Network Service Abstractions for a Mobility-Centric Future Internet Architecture*

Francesco Bronzino, Kiran Nagaraja, Ivan Seskar, Dipankar Raychaudhuri
WINLAB, Rutgers University
671 US1 South, North Brunswick, NJ 08902
{bronzino, nkiran, seskar, ray}@winlab.rutgers.edu

ABSTRACT

The increasing composition of mobile devices and mobile applications in the Internet requires us to revisit the traditional principles of fixed, host-centric communications, when designing a next-generation architecture. To support this major shift, we define in this paper a set of basic service abstractions that should be afforded by a future Internet that is centered upon the notion of *self-certifying globally unique IDs (GUID)* for all network principals - hosts, content, services, etc. alike. We followup with a specific set of network service APIs that provide full access to the proposed abstractions, and implement these on Linux and Android hosts that connect to an instantiation of the future Internet architecture proposal - MobilityFirst [5]. Using performance benchmarks and the implementation of representative use cases we show that the API is flexible and can enable efficient and robust versions of present and future applications.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Internet; mobility; trust; network services; application programming interface; Future Internet Architecture

1. INTRODUCTION

The Internet today is very different from its original concept when the architecture and protocols were developed around the abstraction of communications between fixed end hosts. Growing levels of mobility characterize today's communications; mobile wireless devices have outnumbered fixed end hosts and even service end-points have different levels of mobility (not an uncommon scenario in data centers). Even though support for seamless mobility is a growing requirement for the Internet as a whole, past proposals and current solutions are either only applicable within limited environments (e.g., cellular [21]) or are inefficient when applied to the Internet (e.g., MobileIP [2]). A few recent scalable approaches to support mobility have been proposed, but these are not standalone and require changes to the routing plane and/or protocol stack defined in TCP/IP [1, 3, 16, 20]. A few solutions may be applied by patching current systems, but benefits under fine-grained mobility are still unclear [18, 13].

A second serious shortcoming of the Internet is its host-centricity. Principals such content, services, and context have over the years gained at least as much importance as hosts. However, since they are not first-class network objects, they are not directly addressable. Direct addressability and location-independence, would enable seamless content/service mobility, and help with building efficient delivery networks without resorting to DNS-based tricks used by present-day CDNs. Direct addressability is also important for services offered by the core and edge of the network, not just end-points. While the clear boundaries of network pipes and compute end-points that were fundamental to end-to-end transport protocols (e.g., TCP) are starting to fade, CDNs and hosting platforms are starting to place storage and compute clouds closer to consumers [17] and this strategy is increasingly being co-opted by ISPs [6] eager to provide value-add services. Direct addressability for these resources and services would benefit both deployment and access strategies.

A third important consideration for the Internet is the need for supporting efficient one-to-many communication mechanisms. This need is foremost for the emerging context (e.g., location-aware messaging), collaborative (e.g., multiplayer on-line games) and crowd-sourcing applications. The related aspect of device-level multihoming where devices may simultaneously attach to two or more networks for performance or robust connectivity will also benefit from native support for multi-point communication [1].

Finally, along with the exponential growth in size and economic importance of the Internet, the scale of security threats has grown too. Establishing network trustworthiness and avoiding spoofing and identity hijacking incidents are priorities today that are not completely met by patched solutions such as IPSEC or DNSSEC, mostly due to their partial deployment or adoption. A clean approach that ensures uniform deployment at a basic level for all network

*Research supported by NSF Future Internet Architecture (FIA) grant CNS-1040735

MobiArch'13, October 4, 2013, Miami, Florida, USA

http://dx.doi.org/10.1145/2505906.2505908 ...$15.00.

principals such as the PKI-based self-certified identifier proposal is a potential solution [7].

In MobilityFirst [5], starting from a clean slate approach to the design of the future Internet architecture, we address the above stated challenges and trends collectively. Towards this end, this paper makes the following contributions:

i. We define a set of basic network service abstractions that can support the needs of the future mobility-centric Internet applications. These abstractions are: name-based services, direct addressability for content and services, trusted identities, multi-point addressability and in-network hosting of storage and compute services. [Section 2]
ii. We design and implement a complete network service API, basically an alternate socket library, that provides access to the MobilityFirst future Internet Architecture services. [Sections 3, 4]
iii. Through performance benchmarks of the API/protocol stack implementation and evaluation of sample applications built using the new API, we validate the effectiveness of the network architecture. [Sections 5, 6]

2. NETWORK SERVICE ABSTRACTIONS

We think that a future Internet should support the below set of service abstractions. The implication of each on the network architecture (NetArch) design is presented alongside.

A. Name-based Services. Communication with a mobile end-point should be no different than that with a fixed end-point. The current 'practical' approach results in undesirable asymmetry, where mobile end-points are always responsible for re-establishing connections. The situation is doubly vexing when both end-points are mobile. A basic service abstraction that allows addressing a network end-point by its unique name and not its current location will establish a uniform approach to dealing with fixed and mobile end-points alike, enabling seamless mobility.
NetArch implications: To support a name-based network service that seamlessly handles mobility, the network requires native support for *dynamic and fine-grain location resolution*. Some have proposed protocol interposition approaches that dynamically substitute local addresses for end-points with dynamically resolved ones [16, 1]. Efficiency extensions in MobileIPv6 attempt to signal end-points with address updates [4] to avoid triangular routing through fixed home agents. However, we think that besides end-hosts, the network routing fabric must also be able to dynamically resolve and re-bind in-flight packets.

B. Direct Addressability for All Network Principals. Host-centric abstractions to network services were a solid building block during the conception and early advancement of the Internet. Though, other important principals have emerged since then. While content and services are two established principals (besides hosts), others such as sensors and actuators, as also more abstract ones such as context are quickly gaining traction. Since few foresaw today's usage of the Internet with any accuracy, allowing for a broader definition should be the path forward. In that vein, direct addressability for all principals eliminates any unnecessary bindings of one principal with another. For instance, content should be addressable both directly and independently of where it may be located physically.
NetArch implications: Allowing for direct addressability for not only content and services, but also other emerging first class entities requires that the *name space be practically inexhaustible*. For instance, 256 bits to encode the name would last us for a long time to come. To put it in perspective, 2^{270} is the ballpark for number of atoms in the observable universe. A larger namespace implies engineering challenges to implementing network elements - whenever forwarding engines inspect, lookup or classify on names - and scalability of network support services such as a name-to-address resolution service.

C. Trusted Identities. Stronger security and network trust is possible if self-certifying names were used for addressing principals, as shown in AIP [7]. Present-day approaches to authenticate and establishing trust is based on principals that maintain trust credentials (e.g., a PKI certificate) separate from their network identity (an IP address), and where the credential establishes the linkage of principal's identity to the network identifier. The linkage itself is certified by a mutually trusted entity (e.g., a certificate authority). The adoption today is far from pervasive and particularly challenging for mobile entities where the conflated network identifier may change. Using a public-key as the self-certifying identifier to address principals, can ensure both location independence and greater trust by preventing hijacking and spoofing problems.
NetArch implications: Embedding trust within the network names requires names to be longer and also to be flat. When using a PKI public key of a reasonable strength X, would require a minimum of 2X bits (= 256 bits for the current recommended 128-bit strength) as when using Elliptic Curve Cryptography. Also, since names form the basis of all network services, there must exist organizations or other easily accessible mechanisms by which names are produced and assigned to network principals in a reasonable manner. Finally, flat names imply they cannot be aggregated as is for IP addresses, creating engineering challenges when routing/forwarding operations need be performed on names.

D. Multi-Point Addressