†], Scott Shenker[†⋆]
[†]University of California at Berkeley, [⋆]International Computer Science Institute

## Abstract

It is well known that BGP convergence can cause widespread temporary losses of connectivity resulting from inconsistent routing state. In this paper, we present Anomaly-Cognizant Forwarding (ACF) - a novel technique for protecting end-to-end packet delivery during periods of convergence. Our preliminary evaluation demonstrates that ACF succeeds in eliminating nearly all transient disconnection after a link failure without the use of precomputed backup routes or altering the dynamics of BGP.

## 1 Introduction

It is widely known that BGP, the core Internet interdomain routing protocol, is susceptible to temporary connectivity failures during periods of convergence. A single event, such as a link failure or a policy change, can trigger a lengthy and complex sequence of route recomputations, during which neighboring ASes exchange updates and converge on a new globally-consistent set of routes. During this process, routers operate upon potentially inconsistent local views, which can lead to the emergence of temporary anomalies such as *loops* and *blackholes*. Both of these are considered undesirable, as they result in temporary losses of connectivity to the set of destinations affected by the event.

In order to prevent explosive growth of control traffic during the convergence process, BGP routers are typically configured to constrain the maximum rate of update propagation via the MRAI timer and [1] recommends setting its value to 30 seconds. Inevitably, limiting the rate of update dissemination lengthens the period of exposure to routing anomalies and several studies have reported prolonged and noticeable bursts of packet loss caused by BGP convergence. It has been shown that a single route change can produce up to 30% packet loss for two minutes or more [9]. Further, [15] reports loss bursts that last up to 20 seconds after a single route failure and up to 8 seconds after a route recovery event.

Today's Internet applications such as online games, streaming video delivery, and VoIP demand continuous end-to-end reachability and consistent performance. Hence, this problem has received considerable research attention and previous approaches can be broadly categorized into (a) those that attempt to expedite protocol convergence [3, 12] and (b) those that seek to protect end-to-end packet delivery from the adverse effects of convergence. It has been suggested that mechanisms in the former category face an inherent limitation given the current scale of the Internet on the one hand and stringent demands of today's applications on the other. A scalable policy-based routing protocol that converges fast enough for applications such as interactive voice delivery still remains an elusive goal.

The second category of proposals includes mechanisms such as R-BGP [8], which advocates the use of precomputed failover paths for ensuring connectivity during periods of convergence. This scheme offers provable guarantees of reachability for single link failures, but these guarantees come at the cost of additional forwarding state and protocol complexity associated with the maintenance of backup routes and ensuring loop-free convergence. Further, preserving connectivity in the face of multiple concurrent routing events would require routers to compute and maintain additional link-disjoint paths and the forwarding state requirements would present a serious scalability challenge.

Most recently, Consensus Routing [7] proposes to address transient disconnectivity by requiring BGP routers to agree on a globally-consistent "stable" view of forwarding state. In this context, stability means that a source domain can adopt a route to some destination in a given epoch only if each of the intermediate routers on the path adopts the respective route suffix in the same epoch, which guarantees absence of loops. In each epoch, routers participate in a distributed snapshot and consensus protocol in order to identify the set of "complete" BGP updates that satisfy stability. In contrast to much of prior work directed at reducing the duration of convergence, this scheme intentionally delays the adoption of BGP updates, so as to preserve the stability invariant. In the absence of a stable forwarding path, consensus routing fails over to a transient forwarding mode that implements a heuristic such as detouring, backtracking, or backup paths.

In this paper, we present *Anomaly-Cognizant Forwarding* (ACF) - a new and complementary approach to improving Internet path availability and reducing transient disconnection. Rather than attempting to eliminate anomalous behavior by enforcing global consistency or shrinking the convergence time window, we accept inconsistent routing state as an unavoidable fact and instead develop a mechanism for *detecting* and *recovering* from such inconsistencies on the data path. While much of prior work has focused on extending BGP to improve its consistency and convergence properties, in this paper we consider a somewhat more disruptive approach that involves adding several fields to the packet header and inspecting them on the forwarding path. Our main hypothesis is that a single nearly trivial extension to conventional IP forwarding suffices to eliminate a dominant fraction of convergence-related disconnectivity. Our approach does not require routers to maintain multiple forwarding tables, nor does it require extending BGP or altering its timing dynamics.

## 2 Approach Overview

In broad terms, we view inconsistent BGP state and routing anomalies as unavoidable facts and approach the problem by extending the forwarding plane with a small amount of functionality that enables us to detect and recover from these anomalies. Toward this end, we augment the packet header with two additional pieces of state. First, a packet $p$ originating in $AS_s$ and destined to $AS_d$ carries a *path trace* (denoted $p.pathTrace$) - a list of AS-level hops encountered by $p$ on its path toward $AS_d$. At each hop, the border router inspects this field and appends its own AS identifier. The content of this field enables routers to detect and recover from occurrences of loops via a process that we describe more fully below. Second, each packet carries a *black list* (denoted $p.blackList$) containing an encoding of AS identifiers that are known to have possessed *deficient* routing state for $p$'s destination at some point after packet's origination.

We say that a transit domain $AS_t$ has *deficient* routing state for a destination $AS_d$ at a particular instant in time if at that instant (a) $AS_t$ lacks a valid policy-compliant path to $AS_d$, or (b) the path adopted by $AS_t$ for destination $AS_d$ results in a routing loop that causes packets to return back to $AS_t$.

At a high level, ACF packet forwarding proceeds as follows: a router first inspects $p.pathTrace$ and checks it for the presence of its local AS identifier, which would indicate a loop. If no loop is detected, the packet is forwarded as usual along the adopted route. Otherwise, the loop is viewed as evidence of deficient state and the router acts upon it by moving every AS identifier belonging to the loop (which it knows from $p.pathTrace$) to $p.blackList$ and invoking the control plane, where the RIB is searched for the presence of alternate paths that do not traverse any of the blacklisted domains.

The second core component of our design is an alternate mode of packet delivery, which we term *recovery forwarding*. This mode helps ensure connectivity in situations where a router is unable to forward a packet because it does not possess a valid non-blacklisted path.

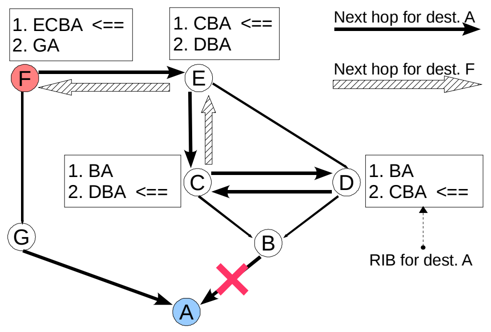


Figure 1: A sample AS topology with a transient loop.

Forwarding in recovery mode is facilitated by a set of *recovery destinations*. When a transit router chooses to initiate recovery forwarding for a packet $p$, it adds the local AS identifier to $p.blackList$, copies $p$'s destination address to an alternate location in the header, and redirects the packet to the address of some recovery destination, chosen at random from a well-known static set of potential destinations. In our current design and simulations, we assign the recovery destination role to a group of 10 well-connected Tier-1 ISPs[1].

The basic intuition that motivates this scheme is that the chosen recovery destination $AS_r$ (or some intermediate router along the path to $AS_r$) is likely to possess a valid non-blacklisted route to packet's original destination. As the packet travels toward $AS_r$ in recovery mode, each router on the path first attempts to forward it to the original destination $AS_d$. If a usable non-blacklisted path is known, the router takes the packet off the recovery path and resumes normal-mode forwarding. Otherwise, the packet is sent to the next hop for destination $AS_r$. If, after reaching the recovery destination, the packet cannot be taken off the recovery path because $AS_r$ does not possess a usable route to $AS_d$, the packet is dropped. Alternatively, in an effort to ensure eventual delivery, $AS_r$ can re-initiate recovery forwarding via another destination. In the latter scenario, the process repeats until (a) the packet is taken off the recovery path by some destination that knows a working route to $AS_d$, (b) the packet is dropped because no recovery destination has such a route, or (c) the packet is dropped because its TTL expires.

We illustrate our scheme using the AS topology in Figure 1. Suppose that initially, domains $C$ and $D$ both use $B$ as the next hop for destination $A$. In this example, failure of the inter-AS link $\langle A-B\rangle$ would cause $B$ to send

[1]Internet service providers can offer recovery forwarding as a paid service for customers that wish to safeguard themselves from BGP-related connectivity failures.

a withdrawal notification to its neighbors. Upon receiving the withdrawal, $C$ and $D$ would immediately switch to alternate paths $\langle D \rightarrow B \rightarrow A\rangle$ and $\langle C \rightarrow B \rightarrow A\rangle$, respectively. With conventional BGP, domain $C$ has no way of determining that the newly-adopted path is invalid until it receives a withdrawal from $D$ and, analogously, $D$ considers $\langle C \rightarrow B \rightarrow A\rangle$ to be a valid route and adopts $C$ as its next hop, thus causing a transient loop to emerge.

Suppose that domain $C$ wishes to send a packet to an address in domain $A$. With ACF, packet forwarding proceeds as follows: Initially, $C$ adds its local AS identifier to $p.pathTrace$ and forwards the packet to its next hop - domain $D$. Upon receiving the packet, $D$ appends its identifier to $p.pathTrace$ and sends the packet back to $C$, which inspects $p.pathTrace$ and detects a loop. It truncates $p.pathTrace$ and, for each non-local AS identifier belonging to the loop (in this example only $D$), adds a corresponding entry to $p.blackList$. Next, $C$ reattempts to forward the packet, this time avoiding the blacklisted forwarding table entry and discarding the corresponding route. In the example shown, $C$ has no alternative working routes for destination $A$, so it adds itself to $p.blackList$ and invokes recovery forwarding, choosing domain $F$ as the recovery destination. $C$ forwards the packet in recovery mode to $E$ (its next hop for $F$) and the packet arrives with $p.pathTrace = \langle C\rangle$, $p.blackList = \langle C,D\rangle$. Upon receiving the packet, $E$ first attempts to forward $p$ to its original destination ($A$), but discovers that both its current next hop ($C$) and the alternate path through $D$ are blacklisted in the packet's header and discards the respective routes. Lacking other alternate paths, $E$ adds itself to $p.blackList$ and forwards the packet further along the recovery path to its peer $F$. Analogously, $F$ determines from the blacklist that its next hop $E$ does not posses a valid path and purges the respective route from its RIB. However, $F$ knows of an alternate working route $\langle G \rightarrow A\rangle$ and adopts it, causing $p$ and all subsequent packets destined to $A$ to be forwarded via $G$. Eventually, BGP path withdrawals will propagate through the topology and reach $F$, causing it to expose $\langle F \rightarrow G \rightarrow A\rangle$. During the transient period of inconsistency, however, the *pathTrace* and *blackList* state being propagated on the data path enables us to discover a valid alternate route and preserve end-to-end packet delivery.

Before we proceed to a detailed description of the design, we make two high-level observations about our approach. First, since ACF utilizes two distinct modes of forwarding (i.e., *normal* and *recovery* modes), it can cause some packets to traverse multiple distinct paths to the destination during periods of convergence. For example, $AS_s$ may initially attempt to send a packet to $AS_d$ via a path $P_1$, but one of the intermediate hops may decide to re-route it via a recovery destination, which, in turn, can choose to forward the packet via $P_2$ - an alternate path toward $AS_d$ that is link-disjoint from $P_1$. Unlike earlier work on failover BGP paths [8], our mechanism does not require routers to construct an explicit set of backup routes and to maintain multiple forwarding table entries. In ACF, the two modes make use of the same forwarding table and we try to *discover* healthy alternate paths dynamically by extending the forwarding plane.

Second, we do not assume that the set of paths to recovery destinations is stable and that every AS possesses a working loop-free route to some recovery destination at all times. Indeed, certain failure scenarios (e.g, a core link failure) can result in disruption of paths to multiple endpoints, including those that serve as recovery destinations, and clearly, our design must succeed in retaining end-to-end connectivity in the face of such failures. Thankfully, there is a simple and effective solution that enables us to handle such cases - we protect recovery-path forwarding against routing anomalies using precisely the same mechanism that we use to safeguard packet delivery on the normal forwarding path, i.e., using the *pathTrace* and *blackList* fields in the packet header.

## 3 The Design of ACF

### 3.1 Packet header state

ACF adds the following fields to the packet header:

*recoveryMode*: A single-bit flag indicating the current forwarding mode (*normal* or *recovery*).

*finalDestAddr*: In recovery mode, this field carries the packet's actual destination address (i.e., its destination prior to redirection).

*pathTrace*: An ordered list of AS-level hops traversed by the packet in the current forwarding mode.

*blackList*: A set of AS identifiers that are known to possess deficient routing state for the packet's original destination.

*blackListRecov*: A set of AS identifiers that are known to possess deficient routing state for the packet's designated recovery destination.

In our current design, *pathTrace* is represented as a linear list of 16-bit AS numbers, while *blackList* and *blackListRecov* and represented using a space-efficient Bloom filter encoding (note that AS identifiers are never removed from blacklists).

### 3.2 Forwarding algorithm

When a packet $p$ arrives at a router, its *recoveryMode* flag is inspected to determine the appropriate forwarding mode. In the normal mode, the router first checks the *pathTrace* field for the presence of its local AS number. If a loop is detected, all AS components of the loop are added to $p.blackList$ and the path trace is truncated to exclude the loop. Next, the forwarding table is consulted to obtain the next hop for $p$'s destination and the content of $p.blackList$ is inspected. If the next-hop AS is present in $p.blackList$, the current route is discarded and the control plane ($FindAlternateRoute$) is invoked to find and install an alternate non-blacklisted route.

In $FindAlternateRoute$ the standard BGP route selection process is invoked to identify a new preferred path and, crucially, all blacklisted routes are excluded from consideration during this process. We investigated and evaluated two alternative methods for deciding whether to exclude a particular candidate route $R = \langle AS_1^R, AS_2^R, ...AS_k^R \rangle$ for a given packet $p$:

1. Exclude $R$ iff $AS_1^R \in p.blackList$.
2. Exclude $R$ iff $\exists i$ such that $AS_i^R \in p.blackList$.

Consider a scenario, in which $AS_s$ knows of two distinct routes to $AS_d$, namely $\langle AS_1 \rightarrow AS_2 \rightarrow AS_d \rangle$ and $\langle AS_3 \rightarrow AS_2 \rightarrow AS_d \rangle$. Initially, it tries to forward the packet via $AS_1$, but the packet returns with $\langle AS_s, AS_1, AS_2, AS_4 \rangle$ in its $pathTrace$, causing $AS_s$ to blacklist $AS_1$, $AS_2$, and $AS_4$. With method 1, $AS_s$ would next attempt to forward via $AS_3$, but this would result in wasted effort if $AS_3$ does not know of any alternate paths to $AS_d$ that do not go through $AS_2$. Conversely, scheme 2 would require $AS_s$ to discard its path through $AS_3$ and invoke recovery forwarding due to absence of other alternatives. In this situation, skipping $AS_3$ can result in a lost opportunity to forward via an efficient alternate route if $AS_3$ does indeed possess such a route.

We examined both alternatives and found that the second method is substantially more effective in reducing transient packet loss for the set of failure cases we considered. It allows problematic paths to be detected and discarded more quickly and reduces the number of hops it takes for a packet to home in on a valid alternate route. Note that as a further optimization, we could also evaluate the criterion of method (2) on the data path (currently, we check only the next hop), but this improvement would come at the expense of additional processing overhead and forwarding state. Hence, our current design adopts a compromise by validating only the next hop on the data plane and performing full AS-PATH inspection only upon evidence of anomalous behavior.

If $FindAlternateRoute$ fails to identify a working route, recovery forwarding is invoked. The router adds its local AS number to $p.blackList$, clears $p.pathTrace$ and $p.blackListRecov$, chooses a non-blacklisted recovery destination, and looks up the corresponding next hop. Forwarding in recovery mode proceeds analogously and we refer the reader to [5] for additional details and pseudocode.

## 4 Preliminary Evaluation

The preliminary evaluation we present in this section focuses on addressing three key questions: (1) How effective is ACF at sustaining end-to-end connectivity during convergence? (2) In the absence of precomputed backup routes, how long does it take to recover a packet from an anomalous path and find an alternate working route? (3) How significant is the header overhead incurred by ACF?

**Methodology:** To answer these questions, we implemented an event-driven parallel simulator that enables us to study the dynamics of BGP convergence in realistic Internet-scale AS topologies and simulate packet forwarding at an arbitrary point in time during convergence. Our initial experiments examine the effects of inter-AS link failures on end-to-end reachability and focus on failures of access links that connect to a multi-homed edge domain. We use the CAIDA AS-level topology from May 12, 2008 [2] annotated with inferred inter-AS relationships. The topology contains 27969 distinct ASes and 56841 inter-AS links. Following standard convention, our simulator implements "valley-free" route propagation policies [6] and customer routes are always preferred over peer and provider routes.

The topology includes 12937 multihomed edge ASes and a set of 29426 adjacent provider links. We conduct a failure experiment for each provider link $\langle AS_p - AS_d \rangle$ in this set. We begin by simulating normal BGP convergence that results in adoption of consistent policy-compliant paths toward the destination $AS_d$. Next, we fail its link to $AS_p$, simulate packet forwarding from each AS to $AS_d$ during the period of reconvergence, and identify the set of ASes that experience temporary loss of connectivity to $AS_d$ during this period. With traditional forwarding, a source domain is considered disconnected if an intermediate router on its path to $AS_d$ drops a packet because it does not possess a route or if the packet's TTL (initially set to 32 hops) expires, indicating a forwarding loop. With ACF, a domain is disconnected if its packet is dropped at the recovery destination and upon TTL expiration.

**Transient disconnection after link failures:** As expected, we found that BGP with conventional forwarding exhibits a substantial amount of transient disconnectivity. 51% of failures cause some of the ASes to experience connectivity loss and 17% of failures cause at least half of all ASes in the topology to lose connectivity. Figure 2 plots the fraction of disconnected domains for the cumulative fraction of failure cases and demonstrates the effectiveness of ACF. In 84% of failure cases that produce some disconnectivity with conventional forwarding, ACF fully eliminates unwarranted packet loss and further, in 96% of such cases no more than 1% of all ASes experience disconnection. The figure also illustrates that recovery forwarding plays a pivotal role in protecting packet delivery

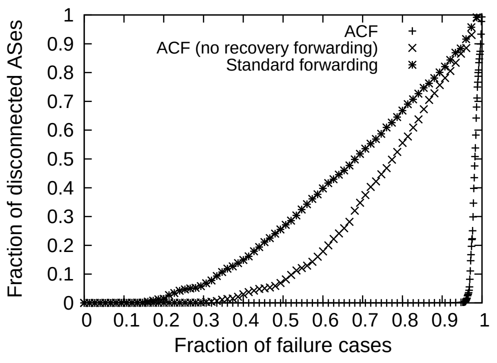


Figure 2: Prevalence of transient disconnection after a single provider link failure. The x-axis denotes the fraction of failure cases that cause some disconnectivity with traditional forwarding.

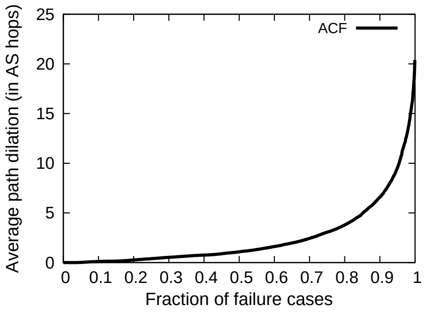


Figure 3: Average path dilation with ACF.

and ensuring connectivity in the face of anomalies. For a small fraction of failure cases (0.2%), our scheme offers little or no measurable improvement, leaving over 90% of the topology disconnected, and further inspection revealed that in most of these cases packets fail to discover a working route within 32 hops.

**Path efficiency:** By not maintaining a precomputed set of efficient alternate routes and instead letting packets discover them dynamically, our scheme can increase the number of hops a packet traverses during periods of instability. This overhead can be attributed to the fact that packets can encounter loops and that finding a working path can require detouring via a recovery destination. We measured this overhead in the above experiment and Figure 3 plots the path dilation (averaged over all ASes) for the cumulative fraction of failure cases. This quantity is computed by subtracting the length of the final route (in AS hops) adopted after reconvergence from the length of the longest path a packet would have to traverse under ACF before reaching its destination. In 65% of failures that cause loss under traditional forwarding, ACF recovers packets using no more than two extra AS hops and only 9% of failures incur the cost of 7 hops or more.

**Packet header overhead:** Table 1 shows the maximum number of entries in the *pathTrace* and *blackList*

| **% disconnected** | 0% | 0.09% | 0.9% | 9% | 90% |
|---|---|---|---|---|---|
| *pathTrace* len. | 11 | 16 | 16 | 20 | 13 |
| *blackList* len. | 4 | 11 | 9 | 11 | 16 |

Table 1: Maximum number of *pathTrace* and *blackList* entries in a representative sample of failures cases.

ACF header fields for a representative sample of failure cases corresponding to 0%, 0.09%, 0.9%, 9%, and 90% transient disconnection with ACF[2]. In the worst case, *pathTrace* consumes 40 bytes assuming that each entry is a 16-bit AS number. Up to 16 entries are added to *blackList* and a Bloom filter representation with 1% lookup error rate would require 10 bytes.

In summary, our initial evaluation suggests ACF to be a promising approach that significantly reduces transient packet loss and incurs reasonable bandwidth and latency overheads. However, the results presented here are only a first step toward understanding its full behavior in a complex Internet-scale environment and future work will include evaluating ACF under a broader range of scenarios that include failures of transit links, multiple concurrent failures, link recovery, and BGP policy changes.

## 5 Discussion and Future Work

**Feasibility of deployment:** ACF introduces several changes to the core mechanisms of IP forwarding and can thus be seen as facing a substantial barrier to adoption. More concretely, ACF requires adding several fields to the packet header, as well as introducing additional logic on the forwarding path. While clearly non-trivial, we believe that packet format issues can be addressed via the use of IP options and/or shim headers. Investigating these issues in detail and proposing a viable path toward deployment are two essential topics of future work.

**Packet processing overhead:** Our scheme adds complexity and computational overhead to the forwarding plane. We note that *FindAlternateRoute* - the most significant source of overhead in ACF - is invoked only during periods of instability and only for the purpose of replacing a broken route whose continued usage would otherwise result in packet loss. In the common case, the overhead reduces to checking *blackList* and *pathTrace* for the presence of the local AS number - operations that incur the cost of a single Bloom filter lookup and a linear scan, respectively. Both operations admit efficient implementation in hardware and parallelization. Finally, if the cost of a vector scan at each hop is deemed unacceptable, loop detection and recovery can be deferred until TTL expiration and handled at the control plane.

[2]*recovBlackList* is not shown because recovery destination paths remain stable in this experiment.

**ACF and routing policies:** Due to recovery forwarding, packets in ACF can be forwarded along a path which violates ISP export policies when viewed from an end-to-end perspective. At the same time, each individual forwarding decision in ACF respects policies by considering only the set of exported routes available in the RIB. In particular, only policy-compliant paths are used in recovery mode to guide a packet toward a recovery destination. ACF envisions the emergence of a new inter-ISP relationship landscape, where a group of highly-connected Tier-1 networks would provide the recovery destination service to multihomed clients that wish to safeguard themselves from the adverse effects of routing convergence. Viewed in this manner, our scheme can be said to provide policy-compliant forwarding via an intermediate destination.

## 6 Related Work

The adverse side-effects of BGP convergence have been studied extensively through measurements and simulations [9,10,13,15] and prior work on addressing this problem includes a family of protocol extensions and heuristics for accelerating convergence [3,4,14].

Another set of techniques, to which our scheme belongs, focuses on protecting end-to-end packet delivery from the unwanted effects of convergence and recent work in this area includes Resilient BGP [8] and Consensus Routing [7]. Analogously to both schemes, ACF implements two logically distinct modes of forwarding, which are differentiated using an extra bit in the packet header. R-BGP advocates the use of precomputed failover paths and requires routers to maintain multiple forwarding table entries for some destinations. To achieve loop-freeness, R-BGP introduces an assumption regarding route selection preferences and augments BGP update messages with root cause information. In contrast, our scheme works with general preference policies and requires no changes to the routing protocol, but does not offer provable guarantees of reachability. Consensus Routing ensures loop-freedom by enforcing a globally-consistent view of forwarding state, achieved by strategically delaying the adoption of BGP updates. Several transient forwarding modes are used to provide high availability and our approach borrows the idea of detouring via a highly-connected domain. Consensus Routing also modifies the forwarding path and the per-hop packet encapsulation used in the backtracking transient mode is conceptually analogous to ACF's *pathTrace*. Our main insight is that carrying the list of prior hops on the data path also provides the ability to detects loops and thus, global consistency is extraneous if packets can be recovered from loops and redirected via a reasonably efficient path. Consensus Routing delays the adoption of new routes by up to several minutes which, in certain scenarios, can have an adverse effect on end-to-end reachability.

Failure-Carrying Packets [11] is a recent proposal for link-state protocols that protects end-to-end delivery by augmenting packets with information about link failures. ACF adopts an analogous approach, but focuses on interdomain policy routing. We compared ACF with the strawman design for path-vector FCP presented in [11] and found that FCP improves end-to-end path availability for only a fraction of failure cases, demonstrating an improvement comparable to ACF without recovery-mode forwarding. Compared to FCP, our scheme does not require routers to precompute and cache a set of alternate forwarding table entries and incurs a smaller per-packet processing overhead (control-plane path reselection is invoked much less frequently). A detailed performance comparison with FCP is a topic of future work.

## 7 Acknowledgments

We are grateful to Philip Brighten Godfrey for his insightful comments on earlier versions of this paper and the anonymous reviewers for their valuable and constructive feedback.

## References

[1] A border gateway protocol 4 (BGP-4). `http://www.ietf.org/rfc/rfc4271.txt`.

[2] CAIDA inferred AS relationships dataset. `http://www.caida.org/data/active/as-relationships/`.

[3] BREMLER-BARR, A., AFEK, Y., AND SCHWARZ, S. Improving BGP convergence via ghost flushing. In *INFOCOM'03*.

[4] CHANDRASHEKAR, J., DUAN, Z., ZHANG, Z. L., AND KRASKY, J. Limiting path exploration in BGP. In *INFOCOM'05*.

[5] ERMOLINSKIY, A., AND SHENKER, S. Reducing transient disconnectivity using anomaly-cognizant forwarding. Tech. Rep. UCB/EECS-2008-120, EECS Department, University of California, Berkeley, Sep 2008.

[6] GAO, L., AND REXFORD, J. Stable internet routing without global coordination. In *SIGMETRICS'00*.

[7] JOHN, J. P., KATZ-BASSETT, E., KRISHNAMURTHY, A., AND ANDERSON, T. Consensus routing: The internet as a distributed system. In *NSDI'08*.

[8] KUSHMAN, N., KANDULA, S., KATABI, D., AND MAGGS, B. R-BGP: Staying connected in a connected world. In *NSDI'07*.

[9] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed internet routing convergence. In *SIGCOMM'00*.

[10] LABOVITZ, C., AHUJA, A., AND JAHANIAN, F. Experimental study of internet stability and backbone failures. In *FTCS'99*.

[11] LAKSHMINARAYANAN, K., CAESAR, M., RANGAN, M., ANDERSON, T., SHENKER, S., AND STOICA, I. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM'07*.

[12] LUO, J., XIE, J., HAO, R., AND LI, X. An approach to accelerate convergence for path vector protocol. In *GLOBECOM'02*.

[13] MAO, Z. M., GOVINDAN, R., VARGHESE, G., AND KATZ, R. H. Route flap damping exacerbates internet routing convergence. In *SIGCOMM'02*.

[14] PEI, D., AZUMA, M., MASSEY, D., AND ZHANG, L. BGP-RCN: improving BGP convergence through root cause notification. *Comput. Netw. ISDN Syst. 48*, 2 (205), 175–194.

[15] WANG, F., MAO, Z. M., WANG, J., GAO, L., AND BUSH, R. A measurement study on the impact of routing events on end-to-end internet path performance. In *SIGCOMM'06*.

# Pathlet Routing

P. Brighten Godfrey, Scott Shenker, and Ion Stoica
{pbg,shenker,istoica}@cs.berkeley.edu
University of California, Berkeley

## Abstract

Source-controlled multipath routing can be highly beneficial to both sources and to network providers. For a source, the flexibility to choose among multiple paths can improve reliability and path quality. To a network provider, source-controlled routing could be a salable service. Unfortunately, the Internet's interdomain routing protocol, BGP, offers no multipath routing mechanism. Several proposals offer multiple paths, but are limited in the paths they can expose.

This paper introduces a new scheme, pathlet routing, in which networks advertise fragments of end-to-end paths from which a source can assemble an end-to-end route. Pathlet routing is a flexible mechanism that, we show, can emulate a number of existing routing protocols, including BGP and unrestricted source routing. It also enables a new type of routing policy, local transit (LT) policies, which allows networks to control the portions of routes which transit across them, while giving a large amount of flexibility to sources. Finally, we show that LT policies have much better scalability than BGP.

## 1 Introduction

Multipath routing, in which a packet's source selects its path from among multiple options, would be highly beneficial to both end hosts and network providers on the Internet.

For an end host, multipath routing is a solution to two important deficiencies of the Border Gateway Protocol (BGP) [12]: poor reliability [1, 7, 9] and suboptimal path quality, in terms of metrics such as latency, throughput, or loss rate [1, 13]. Both reliability and path quality could be improved via multipath routing. End-hosts (or perhaps edge routers) can observe end-to-end failures quickly and react by switching to a different path, and can observe end-to-end path quality in real time and make decisions appropriate to each application. Greater routing flexibility may bring other benefits as well, such as enabling competition and encouraging "tussles" between different parties to be resolved within the protocol [5].

For a network provider, multipath routing represents a new service that can be sold to customers who desire the benefits described above. In fact, commercial route control products exist today which dynamically select paths based on availability, performance, and cost for multi-homed edge networks [3]; exposing more flexibility in route selection would improve the effectiveness of such products.

Unfortunately BGP has very limited *policy expressiveness*: it greatly constrains the routing policies that a network owner can encode—despite its position as the dominant policy-aware routing protocol! For example, consider a very permissive policy in which a network allows any possible route involving it to be used. Even if a network decided to adopt this policy, perhaps because it had been paid sufficiently, it could not be expressed in BGP.

Several proposals [16, 17] give networks the ability to offer multiple paths, but we argue they are still relatively limited. For example, MIRO uses BGP routes by default, with negotiation between autonomous systems for each additional path; offering too many paths thus involves a prohibitively large amount of state. NIRA [17] allows networks to offer any valley-free path, but *only* valley-free paths, thus making it in that respect more limited than BGP. It also requires assumptions about the network topology.

Can we design a protocol which has rich policy expressiveness, thus allowing network operators to offer a service of greater routing flexibility and hence greater reliability and path quality?

This paper addresses that question with a novel scheme called **pathlet routing**. In pathlet routing, networks advertise *pathlets*—fragments of end-to-end paths—along which they are willing to route. A sender concatenates its selection of pathlets into a full end-to-end source route, which is specified in each packet. Pathlet routing is a simple generalization of both path vector routing and source routing, in terms of the end-to-end paths it can allow and disallow. If each pathlet is a full end-to-end route, the scheme is equivalent to path vector routing. If the pathlets are short, one-hop fragments corresponding to links, then senders can use any of the exponentially large number of paths in the network, as in unrestricted source routing.

Pathlet routing has significant advantages over BGP, including (1) highly expressive policies, and in particular, (2) enabling a new type of routing policy which would offer dramatic improvements in router scalability and in route flexibility for senders. In more detail, we evaluate pathlet routing as follows:

- To demonstrate its policy expressiveness, we show that pathlet routing can efficiently emulate unrestricted source routing, path vector routing (BGP), and two recent multipath routing proposals, MIRO [16] and NIRA [17]. On the other hand there exist protocols like FBR [18] which pathlet routing cannot emulate.

- We show that pathlet routing enables a new class of policies, *local transit (LT) policies*, that allow networks to control the portions of routes which transit across their own networks. Subject to these restrictions, LT policies expose the full flexibility of the Internet's autonomous system (AS)-level routes to sources. The exponentially large set of paths dramatically increases route flexibility relative to BGP and many other policy-aware routing protocols.
- We argue that pathlet routing with LT policies has much better scalability than BGP and MIRO where it matters most—forwarding plane memory usage—and in other scalability metrics is comparable to or better than BGP.

The remainder of the paper proceeds as follows. In Sec. 2, we introduce our pathlet routing mechanism. We evaluate its policy expressiveness in Sec. 3 by comparison with other protocols. Finally, we introduce and evaluate LT policies in Sec. 4.

## 2 Pathlet routing

This section defines pathlet routing, beginning with its building blocks of vnodes and pathlets, and continuing with the control and data planes.

### 2.1 Building blocks

Pathlet routing is built upon virtual nodes, or **vnodes**, which are arbitrary abstract nodes created by an AS to represent the structure of routes within its own network. This virtualization enables flexible control. A vnode might variously represent an entire AS, a presence in a geographic area, a physical router, a type of service within a router, or other granularities that we will demonstrate later. There can be multiple routers for each vnode, and multiple vnodes at each router.

Vnodes need to be associated with physical routers only at peering points between ASes, where neighboring routers announce their vnodes to each other. For other vnodes, the mapping to physical routers is not exposed in the protocol.

Finally but importantly, a vnode can be tagged with a destination, such as an IP prefix.

A **pathlet** represents a sequence of vnodes along which the announcing AS $x$ is willing to route. The first vnode should be in $x$'s own network, but this may be followed by vnodes in $x$ or in other ASes as the pathlet may continue outside $x$'s network.

A pathlet announcement consists of the following information: (1) The path that packets using this pathlet will traverse, given as a sequence of vnodes. Vnode identifiers are local to each AS, so the path lists a pair $(AS, vnode)$ to globally identify each hop. (2) A forwarding identifier sequence (**FIDseq**): a sequence of forwarding identifiers (**FIDs**) to be placed in the packet to instruct the first vnode to use this pathlet.

The first entry of the FIDseq is a FID that uniquely identifies pathlet $p$ within the first vnode in $p$. Importantly, it need not be globally unique like the identifiers in IP source routing, or even unique within an AS. The result is very compact FIDs; for example a vnode handling 256 or fewer unique pathlets could use one-byte FIDs. The remaining FIDs in the FIDseq, if any, identify other pathlets that are used to effect forwarding along this pathlet. (We will see examples in §3.)

### 2.2 Control plane

**Pathlet construction.** A router $r$ announces to each neighbor $r'$ its AS number and a vnode identifier $v$, indicating that every packet sent from $r'$ to $r$ will be interpreted as being directed to vnode $v$. Thus, initially, a router can construct pathlets that include its own vnodes and those of its neighbors' peering points. After learning other ASes' pathlets, it can concatenate multiple pathlets to produce new pathlets spanning multiple ASes.

**Pathlet dissemination.** Any pair of routers, regardless of physical location, may open a control plane connection to disseminate pathlets. Presumably this will be done *at least* by physically adjacent routers. Disseminating information in distributed systems generally can be described as either "push" or "pull", and we will find it useful to include both of these fundamental communication patterns. Intuitively, pushing state is useful at least for bootstrapping, while pulling allows additional state to be created on demand.

First, a router may **push** some subset of the pathlets it knows, according to a local export policy. For example, in several cases we will use a gossiping policy, where each router pushes to its neighbors all the pathlets it has constructed or learned. Second, a router may **pull** pathlets by requesting certain pathlets from a router, such as those relevant to a specified destination. We will use this pull dissemination pattern to emulate both MIRO and NIRA.

### 2.3 Data plane

**Route selection.** Once a router has learned a set of pathlets, it can select from among these a route for each packet. The schemes by which routers learn dynamic path quality and availability and select routes are decoupled from the pathlet routing protocol. Separating these components, as in [17, 18] but unlike BGP, gives room for a wide range of solutions to coexist, such as each network operating a route monitoring service for its users [17], commercial route selection products [3], individual sources probing paths, or a global "Internet weathermap" service.

However, it is likely useful to include a basic availability-monitoring protocol. In the rest of the paper, including the scalability evaluation in Section 4, we will assume the following. We run a global link-state protocol disseminating the state of all links between adjacent vnodes (where adjacency is defined by advertised pathlets). A router keeps an *active* *set* of pathlets, all of whose links are currently available. To find a path to a destination, it can then run any shortest-path algorithm on a graph whose edges are pathlets in the active set.

Note it would also be possible to withdraw and re-advertise pathlets in response to failure and recovery, instead of using a link state protocol. This may incur more overhead if many pathlets use a single link between vnodes.

**Packet forwarding.** The data structure routers use for forwarding is as follows. For each vnode at a router, there is an exact-match table that maps each valid FID $f$ to two pieces of information: (1) the FIDseq of the pathlet corresponding to $f$, and (2) a *next hop rule* to send the packet to the next pathlet routing-aware hop or its destination. Examples include transferring the packet to another vnode at the same router; sending it on a certain outgoing interface; or tunneling it across an intradomain routing protocol like OSPF to the IP address of an egress point.

We now describe the forwarding procedure. A packet contains a sequence of FIDs $(f_1, f_2, \ldots, f_n)$ of the pathlets forming the route to its destination. Initially, this is set by the sender to the selected route. When a packet is sent from router $r'$ to $r$, it is interpreted as arriving at a specific vnode $v$ at $r$ (which $r$ declared to $r'$ in the control plane). Router $r$ checks for $f_1$ in the forwarding table for vnode $v$. If no entry is found, the packet is malformed and is dropped. Otherwise, the table maps $f_1$ to the FIDseq $(f'_1, f'_2, \ldots, f'_k)$ for the pathlet, and a next hop rule. The router verifies that the FIDseq is a prefix of the FIDs in the packet, i.e., $f_i = f'_i$ for $i \in \{1, \ldots, k\}$, to ensure that the route is legitimate for this pathlet, dropping the packet if the match fails. Otherwise, the router pops the first FID off of the route and forwards the packet according to its next hop rule.

In the next section, we will see several examples of the protocol in action.

## 3 Comparison with other protocols

A key goal for pathlet routing is that it enables ASes to express a broad range of routing policies. In this section, we give evidence for this policy expressiveness by showing that pathlet routing can emulate several existing protocols: unrestricted source routing (USR), BGP, MIRO, and NIRA. By "emulate" we mean that pathlet routing can match the same set of end-to-end allowed and prohibited paths. Note that these protocols have substantially different forwarding architectures, but all are special cases of pathlet routing. Finally, we compare pathlet routing and Feedback Based Routing, neither of which can emulate the other.

These sections also serve to illustrate the mechanisms of pathlet routing described in Section 2, to which end we give a fairly detailed description of pathlet routing's emulation of USR and BGP.

**Unrestricted source routing (USR)** at the AS level disseminates the entire topology globally, and lets a source AS use any path in the graph by putting a sequence of ASes in each packet.

At a high level, pathlet routing can emulate USR by using one pathlet for each directed link. We give a more detailed description using the AS-level topology in Fig. 1. Each AS has a vnode representing it, which for convenience we will name $a, b, c, d, e$, each tagged with a destination (IP prefix). Each AS discovers its neighbors' vnodes, and creates pathlets to them. We will write pathlets in the form $(P : F)$ where $P$ is the path of vnodes and $F$ is the FIDseq to be installed in the packet to direct it along $P$. Then node $b$, for example, creates pathlets $(b, a : 1)$, $(b, c : 2)$, and $(b, d : 3)$. Here the FIDseqs have just a single entry because each pathlet is just one hop. The FIDs themselves are arbitrary identifiers unique to each vnode. Suppose the other ASes also announce pathlets for each outgoing link and each terminal vnode, such as $(a, b : 1)$, $(c, d : 2)$. All these pathlets are gossiped globally.

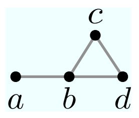


**Figure 1**: Example AS-level network topology.

We can now send a packet at AS $a$ with the route $(1, 2, 2)$ to use the path $(a, b, c, d)$. At vnode $a$ the router looks up index 1 in its forwarding table, finding the pathlet $(a, b : 1)$ and the appropriate outgoing interface along which to forward the packet. It pops off the first FID and sends the packet with route $(2, 2)$ to node $b$. This process repeats until the packet reaches its destination $d$.

**BGP.** Pathlet routing can emulate BGP's routing policies using pathlets which extend all the way to a destination. We give a simple example, again using the topology of Fig. 1. Suppose that in BGP, $d$ advertises a destination IP prefix; $d$ exports routes only to $c$ but all other ASes export all routes; and in the BGP decision process, all ASes select the shortest of their available routes.

We emulate this in pathlet routing as follows. To allow selective route exporting, each AS has one vnode per neighbor, e.g. $d$'s vnodes are $d_b$ and $d_c$, as well as a terminal vnode $d_\bullet$ tagged with its IP prefix(es). It creates a pathlet $(d_c, d_\bullet : 1)$, but no pathlet from $d_b$ to $d_\bullet$: any packet arriving from $b$ must therefore be invalid and hence will be dropped. The other pathlets created are $(c_b, d_c, d_\bullet : 7, 1)$, $(b_a, c_b, d_c, d_\bullet : 4, 7, 1)$. Here the FIDseq has multiple entries, unlike the USR example above. (Note again that the FIDs $4, 7, 1$ are arbitrary.)

A packet can now be sent from AS $a$ to AS $b$ with route $(4, 7, 1)$ to use the path $(b_a, c_b, d_c, d_\bullet)$. The vnode $b_a$ looks up index 4 in its forwarding table, verifies that the associated pathlet's path is a prefix of the packet's path, and forwards the packet to $c_b$ with route $(7, 1)$—which, in turn, forwards it to $d_c$ with route $(1)$, which forwards it to $d_\bullet$ with the empty route $()$, where it is delivered.

**MIRO** [16] is a multipath extension of BGP. In addition to using BGP's paths, a MIRO router $r_1$ can contact any other MIRO router $r_2$ and request alternate routes to a given destination $d$, potentially including preferences regarding which alternate routes are returned. Router $r_2$ responds with some subset $P$ of the alternate routes that $r_2$ has available to $d$, from which $r_1$ picks one route $p \in D$. Finally, a tunnel is constructed: $r_1$ can send a packet along an existing path to some IP address specified by $r_2$, which forwards them to $p$.

MIRO's tunneling is easy to emulate using source routing over pathlets, by placing two pathlets in a packet's route: one representing the tunnel, and a second representing the remainder of the alternate route. To obtain the alternate-route pathlet, a pathlet router can contact any other and pull routes to a specified destination, similar to MIRO. We omit the details.

Pathlet routing can emulate MIRO; is the converse true? MIRO can represent any possible end-to-end path with the appropriate tunnels. But each allowed end-to-end route is constructed and represented explicitly, so there are network topologies for which MIRO would require $\Theta(n!)$ state to represent all possible routes in a graph of $n$ nodes. Thus MIRO cannot scalably emulate pathlet routing (or USR).

**NIRA** [17] offers more choice to sources than BGP, while simultaneously reducing control plane state. NIRA supports routing along so-called *valley-free* paths, which consist of an "up" portion along provider links, then potentially a peering link, and finally a "down" portion along customer links. Each AS learns all of its "up" routes and publishes them at a well-known Name-to-Route Lookup Service (NRLS). A source constructs the first half of a path by choosing a route from its own up-graph, and the second half from the reverse of a route in the destination's up-graph, which it obtains by querying the NRLS.

Pathlet routing can emulate NIRA's routing policy, including its compact routing state. We again use the "pull" mode of obtaining pathlets in place of the NRLS. We concatenate appropriately constructed pathlets representing the up route, a short route across the core, and the down route. We omit details due to space constraints.

MIRO and BGP cannot scalably emulate NIRA because NIRA can compactly represent a very large number of paths by allowing any up-route to be paired with any down-route. On the other hand, NIRA cannot emulate USR, MIRO, BGP, or pathlet routing since it is limited to valley-free routes.

**Feedback Based Routing** (FBR) [18] is an example of a protocol which is incomparable with pathlet routing in the sense that neither protocol can emulate the other. FBR is source routing at the AS level, with each link tagged with an access control rule, which either whitelists or blacklists a packet based on its source or destination IP address. Pathlet routing cannot emulate FBR for two reasons. First, a pathlet router can decide whether to accept a packet based only on its immediately previous hop and on its remaining hops—not based on the full end-to-end path including the source. Second, FBR has both blacklisting and whitelisting, while pathlet routing effectively has only whitelisting, meaning FBR can represent some policies more efficiently.

However, FBR cannot emulate pathlet routing, either. For example, controlling access based on source and destination address ignores intermediate hops which can be taken into account by pathlet routing.

Zhu et al [18] suggested that the access control rules could be more complex than source/destination address matching. Similarly, it is possible that pathlet routing could be extended to include matches on the full end-to-end path and blacklisting; this may be an interesting area of future research.

## 4 Local transit policies

In the previous section, we saw that pathlet routing can efficiently express a wide variety of routing policies, emulating a number of past schemes. In this section we discuss a new class of policies enabled by pathlet routing, **local transit (LT) policies**. LT policies allow networks to control what is arguably the most important aspect of routing policy: the portions of routes which transit across their own networks.

We first define LT policies (§4.1) and argue that LT policies are useful (§4.2) and offer a large amount of route flexibility to sources (§4.3). We then show how LT policies can be implemented in pathlet routing (§4.4) and that this implementation has much better scalability than BGP (§4.5).

### 4.1 Definition

We define **local transit policies** as those policies in which a network $x$'s willingness to carry traffic following some path across its network depends only on the portion of the path that crosses $x$'s network. In other words, under an LT policy the permissibility and cost of some path, according to $x$, is a function only of the ingress and egress point of the path in $x$. Note that LT policies are independent of $x$'s preferences on the paths taken by traffic that $x$ sends, which may be arbitrary.

Consider Fig. 1. If AS $b$ follows an LT policy and allows the path $(a,b,c)$, then it must also allow the path $(a,b,c,d)$, but possibly not $(a,b,d)$ or $(c,b,a)$ which have different ingress or egress points. Essentially, in LT policies, pathlets do not extend beyond a network's ingress/egress points.

### 4.2 Capturing policies and costs

We argue that LT policies represent an important class of routing policy, with two points. First, the *direct* costs to a network of carrying traffic is incurred between its ingress and egress points; for example, the path a packet follows before it arrives at an AS $x$ and after it leaves $x$ do not affect the congestion on $x$'s network. Second, a special case of LT policies—namely valley-free routes [6]—is a common route export policy in BGP today. Valley-free routes can be defined as follows: each neighbor is labeled as a customer, provider, or peer; a BGP route is exported to a neighbor $x$ if and only if either the next hop of the route is a customer or $x$ is customer. This is a function of the ingress and egress point, and hence is an LT policy.

Of course, valley-freeness defines which routes are allowed, but not which ones are preferred. In BGP, this is handled by picking the single most preferred route for each destination as the only route. This brings us to a key challenge: providing the incentive for a transit AS to offer more than the single end-to-end path which is most convenient for that AS.

We envision two ways this incentive could be provided. The first is simply payment between ASes for a multipath routing service which does not discriminate the cost of each path—much as today's transit services have a fixed rate regardless of a packet's destination.

Second, a more discriminatory service could differentiate prices based on the path used. In pathlet routing, it would be easy to annotate each pathlet with a cost. This does not effect a monetary payment, but it does permit the communication of prices to sources, so that payment can happen via some mechanism external to the protocol. Designing such a payment mechanism is outside the scope of this paper. However, note that measurements indicate that ASes' routing preferences are based on next-hops (i.e., egress point) for 98% of IP prefixes [15]. Thus, once a payment mechanism is in place, it would be possible to represent those preferences as LT policies.

It is also possible that ASes would be reluctant to use LT policies because their policies become more explicitly visible. But high level routing policies such as business relationships between neighbors can be inferred from publicly available data from BGP routers today. [14]

## 4.3 Enabling route flexibility

LT policies can provide a dramatic amount of flexibility for sources—potentially, any AS-level path—because the networks' pathlets can be concatenated in an exponentially large number of ways. But this flexibility is limited by the specific LT policies that are chosen.

For example, networks could limit routes to being valley-free. In this case there would still be vastly more flexibility than BGP, but no more than in NIRA. Unlike NIRA, pathlet routing would not be limited to Internet-like topologies. Also, handling exceptions to policies is more difficult in NIRA: a special source routing option is required in the data plane and in the control plane, no mechanism for discovering non-valley-free routes is provided. In pathlet routing, if for example an AS wished to provide transit service between two peers or two providers, this would simply involve advertising two additional pathlets (one in each direction).

However, much more flexibility than valley-free routes may be available. Some incentive for this flexibility exists: for example, a path which is not the cheapest may be worthwhile for real-time applications, or when cheaper paths have failed. Whether ASes choose to expose these routes depends on business decisions and payment mechanisms, but pathlet routing makes it feasible at a technical level.

## 4.4 Implementation

LT policies are easy to implement in pathlet routing: for each ingress $x$ and egress $y$ for which an AS allows routing from $x$ to $y$, it announces a pathlet $(x,y)$, in addition to pathlets

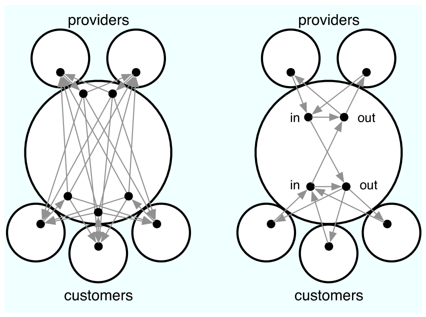


**Figure 2**: vnodes and pathlets for a full LT policy (left) and a class-based LT policy (right) in a single AS with two providers and three customers.

that terminate at its own IP prefixes. However, this results in $d^2$ pathlets for a network with $d$ neighbors. Thus, for large networks, it may be more appropriate to use what we call *class-based LT policies*, in which each neighbor is assigned to a class (such as a geographical region, or business relationship) represented by ingress and egress vnodes, and we use full LT policies between only these class vnodes. These two options are depicted in Fig. 2.

To the best of our knowledge, other policy-aware routing proposals cannot efficiently implement the same set of possible paths that is exposed by LT policies. BGP cannot represent multiple paths per neighbor; MIRO must set up each end-to-end path explicitly, resulting in exponentially more state; NIRA represents only valley-free routes; and FBR examines a packet's source and destination IP address, which does not include information about the intermediate hops, such as a transit from peer to customer vs. peer to provider.

## 4.5 Scalability

We evaluate the scalability of LT policies, which is similar to that of unrestricted source routing. It has been suggested [16] that source routing schemes may not scale since each router must have current knowledge of the entire network. On the contrary, we argue that pathlet routing with LT policies in fact has much better scalability than BGP (and, hence, MIRO [16]) where it matters most—forwarding plane memory usage—and in other scalability metrics is comparable to or better than BGP.

In this evaluation, we assume class-based LT policies with three classes representing the customer, provider, and peer business relationships. (Our conclusions would be substantially similar with full LT policies on all ASes except for the very high degree (top 1%) ASes, or with other class-based LT policies with a limited number of classes.) Following the pattern in Fig. 2 which depicts two classes, using three classes results in $6+d$ pathlets created by each AS with $d$ neighbors, plus 3 pathlets to link each class to a destination vnode with associated IP prefixes. In fact, this may overestimate the number of pathlets, e.g. if the provider$\rightarrow$provider pathlet is omitted in order to disallow transit between providers.

We produce numerical results by analyzing an AS-level topology of the Internet generated by CAIDA [4] and data from APNIC on global IP prefix allocation [2], both from August 18, 2008.

**Forwarding plane memory.** Because it has to operate at high speeds and often uses SRAM rather than commodity DRAM, memory that stores a router's Forwarding Information Base (FIB) is arguably more constrained and expensive than other resources in a router [10]. LT policies dramatically reduce the necessary size of the FIB relative to BGP. Using class-based LT policies as described above in this topology results in a maximum of $2,317$ and a mean of only $6.1$ pathlets to represent an AS. This is also an upper bound on the number of pathlets per router assuming at least one router per AS. In comparison, BGP FIBs would need to store entries for $266,073$ IP prefixes.

**Control plane memory.** In the AS-level measured topology, there are a total of $157,454$ pathlets; tagging vnodes with IP prefixes brings the total to $423,527$ entries. In comparison, the RIB in a BGP router with $d$ neighbors advertising a route to every prefix would contain $266,073 \cdot d$ entries, which is already worse than pathlet routing for $d \geq 2$ and can become problematic in practice for larger $d$ [8].

**Control plane messaging.** Here we employ simple analysis. Consider the effect of a single link failure in a AS-level topology of $n$ nodes, mean degree $d$, and mean path length $\ell$. In pathlet routing, a standard gossiping protocol (§2.3) results in a link state update being sent once along every edge in the graph, i.e., $dn/2$ messages.

In BGP, the number of updates is *at least* the number $N$ of source-destination pairs that were using the failed link. Suppose a random link fails, and let $\ell_{st}$ be the length of a path $s \to t$. Then we have

$$\begin{aligned} E[N] &= \sum_{s,t} \Pr[\text{failed link} \in \text{path } s \to t] \\ &= \sum_{s,t} \frac{\ell_{st}}{nd/2} \\ &= \frac{2(n-1)\ell}{d}. \end{aligned}$$

In the Internet AS-level topology, we roughly have $\ell \approx d \approx 4$, making the messaging cost for both protocols close to $2n$. Moreover, BGP's messaging cost would likely be substantially higher than this lower bound for two reasons. First, because BGP operates at the IP prefix level instead of the AS level, it has in effect about $9n$ destinations [2] rather than $n$. Second, BGP's path exploration can result in much more than one message per source-destination pair with a failed link. [11]

## REFERENCES

[1] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. 18th ACM SOSP*, October 2001.

[2] Routing table report. http://thyme.apnic.net/ap-data/2008/08/18/0400/mail-global.

[3] Avaya. Converged network analyzer. http://www.avaya.com/master-usa/en-us/resource/assets/whitepapers/ef-lb2687.pdf.

[4] CAIDA AS ranking. http://as-rank.caida.org/.

[5] David Clark, John Wroclawski, Karen Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's Internet. In *SIGCOMM*, 2002.

[6] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, December 2001.

[7] Krishna P. Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David Wetherall. Improving the reliability of internet paths with one-hop source routing. In *Proc. OSDI*, 2004.

[8] Elliott Karpilovsky and Jennifer Rexford. Using forgetful routing to control BGP table size. In *CoNEXT*, 2006.

[9] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. In *Computer Communication Review*, 2007.

[10] D. Meyer, L. Zhang, and K. Fall. Report from the iab workshop on routing and addressing. In *RFC2439*, September 2007.

[11] Ricardo Oliveira, Beichuan Zhang, Dan Pei, Rafit Izhak-Ratzin, and Lixia Zhang. Quantifying path exploration in the Internet. In *Proc. Internet Measurement Conference*, October 2006.

[12] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). In *RFC4271*, January 2006.

[13] Stefan Savage, Thomas Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: Informed Internet routing and transport. In *IEEE Micro*, January 1999.

[14] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.

[15] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *IMC*, 2003.

[16] Wen Xu and Jennifer Rexford. MIRO: Multi-path Interdomain ROuting. In *SIGCOMM*, 2006.

[17] Xiaowei Yang, David Clark, and Arthur Berger. NIRA: a new inter-domain routing architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, 2007.

[18] Dapeng Zhu, Mark Gritter, and David Cheriton. Feedback based routing. *Computer Communication Review (CCR)*, 33(1):71–76, 2003.

# Towards A New Internet Routing Architecture: Arguments for Separating Edges from Transit Core

Dan Jen, Michael Meisel
UCLA
{jenster,meisel}@cs.ucla.edu

He Yan, Dan Massey
Colorado State University
{yanhe,massey}@cs.colostate.edu

Lan Wang
University of Memphis
lanwang@memphis.edu

Beichuan Zhang
University of Arizona
bzhang@arizona.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

## ABSTRACT

In recent years, the size and dynamics of the global routing table have increased rapidly along with an increase in the number of edge networks. The relation between edge network quantity and routing table size/dynamics reveals a major limitation in the current architecture; there is a conflict between provider-based address aggregation and edge networks' need for multihoming. Two basic directions towards resolving this conflict have surfaced in the networking community. The first direction, which we dub *separation*, calls for separating edge networks from the transit core, and engineering a control and management layer in between. The other direction, which we dub *elimination*, calls for edge networks to adopt multiple provider-assigned addresses to enable provider-based address aggregation. In this paper, we argue that separation is a more promising approach to scaling the global routing system than elimination, and can potentially be leveraged to bring other architectural improvements to today's Internet that an elimination approach cannot.

## 1. INTRODUCTION

A recent workshop report by the Internet Architecture Board (IAB) [16] revealed that Internet routing is facing a serious scalability problem. The current global routing table size in the default-free zone (DFZ) has been growing at an alarming rate over recent years, despite the existence of various constraints such as a shortage of IPv4 addresses and strict address allocation and routing announcement policies. Though the deployment of IPv6 will remove the address shortage, there is an increasing concern that wide-scale IPv6 deployment could result in a dramatic increase of the routing table size, which may exceed our ability to engineer the operational routing system.

A major contributor to the growth of the routing table is site multihoming, where individual edge networks connect to multiple service providers for improved availability and performance [25]. In the presence of network failures, a multihomed edge network remains reachable as long as any one of its providers remains functioning. In the absence of failures, the edge network can utilize multiple-provider connectivity to maximize some locally defined goals such as higher aggregate throughput, better performance, and less overall cost. However, for an edge network to be reachable through any of its providers, the edge network's address prefix(es) must be visible in the global routing table. In other words, no service provider can aggregate a multihomed edge network's prefix into its own address prefix, even if the edge network may be using a provider-assigned (PA) address block. In addition, more and more edge networks are getting provider-independent (PI) address allocations that come directly from the Regional Internet Registries to avoid renumbering when changing providers. In short, multihoming destroys topology-based prefix aggregation by providers and leads to fast global routing table growth.

Routing table size is not the only scalability concern. Equally important is the amount of updates the system must process. Under the current, flat inter-domain routing system, a connectivity flap to *any* destination network may trigger routing updates to propagate throughout the entire Internet, even when no one is communicating with the unstable destination network at the time. Several measurement studies have shown that the overwhelming majority of BGP updates are generated by a small number of edge networks [12, 20]. Unfortunately, a large-scale, decentralized system such as the Internet will surely contain a small number of poorly managed or even suspicious components.

A number of solutions to the routing scalability problem have been proposed, most recently in the IRTF Routing Research Group [1]. Though all the proposals share a common goal of bringing routing scalability under control by removing PI prefixes and de-aggregated PA prefixes from the global routing system, they differ in how to achieve this goal. We observe that all the proposals fall into one of two categories: *separation* or *elimination*. Solutions in the *separation* category insert a control and management layer between edge networks and today's DFZ, which we refer to as the Internet's *transit core*; edge networks would no longer participate in transit core routing nor announce their prefixes into it. Solutions in the *elimination* category require that edge networks take address assignments from their providers; as a result a multihomed edge network will use multiple PA addresses internally and must modify end hosts to support multihoming.

The purpose of this paper is to compare the two approaches described above and articulate our arguments for supporting the *separation* direction towards routing scalability. Note that, *if fully deployed*, each of the two approaches can be effective in achieving routing scalability in a pervasively multihomed environment. Therefore, our comparison is based on the following high-level criteria to determine the actual impact of a proposed solution: (a) the difficulty in realizing the solutions in the Internet; not only does this involve design issues, but also deployment issues such as the ability to accommodate heterogeneity in the uncertain future, alignment of costs and benefits, and effectiveness in partial deployment; (b) architectural benefits other than scalability – we believe that IP routing and addressing play an essential role in the overall architecture, and that the right kind of changes could help rectify other problems that stem from the same architectural deficiencies.

## 2. Separation

The root cause of the routing scalability problem facing us today is the fact that all the networks operate in the same routing and addressing space. As a result, edge growth is directly reflected in the core routing table size, and unstable edge networks can flood the entire Internet with frequent updates. The separation approach addresses this root cause by separating edge networks from the transit core in the routing architecture. Generally speaking, Internet service providers (ISPs) fall into the category of *transit networks* who operate in the transit core. The business of transit networks is to provide packet transport services for other networks. End-user sites are generally *edge networks*, which only function as sources and sinks of IP packets. After these two types of networks are separated, edge network prefixes are eliminated from the DFZ routing table. Thus, the DFZ routing table will grow with the number of ISPs, which is much smaller and grows slower compared to that of edge networks. More importantly, the separation enables aggregation of routing announcements on a per-ISP basis. Since most routing dynamics are generated by edge networks, separation will also greatly reduce routing churn in the core. A previous study estimates that removing edge networks from the core routing system can reduce the routing table size and routing dynamics by an order of magnitude [15]. However, due to the absence of edge-

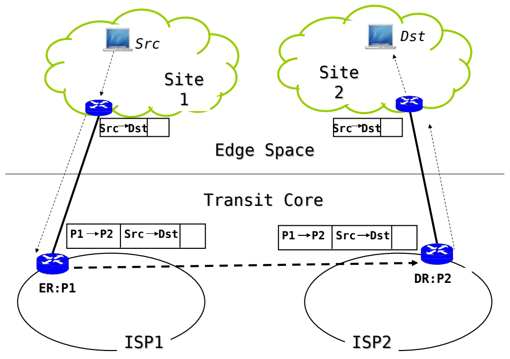


**Figure 1**: Separation via Map & Encap

prefixes from the DFZ, end-to-end data delivery requires mapping a destination edge prefix to one or more transit addresses that correspond to that edge network's attachment points to the transit core.

One realization of separation is Map & Encap [4, 11], which uses IP-in-IP encapsulation to carry packets across the transit core. As shown in Figure 1, each ISP has border routers that perform encapsulation (*Encapsulation Router or ER*) and ones that perform decapsulation (*Decapsulation Router or DR*). When an ER receives a data packet, it must discover the mapping from the packet's destination address to the corresponding DR address. It then encapsulates the packet and forwards it directly to the DR, who decapsulates and delivers the packet to the final destination. Internal ISP routers or routers connecting two ISPs do not need to understand the encapsulation/decapsulation mechanism; they function the same way as they do today, only with a much smaller routing table.

A number of Map & Encap schemes are under active development and discussion in the IRTF Routing Research Group community, including APT [13], LISP [6], and TRRP [10]. There also other types of separation solutions besides Map & Encap. For example, Six-One Router [23] and GSE [19] use address rewriting, which rewrites the packet header to include information about the destination's attachment point to the transit core. A common requirement of all the separation solutions is a mapping system that associate an edge prefix with the corresponding transit addresses.

Designing a mapping system is a challenging problem. Because failures of the mapping system can disrupt packet delivery, it is vitally important to make the mapping system robust against failures and attacks. Other issues include the difficulty of handling a large mapping database and the potential overhead and delay introduced by the mapping and encapsulation process. Note, however, that compared with routing data, mapping data has several characteristics that make it easier to scale and secure. First, a piece of mapping data reflects a long-term

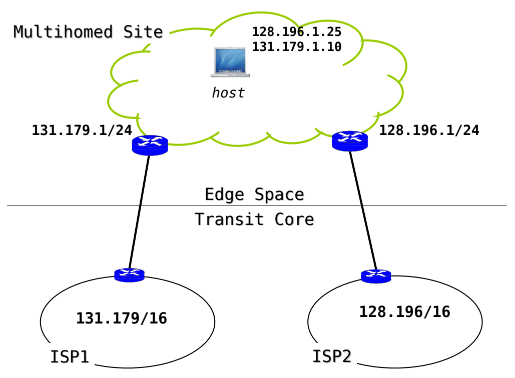


**Figure 2**: Elimination: Pushing multiple PA addresses to hosts

business relationship, so its changes should occur over a relatively longer time scale (*e.g.*, on a monthly basis). Second, the change of one edge network's provider only affects that edge network's mapping data, whereas link failures in the routing system may affect many prefixes.

Several mapping systems designs have been proposed. APT [13] propogates the full mapping table to each ISP. ERs in each ISP use caching internal mapping queries to deliver data. TRRP [10] proposes to set up another DNS to serve mapping information. On the other hand, LISP has proposed a number of different mapping system designs, including LISP-CONS [2], LISP-ALT [5], and LISP-NERD [14]. LISP-NERD distributes the full mapping table to every ER, while LISP-CONS and LISP-ALT build a DNS-like hierarchical overlay to retrieve mapping data when needed. Each design has its own pros and cons in terms of scalability, controllability, cost, performance and security.

## 3. Elimination

In order to achieve routing scalability, the elimination approach enforces provider-based address aggregation by eliminating all PI prefixes and de-aggregated PA prefixes. Each multihomed edge network will receive from each of its providers an address block out of a larger, aggregated block announced by the provider. The multihomed site does not inject PI prefixes or more specific PA prefixes into the routing system. Instead, each host in a multihomed site is given multiple PA addresses. For example, as shown in Figure 2, the host obtains two addresses, one from each of its network's ISPs.

In the elimination approach, each host in a multihomed site must be upgraded to understand how to utilize multiple addresses for packet delivery. Each host must also be able to detect and handle potential failures of its upstream connections to its providers. Otherwise, the benefits of multihoming are lost. One elimination scheme, Shim6 [18], proposes to augment the IP layer for this purpose. Shim6 defines a shim sublayer, placed in the IP layer, which ensures that the transport layers at both ends of a given communication sees the same IP identifiers, even though different IP addresses can be used to forward packets along different paths. Prompt failure detection at the IP layer, however, has proven to be difficult and involves a complex tradeoff between overhead, recovery delay, and impact on transport layers [3].

Elimination can also be achieved through multipath transport [9] [21] which can overcome the above-mentioned issues associated with Shim6. Multipath transport works as follows. To communicate with a destination in a multihomed site, a source first uses DNS to find at least one address for the destination. During the initial three-way TCP handshake, the sender and the receiver exchange all of their addresses. The transport layer then creates multiple subflows from all sender addresses to all receiver addresses. Each subflow performs its own congestion control, and subflows may cooperate with each other. That is, if a packet gets dropped due to one subflow being congested, it can be resent on another uncongested subflow. Assuming transport protocols provide reliable delivery, their closed-loop data exchange provides automatic failure detection. At the same time, the use of multiple paths simultaneously reduces the dependancy on any specific paths. By choosing different (source,destination) address pairs, hosts can utilize the end-to-end paths to achieve higher throughput, better performance and faster failure handling.

Multipath transport realization also faces a number of challenges. Being able to effectively gauge the status of multiple paths requires transmitting a large quantity of data and sophisticated subflow control; not all applications can continuously send large quantities of data (*e.g.*, VoIP connections), and not all end points are suited to perform complex control (*e.g.*, small sensors). It also remains an open question whether all multihomed edge sites are willing to handle multiple PA addresses internally and perform renumbering when changing providers. Moreover, since providers announce aggregated prefixes, failures of links to individual edge networks will no longer be reflected in the routing system; thus even long after a link has failed, new nodes may still attempt to use the failed link because individual transport connections detect failures individually. A single failure inside the core may also affect a large number of transport connections, potentially triggering synchronized recovery attempts by all of them. How to make effective use of multiple addresses and how to detect and recover from failures are open challenges when designing an elimination scheme.

## 4. Why Separation?

If fully deployed, both the separation approach and the elimination approach can achieve the same goal of routing scalability. However, there are important differences that reveal separation to be a better direction than elimination towards routing scalability.

### 4.1 Aligning Cost with Benefits

For any significant change to happen on the Internet, the cost of deployment must align with the benefits of the deployment. Since it is the transit networks that are facing the routing scalability problem, naturally they would have incentive to deploy a solution once it is available. With the separation approach, transit networks can deploy the solution directly and receive the benefits mentioned in the previous section. In other words, the parties responsible for fixing the problem are also the parties who suffer the negative effects if the problem goes unaddressed.

The elimination approach does not change the routing architecture per se; it requires changes of network operations in edge networks and software upgrade at end hosts. At first glance it may appear simpler than the separation approach because it does not need the development of a mapping system. However to remove any of the PI prefixes from the global routing table, the edge networks using PI prefixes must agree to relinquish them and accept PA addresses from their providers instead. The amount of routing table size reduction depends on the number of edge networks that choose to give up their PI prefixes. Under Elimination, transit networks can do nothing but wait for a unanimous action by all the edge networks before the routing table begins to scale. Unfortunately, the routing system has no control over edge site deployment of new solutions. By the time a significant portion of edge sites deploy the new Elimination-based solution(assuming that time ever comes), the routing table may have already grown beyond critical mass.

### 4.2 Accommodating Heterogeneity

The separation approach has the ability to accommodate heterogeneity in network operations. Different networks have different operational practices and considerations. The elimination approach requires all edge networks to use PA addresses, but some networks may not want to do so – it may cause them trouble in renumbering when they switch providers, or they may not want to give end hosts the ability to affect traffic engineering within their network. Since the elimination approach pushes multiple PA addresses all the way to end hosts, what an edge site does within its network can impact the deployment and effectiveness of the elimination approach. On the contrary, the separation approach is flexible in that it does not enforce any particular operational practices within edge networks. Some may choose to give hosts multiple addresses to improve user experience, while others may choose not to in order to tighten traffic control. Both can be accommodated by the separation approach because what an edge site does within its network will not affect the transit core. The Internet is inherently heterogeneous. A main reason for the success of the original Internet design is its ability to accommodate heterogeneity at many different levels, and we believe we must continue to accommodate heterogeneity in any new architecture.

### 4.3 Other Architectural Benefits

Separating edges from the transit core provides additional features that are sorely missing in today's Internet. With separation, an end host can send packets through the transit core, but can no longer address a packet to any specific device inside the transit core. Although the separation does not eliminate any specific security threat, it raises the bar against malicious attacks targeted at the global routing infrastructure. In addition, the mapping layer between edge and core networks can serve as a mounting point for badly-needed control and protection mechanisms, and can also act as a cushion layer between the edge and core, allowing each side to deploy innovations without any involvement of the other side. We now elaborate on each of these benefits.

**Rolling out new protocols.** Internet user innovations don't just happen at the application layer; they also occur at transport and network layers. Intserv/Diffserv, IPv6, and IP multicast, are just a few examples of this. Currently, those innovations require changes to the transit core. In other words, users cannot roll out their new transport and network layer protocols without actions from ISPs which may not have financial incentive to support them.

Separation allows edge networks to develop and deploy new innovative address structures and new protocols. For example, suppose two edge networks $Site_1$ and $Site_2$ could develop a new IPvX address structure. The process of sending an IPvX packet from $Site_1$ to $Site_2$ works as follows. First, the $Site_1$ network routes the IPvX packet to one of its border routers. The router then encapsulates the packet with one of the transit core addresses associated with $Site_2$ (selected by the mapping service). It is essential to note that global agreement on IPvX is not required. Only the mapping service needs to know how to translate an IPvX address to one or a set of transit core addresses.

**DDoS mitigation.** DDoS attacks abuse the open nature of the Internet architecture by sending attack traffic from multiple compromised hosts to a single, overwhelmed target. In the last few years, a number of efforts have been devoted to developing DDoS mitigation solutions [24].

As described in [17], the DDoS mitigation solution space has become increasingly complex over time. One critical question is where to install the various traffic identification, filtering and blocking functions proposed by the solutions. Various proposals place the needed functions at the victim, the victim network entry point, some intermediate point along the path, the source network, and/or the source. We believe that the fundamental reason for this diversity is due to the lack of a common architectural framework for solution development. The existing Internet architecture has no convenient hinges or plug-in points where a defense layer could be easily mounted when needed.

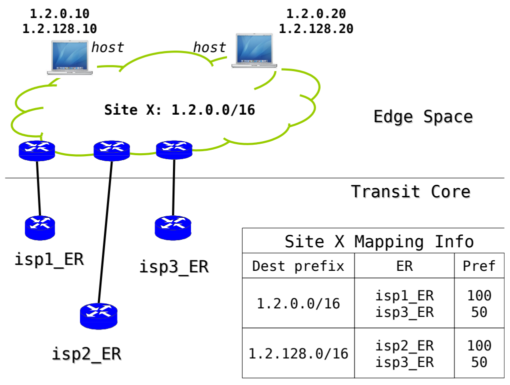
The mapping layer provides such a mounting point. CIRL[7] is one example of approach that leverages the mapping layer. The encapsulation of end-user packets makes it easy to trace attack packets back to the ER, even if they have spoofed source addresses, since the encapsulation header records the addresses of the ER and DR. CIRL lets ERs perform rate-limiting on the traffic going to each specific DR in a way adopted from TVA [24], but without requiring symmetric routing or host changes. Feedback can be provided from DR to ER to adapt the control parameters used for rate limiting.

**Ingress traffic engineering.** Today, multihomed edge sites already have the ability to forward outgoing traffic to whichever of their providers they prefer. However, edge sites may also want control over their inbound traffic flow for load balancing or using a particular provider only as a backup. Today, edge sites' options are limited – they must resort to prefix splitting and BGP trickery.

Under separation, with the help of the mapping service, an edge site can explicitly express its ingress traffic engineering preferences in its mapping information. For example, say edge site $Site_1$ wants to communicate with multihomed edge site $Site_2$. When packets from $Site_1$ to $Site_2$ enter the transit core, the mapping system will need to select one of $Site_2$'s connections to the transit core as the exit. The mapping system has $Site_2$'s explicit preferences for this selection, and can therefore make this decision based on some combination of $Site_1$ and $Site_2$'s preferences. Though these preferences may be in conflict, this tussle between $Site_1$, $Site_2$, and their respective providers plays out only in the mapping service's selection mechanism. That is to say, this decision takes place at the edges of the network and remains distinct from the specialized problem of transporting packets across the core in the most efficient manner.

## 5. Separation is Compatible with Multipath Transport

Multipath transport can actually be a great feature for the transport layer. As multihoming (both host multihoming and site multihoming) becomes more and more prevalent, there is an increasing need for TCP to explicitly select among multiple end-to-end paths. For example, TCP may use multiple paths simultaneously to improve throughput or switch from one path to another to avoid congestion or decrease latency. If hosts have multiple addresses, each of which corresponds to a network attachment point, then they can use different (source,destination) address pairs to utilize all available paths.



| Dest prefix | ER | Pref |
|---|---|---|
| 1.2.0.0/16 | isp1_ER<br>isp3_ER | 100<br>50 |
| 1.2.128.0/16 | isp2_ER<br>isp3_ER | 100<br>50 |

**Figure 3**: Adding Multipath Transport to Separation

One misconception is that multipath transport is inseparably tied to the elimination approach. On the contrary, multipath transport is orthogonal to elimination, and can be used with PI addresses under separation as well. Each edge network can split its provider-independent (PI) prefix into multiple, longer subprefixes, mapping each subprefix to different network attachment points (*e.g.*, a provider's router or an Internet exchange point). Those hosts that desire multipath transport are assigned multiple addresses, one from each subprefix. In this way, hosts get multiple source-destination address pairs providing multiple end-to-end transport paths.

Additionally, the use of PI prefixes for multipath transport provides an opportunity for edge site operators to constrain an end user's path selection. Figure 3 illustrates how this can be done. In the figure, $Site_X$ has a PI prefix 1.2.0.0/16 and is multihomed with three providers, $isp1$, $isp2$, and $isp3$. $Site_X$ only intends to use $isp3$ as a backup – that is, $isp3$ should be used only if the link to $isp1$ or $isp2$ fails. However, $Site_X$ would still like to offer its users some degree of path selection. Thus, $Site_X$ simply splits its prefix into two subprefixes, 1.2.0.0/17 and 1.2.128.0/17, and assigns each end host two addresses. In the mapping table, $Site_X$ explicitly maps 1.2.0.0/17 to $isp1$ with $isp3$ as a backup, and maps 1.2.128.0/17 to $isp2$ with $isp3$ as a backup.

## 6. Summary

In the last few years a number of research efforts have independently reached, or rediscovered, the same basic idea: add a new *layer of indirection* in routing and addressing [15, 22, 26]. In addition to solving the routing scalability problem, this separation solution offers a number of other advantages explained earlier in the paper: enabling end-path selection and multipath routing, raising the barrier against malicious attacks to the routing infrastructure, allowing the edges and the core to freely evolve independently from each other, and providing a boundary around the transit core in the form of a mapping service, where various new security and control functions can be easily implemented.

Host-based solutions, such as Shim6 [18] and multipath transport [9], can be used to realize the elimination approach to the routing scalability problem. An ongoing debate in the IRTF Routing Research Group involves whether separation is still necessary, if and once the multipath transport solution is deployed. In this paper, we point out that the current proposal is actually a combination of two pieces: multipath transport for better transport performance as the primary goal, and elimination of PI prefixes for better routing scalability as a consequence. We explained why separation is preferable over elimination to solve the scalability problem, and sketched out how multipath transport can be incorporated into separation solutions.

In his 1928 article, "Being the Right Size" [8], J.B.S. Haldane illustrated the relationship between the size and complexity of biological entities and concluded that, "for every type of animal there is a most convenient size, and a large change in size inevitably carries with it a change of form." We believe that the same holds true for the Internet. It would not have made any sense to have the original routing system design split the network into two parts, core and edges, with the added complexity of a mapping service in the middle. However, the Internet has grown so large over time that it is now technically and economically infeasible to have all IP devices continue to live in the same address and routing space. Hence, a separation, along with a new mapping service, is both necessary and justified.

## 7. REFERENCES

[1] IRTF Routing Research Group. `http://www.irtf.org/charter?gtype=rg\&group=rrg`.

[2] S. Brim, N. Chiappa, D. Farinacci, V. Fuller, D. Lewis, and D. Meyer. LISP-CONS: A Content distribution Overlay Network Service for LISP. draft-meyer-lisp-cons-04, April 2008.

[3] A. de la Oliva1, M. Bagnulo, A. Garca-Martnez, and I. Soto1. Performance analysis of the reachability protocol for ipv6 multihoming. *Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN 2007).*

[4] S. Deering. The Map & Encap Scheme for Scalable IPv4 Routing with Portable Site Prefixes. Presentation, Xerox PARC, March 1996.

[5] D. Farinacci, V. Fuller, and D. Meyer. LISP Alternative Topology (LISP-ALT). draft-fuller-lisp-alt-02, April 2008.

[6] D. Farinacci, V. Fuller, D. Oran, D. Meyer, and S. Brim. Locator/ID Separation Protocol (LISP). draft-farinacci-lisp-08, July 2008.

[7] C. Frost and M. Mammarella. CIRL: DDoS mitigation in eFIT. Work in progress, 2007.

[8] J. B. S. Haldane. Being the Right Size. `http://irl.cs.ucla.edu/papers/right-size.html`, 1928.

[9] M. Handley, D. Wischik, and M. B. Braun. Multipath Transport, Resource Pooling, and implications for Routing. Presentation at IETF-71, `http://www.cs.ucl.ac.uk/staff/M.Handley/slides/rpool-rrg.pdf`, July 2008.

[10] W. Herrin. Tunneling Route Reduction Protocol (TRRP). `http://bill.herrin.us/network/trrp.html`.

[11] R. Hinden. New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG. *RFC 1955*, 1996.

[12] G. Huston. 2005 – A BGP Year in Review. APNIC 21, March 2006.

[13] D. Jen, M. Meisel, D. Massey, L. Wang, B. Zhang, and L. Zhang. APT: A Practical Tunneling Architecture for Routing Scalability. Technical Report 080004, UCLA, 2008.

[14] E. Lear. NERD: A Not-so-novel EID to RLOC Database. draft-lear-lisp-nerd-04, April 2008.

[15] D. Massey, L. Wang, B. Zhang, and L. Zhang. A Scalable Routing System Design for Future Internet. In *Proc. of ACM SIGCOMM Workshop on IPv6*, 2007.

[16] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. *RFC 4984*, 2007.

[17] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attacks and Defense Mechanisms. *SIGCOMM CCR*, 34(2):38–47, 2004.

[18] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. draft-ietf-shim6-proto-09, October 2007.

[19] M. O'Dell. GSE – An Alternate Addressing Architecture for IPv6. draft-ietf-ipngwg-gseaddr-00, February 1997.

[20] R. Oliveira, R. Izhak-Ratzin, B. Zhang, and L. Zhang. Measurement of Highly Active Prefixes in BGP. In *IEEE GLOBECOM*, 2005.

[21] P. F. Tsuchiya. Efficient and robust policy routing using multiple hierarchical addresses. *SIGCOMM Comput. Commun. Rev.*, 21(4):53–65, 1991.

[22] P. Verkaik, A. Broido, kc claffy, R. Gao, Y. Hyun, and R. van der Pol. Beyond CIDR Aggregation. Technical Report TR-2004-1, CAIDA, 2004.

[23] C. Vogt. Six/One Router: A Scalable and Backwards-Compatible Solution for Provider-Independent Addressing. In *ACM SIGCOMM MobiArch Workshop*, 2008.

[24] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting Network Architecture. *SIGCOMM 2005.*

[25] L. Zhang. An Overview of Multihoming and Open Issues in GSE. *IETF Journal*, 2006.

[26] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling IP Routing with the Core Router-Integrated Overlay. *ICNP 2006.*

# ViAggre: Making Routers Last Longer!

Hitesh Ballani, Cornell University
Paul Francis, Cornell University
Tuan Cao, Cornell University
Jia Wang, AT&T Labs – Research

## Abstract

*This paper presents ViAggre (Virtual Aggregation), a "configuration-only" approach to shrinking the routing table on routers. ViAggre applies to legacy routers and can be adopted independently and autonomously by any ISP. ViAggre is effectively a scalability technique that allows an ISP to modify its internal routing such that individual routers in the ISP's network only maintain a part of the global routing table. We find that ViAggre can shrink the routing table on routers by more than an order of magnitude while imposing negligible traffic stretch.*

## 1 Introduction

The Internet default-free zone (DFZ) routing table has been growing at a rapid rate for the past few years [1]. Looking ahead, there are concerns that as the IPv4 address space runs out, hierarchical aggregation of network prefixes will further deteriorate resulting in a substantial acceleration in the growth of the routing table [2]. A growing IPv6 deployment would worsen the situation even more [3].

The increase in the size of the DFZ routing table has several harmful implications for inter-domain routing. At a technical level, increasing routing table size may drive high-end router design into various engineering limits. For instance, while memory and processing speeds might just scale with a growing routing system, power and heat dissipation capabilities may not [4]. On the business side, it makes networks less cost-effective by increasing the cost of forwarding packets [5] and making it harder to provision networks, not to mention the cost of actually upgrading the routers to account for larger routing tables. As a matter of fact, instead of upgrading their routers, a few ISPs have resorted to filtering out some small prefixes (mostly /24s) which implies that parts of the Internet don't have reachability to each other [6]. It is a combination of these possibilities that led a recent Internet Architecture Board workshop to conclude that scaling the routing system was one of the most critical challenges of near-term Internet design [4].

The severity of the routing scalability problem has also meant that a number of proposals have focussed on reducing the size of the DFZ routing table [3,7–14]. However, all these proposals require changes in the routing and addressing architecture of the Internet and perhaps this has contributed to the fact that none of them have seen deployment.

An alternative is to tackle the routing scalability problem through a series of incremental, cost-effective upgrades. Guided by this, we propose Virtual Aggregation or *ViAggre*, a "configuration-only" solution that shrinks the routing table on routers.[1] ViAggre *applies to legacy routers*. Further, it can be *adopted independently and autonomously by any ISP* and hence the bar for its deployment is much lower. In effect, ViAggre is a scalability technique that allows an ISP to modify its internal routing such that individual routers in the ISP's network only maintain a part of the global routing table. In this paper, we briefly discuss two deployment options through which an ISP can adopt ViAggre.

Preliminary results show that ViAggre can reduce the size of routing tables on routers by more than an order of magnitude while imposing negligible stretch on traffic. However, several important questions remain unanswered. These include the impact of an ISP adopting ViAggre on router load, network complexity and network robustness. We discuss ongoing work that aims to answer these questions. In spite of these questions, we believe that its simplicity makes ViAggre an attractive short-term alternative that can be used by ISPs to cope with the growing routing table till more fundamental, long-term architectural changes can be agreed upon and deployed in the Internet.

## 2 ViAggre design

*ViAggre* allows individual ISPs in the Internet's DFZ to do away with the need for their routers to maintain routes for all prefixes in the global routing table. An ISP adopting ViAggre divides the global address space into a set of *virtual prefixes* that are larger than any aggregatable prefix in use today. For instance, an ISP could divide the IPv4 address space into 128 parts with a /7 representing each part (0.0.0.0/7 to 254.0.0.0/7). Note that such a naïve allocation would yield an uneven distribution of real prefixes across the virtual prefixes. However, the virtual prefixes need not be of the same length and as long as the virtual prefixes together cover the complete address space, the ISP can choose them such that they contain a comparable number of real prefixes.

[1]Specifically, we focus on the router Forwarding Information Base (FIB).

The virtual prefixes are not topologically valid aggregates, i.e. there is not a single point in the Internet topology that can hierarchically aggregate the encompassed prefixes. ViAggre makes the virtual prefixes aggregatable by organizing *virtual networks*, one for each virtual prefix. In other words, a virtual topology is configured that causes the virtual prefixes to be aggregatable, thus allowing for routing hierarchy that shrinks the routing table. To create such a virtual network, some of the ISP's routers are assigned to be within the virtual network. These routers maintain routes for all prefixes in the virtual prefix corresponding to the virtual network and hence, are said to be *aggregation points* for the virtual prefix. A router can be an aggregation point for multiple virtual prefixes and is required to only maintain routes for prefixes in the virtual prefixes it is aggregating.

Given this, a packet entering the ISP's network is routed to a close by aggregation point for the virtual prefix encompassing the actual destination prefix. This aggregation point has a route for the destination prefix and forwards the packet out of the ISP's network. In figure 1 (figure details explained later), router C is an aggregation point for the virtual prefix encompassing the destination prefix and B → C → D is one such path through the ISP's network.

## 2.1 Design Goals

The discussion above describes ViAggre at a conceptual level. However, the design space for organizing an ISP's network into virtual networks is characterized by several dimensions. For example, the flexibility to change the ISP's topology or to change the routers themselves lead to very different architectures, all of which allow for virtual prefix based routing. However, this paper aims for deployability and hence is guided by two major design goals:

1. *No changes to router software and routing protocols*: The ISP should not need to deploy new data-plane or control-plane mechanisms.

2. *Transparent to external networks*: An ISP's decision to adopt the ViAggre proposal should not impact its interaction with its neighbors (customers, peers and providers).

These goals, in turn, limit what can be achieved through the ViAggre designs presented here. Routers today have a Routing Information Base (RIB) generated by the routing protocols and a Forwarding Information Base (FIB) that is used for forwarding the packets. Consequently, the FIB is optimized for looking up destination addresses and is maintained on fast(er) memory, generally on the line cards themselves. All things being equal, it would be nice to shrink both the RIB and the FIB for all ISP devices, as well as make other improvements such as speed up convergence time.

While the basic ViAggre idea can be used to achieve these benefits (section 5), we have not been able to reconcile them with the aforementioned design goals. This paper takes the position that given the performance and monetary implications of the FIB size for routers, an immediately deployable solution that reduces FIB size is useful. Actually, one of the presented designs also shrinks the RIB on routers; only components that are off the data path need to maintain the full RIB. The rest of this section abuses terminology and uses the term "ViAggre" to refer to the specific design being presented.

## 2.2 Design-I

This section details one way an ISP can deploy virtual prefix based routing while satisfying the goals specified in the previous section. The discussion below applies to IPv4 (and BGPv4) although the techniques detailed here work equally well for IPv6. The key concept behind this design is to operate the ISP's routing untouched and in particular, to populate the RIB on routers with the full routing table but to suppress most prefixes from being loaded in the FIB of routers. A standard feature on routers today is to prevent routes for individual prefixes in the RIB from being loaded into the FIB. We have verified this as part of our ViAggre deployment on Cisco 7300 and 12000 routers. Documentation for Juniper [15] and Foundry [16] routers specify this feature too. We use this as described below.

The ISP does not modify its routing setup – the ISP's routers participate in an intra-domain routing protocol that establishes internal routes through which the routers can reach other while BGP is used for inter-domain routing just as today. For each virtual prefix, the ISP designates some number of routers to serve as aggregation points for the prefix and hence, form a virtual network. Each router is configured to only load prefixes belonging to the virtual prefixes it is aggregating into its FIB while suppressing all other prefixes.

Given this, the ISP needs to ensure that packets to any prefix can flow through the network in spite of the fact that only a few routers have a route to the prefix. This is achieved as follows:

– *Connecting Virtual Networks.* Aggregation points for a virtual prefix originate a route to the virtual prefix that is distributed throughout the ISP's network but not outside. Specifically, an aggregation point advertises the virtual prefix to its iBGP peers. A router that is not an aggregation point for the virtual prefix would choose the route advertised by the aggregation point closest to it and hence, forward packets destined to any prefix in the virtual prefix to this aggregation point.[2]

– *Sending packets to external routers.* When a router receives a packet destined to a prefix in a virtual prefix it is aggregating, it can look up its FIB to determine the route for the packet. However, such a packet cannot be forwarded in the normal hop-by-hop fashion since a router that is not an aggregation point for the virtual prefix in question might forward the packet back to the aggregation point, resulting in a loop. Hence, the packet must be tunneled from the aggregation point to the external router that advertised the prefix. While the ISP can probably choose from many tunneling technologies, the description in the rest of this paper assumes the use of MPLS Label Switched Paths (LSPs) for such tunnels.

However, an LSP from the aggregation point to an external router would require cooperation from the neighboring ISP. To avoid this, every edge router of the ISP initiates a LSP for every external router it is connected to. Thus, all the ISP routers need to maintain LSP mappings equal to the number of external routers connected to the ISP, a number much smaller than the routes in the DFZ routing table. Note that even though the tunnel endpoint is the external router, the edge router can be configured to strip the MPLS label from the data packets before forwarding them onto the external router. This, in turn, has two implications. First, external routers don't need to be aware of the adoption of ViAggre by the ISP. Second, even the edge router does not need a FIB entry for the destination prefix, instead it chooses the external router to forward the packets to based on the MPLS label of the packet. The behavior of the edge router here is similar to the penultimate hop in a VPN scenario and is achieved through standard configuration.

We now use a concrete example to illustrate the flow of packets through an ISP network that is using ViAggre. Figure 1 shows the relevant routers. The ISP is using /7s as virtual prefixes and router C is an aggregation point for one such virtual prefix 4.0.0.0/7. Edge router D initiates a LSP to external router E with label $l$ and hence, the ISP's routers can get to E through MPLS tunneling. The figure shows the path of a packet destined to prefix 4.0.0.0/24, which is encompassed by 4.0.0.0/7, through the ISP's network. The path from the ingress router B to the external router E comprises of three segments:

1. VP-routed: Ingress router B is not an aggregation point for 4.0.0.0/7 and hence, forwards the packet to aggregation point C.

2. MPLS-LSP: Router C, being an aggregation point for 4.0.0.0/7, has a route for 4.0.0.0/24 with NEXT-HOP set to E. Further, the path to router E involves tunneling the packet with MPLS label $l$.

3. Map-routed: On receiving the tunneled packet from router C, egress router D looks up its MPLS label map and forwards the packet to external router E after stripping off the MPLS header.

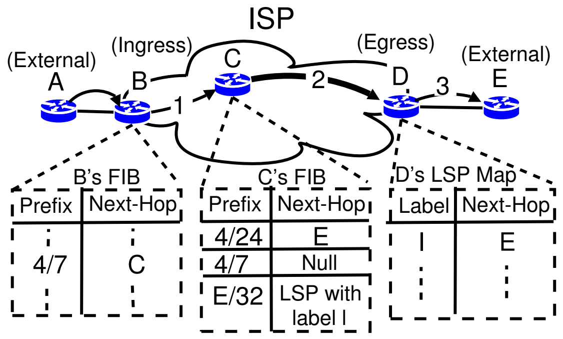


**Figure 1**: Path of packets destined to prefix 4.0.0.0/24 (or, 4/24) between external routers A and E through an ISP with ViAggre. Router C is an aggregation point for virtual prefix 4.0.0.0/7 (or, 4/7).

The description above suggests that all of the ISP's traffic would need to be routed through some aggregation point. However, several past studies from as early as 1999 have shown that a large majority of Internet traffic is destined to a very small fraction of prefixes [17–20]. Consequently, routes to these *popular prefixes* will be maintained by all routers so that ViAggre's impact on the ISP's traffic is minimal.

## 2.3 Design-II

The second design offloads the task of maintaining the full RIB to devices that are off the data path. ISPs commonly use route-reflectors for scalable internal distribution of BGP prefixes and we require only these route-reflectors to maintain the full RIB. For ease of exposition, we assume that the ISP is already using per-PoP route reflectors that are off the data path, a common deployment model.

In the proposed design, the external routers connected to a PoP are made to peer with the PoP's route-reflector.[3] This is necessary since the external peer may be advertising the entire DFZ routing table and all these routes obviously cannot reside on any given router. The route-reflector also has

[2]All other attributes for the routes to a virtual prefix are the same and hence, the decision is based on the IGP metric to the aggregation points. Hence, "closest" means closest in terms of IGP metric.

[3]Note that these will be eBGP multihop peerings since the route-reflector is not directly connected to the external routers.

iBGP peerings with other route-reflectors and with the routers in its PoP. Egress filters are used on the route-reflector's peerings with the PoP's routers to ensure that a router only gets routes for the prefixes it is aggregating. This shrinks both the RIB and the FIB on the routers. The data-plane operation and hence, the path of packets through the ISP's network remains the same as with the previous design.

## 2.4 Design Comparison

As far as the configuration is concerned, configuring suppression of routes on individual routers in design-I is comparable, at least in terms of complexity, to configuring egress filters on the route-reflectors. In both cases, the configuration can be achieved through a BGP route-map; in design-I, the route-map is applied at individual routers while in design-II, it is applied to the iBGP peerings of the route-reflectors.

Design-II, apart from shrinking the RIB on the routers, does not require the route suppression feature on routers. However, it does require the ISP's eBGP peerings to be reconfigured which could represent a substantial overhead. It may also seem that the second design impacts the ISP's robustness since the failure of a route-reflector in a PoP would severely impact the PoP's routers. However, this is not qualitatively any different from the use of route-reflectors today and is typically accounted for by using redundant route-reflectors.

## 3 ViAggre Impact

ViAggre causes packets to take paths longer than native paths. Apart from the stretch imposed on traffic, this leads to extra load on the ISP's routers and links. In the first part of this section, we study how an ISP may choose the aggregation points for its virtual prefixes so as to shrink the FIB on its routers while constraining traffic stretch. We comment on the load increase issue in section 3.3.

## 3.1 Assigning Aggregation Points

Ideally, an ISP would like to deploy an aggregation point for all virtual prefixes in each of its PoPs such that for every virtual prefix, a router chooses the aggregation point in the same PoP and hence, the stretch imposed on the ISP's traffic is minimal. However, this is often not possible in practice. This is because ISPs, including tier-1 ISPs, often have some small PoPs with just a few routers and therefore there may not be enough cumulative FIB space in the PoP to hold all the actual prefixes.

Hence, the ISP needs to be smart about the way it designates routers to aggregate virtual prefixes. To this effect, we have implemented a very simple tool that uses an ISP's topology and information about router memory constraints to determine an

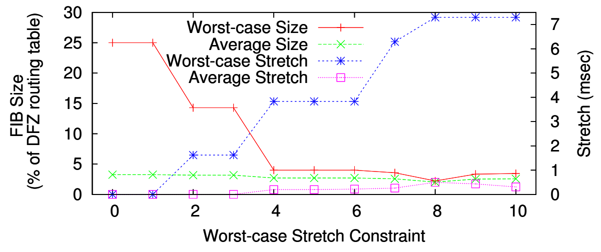


**Figure 2**: Variation of FIB size and stretch with the constraint on the worst-case stretch.

assignment of aggregation points to the ISP routers. This tool lets us explore the trade-off between traffic stretch and FIB size offered by ViAggre. Specifically, the parameters of interest here include the *Worst-case FIB size* which refers to the largest FIB across the ISP's routers and the *Worst-case stretch* which refers to the maximum stretch imposed across traffic to all destination prefixes from all PoPs. We also define *Average-case stretch* as the average of the stretch imposed on traffic across all PoPs. The tool uses a greedy algorithm to assign the ISP's routers to aggregate virtual prefixes so as to minimise the worst-case FIB size while ensuring that the worst-case stretch is within a specified bound. While trivial, such a constraint would probably be critical for a practical deployment so that the ISP can ensure that its existing SLAs with managed Internet customers are not breached due to ViAggre. In the interest of brevity, we don't discuss the details of our algorithm here; however, below we discuss the application of this tool.

## 3.2 Tier-1 ISP study

We used the router-level topology and BGP routing tables of a tier-1 ISP to determine the impact of the ISP adopting ViAggre. Instead of using virtual prefixes of the same length, we programmatically selected the virtual prefixes such that the distribution of real prefixes across them is relatively uniform. This led to a total of 1024 virtual prefixes that are in the FIB of every router.

We then used the aforementioned algorithm to determine an assignment of aggregation points that minimizes the worst-case FIB size given a constraint on the worst-case stretch. Figure 2 shows the (average and worst-case) FIB size and stretch for different constraints. As expected, the worst-case FIB size reduces as the stretch constraint is relaxed. For the ISP being studied, ViAggre can yield a more than 20x reduction in FIB size while ensuring that the worst-case stretch is less than 4 msec and the average stretch is less than 0.2 msec. Note that choosing the virtual prefixes such that the distribution of actual prefixes across them is not skewed provides the algorithm with greater flexibility in choosing aggregation points. For instance, simply using /7s as virtual prefixes yields a reduction of ≈12x with the same 4 msec constraint.

### 3.3 Router Load

A naïve ViAggre deployment can cause a significant increase in traffic load across the ISP's routers and links, not to mention the resulting interference with the ISP's traffic engineering. For instance, for the ISP discussed in section 3.2, calculations using the ISP's traffic matrix yielded that a deployment with worst-case stretch constrained to 4 msec would reduce the FIB size by more than 20x but would also cause a median increase in router load by 31.3%.

As mentioned earlier, the ISP can alleviate the load concern by taking advantage of the skewed distribution of traffic across Internet prefixes, which also holds for the ISP we studied. For instance, we found that 5% of the most popular prefixes were carrying 96.7% of the ISP's traffic. Hence, the ISP can maintain routes to these popular prefixes on all its routers to greatly reduce both the load increase and the amount of traffic that gets stretched due to ViAggre. While we don't present the details of our load analysis, considering 5% of the prefixes to be popular would drop the median and the worst-case load increase across the routers to less than 1% of the router's native load.

## 4 Related Work

A number of efforts have tried to directly tackle the routing scalability problem through clean-slate designs. One set of approaches try to reduce routing table size by dividing edge networks and ISPs into separate address spaces [3,7–9,13]. Alternatively, it is possible to encode location information into IP addresses [10–12] and hence, reduce routing table size. Finally, an interesting set of approaches that trade-off stretch for routing table size are *Compact Routing* algorithms; see [21] for a survey of the area.

The use of tunnels has long been proposed as a routing scaling mechanism. VPN technologies such as BGP-MPLS VPNs [22] use tunnels to ensure that only PE routers need to keep the VPN routes. As a matter of fact, ISPs can and probably do use tunneling protocols such as MPLS and RSVP-TE to engineer a BGP-free core [23]. However, edge routers still need to keep the full FIB. With ViAggre, none of the routers on the data-path need to maintain the full FIB. A number of techniques are being used by router vendors to alleviate the impact of routing table growth, including FIB compression [23] and route caching [23]. In recent work, Kim et. al. [24] use relaying, similar to ViAggre's use of aggregation points, to address the VPN routing scalability problem.

Over the years, several articles have documented the existing state of inter-domain routing and delineated requirements for the future [25–27]; see [26] for other routing related proposals. RCP [28] and 4D [29] argue for logical centralization of routing in ISPs to provide scalable internal route distribution and a simplified control plane respectively. We note that ViAggre fits well into these alternative routing models. As a matter of fact, the use of route-reflectors in design-II is similar in spirit to RCSs in [28] and DEs in [29].

## 5 Discussion and Future work

**Pros.** The ViAggre design presented in this paper can be *incrementally deployed* by an ISP since it does not require the cooperation of other ISPs and router vendors. What's more, an ISP could experiment with ViAggre on a limited scale (a few virtual prefixes or a limited number of routers) to gain experience and comfort before expanding its deployment. Also, the use of ViAggre by the ISP does not restrict its routing policies and route selection. Actually, design-I does not modify the ISP's routing setup and hence all properties such as convergence times, etc. remain the same. Finally, there is *incentive for deployment* since the ISP improves its own capability to deal with routing table growth.

**Management Overhead.** ViAggre imposes a significant configuration burden on the ISP. For the first design, this includes configuring route suppression on individual routers and configuring LSP advertisements on the border routers. Further, the ISP needs to make a number of deployment decisions such as choosing the virtual prefixes to use, deciding where to keep aggregation points for each virtual prefix, which prefixes to consider popular, and so on. Apart from such one-time or infrequent decisions, ViAggre may also influence very important aspects of the ISP's day-to-day operation such as maintenance, debugging, etc.

To study this overhead, we have deployed ViAggre on the WAIL testbed [30] comprising of Cisco 7300 routers. We have already developed a tool that extracts information from existing router configuration files and other ISP databases to generate the configuration files that would be needed for ViAggre deployment. We are also developing a planning tool that would take constraints such as stretch and load constraints and other high-level goals as its input and generate ways that an ISP can deploy ViAggre so as satisfy these. While these tools are specific to the routers, ISP data and other technologies in our deployment, we believe that they can buttress our argument that ViAggre offers a good trade-off between the management overhead and increased routing scalability.

**Router changes.** Routers can be changed to be ViAggre-aware and hence, make virtual prefixes first-class network objects. This would do away with the configuration complexity that ViAggre entails and hence, make it more palatable for an ISP. We, in cooperation with a router vendor, are exploring this option [31].

**Clean-slate ViAggre.** Applying the basic concept of virtual networks in an inter-domain setting to induce a routing hierarchy that is more aggregatable can accrue benefits beyond shrinking the router FIB. The idea here is to have virtual networks for individual virtual prefixes span domains such that even the RIB on a router only contains the prefixes it is responsible for. This would reduce both the router FIB and RIB and in general, improve routing scalability.

To summarize, preliminary results show that an ISP can use ViAggre to substantially shrink the FIB on its routers and hence, extend the lifetime of its installed router base. The ISP may have to upgrade the routers for other reasons but at least it is not driven by DFZ growth over which it has no control. While it remains to be seen whether most, if not all, of the configuration and management overhead introduced by ViAggre can be eliminated through automated tools, we believe that the simplicity of the proposal and its possible short-term impact on routing scalability suggest that is an alternative worth considering.

## References

[1] G. Huston, "BGP Reports," Dec 2007, http://bgp.potaroo.net/.

[2] T. Narten, "Routing and Addressing Problem Statement," Internet Draft draft-narten-radir-problem-statement-01.txt, Oct 2007.

[3] D. Massey, L. Wang, B. Zhang, and L. Zhang, "A Proposal for Scalable Internet Routing & Addressing," Internet Draft draft-wang-ietf-efit-00, Feb 2007.

[4] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," Internet Draft draft-iab-raws-report-02.txt, Apr 2007.

[5] T. Li, "Router Scalability and Moore's Law," Oct 2006, http://www.iab.org/about/workshops/routingandaddressing/Router_Scalability.pdf.

[6] D. Hughes, Dec 2004, PACNOG list posting http://mailman.apnic.net/mailing-lists/pacnog/archive/2004/12/msg00000.html.

[7] S. Deering, "The Map & Encap Scheme for scalable IPv4 routing with portable site prefixes," March 1996, http://www.cs.ucla.edu/~lixia/map-n-encap.pdf.

[8] D. Farinacci, V. Fuller, D. Oran, and D. Meyer, "Locator/ID Separation Protocol (LISP)," Internet Draft draft-farinacci-lisp-02.txt, July 2007.

[9] M. O'Dell, "GSE–An Alternate Addressing Architecture for IPv6," Internet Draft draft-ietf-ipngwg-gseaddr-00.txt, Feb 1997.

[10] P. Francis, "Comparision of geographical and provier-rooted Internet addressing," *Computer Networks and ISDN Systems*, vol. 27, no. 3, 1994.

[11] S. Deering and R. Hinden, "IPv6 Metro Addressing," Internet Draft draft-deering-ipv6-metro-addr-00.txt, Mar 1996.

[12] T. Hain, "An IPv6 Provider-Independent Global Unicast Address Format," Internet Draft draft-hain-ipv6-PI-addr-02.txt, Sep 2002.

[13] X. Zhang, P. Francis, J. Wang, and K. Yoshida, "Scaling Global IP Routing with the Core Router-Integrated Overlay," in *Proc. of ICNP*, 2006.

[14] P. Verkaik, A. Broido, kc claffy, R. Gao, Y. Hyun, and R. van der Pol, "Beyond CIDR Aggregation," CAIDA, Tech. Rep. TR-2004-1, 2004.

[15] "JunOS Route Preferences," Jul 2008, http://www.juniper.net/techpubs/software/junos/junos60/swconfig60-routing/html/protocols-overview4.html.

[16] "Foundry Router Reference," Jul 2008, http://www.foundrynetworks.co.jp/services/documentation/srcli/BGP_cmds.html.

[17] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. of Internet Measurment Workshop*, 2002.

[18] W. Fang and L. Peterson, "Inter-As traffic patterns and their implications," in *Proc. of Global Internet*, 1999.

[19] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: methodology and experience," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, 2001.

[20] N. Taft, S. Bhattacharyya, J. Jetcheva, and C. Diot, "Understanding traffic dynamics at a backbone PoP," in *Proc. of Scalability and Traffic Control and IP Networks SPIE ITCOM*, 2001.

[21] D. Krioukov and kc claffy, "Toward Compact Interdomain Routing," Aug 2005, http://arxiv.org/abs/cs/0508021.

[22] E. Rosen and Y. Rekhter, "RFC 2547 - BGP/MPLS VPNs," Mar 1999.

[23] J. Scudder, "Router Scaling Trends," APRICOT Meeting, 2007, http://submission.apricot.net/chatter07/slides/future_of_routing.

[24] C. Kim, A. Gerber, C. Lund, D. Pei, and S. Sen, "Scalable VPN Routing via Relaying," in *Proc. of ACM SIGMETRICS*, 2008.

[25] E. Davies and A. Doria, "Analysis of Inter-Domain Routing Requirements and History," Internet Draft draft-irtf-routing-history-07.txt, Jan 2008.

[26] N. Feamster, H. Balakrishnan, and J. Rexford, "Some Foundational Problems in Interdomain Routing," in *Proc. of Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[27] Z. M. Mao, "Routing Research Issues," in *Proc. of WIRED*, 2003.

[28] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and Implementation of a Routing Control Platform ," in *Proc. of Symp. on Networked Systems Design and Implementation (NSDI)*, 2005.

[29] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communications Review*, October 2005.

[30] P. Barford, "Wisconsin Advanced Internet Laboratory (WAIL)," Dec 2007, http://wail.cs.wisc.edu/.

[31] P. Francis, X. Xu, and H. Ballani, "FIB Suppression with Virtual Aggregation and Default Routes," Internet Draft draft-francis-idr-intra-va-01.txt, Sep 2008.

# Good Things Come to Those Who (Can) Wait

*or how to handle Delay Tolerant traffic and make peace on the Internet*

Nikolaos Laoutaris
Telefonica Research
nikos@tid.es

Pablo Rodriguez
Telefonica Research
pablorr@tid.es

## ABSTRACT

Recent revelations that ISPs selectively manipulate P2P traffic have sparked much public discussion. Underlying this issue is the misalignment of interests between consumers on one hand who desire bulk transfers at flat rates, and ISPs on the other hand who are bound by budget and capacity constraints. Our thesis is that much of the tension can be alleviated by time-shifting traffic away from peak hours taking advantage of its *Delay Tolerant* (DT) nature. We propose two solutions for doing this. The first one offers incentives to end-users to shift their DT traffic and yet be compatible with flat-rate charging schemes. The second one posits augmenting the network with additional storage in the form of Internet Post Offices which can let ISPs perform store-and forward relaying of such DT traffic.

## 1. INTRODUCTION

The long term planning and deployment of infrastructure has always been a challenging task that requires predicting variables and future events that are unknown when the planning decisions are made – *"what will be the car usage in 5 or 10 years?"*, or, *"which areas in the vicinity of a large metropolis will develop more and thus require new roads and train connection to the city center'?'*. Similar questions are asked in the domain of networks for things like the future rate needs of residential and corporate connections, the dimensioning of backbones, and the peering agreements between ISPs.

**Applications, Access, Backbone – Chasing the ever moving target:** Much like in the previous examples taken from transportation, coming up with accurate predictions for the dimensioning of a network is a very hard task since there is too much uncertainty involved. For example, technological advances in access and backbone links often occur independently thus moving the bottlenecks anywhere between the end-user premises and the the network core [2]. At the application layer, the continuous introduction of new applications like P2P systems, user generated content websites, and multiplayer online games keeps changing the shape of network traffic matrices over increasingly shrinking time scales. Further, the difficulty of making accurate predictions is made worse by the fact that end-users are becoming increasingly involved in the introduction of new high bandwidth consuming applications and data. All the above points illustrate the difficulty of accurately predicting the future resource requirements of next generation networks. Therefore, bottlenecks are expected to keep appearing at one point of the network or the other and identifying them will continue being a chase of an ever moving target.

In such a volatile environment, it is important to have tools for relieving bottlenecks promptly and thus make time for network dimensioning to come up with more long term solutions. A prime objective of such tools would be to promote further the efficient usage of resources under the current triplet of applications, access, and backbone technology. Resource wastage – often referred to as "fat" in economics jargon – should be identified and removed promptly. But where can we find "fat" on the current Internet?

**Delay tolerant applications and traffic:** Consider the familiar example of a user who on receiving a suggestion, or after browsing a collection of media or applications, starts a large download that can take anywhere from a few to several hours. This is typically followed by additional time before the end-user really makes use of the information, e.g., watch the movie or install and start using the application. Such *Delay Tolerant* (DT) applications and their traffic allow much room for flexible scheduling and transmission, unlike interactive ones, like web browsing or video streaming, where requests and transmissions have to occur nearby in time.

*DT applications therefore permit for a time-expansion of basic Internet scheduling and routing mechanisms.* Internet routing has in the last few years gone through a *spatial-expansion* through technologies like overlay routing [3], anycast routing [4, 5], locality aware P2P network formation [1, 13, 6], etc. Scheduling, however, has not yet seen its own expansion, as it has been severely limited within the tight time scales imposed by congestion avoidance through TCP. The latter was designed under the overarching assumption that communication is interactive and intolerant to delay, which is not true for the aforemention class of DT applications.

As a consequence, both end-users and the network treat DT traffic like ordinary interactive traffic. Bulk downloads are initiated and accepted in the network during the hours of peak load despite the fact that the information they carry may be consumed several hours later. In the domain of transportation, such issues have been resolved through legislation. For example, in many places supply trucks are not allowed to make deliveries during commute hours, or access some highways during peak weekend traffic. On the Internet, however, there is no mechanism to prohibit DT applications from using limited resources during peak hours that interactive applications would value more. In that sense, DT traffic appears as "fat" in the pipes of ISPs.

**Our contribution:** In this paper we start by first identifying two basic causes behind the currently inefficient handling of DT traffic. The first one is the *lack of appropriate incentives for end-users* to self-select and schedule efficiently the transmission of DT traffic, e.g., postpone it until non-peak hours. This is a direct consequence of the prevailing flat-rate charging scheme that does not reward residential users that make efficient usage of network resources. Secondly, we point to a *lack of mechanisms on the part of the network* for identifying and handling DT traffic independently of how and when it is injected by the end-users. We propose two fixes with different pros and cons.

- Provide incentives under flat-rate charging: We argue that it is possible to keep flat-rate charging but still be able to incentivize the end-users to postpone their DT transfers until times of low utilization. The trick is to reward them for keeping their traffic low during peak hours, by providing them with bonus "higher-than-the-purchased" access rates during non-peak hours. For ISPs this makes sense since unutilized bandwidth costs nothing, whereas additional bandwidth during peak hours requires more investment in equipment.
- Allow the network to time-shift the DT traffic: We propose network attached storage in the form of *Internet Post Offices* (or IPOs) that will collect DT traffic in an opaque way from the end-users, and perform efficient transmission and scheduling based on the background load and the peering relationships between ISPs. We discuss two scenarios, one in which the local ISP operates the local IPO, and one in which IPOs are installed and operated by CDNs specializing in DT transfers. Such CDNs can become the catalyst for resolving tensions between ISPs and heavily consuming end-users.

Both solutions modify the flow of DT traffic, the first one at the source and the second inside the network. In the remainder of the article, we first discuss the impact of flat-rate charging on the way that end-users generate and transmit DT traffic, and then move on to elaborate on our proposals.

## 2. FLAT-RATE BROADBAND ACCESS

Despite the strong arguments [7] that economists have presented against flat-rate charging and in favor of more elaborate usage-based charging, flat-rate remains ubiquitous and has become a defacto standard for residential broadband access [10]. Undeniably, most of the appeal of flat-rate charging stems from its simplicity. It is easily communicable to the end-users who, in addition, feel safe by not having to worry about unpleasant surprises at the end of the month when the bill arrives, something not at all uncommon under usage-based charging schemes for other services like electricity and gas. For network operators, flat-rate charging obliterates the need to perform complex computations for calculating the charged amount of each user. On the negative side, flat-rate introduces the following problems.

- Unfairness: The common monthly amount that an ISP charges all users depends in the long run from the level of consumption of individuals and thus light users end up subsidizing the bandwidth of heavy users. When the difference between minimum and maximum consumption is not large, e.g., as in "all-you-can-eat" restaurants where the size of the human stomach puts very rigid bounds, then this is not much of a problem. In broadband access, however, as the rates increase, so does the maximum amount of unfairness due to cross-subsidy.
- Lack of incentives for efficient use of resources: Flat rate does not offer any incentives to end-users for making efficient use of network resources. Thus, even if a user knows that he won't be able to watch a movie until late at night or the weekend, there is no incentive for him not to start the download immediately. The reason is that postponing the download would place on the user the burden of having to remember to initiate it after the peak hours. Such wasteful usage habits combined with multimegabit Fiber-To-The-Home (FTTH) technologies can put an all too heavy strain on the infrastructure of an ISP. This partially explains why some ISPs have not yet released FTTH despite it being already a mature technology for the access.

There exists some partial solutions to these limitations. For example, usually one has the choice of multiple classes of flat-rate [7], each with different transmission rate and monthly cost. This however requires users to be able to predict accurately their bandwidth requirement and be willing to commit to it, as changing

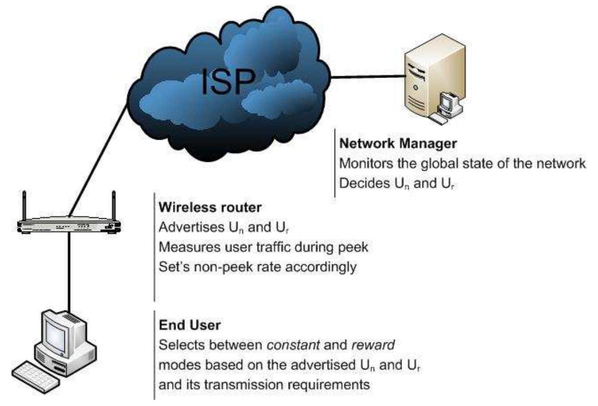

ISP
Network Manager
Monitors the global state of the network
Decides $U_n$ and $U_r$
Wireless router
Advertises $U_n$ and $U_r$
Measures user traffic during peek
Set's non-peek rate accordingly
End User
Selects between *constant* and *reward*
modes based on the advertised $U_n$ and $U_r$
and its transmission requirements

**Figure 1: Architecture for implementing the *reward/constant* incentive scheme.**

plans frequently based on usage habits is cumbersome. Similarly, some ISPs provide a capped download volume per month during peak hours and uncapped during off-peak hours. Although this allows for some flexibility (e.g., DT downloads can be put on crontab), it still ties the user to a particular daytime volume, and prohibits any kind of dynamic adjustment based on current usage habits. Unlike these two schemes, our proposal in the next section gives the end-user a very basic ability to modulate his available maximum rate according to his daily usage habits. Borrowing a term from the area of randomized algorithms, we will argue that network resource efficiency has much to gain from such an incentive scheme that embodies the *power of two choices*.

## 3. BUILDING INCENTIVES IN FLAT-RATE

In this section we show how to use the maximum allowed daily download volume as the means for building an incentive scheme into flat-rate charging.

### 3.1 Basic idea

A user pays a flat monthly amount for broadband access which entitles him to two different usage modes. In the first one (we will call it *constant*) the maximum allowed transmission rate has a constant value $U$ throughout the duration of a day.[1] In the second one (we will call it *reward*) the maximum allowed transmission rate has value $U_n < U$ during the $B$ "busy hours" of the network, and $U_r > U$ during the remaining $24-B$ hours of the day. The user can switch between the two modes on a day-by-day basis as will be explained next. *Reward* is designed to incentivize users to move all or part of their delay tolerant traffic away from the busy hours. The idea is pretty simple: by being "nice" to the network and keeping your rate below $U_n$ (hence the subscript $_n$), you get "rewarded" with a higher rate $U_r$ during the non-busy hours (hence the subscript $_r$). The values $U, U_n, U_r, B$ must satisfy $U_n \cdot B + U_r \cdot (24-B) >> U \cdot 24$, i.e., permit a much higher overall daily transferred volume under *reward* than under *constant* with 100% utilization. P2P users with some ability for "Delayed Gratification" [11] would naturally respond to such a scheme.

The aforementioned example involving only 2 values $(U_n, U_r)$ other than the standard one $U$, is the simplest possible reward scheme and as such it has the advantage of being the most easily explainable to the end-users. The idea, however, can certainly be generalized by making the non-standard rates a function of time, i.e., have $U_n(t)$ and $U_r(t)$ instead of constant values. In this case, the necessary condition for incentivizing the users to move their delay tolerant traffic away from the busy hours becomes: $\int_0^B U_n(t)dt + \int_B^{24-B} U_r(t)dt >> U \cdot 24$.

### 3.2 Architecture

The previous scheme can be fixed with respect to the values $U, U_n, U_r, B$, which would be decided once upon the establishment of a contract between a user and the ISP. It can be implemented very simply with the integration of minimal functionality on the user (PC) and ISP side (wireless router/gateway). For example, a simple button can be integrated to the user interface, allowing the end-user to select between *constant* and *reward*. Selecting the *reward* choice would set a self imposed cap of $U_n$ during the busy hours through the OS and thus help the end-user meet the condition for receiving the *reward* rate during the non-busy hours. On the network side, all that is needed is to measure the transmission rate during the busy hours, and if it stays below $U_n$, then reward the user by increasing its allowed rate to $U_r$ for the rest of the day. This is much simpler than trying to identify and shape DT traffic using elaborate deep packet inspection equipment. It leads to a win-win situation in which users are able to download more content, whereas ISPs do not need to over-dimension.

Another possibility is to keep only $U$ fixed (going into to the contract) and communicate $U_n, U_r, B$ dynamically to the end-user, letting him select accordingly. Fig. 1 shows the envisioned architecture. Such a scheme gives the ISP greater flexibility than the static one. For example, upon observing high utilization at some part of the network, the ISP can advertise lucrative "offers" for high $U_r$ in an attempt to convince as many nearby users as possible to settle for a lower $U_n$.

One might argue that a similar scheme can be implemented only at the application layer, e.g., within downloaders and P2P clients, thereby obliterating the need for any kind of accounting on the network side. The problem of such an approach is that it cannot be enforced, as there will always be users that will hack the application and try to get $U_r$ during the entire day.

[1]Henceforth, whenever we refer to the capacity of a link we mean the maximum of either direction.

## 4. ADDING STORAGE TO THE ISP: INTERNET'S POST-OFFICE

The previous incentive-based scheme requires minimal change in the infrastructure and the protocols used by an ISP. It rationalizes the use of network resources by incentivizing the end-users to time-shift their DT high rate transfers until appropriate times for the network. The price paid for not having to change the network, is that it requires end-users to pay some attention and e.g., decide whether they want to do P2P immediately or delay it to get higher daily volume. In the first case they would select the *reward* scheme through their user interface, otherwise they would continue with *constant*. In this section we look at ways to hide time-shifts from the end-users.

### 4.1 A storage enabled ISP architecture

In Fig. 2 we show a high level architecture for a storage enabled network involving the following two new elements. *Internet Post Offices* (or IPOs) which are just storage repositories at the access ISP, i.e., near the end-users. Since they are co-located, the IPOs can communicate with the end-users as fast as the access technology of the latter allows. There exists no other bottleneck or need for further investment to support such high rate transfers between the two.

Additionally, there exist *Transit Storage Nodes* (or TSNs) located at some PoPs at the backbone of the ISP, preferably near to peering-points with other networks. Of course, between end-user and TSNs, or between IPOs and TSNs, there can be all sorts of possible bottlenecks arising either due to congestion [2], or due to traffic engineering policies [12]. The key idea here, is to *use the IPOs and TSNs to time-shift bulk DT transfers, and thus avoid congestion and ISP-throttling, while making the time-shift transparent to the end-users.* The idea makes use of the fact that the price of storage is declining much faster than the price of bandwidth [8], especially at the access network. This approach is significantly different from previous attempts to tap on unutilized bandwidth (e.g., QBone's *Scavenger Service*[2]) that require changing the routers and cannot perform in-network time-shifting as they lack network attached storage.

### 4.2 A *fire-and-forget* approach to DT transfers

Imagine a user who wants to share with his friends a large collection of high resolution photographs and videos from his latest trip or vacations. Large numbers of such users having FTTH high rate access pose a formidable challenge to existing networks that are not dimensioned for such access rates and content sizes.

[2]http://qos.internet2.edu/wg/wg-documents/qbss-definition.txt

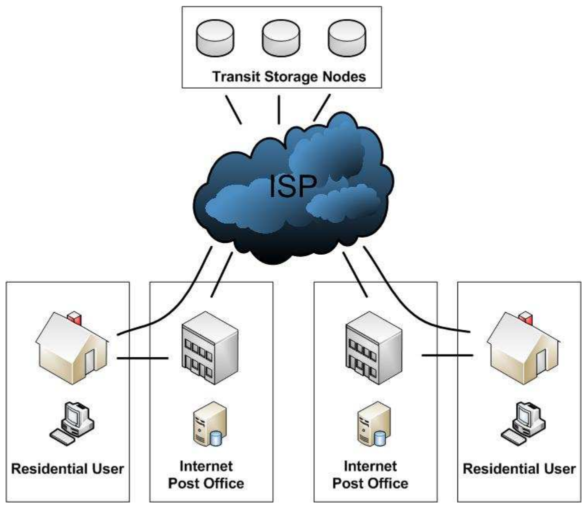


**Figure 2: High level architecture of a storage enabled network.**

Without substantial investment in upgrades of the backbone, the uplinks of DSLAMs, and the peering points to other networks, an easy solution for ISPs is to roll out FTTH and *police it heavily* when DT transfers like the above get into the way of servicing interactive, non-DT traffic. Of course, this would immediately trigger complaints from end-users expecting full FTTH rates at all times. Are there any other possibilities?

The aforementioned storage enabled architecture based on IPOs and TSNs suggests one. An end-user can push his collection of voluminous DT media to a local IPO at full FTTH rate. Since IPOs may connect directly to the DSLAMs, this does not put any strain on the rest of the network. Then the entry IPO can coordinate with other IPOs and TSNs to see that the collection reaches the intended recipients. This resembles snail (electronic) mail, where the end-user just hands in his mail to the local post office (SMTP server), at which point his direct involvement in the transfer comes to an end. A new breed of P2P applications can also be developed to make use of IPOs and TSNs. There are advantages from this for both the end-user and the network.

**The end user:** Benefits by pushing the data out of his computer at full FTTH rate. The end-to-end delivery has not been completed yet, but since the data are DT, what matters for the sender is how soon they will clear out from his computer and access line. After that, the user gets back his full CPU and uplink capacity for interactive tasks that would otherwise suffer from resource contention with slow and therefore long lived transfers. If the computer is a portable one, the user is free to disconnect and move. Last but not least, the user can shut the computer down much sooner, thus saving energy.

**The ISP:** Benefits by taking full control of the bulk transfer from the entry IPO and onwards, i.e., where most problems currently exist. The ISP can use IPOs and TSNs to schedule transfers at times of low background load. If the receiver is on an access network attached to the same ISP, then it can use the receiver's local IPO to bring the data down at a time of low utilization for the access network. If the flow has to cross to a different transit ISP, then this can be done when the corresponding peering point is least loaded.

## 5. CDNS AND DT TRAFFIC

The discussion up to now has been limited to ISPs and end-users. Next, we examine the potential gains for CDNs from handling DT traffic. We look at two scenarios based on the source of the DT traffic.

### 5.1 A CDN for Delay Tolerant Bulk data

Consider a CDN for servicing terabyte-sized *Delay Tolerant Bulk* (DTB) data, including scientific datasets, digitally rendered scenes from movie production studios, massive database backups, etc. Such a CDN installs storage nodes at access and transit ISPs from which it buys bandwidth according to a standard 95-percentile charging scheme [9]. Store-and-Forward scheduling is used to transfer DTB data between IPOs with the help of intermediate TSNs.[3] Our initial results based on real traffic traces from more than 200+ interconnection points of a large transit ISP show that SnF policies can reduce dramatically the transit costs incurred by End-to-End (E2E) policies that don't employ network storage. For example, with SnF we can transfer 100 Tbits of data from Latin America to Europe in 48 hours at zero transit cost, whereas an E2E stream of average rate of around 0.5 Gbps increases the monthly transit cost by tens of thousands of dollars under current bandwidth prices. The advantage of SnF lies on the fact that it can solve the problem of *non-coinciding load valleys* between the uplink of the sender IPO, and the downlink of a receiver IPO on a different time-zone. We explain the proposal through an example.

The top row of Figure 3 illustrate the 5-minute aggregate load on the uplink of an ISP in Latin America (LAT) hosting a sender IPO. The second and third rows depict the load on the downlinks of two ISPs in Europe (EU) and China (CH) hosting receiver IPOs. We have annotated with $uvalley(LAT)$ the time at which the uplink of LAT is least loaded and similarly for the downlinks of EU and CH. One can easily observe that due to time-zone differences, these valleys do not coincide. In the case of LAT and CH, the load valley

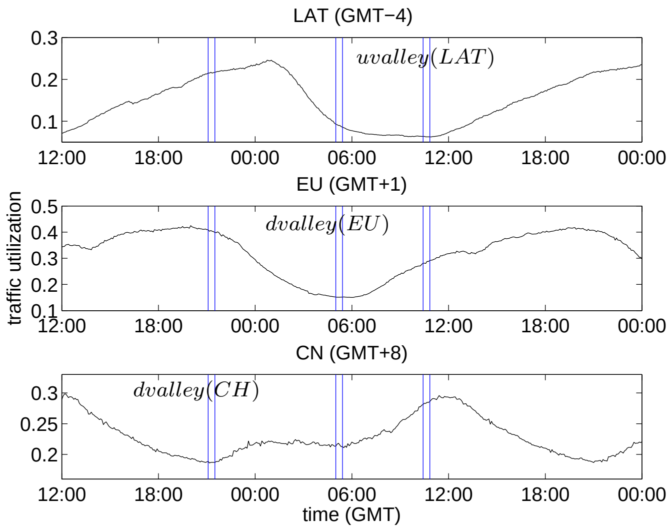


**Figure 3: Time series plot of the uplink load of a sender in LAT and receivers in EU and CH. The uplink valley of LAT finds EU with substantial load and CH with peak load.**

of the sender actually coincides with the peak of the downlink of the receiver. In this setting, E2E transfers will have to overlap with either a highly loaded uplink, or a highly loaded downlink and, thus, create either additional monetary costs by pushing the 95-percentile load based on which ISPs pay for transit, and/or obstruct the QoS of other interactive traffic with which the DTB traffic gets multiplexed. An SnF transfer through a TSN can do much better in this setting. It uses the uplink load valley to push data from LAT to a TSN on the transit ISP. The DTB data remain buffered there until the beginning of the load valley of the downlink of the receiver, at which point they are pushed to their final destination (EU or CH).

Examining all the pairs from the 200+ peering points of our transit provider we found that more than 50% of the busiest pairs had valleys that were apart for at least two hours, and thus cases like the aforementioned example were not at all uncommon. Non-coinciding valleys appear frequently, even within the same or nearby time-zones. This happens because networks of different type can peak at different hours, e.g., a corporate network typically peaks during work-hours, whereas an ADSL access network typically peaks in the late evening.

### 5.2 A CDN for Delay Tolerant End-User data

Next we look at what CDNs can do for residential end-user DT traffic. The model is similar to Fig. 2 with the difference that storage nodes are not managed by the ISP, but by an independent CDN which, unlike the ISP, has global coverage with PoPs on multiple net-

[3]For this example we have assumed that there are no bottlenecks inside the transit provider and thus it suffices to consider a single TSN.

works. Again IPO nodes are used for collecting end-user DT data at full FTTH rate, whereas other intermediate IPOs and TSNs help complete the delivery. The CDN can sell this service to content creators and give it for free to end-users. In addition to its obvious benefits for content creators and end-users, the operation of such a CDN adds value to the ISPs. The reason is that since it receives end-user DT traffic, the CDN can transmit it in an ISP-friendly manner unlike most end-user applications. For example the CDN can:

**Prefer peering to transit links:** Having post offices at multiple ISPs, the CDN can try to create an end-to-end path between a sender and a receiver that involves mostly peering links between neighboring ISP, over which traffic is exchanged without monetary transit costs. Transit links can be used only in cases that alternative paths through peering links do not exist or are severely congested.[4]

**Avoiding the hours of peak load:** The CDN can take advantage of the DT nature of the traffic to avoid times of high utilization. In the case of transit links this protects against increases of the 95-percentile of send traffic, and corresponding increases of monthly charging bills. In the case of peering links, it preserves the QoS of the background traffic and avoids the need to upgrade the link and incurring additional equipment and maintenance costs.

We believe that establishing and demonstrating the above practices would permit a CDN operator to achieve a symbiotic relationship with the ISP. The ISP would benefit by having the heat of end-user DT traffic taken away from it thanks to the CDN. The CDN would benefit by obtaining cheap flat-rate or even free access to ISP bandwidth under the conditions of ISP-friendliness discussed above. Notice that unlike Sect. 5.1 in which the CDN was introducing new *exogenous* DTB traffic to the ISP, and thus had to pay according to 95-percentile charging for it, now the CDN is just servicing *endogenous* end-user DT traffic, including high definition video from P2P, that already flows in the ISP.

## 6. CONCLUSIONS

In this article we claim that many of the tensions that currently exist on the Internet are due to wasteful usage of resources during the hours of peak load. A first step towards reducing such wastage is to shift what we define as Delay Tolerant traffic to non-peak load hours. We have proposed two solutions for this, one by offering *flat-rate compatible incentives to the end users*, and a second one based on the addition of *network attached storage*. The first solution has the advantage of requiring minimal change to the existing network, but requires a small involvement from the end-users. The second solution is completely transparent to the end-users but requires the addition of network attached storage. The latter proposal becomes economically efficient since the price of storage has been declining much faster than the price of network equipment, especially at the access and regional network.

[4]Notice that although no immediate transit cost is paid for crossing peering links, there still exist implicit, albeit real costs. For example, if the peak utilization becomes too high due to the additional delay tolerant traffic then the ISPs will have to upgrade the speed of their peering and thus incur capital and maintenance costs in order to preserve the QoS offered to their clients. We consider this next.

## 7. ACKNOWLEDGEMENTS

This work is supported in part by the NANODATACENTERS program (FP7-ICT-223850) of the EU. The authors would like to acknowledge the help of P. Chhabra, V. Erramilli, G. Smaragdakis, R. Sundaram as well as the anonymous reviewers in improving the paper.

## 8. REFERENCES

[1] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can ISPs and P2P users cooperate for improved performance? *ACM SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.
[2] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *Proc. of ACM SIGMETRICS '03*, pages 316–317, 2003.
[3] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. of ACM SOSP'01*, Banff, Canada, Oct 2001.
[4] Hitesh Ballani and Paul Francis. Towards a global IP Anycast service. In *Proc. of ACM SIGCOMM'05*.
[5] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. A Measurement-based Deployment Proposal for IP Anycast. In *Proc. of ACM IMC'06*, October 2006.
[6] David R. Choffnes and Fabián E. Bustamante. Taming the Torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *ACM SIGCOMM'08*.
[7] Costas Courcoubetis and Richard Weber. *Pricing Communication Networks: Economics, Technology and Modelling*. Wiley, 2003.
[8] Bradford DeLong. Shifting into overdrive, what happens when mass storage leaves microchips in the dust. *WIRED Magazine*, (11), 2005.
[9] David K. Goldenberg, Lili Qiuy, Haiyong Xie, Yang Richard Yang, and Yin Zhang. Optimizing cost and performance for multihoming. In *Proc. of ACM SIGCOMM '04*, pages 79–92, 2004.
[10] Andrew M. Odlyzko. Internet pricing and the history of communications. *Computer Networks*, 36:493–517, 2001.
[11] Y. Shoda, W. Mischel, and P. K. Peake. Predicting adolescent cognitive and self-regulatory competencies from preschool delay of gratification: Identifying diagnostic conditions. *Developmental Psychology*, 26(6):978–986, 1990.
[12] Slashdot. Comcast Hinders BitTorrent Traffic, Aug 2007.
[13] Haiyong Xie, Yang Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. P4P: Provider portal for applications. In *Proc. of ACM SIGCOMM'08*.

# Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs

Ying Zhang University of Michigan
Z. Morley Mao University of Michigan
Ming Zhang Microsoft Research

## Abstract

On the Internet today, a growing number of QoS sensitive network applications exist, such as VoIP, imposing more stringent requirements on ISPs besides the basic reachability assurance. Thus, the demand on ISPs for Service Level Agreements (SLAs) with better guarantees is increasing. However, despite overprovisioning in core ISP networks, resource contention still exists leading to congestion and associated performance degradations. For example, residential broadband networks rate-limit or even block bandwidth intensive applications such as peer-to-peer file sharing thereby violating network neutrality. In addition, traffic associated with specific applications, such as Skype, could also be discriminated against for competitive business reasons.

So far, little work has been done regarding the existence of traffic discrimination inside the core of the Internet. Due to the technical challenges and widespread impact, it seems somewhat inconceivable that ISPs are performing such fine-grained discrimination based on the application content. Our study is the first to demonstrate evidence of network neutrality violations within backbone ISPs. We used a scalable and accurate monitoring system – NVLens – to detect traffic discrimination based on various factors such as application types, previous-hop, and next-hop ASes. We discuss the implication of such discrimination and how users can counter such unfair practices.

## 1 Introduction

The topic of network neutrality on today's Internet is a highly contentious one. Previously, users assumed ISP networks are neutral to carry traffic without any preferential treatment. Edge customers can instrument their own policies for traffic management by for example blocking certain traffic using firewalls at the edge of the Internet. So customers expected that ISPs would not treat traffic differently based on properties other than the basic information required for forwarding, *e.g.,* destination IP address. In violation of network neutrality, traffic properties suspected to be used to perform discrimination include application types inferred from port numbers or payload data, previous-hop network, and next-hop network.

Various residential broadband networks, such as Comcast, are known to be violating network neutrality, by restricting the bandwidth usage of peer-to-peer file sharing applications. Network neutrality has different technical definitions and feasibility in various types of network models [1, 2]. Several research proposals exist for counteracting discrimination relying on encryption and multipath routing [3, 4], along with ideas to block traffic via auctions under the bandwidth shortage [5]. Given the potential detrimental effect on traffic which can be given lower priority, it is critical for end-users to first *detect* which ISP is violating network neutrality and to understand the policies for discriminating against specific traffic types. Beverly *et al.* presented the first study of the port blocking behavior that violates neutrality [6]. Related to our work, POPI is a tool for determining the router forwarding policy via end host measurements [7], but it only focuses on preferential treatment based on port numbers. Their methodology of saturating the link with high traffic volume is unsuitable for backbones.

No detailed and comprehensive study on the current practice of traffic discrimination, particularly inside the core ISPs, currently exists. And yet, traffic differentiation in the core has a much wider scope of impact, as such policies affect much more traffic compared to policies near the edge of the Internet. Knowing which ISPs perform discrimination and how they perform it is a critical first step towards identifying alternatives to address the network neutrality issues.

Our work is the first to demonstrate concrete evidence of network neutrality violations in backbone ISPs and analyze the extent of their violations. We developed a scalable and accurate distributed measurement methodology called NVLens(*Neutrality Violation Lens*[1]) to monitor ISP's loss and delay behavior in order to identify traffic discrimination based on factors such as applications, previous-hop and next-hop ASes. Given the initial report on discrimination, we performed selective drill-down to deduce how discrimination is implemented.

Unlike ISP-centric SLA monitoring, which requires access to proprietary data, NVLens relies on minimal network cooperation and is entirely end system based, leading to easy deployment and accurate observation from the end system's perspectives. NVLens can be used as a simple tool by end users to detect network neutrality violation and similarly SLA compliance of any ISP. By studying 19 large ISPs covering major continents including North America, Europe, and Australia over several weeks, we discovered ISPs some-

[1]The common translation of "night vision lens" is also relevant here, as our monitoring continuously covers both day and night.

| Type | Examples |
|---|---|
| Application types | packet header field (*e.g.,* src/dst port numbers, protocol type) |
| Application properties | data content, application protocol header (*e.g.,* HTTP header, IPSec header) |
| Network policies | routing info (previous-hop, next-hop AS, routing entry) |
| Traffic behavior | flow rate, packet size, flow duration, fragment bit |
| Available resources | router state (load, memory), time of day, location (*e.g.,* PoP) |

**Table 1**: Information commonly used to determine policies for discrimination.

times do give different priority to traffic coming from different neighbors (previous-hop ASes). Discrimination based on the next-hop AS is less common. We also observed different priority for traffic associated with UDP and specific applications such as BitTorrent compared to HTTP traffic. The loss rate increase for discriminated traffic can be as high as 8% with up to 500ms increase in RTT.

## 2 Net neutrality violation

In this study, we define network neutrality as ISPs giving equal treatment to packets regardless of their application content, application types, and packet sources or destinations. Any differentiation behavior that violates network neutrality is called *discrimination*. Note that we broaden the previous definition [2] by not singling out customers who may receive better treatment. Therefore, the observed performance difference can result from distinct business contracts between provider and its customers. It is debatable that whether this type of discrimination should be considered as neutrality violation. In this work we also report such discrimination to enable different interpretations.

Packets contain plenty of information that an ISP can use to construct discrimination policies. Table 1 shows the potential factors used to determine the discrimination policy. First, an ISP may provide differentiated service depending on the application type for security or business reasons. Application types can be determined from transport layer protocol fields or application layer content information [8]. Even with encrypted traffic, such discrimination can be made using more sophisticated traffic flow information [9]. Second, an ISP can discriminate against traffic due to business relationships, based on their source/destinations or incoming/outgoing networks. This information can be easily gathered from packet headers and routing information. Third, an ISP can selectively enable discrimination depending on the resource conditions, *e.g.,* when resources are limited as indicated by high link utilization.

The feasibility of implementing packet discrimination in a backbone ISP network with many high-speed links is questionable due to the need to perform additional per packet processing. We discuss several techniques that an ISP can employ to implement relevant policies today.

Today's router already has support for various queuing mechanisms to fulfill the need of traffic engineering, ensuring quality of service and security guarantees. Figure 1 illustrates a common architecture for implementing the discrimination within an ISP. The ingress border routers perform traffic classification by marking packets according to priorities, which are determined by packet fields such as protocol, source, and destination. The marking usually occurs on the Type-of-Service (TOS) field in the IP header. The internal routers can carry out different queuing and dropping decisions according to the packet classification encoded within TOS by the border routers [10]. Different queuing mechanisms provide various services to traffic based on its priority, *e.g.,* priority queuing, proportional share scheduling, and policing [11]. These mechanisms differ in details of how and when the differentiation is carried out.

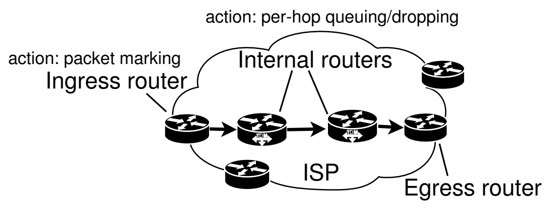


**Figure 1**: An example of discrimination implementation.

Besides router based mechanisms relying on packet header information, deep packet inspection (DPI) tools [12] allow ISPs to classify applications using packet content to understand application types. Although DPI devices are usually too expensive to be widely deployed, some current products claim to support up to 100 Gps links [13, 14] capable of searching for patterns in the payload using hardware support.

Given the feasibility of discrimination deployment, we studied all the factors shown in Table 1 except for the discrimination based on traffic behavior due to the limited resource of end-host based probing. This type of discrimination is also more difficult to implement by ISPs due to required per flow state information. NVLens enables us to discern which factor(s) may influence ISP's policies for preferential treatment of different classes of traffic. The design is extensible to other factors once they are known. The goal of detecting all these types of discrimination guides the methodology design of probing strategy and the probe packet composition in NVLens .

## 3 Measurement methodology

This section describes the design of NVLens and illustrates how to monitor networks for neutrality compliance from end systems without any ISP cooperation. NVLens has the capability to detect three main types of network neutrality violations. Figure 2 illustrates the collaborative probing used to detect neutrality violations by a particular ISP based on factors such as application types and network policies (described in Table 1). Multiple ISPs were probed in parallel simultaneously to allow for accurate comparison. As shown in the figure, discrimination detection focuses on *ISP W* based on different traffic properties, *i.e.,* towards different next-hop ASes, from different previous-hop ASes, or based on differ-

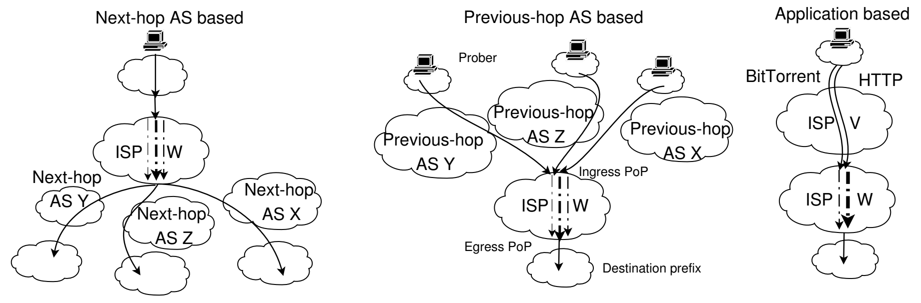


**Figure 2**: Collaborative probing to discover neutrality violations of different types.

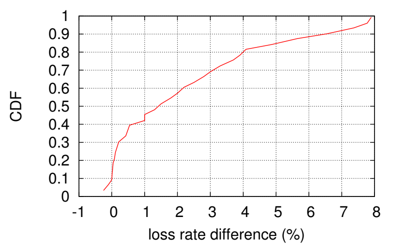


**Figure 3**: Loss rate difference between path pairs passing the test

ent application types.

Note that we focus on differences in performance metrics observed to identify traffic differentiation performed by the routers in ISPs. Many confounding factors could also cause differences in observed performance. First, different network paths in one ISP have different load leading to different performance observed. Even from one ingress to same egress, many equal-cost paths exist. Second, different application properties, *e.g.,* packet size, packet rate, can result in different performance measured. Third, external measurement artifacts, *e.g.,* heavily-loaded probing hosts, lossy reverse path, are also likely to create differences.

To rule out the impact of all these factors, we design our methodology carefully to eliminate the impact of most factors. For factors that are difficult to control, *e.g.,* impact of equal-cost paths, we use controlled experiments to confirm they would not introduce any systematic bias. In the following, we introduce our novel methodology to detect neutrality violation with low overhead.

## 3.1 Collaborative probing optimization

Probing overhead is always a concern in any active measurement study. For the purpose of discovering neutrality violation, it is particularly important to keep probing hosts lightly-loaded and to ensure short probing intervals. Otherwise, different performance might be caused by the heavily-loaded hosts or measurement conducted at different time periods. We use collaborative probing to ensure low probing overhead.

A typical backbone ISP consists of multiple PoPs (Points of Presence) at several geographic locations. In order to quantify the overall network neutrality compliance of an ISP and avoid the potential bias introduced by any particular path, NVLens should cover a reasonably large fraction of paths between distinct PoP pairs. Therefore, path selection strategy is a key issue for NVLens. Given a list of backbone ISPs, we couldn't afford to continuously probe all the destination prefixes on the Internet from all the probers. Instead, we devised an intelligent path selection strategy as follows for a probing interval: 1) Each three-tuple path $(P_i, P_e, d)$ is traversed at least $n$ times by probes from different probers; and 2) A prober does not conduct more than $m$ probes. Here, $s$ is a prober, $d$ is a destination IP address, and $P_i$ and $P_e$ are the ingress and egress points in the target ISP respectively. Previous work [15] has shown this problem is an instance of the set covering/packing problem [16] for which we use a greedy algorithm as an approximation.

## 3.2 Loss rate and RTT measurement

NVLens measures both loss rate and roundtrip time (RTT) of a path which are simple performance metrics. To comply with the resource limits at each host, we take two steps to reduce probing overhead. First, NVLens only probes the hops that map to an ingress or an egress in one of the target ISPs instead of probing all the hops along a path. Since we are only interested in identifying ISP internal traffic discrimination between ingress-egress pairs, there is no need to probe other hops. Second, to measure the loss rate and RTT to a particular hop, NVLens sends probe packets with pre-computed TTL value which is expected to trigger ICMP time exceeded response from the corresponding router. In essence, the packet is similar to traceroute probes. However, since loss may occur in both directions, we use relatively large probe packets to increase the likelihood of inducing loss on forward paths only, which has been widely adopted in previous studies [17, 18]. NVLens probes each hop 200 times so that it can detect minimum loss rate of 0.5%. To reduce the chance of triggering ICMP rate limiting, NVLens probes each hop only at most once per second.

## 3.3 Application-specific probing

We use NVLens to explore how backbone ISPs preferentially treat various real-time and QoS sensitive applications. We choose five representative applications with distinct traffic characteristics in our study: UDP, HTTP, BitTorrent (P2P file sharing), Skype (VoIP), and World of Warcraft or WoW (online gaming). To avoid the overhead of comparing each pair of applications, we use HTTP traffic as the baseline. Since HTTP is the most widely-used Internet application, we assume it does not receive any preferential treatment, *i.e.,* representing the normal performance that most applications will experience.

The following steps are taken to eliminate the impact of all possible confounding factors we can think of. First, we classify applications into two groups: large packets of 200 bytes (HTTP, UDP, BitTorrent), small packets of 80 bytes (HTTP, Skype, World of Warcraft). This classification is based on empirical observation of corresponding applications, while observing the bandwidth constraints of probe hosts. We use controlled experiments to verify that most observed packet loss occurred on forward paths. We measure three types of application at the same time, using the same probe hosts, for the same paths, with the same packet size.

To accurately represent the application behavior, we construct application-specific packets with the corresponding payload captured from real application traces. This is especially important for proprietary applications such as Skype or WoW whose protocols are not not publicly known. Given that data packets are often exchanged after a few control packets, we first transmit 9 to 25 small application-specific control packets at one packet per second rate. These packets help ISPs identify and potentially discriminate subsequent data packets. Control packets are identified using either known protocol specification (*e.g.,* for BitTorrent) or timing and packet size behavior (*e.g.,* for Skype), as there is usually a large gap between the exchange of control and data packets in both interpacket timing and packet size. Also note that control packets are constructed with sufficiently large TTLs, meaning all the routers along the path up to the last ISP's egress router can observe the control packets in case routers use such information to store state needed for traffic discrimination[2].

## 4 Experimental results

This section presents our experimental results that provide insights on how network neutrality is violated on today's Internet. To ensure precise and accurate analysis, we perform statistical tests on a large number of samples to detect traffic discrimination. This section provides concrete evidence of discrimination in several large ISPs based on routing and traffic content. The next section examines in greater depth the mechanisms and policies used for carrying out traffic differentiation.

### 4.1 Data processing

Each data point is obtained by sending 200 packets from a probing source host $s$ to a destination IP address $d$, traversing a target ISP $I$ using packets representing a particular application $a$. A data point at time interval $i$ is denoted as $l_{s,d,I,a,i}$ (percentage of lost packets relative to the 200 probing packets) and $d_{s,d,I,a,i}$ (average delay of 200 delay measurements).

We define two key variables: a path $pa$ which defines the smallest path unit for discrimination analysis and an aggregation element $agg$ which excludes certain variables in the definition of corresponding $pa$. An $agg$ helps identify the relevance of some factor in discrimination. For example, for application based discrimination analysis, $pa$=$(s, d, I, a)$ and $agg$=$(s, d, I)$. To detect whether discrimination exists between applications $a_1$ and $a_2$ on the path from $s$ to $d$ in ISP $I$, we compare the performance of $pa_1$=$(s, d, I, a_1)$ and $pa_2$=$(s, d, I, a_2)$. For previous-hop AS based discrimination analysis, $pa$=$(AS_p, P_i, P_e, AS_n)$ and $agg$=$(P_i, P_e, AS_n)$. $AS_p$ and $AS_n$ are the previous-hop and next-hop ASes of $I$ respectively. $P_i$ and $P_e$ are the ingress and egress points of $I$ respectively. These notations will be used in the following analysis.

[2]The TTLs are not too large to avoid potential complaints from edge networks.

Prior to performing discrimination analysis, we filter measurement noise caused by resource competition on or near a host by identifying high loss rates on many paths that share the same host. We also filter noise caused by ICMP rate limiting by identifying high loss rates that exceed the long-term average plus three times the standard deviation.

### 4.2 Statistical test to infer discrimination

Assuming random noise has roughly the same impact on the data points measured on any path, we apply statistical tests on several data points to identify consistent performance differences caused by traffic discrimination rather than due to random noise. There are quite a few standard hypothesis tests that compute the statistical significance of the difference between the mean values of two data sets. T-test, the most commonly-used one, requires the data sets under test to follow normal distribution which may not hold for loss rate and delay distributions. So instead, we apply the Wilcoxon signed-rank test [19] and the permutation test [20]. Neither test relies on any assumption of the input data distribution. This is a standard approach for testing the difference between two distributions without any assumptions on the properties of the distributions.

Our input data consists of two sets of data points for the path pair $pa_1$ and $pa_2$ respectively, where $pa_1$ and $pa_2$ share a common $agg$. First, we calculate the difference between each pair of data points after each set is sorted numerically: $z_i = x_i - y_i$. For the resulting difference set $Z$, we test the hypothesis that $mean_z \neq 0$ using the Wilcoxon test. Then we permute half of the data points and apply Wilcoxon test on the permuted set. The permutation tests are repeated 400 times. If both the Wilcoxon and the permutation tests are passed with 95% significance, we determine that discrimination exists between $pa_1$ and $pa_2$.

### 4.3 Characterization of discrimination

We have implemented NVLens on the PlanetLab testbed [21]. We use all the available PlanetLab hosts, roughly 750 of them, as probers covering about 300 distinct sites. It has been fully operational for more than five weeks to monitor 19 ISPs.

Table 2 illustrates the results based on the loss rate metric. Similar results based on the latency metric are omitted due to the lack of space. For application based discrimination, the baseline application for comparison is HTTP. For each path, we also collect the data points for other applications, *e.g.,* BitTorrent, and compare the loss rate with the HTTP loss rate measured during the same period. For previous-hop AS based discrimination, we compare path pairs that share the same $agg$=$(P_i, P_e, AS_n)$ but from different $AS_p$.

Table 2 summaries our main findings regarding the absolute number and percentage of path pairs that pass the

| ASN | ISP name | Tier | Application/protocol types | | | | Previous-hop | | Next-hop | | Same AS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | BT | UDP | Skype | Game | P-P | P-P-AS | P-P | AS-P-P | AS-P-P-AS |
| 209 | Qwest | | 10 , 1 | 0 | 0 | 0 | 8 , 1 | 36 , 0.2 | 1 , 0.1 | 5 , 0.03 | 6,0.1 |
| 701 | UUNet | | 29 , 0.9 | **90 , 3.6** | 0 | 0 | 89 , 3.5 | **633 , 3.6** | 13 , 0.5 | 38 , 0.2 | 92,0.5 |
| 1239 | Sprint | | 4 , 0.3 | 31 , 3.5 | 3 , 0.2 | 0 | 40 , 2.7 | 315 , 1.1 | 4 , 0.2 | 19 , 0.1 | 0 |
| 1668 | AOL Transit | | 0 | 0 | 0 | 1 , 0.5 | 4 , 1.7 | 24 , 0.9 | 0 | 0 | 20,0.3 |
| 2914 | Verio | 1 | 13 , 1.4 | 66 , 6.8 | 18 , 1.5 | 0 | 33 , 3.4 | 110 , 0.4 | 10 , 1.1 | 38 , 0.1 | 0 |
| 3356 | Level3 | | 0 | 1 , 0.05 | 0 | 0 | **109 , 6** | **746 , 1** | 2 , 0.1 | 7 , 0.01 | 9,0.1 |
| 3549 | Global Crossing | | 14 , 1.7 | 0 | 0 | 2 , 0.2 | 34 , 3.2 | 293 , 0.6 | 30 , 3.1 | 206 , 0.5 | 0 |
| 3561 | Savvis | | 0 | 1 , 0.05 | 0 | 0 | 16 , 2.7 | 254 , 1 | 3 , 0.5 | 25 , 0.1 | 33,0.1 |
| 7018 | AT&T | | 0 | 2 , 0.1 | 0 | 0 | 22 , 1 | 330 , 1 | 0 | 0 | 0 |
| 2828 | XO | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2856 | British Telecom | | 0 | 45 , 4.5 | 0 | 0 | 15 , 1.5 | 45 , 0.4 | 2 , 0.2 | 6 , 0.02 | 40,1 |
| 3257 | Tiscali | | **221 , 8** | 0 | 17 , 1 | 0 | 21 , 3 | **184 , 3** | 2 , 0.2 | 6 , 0.1 | 0 |
| 3320 | Deutsche Telekom | 2 | 6 , 0.4 | 0 | 0 | 0 | 5 , 0.4 | 26 , 0.2 | 0 | 0 | 11,1 |
| 5511 | France Telecom | | 9 , 1 | 0 | 29 , 3 | 0 | 10 , 1 | 38 , 0.3 | 0 | 0 | 13,1 |
| 6395 | Broadwing | | 0 | 0 | 0 | 0 | 2 , 0.2 | 5 , 0.09 | 0 | 0 | 0 |
| 6453 | Teleglobe | | 0 | 68 , 6 | 0 | 11 , 1 | 17 , 1 | 68 , 0.6 | 0 | 0 | 3,0.2 |
| 16631 | Cogent | | 0 | 0 | 4 , 0.05 | 0 | 70 , 4 | 213 , 0.8 | 55 , 3 | 134 , 0.2 | 94 , 0.3 |
| 6461 | AboveNet | 3 | 0 | 24 , 2.5 | 0 | 0 | 8 , 0.8 | 37 , 0.4 | 0 | 0 | 0 |
| 11537 | Abilene | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 2**: Statistical test for loss-based discrimination: discriminated path pairs in absolute number, percentage(%).

statistical test. These two numbers illustrate whether discrimination exists and how widely it is detected in an ISP. Surprisingly, evidence exists for traffic discrimination within backbone ISPs. UUNet, Tiscali, Sprint, Level3, Savvis, and AT&T all have hundreds of path pairs that exhibit previous-hop AS based discrimination. The bold numbers highlight this evidence. Next-hop AS based discrimination is far less prevalent, probably due to the ease of implementation and effectiveness in managing internal resources for the previous-hop based approach. An ingress router can easily mark packets based on their incoming interfaces. There also appears to be application based discrimination, in particular against BitTorrent and UDP traffic. We found one ISP, Tiscali, which exhibits strong evidence of discrimination against BitTorrent traffic. Figure 3 shows significant loss rate difference for the discriminated path pairs: at least 30% of the path pairs have loss rate differences ranging from 3% to 8%.

ISPs usually have incentives to give customers high priority for business reasons. To confirm this claim, for previous-hop based discrimination, we further analyze the relationship between the previous-hop AS and the ISP performing discrimination. We employ the commonly used Gao's relationship inference results [22]. Among the previous-hop discrimination, we found that 51% of path pairs involve ISPs favoring their customers' traffic over peers' traffic. 10% of the path pairs gave traffic from siblings higher priority over customers and peers. We also found many instances of particular peers being given preferential treatment over other peers. For example, among UUNet's peer, Level 3 and Savvis receive better treatment than other peers.

To further confirm that previous-hop discrimination indeed exists, we apply the same statistical tests to path pairs that share the same $(AS_p,\ P_i, P_e, AS_n)$ using UDP traffic, which should not be affected by previous-hop or next-hop AS based discrimination. The last column in Table 2 provides the absolute number and percentage of such path pairs that pass the tests. In most cases, they are much smaller than the numbers in the previous-hop column, suggesting that the loss rate difference between path pairs are more likely caused by previous-hop AS based discrimination as opposed to random noise.

| ASN | % TOS-marked path pairs with discrimination | % discriminated path pairs matching TOS rules |
|---|---|---|
| 209 | 2.1 | 2.9 |
| 701 | 71 | 45 |
| 1239 | 16 | 11 |
| **1668** | 80 | 76 |
| **2914** | 95 | 89 |
| **3356** | 92 | 80 |
| **3549** | 81 | 70 |
| 3561 | 48 | 35 |
| **7018** | 90 | 77 |
| 2856 | 56 | 41 |
| 3257 | 84 | 59 |
| 3320 | 0 | 0 |
| 5511 | 60 | 17 |
| 6453 | 9 | 11 |
| 16631 | 91 | 55 |
| 6461 | 9 | 6 |

**Table 3**: Correlation between loss based discrimination and TOS difference.

## 5 In-depth analysis

Some routers mark the Type of Service (TOS) bit in order to provide different levels of service within an ISP. We study to what extent the loss rate discrimination can be explained by the difference in TOS value. Note that our probing packets trigger ICMP time exceeded messages from routers. These messages include the IP header of the original probing packets, which reveals the TOS value of the original probing packets marked by the routers. This allows us to correlate the loss rate difference with TOS difference for any path pair.

While a large TOS value does not always imply high priority, we assume an ISP has a consistent rule of mapping a TOS value to a fixed priority. Before performing a correlation, we need to determine this rule. Starting with all the path pairs that pass the discrimination tests, we obtain all the distinct TOS values observed in the ISP. We then construct a mapping from TOS value to the priority it stands for. The mapping is constructed in a way to best explain the loss rate difference between all the discriminated path pairs. For example, if TOS value $x$ stands for higher priority, then paths marked with $x$ should experience lower loss rate.

Table 3 illustrates the correlation results between loss rate discrimination and TOS difference. The second column indicates the percentage of TOS-marked path pairs that exhibit the correct loss rate discrimination. And finally, the third column shows the percentage of discriminated path pairs that match the inferred TOS rules. Both percentage numbers are high for a few ISPs, *e.g.,* AS1668, AS2914, AS3356, AS3549, and AS7018, strongly indicating TOS value is used for discriminating against traffic inside these ISPs. We also check the temporal stability of TOS marking and find the marking of 99.9% of the paths does not change within the six-day analysis.

For application based discrimination, we conduct controlled experiments in order to understand how ISPs perform the discrimination. We vary our probing by using a different port, zeroing the application payload, or bypassing the initial control messages. We study the BitTorrent traffic discrimination in Tiscali as an example. We studied the likelihood that the discrimination is performed based on port number. By changing the port from the default BitTorrent port to 80, the number of discriminated path pairs drops by 50%. Zeroing payload or bypassing control messages has a negligible effect.

## 6 Discussion

Besides detecting neutrality violations, NVLens can further identify the policies used by the ISPs to perform traffic differentiation and reveal other relevant information such as the location of enforcement, time-of-day effect, and relative versus absolute differentiation. The technique can be extended to discover other types of discrimination, *e.g.,* IPSec vs. non-IPSec. Such information can be used by end-systems to make more informed decisions for selecting routes and ISPs, applying encryption or routing through proxies to overcome some of this discrimination.

Even if ISPs are aware of techniques used by NVLens to perform neutrality violation detection, they cannot easily evade our probing. The probe packets are constructed using real traffic traces and are difficult to distinguish from actual data traffic. Unless ISPs perform stateful TCP flow analysis, it is challenging to identify and preferentially treat our probe traffic. In the future, we can further use two-end controlled experiments to mimic the TCP states.

## 7 Conclusion

In this paper we presented the design and implementation of the first deployed system to accurately and scalably detect network neutrality violations performed by backbone ISP networks. Using collaborative probing from end hosts with innovative application-specific probing on carefully selected network paths, we demonstrate the surprising evidence of traffic discrimination carried out by today's backbone ISPs. NVLens has been operational on PlanetLab for five weeks and is capable of monitoring 19 large backbone ISPs simultaneously for neutrality violations detection using loss rate and delay as performance metrics. In addition to detecting network neutrality violation, we perform in-depth analysis to further examine the discrimination policies. Our work demonstrates the feasibility of detecting network neutrality violations in backbone ISPs entirely from end systems and presents an important step to attain more accountability and fairness on today's Internet. To devise countermeasures against ISPs' action of network neutrality violations, detection is an indispensable first step, and our proposed system NVLens is a promising approach.

## References

[1] J. Crowcroft, "Net neutrality: the technical side of the debate: a white paper," *ACM Computer Communication Review*, 2007.

[2] X. Yang, G. Tsudik, and X. Liu, "A Technical Approach to Net Neutrality," in *Proceedings of ACM HotNets-V, Irvine*, 2006.

[3] I. Avramopoulos, J. Rexford, D. Syrivelis, and S. Lalis, "Counteracting discrimination against network traffic," Tech. Rep. TR-794-07, Princeton University Computer Science, 2007.

[4] I. Avramopoulos and J. Rexford, "Stealth probing: Efficient data-plane security for IP routing," in *Proceedings of USENIX Annual Technical Conference*, 2006.

[5] X. Yang, "Auction, but Don't Block." Work in Progress.

[6] R. Beverly, S. Bauer, and A. Berger., "The Internet's Not a Big Truck: Toward Quantifying Network Neutrality," in *Proceedings of the 8th Passive and Active Measurement (PAM 2007) Conference*, 2007.

[7] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, C. Y. Cheung, and X. Li, "End-to-end inference of router packet forwarding priority," in *Proc. IEEE INFOCOM*, 2007.

[8] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 50–60, 2005.

[9] C. Wright, F. Monrose, and G. Masson, "On inferring application protocol behaviors in encrypted network traffic," in *Journal of Machine Learning Research (JMLR): Special issue on Machine Learning for Computer Security*, 2006.

[10] V. S. Kaulgud, "Ip quality of service: Theory and best practices." www.sanog.org/resources/sanog4-kaulgud-qos-tutorial.pdf, 2004.

[11] C. S. Inc., "Configuring Priority Queueing." http://www.cisco.com/en/US/docs/ios/12_0/qos/configuration/guide/qcpq.h%tml.

[12] "Deep packet inspection." www.networkworld.com/details/6299.html.

[13] "Arbor Ellacoya e100 ." http://www.arbornetworks.com.

[14] C. Networks, "Complete Packet Inspection on a Chip." http://www.cpacket.com/.

[15] R. Mahajan, M. Zhang, L. Poole, and V. Pai, "Uncovering Performance Differences in Backbone ISPs with Netdiff," in *Proceeding of NSDI*, 2008.

[16] S. G. Kolliopoulos and N. E. Young, "Approximation algorithms for covering/packing integer programs," *Journal of Computer and System Sciences*, vol. 71, no. 4, 2005.

[17] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level Internet Path Diagnosis," in *Proceedings of SOSP*, 2003.

[18] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," in *Proc. Operating Systems Design and Implementation*, 2006.

[19] S. Siegel, *Non-parametric statistics for the behavioral sciences*. McGraw-Hill, 1956.

[20] E. S. Edgington, *Randomization tests*. Marcel-Dekker, 1995.

[21] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology Into the Internet," in *Proc. of ACM HotNets*, 2002.

[22] L. Gao, "On Inferring Autonomous System Relationships in the Internet," in *Proc. IEEE Global Internet Symposium*, 2000.

# NANO: Network Access Neutrality Observatory

Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster
{mtariq,murtaza,feamster}@cc.gatech.edu

## ABSTRACT

We present *NANO*, a system that establishes whether performance degradations that services or clients experience are caused by an ISP's discriminatory policies. To distinguish discrimination from other causes of degradation (e.g., overload, misconfiguration, failure), *NANO* uses a statistical method to estimate causal effect. *NANO* aggregates passive measurements from end-hosts, stratifies the measurements to account for possible confounding factors, and distinguishes when an ISP is discriminating against a particular service or group of clients. Using simulation we demonstrate the promise of *NANO* for both detecting discrimination and absolving an ISP when it is not discriminating.

## 1. Introduction

In late 2005, Ed Whitacre sparked an intense debate on *network neutrality* when he decried content providers "using his pipes [for] free". Network neutrality says that end users must be in "control of content and applications that they use on the Internet" [1], and that the ISPs respect that right by remaining *neutral* in treating the traffic, irrespective of its content or application. This paper does not take a stance in this debate, but instead studies a technical question: Can users in access networks detect and quantify discriminatory practices of an ISP against a particular group of users or services? We define any practice by the ISP that degrades performance or connectivity for a service as discrimination or violation of network neutrality. We refer to such violations as discrimination. We argue that, regardless of whether or not discrimination is ultimately deemed to be acceptable, the network should be *transparent*; that is, users should be able to ascertain the behavior of their access ISPs.

Unfortunately, because ISP discrimination can take many forms, detecting it is difficult. Several ISPs have been interfering with TCP connections for BitTorrent and other peer-to-peer applications [3]. Other types of discrimination may include blocking specific ports, throttling bandwidth, or shaping traffic for specific services, or enforcing traffic quotas. Existing detection mechanisms *actively* probe ISPs to test for specific cases of discrimination: Glasnost [3], automates detection of spurious TCP reset packets, Beverly *et al.* [8] present a study of port-blocking, and NVLens [10] detects the use of packet-forwarding prioritization by ISPs by examining the TOS bits in the ICMP time exceeded messages. The main drawback of these mechanisms is that each is specific to one type of discrimination; thus, each form of discrimination requires a new test. Worse yet, an ISP may either block or prioritize active probes associated with these tests, making it difficult to run them at all.

If end users could instead somehow detect discrimination by observing the effect on service performance using passive, in-band methods, the detection mechanism would be much more robust. Unlike active measurements, ISPs cannot prioritize, block, or otherwise modify in-band measurements. To achieve this robustness, we use a black-box approach: we make no assumptions about the mechanisms for implementing discrimination and instead use statistical analysis primarily based on *in situ* service performance data to quantify the *causal relationship* between an ISP's policy and the observed service degradation.

In this paper, we present the design for Network Access Neutrality Observatory (*NANO*), a system that infers the extent to which an ISP's policy causes performance degradations for a particular service. *NANO* relies on participating end-system clients that collect and report service performance measurements for a service. Establishing such a causal relationship is challenging because many *confounding factors* (or variables) that are unrelated to ISP discrimination can also affect the performance of a particular service or application. For example, a service may be slow (e.g., due to overload at a particular time-of-the-day). A service might be poorly located relative to the customers of the ISP. Similarly, a service may be fundamentally unsuitable for a particular network (e.g., Internet connectivity is not suitable for VoIP applications in many parts of the world).

A necessary condition for inferring a valid causal relationship is to show that *when all the other factors are equal*, a service performs poorly when accessed from an ISP compared to another ISP. The main challenge in designing *NANO* is to create an environment where all other factors are in fact equal. Creating such an environment requires (1) enumerating the confounding factors; (2) establishing a "baseline" level of performance where all factors besides the confounding variables are equal. Unfortunately, the nature of many confounding factors makes it difficult to create an environment on the real Internet where all other factors, except for an ISP's discriminative policy and service, would be equal. Instead, to correctly infer the causal relationship, we must adjust for the confounding factor by creating strata of clients that have "similar" values for all factors except for their access network. Our approach is based on the theory of causal inference, which is applied extensively in other fields, including epidemiology, economics, and sociology.

This paper is organized as follows. In Section 2 we overview necessary background for establishing a causal relationship between ISP policy and service performance degradation and formalize the problem. In Section 3, we describe the steps in the causal inference, the confounding variables for the problem, and the *NANO* architecture for collecting and processing the necessary data. Section 4 presents simulation-based results, and Section 5 concludes with a discussion of open issues and a research agenda.

## 2. Background and Problem Formulation

In this section, we formalize the definitions and basic concepts used for establishing ISP discrimination as the cause of service degradation. We describe the concept of service, service performance, ISP, and discrimination; the inference of causal effect, how it relates to association and correlation; and finally, approaches for quantifying causal effect. We also formalize the application of causality to detecting ISP discrimination.

### 2.1 Definitions

**Service and Performance.** A *service* is the "atomic unit" of discrimination. An ISP may discriminate against traffic for a particular service, e.g., Web search, traffic for a particular domain, or particular type of media, such as video. Such traffic may be identifiable using the URL or the protocol. Similarly, ISPs may target specific applications, e.g., VoIP, or peer-to-peer file transfers. *Performance*, the outcome variable, is specific to the service. For example, we use server response time for HTTP requests, loss, and jitter for VoIP traffic, and average throughput for peer-to-peer traffic.

**ISP and Discrimination.** *Discrimination* against a service is a function of ISP policy. The performance for a service depends on both the properties of the ISP's network, e.g., its location, as well as the policy of treating the traffic differently. Thus, an objective evaluation of ISP discrimination must adjust for the ISP's network as a confounding factor. To differentiate an ISP's network from its discrimination policy, we use the ISP *brand* or *name* as the causal variable referring to the ISP's discrimination policy. In the rest of the paper, when we use ISP as the cause, we are referring to the ISP policy or the brand with which the policy is associated.

We aim to detect whether a certain practice of an ISP results in poorer performance for a service compared to other similar services or performance for the same service through other ISPs. If an ISP's policy of treating traffic differently does not result in degradation of performance, we do not consider it as discrimination.

### 2.2 Background for Causal Inference

Statistical methods offer tools for causal inference that have been used in observational and experimental studies [6, 7]. *NANO* draws heavily on these techniques. In this section, we review basic concepts and approaches for causal inference, and how they relate to inferring ISP discrimination.

**Causal Effect.** The statement *"X causes Y"* means that if there is a change in the value of variable $X$, then we expect a change in value of variable $Y$. We refer to $X$ as the *treatment variable* and $Y$ as the *outcome variable*.

In the context of this paper, accessing a particular service through an ISP is our treatment variable ($X$), and the observed performance of a service ($Y$) is our outcome variable. Thus, treatment is a binary variable; $X \in \{0,1\}$, $X = 1$ when we access the service through the ISP, and $X = 0$ when we do not (e.g., access the service through an alternative ISP). The value of outcome variable $Y$ depends on the performance metric and the service for which we are measuring the performance.

The goal of causal inference is to estimate the effect of the treatment variable (the ISP) on the outcome variable (the service performance). Let's define *ground-truth* value for the outcome random variable as $G_X$, so that $G_1$ is the outcome value for a client when $X = 1$, and $G_0$ is the outcome value when $X = 0$. We will refer to the outcome when not using the ISP ($X = 0$) as the *baseline*—we can define baseline in a number of ways, as we describe in more detail in Section 3.1.2.

We can quantify the *average causal effect* of using an ISP as the expected difference in the ground truth of service performance between using the ISP and the baseline.

$$\theta = \mathbb{E}(G_1) - \mathbb{E}(G_0) \qquad (1)$$

Note that to compute the causal effect, $\theta$, we must observe values of the outcome both under the treatment and without the treatment.

**Association vs. Causal Effect.** In a typical *in situ* dataset, each sample presents only the value of the outcome variable either under the treatment, or under the lack of the treatment, but not both; e.g., a dataset about users accessing a particular service through one of the two possible ISPs, *ISP*$_a$ and *ISP*$_b$, will comprise data of the form where, for each client, we have performance data for either *ISP*$_a$ or *ISP*$_b$, but not both. Such a dataset may thus be incomplete and therefore not sufficient to compute the causal effect, as shown in Equation 1.

Instead, we can use such a dataset to compute correlation or association. Let's define *association* as simply the measure of observed effect on the outcome variable:

$$\alpha = \mathbb{E}(Y|X = 1) - \mathbb{E}(Y|X = 0) \qquad (2)$$

It is well known that association is not a sufficient metric for causal effect, and in general $\alpha \neq \theta$.

**Example.** Tables 1(a) and (b) illustrate the difference between association and causal effect using an example of eight clients (*a–h*). The treatment variable $X$ is binary; 1 if a user uses a particular ISP, and 0 otherwise. For simplicity, the outcome ($Y$) is also binary, 1 indicating that a client observes good performance and 0 otherwise; both $\alpha$ and $\theta$ are in the range $[-1, 1]$.

Table 1(a) shows an *in situ* dataset. In this dataset, clients *a–d* do not use the ISP in question and clients *e–h* use the ISP. Note that for each sample, only one or the other outcome is observable. The association value in this dataset is $\alpha = -3/4$. If we use the association value as an indicator of causal effect, we would infer that using the ISP causes a significant negative impact on the performance.

Table 1(b), on the other hand, presents the ground-truth performance values, $G_0$ and $G_1$, as the performance when not using the ISP and performance when using the ISP for the same client, respectively. These values could be obtained by either subjecting the client to the two cases, or through an oracle. For this set of clients, the true average causal effect $\theta = 1/8$, which is quite small, implying that in reality, the choice of ISP has no or little effect on the performance for these clients. Although the *in situ* dataset is consistent with the ground-truth, i.e., $Y = G_X$, there is a clear discrepancy between the observed association and the true causal effect.

| | (a) Original Dataset | | (b) Ground Truth (Oracle) | | (c) Random Treatment | |
|---|---|---|---|---|---|---|
| | $X$ | $Y$ | $G_0$ | $G_1$ | $X$ | $Y$ |
| a | 0 | 1 | 1 | 1 | 1 | 1 |
| b | 0 | 1 | 1 | 1 | 0 | 1 |
| c | 0 | 1 | 1 | 1 | 1 | 1 |
| d | 0 | 1 | 1 | 1 | 0 | 1 |
| e | 1 | 0 | 0 | 0 | 1 | 0 |
| f | 1 | 0 | 0 | 0 | 0 | 0 |
| g | 1 | 0 | 0 | 0 | 0 | 0 |
| h | 1 | 1 | 0 | 1 | 1 | 1 |
| | $\alpha = -3/4$ | | $\theta = 1/8$ | | $\alpha = 0$ | |

**Table 1: (a) Observed Association ($\alpha$) in a passive dataset (b) True causal effect ($\theta$) in an example dataset: $\alpha \neq \theta$. (c) Association converges to causal effect under random treatment assignment: $\alpha \approx \theta$.**

## 2.3 Approaches for Estimating Causal Effect

This section presents two techniques for estimating the causal effect, $\theta$. The first, random treatment, involves an active experiment, where we randomly assign the treatment to the clients and observe the association. The second, adjusting for confounding variables, is a passive technique, where we work with only an *in situ* dataset and estimate the overall causal effect by aggregating the causal effect across several small strata.

**1. Random Treatment.** Because the ground-truth values $(G_0, G_1)$ are not simultaneously observable, we cannot estimate the true causal effect (Eq. 1) from an *in situ* dataset alone. Fortunately, if we assign the clients to the treatment in a way that is independent of the outcome, then under *certain conditions*, association is an unbiased estimator of causal effect. This property holds because when $X$ is independent of $G_X$, then $\mathbb{E}(G_X) = \mathbb{E}(G_X|X) = \mathbb{E}(Y|X)$; see [9, pp. 254–255] for a proof. In Table 1(c) we randomly assign a treatment, 0 or 1, to the clients and see that association, $\alpha$, converges to the true causal effect, $\theta$.

For association to converge to causal effect with random treatment, all other variables in the system that have a causal association with the outcome variable must remain the as we change the treatment. In the case of the example above, association will converge to true causal effect under random treatment, if and only if the original ISP and the alternative ISP are both similar except for their discrimination policy.

Random treatment is difficult to emulate in the Internet for two reasons. First, it is difficult to make users switch to an arbitrary ISP, because not all ISPs may offer services in all geographical areas, the users may be contractually bound to a particular ISP, and asking users to switch ISPs is inconvenient for users. Second, if changing the ISP brand also means that the users must access the content through a radically different network which could affect the service performance, then we cannot use the mere difference of performance seen from the two ISPs as indication of interference: the association may not converge to causal effect under these conditions because the independence condition is not satisfied. This situation is called *operational confounding*: changing the treatment inadvertently or unavoidably changes a confounding variable.

**2. Adjusting for Confounding Variables.** Because it is difficult to emulate random treatment on the real Internet and control operational confounding, we need to find a way to adjust for the effects of confounding variables. *NANO* uses the well-known stratification technique for this purpose [6].

*Confounding variables* are the extraneous variables in the inference process that are correlated with both the treatment and the outcome variables. As a result, if we simply observe the association between the treatment and the outcome variables, we cannot infer causation or lack of it, because we cannot be certain whether the change is due to change in the treatment variable or a change in one or more of the confounding variables.

With stratification, all samples in a stratum are *similar* in terms of values for the confounding variables. As a result, $X$ and $G_X$ are independent of the confounding variables within the stratum, essentially creating conditions that resemble random treatment. Thus, the association value within the stratum converges to causal effect, and we can use association as a metric of causal effect within a strata.

**2a. Challenges.** This approach presents several challenges. First, we must enumerate the confounding variables and collect sufficient data to help disambiguate the true causal effect from the confounding effects. Second, we must define the stratum boundaries in a way that satisfies the above conditions. Unfortunately, there is no automated way to enumerate all the confounding variables for a given problem; instead, we must rely on domain knowledge. Section 3 addresses these challenges.

**2b. Formulation.** In the context of *NANO*, we have multiple ISPs and services; we wish to calculate the causal effect $\theta_{i,j}$ that estimates how much the performance of a service $j$, denoted by $Y_j$, changes when it is accessed through ISP $i$, versus when it is not accessed through ISP $i$. Let $Z$ denote the set of confounding variables, and $s$ a stratum as described above. The causal effect $\theta_{i,j}$ is formulated as:

$$\theta_{i,j}(s;x) = \mathbb{E}(Y_j|X_i = x, Z \in \mathbb{B}(s)) \quad (3)$$

$$\theta_{i,j}(s) = \theta_{i,j}(s;1) - \theta_{i,j}(s;0) \quad (4)$$

$$\theta_{i,j} = \sum_s \theta_{i,j}(s) \quad (5)$$

$\mathbb{B}(s)$ represents the range of values of confounding variables in the stratum $s$. $\theta_{i,j}(s)$ represents the causal effect within the stratum $s$. A key aspect is the term $\theta_{i,j}(s;0)$ in Equation 4: it represents the *baseline* service performance, or the service performance when the ISP is *not* used; we define this concept in more detail in Section 3.1.2. Note that the units for causal effect are same as for service performance, so we can apply simple thresholds to detect discrimination.

**2c. Sufficiency of Confounding Variables.** Although there is no simple or automatic way to enumerate all the confounding variables for a problem, we can test whether a given list is sufficient in the realm of a given dataset. To do so, we predict the value of the outcome variable using a non-parametric regression function, $f()$, of the treatment variable, $X$, and the confounding variables, $Z$, as $\hat{y} = f(X;Z)$. We then compare the predicted value with the value of outcome variable observed in the given dataset, $y$, using relative error, $|y - \hat{y}|/y$. If $X$ and $Z$ are sufficient to define the outcome $Y$, then the prediction error should be small.

## 3. NANO System Design

This section explains how *NANO* performs causal inference, enumerates the confounding variables required for this inference, and describes the system architecture for collecting and processing the relevant data.

## 3.1 Establishing the Causal Effect

Estimating the causal effect for a service degradation involves three steps. First, we stratify the data. Next, we estimate the extent of causal impact of possible ISP interference within each stratum and across the board. Finally, we try to infer the criteria that the ISP is using for discrimination.

### 3.1.1 *Stratifying the data*

To stratify the data, *NANO* creates bins (i.e., ranges of values) along the dimensions of each of the confounding variables, such that the value of the confounding variable within the bin is (almost) constant. The bin size depends on the nature of the confounding variable. As a general rule, we create strata such that there is a bin for every unique value of categorical variables; for the continuous variables, the bins are sufficiently small, that the variable can be assumed to have essentially a constant value within the stratum. For example, for a confounding variable representing the client browser, all the clients using a particular version and make of the browser are in one stratum. Similarly, we create one hour strata along the time-of-the-day variable.

We use simple correlation to test whether the treatment variable and the outcome variable are independent of the confounding variable within a stratum. We combine adjacent strata if the distribution of the outcome variable conditioned on the treatment variable is identical in each of the stratum; this reduces the total number of strata and the number of samples needed.

### 3.1.2 *Establishing the baseline performance*

A thorny aspect of Equation 4 is the term $\theta_{i,j}(s;0)$, which represents the *baseline* service performance, or the service performance when the ISP is *not* used. This aspect raises the question: What does it mean to not use ISP $i$ to access service $j$? It could mean using another ISP, $k$, but if ISP $k$ is also discriminating against service $j$, then $\theta_{k,j}(s;1)$ will not have the (neutral) ground-truth baseline value. To address this problem, *NANO* takes $\theta_{i,j}(s;0)$ as the average service performance when not using ISP $i$, calculated as: $\sum_{k \neq i} \theta_{k,j}(s;1)/(n_s - 1)$, where $n_s > 2$ is the number of ISPs for which we have clients in stratum $s$.

An important implication of defining the baseline in this way is that *NANO* is essentially comparing the performance of a service through a particular ISP against the average performance achieved through other ISPs, while adjusting for the confounding effects. If all or most of the ISPs across which *NANO* obtains measurements are discriminating against a service, it is not possible to detect such discrimination using the above definition of baseline; in this case, discrimination becomes the *norm*. In such cases, we might consider using other definitions of discrimination, such as the comparing against the best performance instead of the average, or using a performance model of the service obtained from laboratory experiments or mathematical analysis as the baseline.

### 3.1.3 *Inferring the discrimination criteria*

*NANO* can infer the discrimination criteria that an ISP uses by using simple decision-tree based classification methods. For each stratum and service where *NANO* detects discrimination, *NANO* assigns a negative label, and for each stratum and service where it does not detect discrimination, it assigns a positive label. *NANO* then uses the values of the confounding variables and the service identifier as the feature set and uses the discrimination label as the target variable, and uses a decision-tree algorithm to train the classifier.

The rules that the decision tree generates indicate the discrimination criteria that the ISP uses, because the rules indicate the boundaries of maximum information distance between discrimination and the lack of it.

## 3.2 Confounding Variables

Confounding variables are the extraneous factors in inferring whether an ISP's policy is discriminating against a service; these variables correlate, either positively or negatively, with both the ISP brands and service performance. Because there is no automated way to enumerate these variables for particular problem, we must rely on domain knowledge. In this section, we describe three categories of confounding variables and explain how they correlate with both the ISP brands and the service performance. In Section 3.3, we describe the specific variables that we collect to adjust for these confounding variables.

**Client-based.** Client-side applications, as well as system and network setup, can confound the inference. The particular application that a client uses for accessing a service might affect the performance. For example, in the case of HTTP services, certain Web sites may be optimized for a particular Web browser and perform poorly for others. Similarly, certain Web browsers may be inherently different; for example, at the time of this writing, Opera, Firefox, and Internet Explorer use different number of simultaneous TCP connections, and only Opera uses HTTP pipelining by default. For peer-to-peer traffic, various client software may experience different performance. Similarly, the operating system and the configuration of the client's computer and local network, as well as a client's service contract, can affect the performance that the client perceives for a service.

We believe that the above variables also correlate with ISP brand, primarily because the ISP may serve particular communities or localities. As an example, we expect that Microsoft's Windows operating system may be more popular among home users, while Unix variants may be more common in academic environments. Similarly, certain browsers may be more popular among certain demographics and localities than other.

**Network-based.** Various properties of the Internet path, such as location of the client or the ISP relative to the location of the servers on the Internet, can cause performance degradation for a service; such degradation is not discrimination. Similarly, a path segment to a particular service

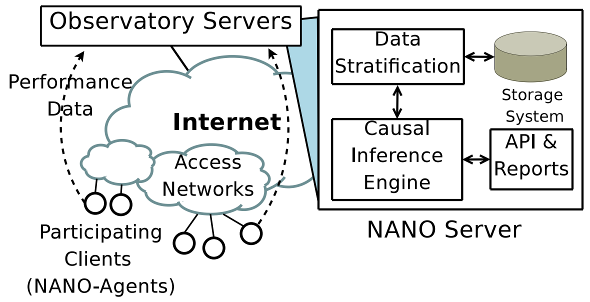


**Figure 1:** ***NANO*** **System Architecture.**

provider might not be sufficiently provisioned, which could degrade service. If we wish to not treat these effects as discrimination, we should adjust for the path properties.

**Time-based.** Service performance varies widely with time-of-day due to changes in utilization. Further, the utilization may affect both the ISPs and the service providers, thus confounding the inference.

## 3.3 Data Collection and Analysis Architecture

An important aspect of *NANO* is collecting the necessary data for facilitating inference. This section describes the criteria for data collection, the features that we collect, and, finally, the data collection mechanism.

**1. Criteria.** The criteria for data collection has two parts. First, the feature should quantify the treatment variable, the outcome variable, or the values of the confounding factors. Second, the data should be unbiased.

The first criterion helps us determine a set of features for which to collect data; this list is explained below. We can collect many of these features through active or passive monitoring. The second criterion, however, suggests that we must take care that the measurements are not biased. As we discussed in Section 1, ISPs may have the incentive to interfere with identifiable active measurements to deter inference of discrimination or improve their rankings. Similarly, we believe that while we could use the data directly from service providers as the "baseline" service performance, such information could be biased in the favor of the service provider. Therefore, to the extent possible, *NANO* relies on passive measurements to determine the values of the features.

**2. Mechanism.** Given the nature of the confounding factors and the desire to collect data passively, we believe the best place to collect this data is using monitoring agents at clients. Figure 1 shows the system architecture. The primary source of data for *NANO* are client-side agents installed on computers of voluntarily participating clients (*NANO*-Agents). Each agent continuously monitors and reports the data to the *NANO* servers. We are developing two versions of this agent. The first is a Web-browser plug-in that can monitor Web-related activities, and the second is a packet-level sniffer that can access more fine-grained information from the client machines.

**3. Dataset Features.** The *NANO*-Agent collects three sets of features for the confounding factors, corresponding to the three classes of confounding variables (Section 3.2)

First, the *NANO*-Agent collects features that help identify the client setup, including the operating system, basic system configuration and resource utilization on the client machine. Second, *NANO*-Agents perform active measurements to a corpus of diverse benchmark sites (PlanetLab nodes) to establish the topological location of the clients and their ISPs. These measurements include periodic short and long transfers with the benchmark sites. These measurements are similar in spirit to ones used by many Internet coordinate systems [5]. *NANO* uses this information to establish the topological properties of the ISP and stratify ISPs with similar topological location to adjust path properties factor. Finally, all the data is time-stamped to allow adjustment for the time-of-day factor.

To identify the treatment variable, i.e., ISP brand, we use the IP address of the client and look it up against `whois` registry servers. To determine the performance for each service, the *NANO*-Agent monitors and logs the information about the ongoing traffic from the client machine for the services that *NANO* monitors. The sniffer version of the agent logs the network round-trip time (RTT) measurements to various destinations for small and large packets. It also collects unsampled flow statistics for the ongoing flows to determine the throughput, and also maintains the applications associated with each flow. The latter is used to disambiguate the performance differences that might be associated with particular applications. The *NANO*-Agent tags this information with a service identifier that it infers by inspecting the packet payloads (e.g., by looking for regular-expression `google.com/search?q=` in the HTTP request message to identify search service), or, if possible, by looking at the protocol and port numbers.

# 4. Simulation

To illustrate how *NANO* can detect ISP discrimination against a particular service, we evaluate the technique in simulation for a specific example. A rigorous validation of the approach will ultimately require a real deployment where there is less control over confounding variables; we discuss this issue and various others in more detail in Section 5.

Our simulation setup comprises three ISPs, $\text{ISP}_A$, $\text{ISP}_B$, and $\text{ISP}_C$, that provide connectivity for two services, $\text{S}_1$ and $\text{S}_2$ for their respective clients. The clients use one of the two types of applications, $\text{App}_1$ and $\text{App}_2$ to access the services $\text{S}_1$ and $\text{S}_2$. Performance for the service $\text{S}_1$ is sensitive to the choice of application: clients perceive better performance when using application $\text{App}_1$, compared to using $\text{App}_2$. The performance of service $\text{S}_2$ is agnostic to the choice of application. The distribution of clients using the two types of applications is different across the ISPs (e.g., due to demographics). In particular, we set the distribution of $\text{App}_1$ to be 60%, 10%, and 90%, across the three ISPs, respectively. This distribution makes the client application a confounding variable because its distribution correlates with both the ISP and service performance.

We set up the experiment such that $\text{ISP}_B$ throttles the bandwidth for all of its clients that access service $\text{S}_1$. To achieve this, we configure the request rates from the clients and the available throttled bandwidth between the ISPs and the services to achieve certain expected utilization levels. In

| (a) Association | | |
|---|---|---|
| | Service $S_1$ | Service $S_2$ |
| Baseline | 7.68 | 2.67 |
| $ISP_B$ | 8.60 | 2.71 |
| Association | 0.92 | 0.04 |

| (b) Causal Effect | | | | |
|---|---|---|---|---|
| | Service $S_1$ | | Service $S_2$ | |
| Confounding Var. | $App_1$ | $App_2$ | $App_1$ | $App_2$ |
| Baseline | 9.90 | 2.77 | 2.61 | 2.59 |
| $ISP_B$ | 11.95 | 7.95 | 2.67 | 2.67 |
| Causal Effect | 2.0 | 5.18 | 0.06 | 0.12 |

**Table 2: Simulation Results: Estimating the causal effect between $ISP_B$ and $S_1$. All the numbers are request completion times in seconds.**

particular, we configure the links so that the expected utilization on all links is about 40%, except for the traffic from $ISP_B$ to service $S_1$, which faces about 90% utilization.

We aim to detect discrimination by $ISP_B$ against the service $S_1$, and establish a causal relationship. We compute the association and causal effect using Equation 2 and Equation 4, respectively. We use the average response time seen through $ISP_A$ and $ISP_C$, combined, as the baseline.

Table 2 presents the association and causal effect estimated for this simulation. Table 2(a) shows that $ISP_B$ has little to no association for either service $S_1$ or $S_2$. However, as we stratify on the application variable, we see in Table 2(b) that $ISP_B$ has significant causal effect for both applications for service $S_1$, but there is still no causal effect for service $S_2$. This example shows that, for this case, *NANO* can establish a causal relationship where one exists ($ISP_B$ and service $S_1$), and rule out where one does not exist, e.g., between $ISP_B$ and $S_2$.

## 5. Summary and Research Agenda

We presented Network Access Neutrality Observatory (*NANO*), a system for inferring whether an ISP is discriminating against a particular service. We have examined only basic criteria for discrimination in a simulation environment; in future work, we will evaluate *NANO*'s effectiveness in the wide area, for a wider range of possible discrimination activities, and in adversarial settings where ISPs may attempt to evade detection. In this section, we pose several questions that are guiding our ongoing work.

**How can *NANO*-agents be deployed?** *NANO* relies on participating clients to collect the data needed to perform causal inference. PlanetLab and CPR [4] nodes are our initial deployment candidates, but in the long run, we wish to incentivize home users to deploy *NANO*-Agents. Because *NANO* inference engine must exclude all extraneous factors, including transient faults to establish ISP discrimination, *NANO*-Agents can additionally act as a network troubleshooting and diagnostics application that users might find useful.

**How can *NANO*-agents protect user privacy while still exposing enough client-side information to expose discrimination?** Because some of the measurements that *NANO*-Agent collects can lead to invasion of user privacy, *NANO* stores the data in a stratified form and does not maintain any client-identifiable data (e.g., client IP addresses or search queries). Further, we are instrumenting *NANO* to give users full control over the services that the a *NANO*-agent can monitor. Finally, we are investigating ways of distributed inference, were we may be able to mitigate the need for aggregating the data at a central repository for inference, instead, the *NANO* server can act as a coordinator and the clients infer the discrimination in a peer-to-peer fashion.

**Is *NANO* general enough to detect diverse, evolving, and adversarial discrimination policies?** ISPs may continue to evolve their policies for distinguishing between various services. One such policy is imposing quotas for Internet traffic from a client [2]; in the future, ISPs may extend this policy by exempting traffic to certain *partner* sites from such quotas, thereby creating a discriminatory environments. We believe that *NANO* can quickly detect such new policies and infer the criteria if we measure sufficient metrics about the state of the network and service.

**How much data is needed to perform inference?** *NANO* requires a collection of sample data inputs from each stratum to be able to adjust for each confounding variable. The greater the variance of the confounding variables, the more data samples are needed to adjust for each of them and establishing confidence bounds. While statistics theory does help determine the number of samples needed for each stratum, each variable will be distributed differently across the set of clients and ISPs, so it is difficult to determine how many clients would need to run *NANO*-Agents to allow sufficient confidence levels for inference. We expect to understand this better as we deploy these agents.

## Acknowledgements

We would like to thank Mostafa Ammar at Georgia Tech and Joseph Hellerstein at UC Berkeley for their valuable feedback that helped improve several aspects of our work. This work is supported in part by NSF Awards CNS-0643974, CNS-0721581, and CNS-0721559.

## REFERENCES

[1] A Guide to Network Neutrality for Google Users. http://www.google.com/help/netneutrality.html.
[2] Comcast Excessive Use Policy. http://help.comcast.net/content/faq/Frequently-Asked-Questions-about-Excessive-Use.
[3] Glasnost: Bringing Transparency to the Internet. http://broadband.mpi-sws.mpg.de/transparency/.
[4] CPR: Campus-Wide Network Performance Monitoring and Recovery. http://www.rnoc.gatech.edu/cpr/, 2006.
[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004.
[6] N. Jewell. *Statistics for Epidemiology.* Chapman & Hall/CRC, 2004.
[7] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2000.
[8] S. B. Robert Beverly and A. Berger. The internet's not a big truck: Toward quantifying network neutrality. In *Passive & Active Measurement (PAM)*, Louvain-la-neuve, Belgium, Apr. 2007.
[9] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference.* Springer, 2003.
[10] Y. Zhang, Z. M. Mao, and M. Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *Proc. 7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, Calgary, Alberta. Canada., Oct. 2008.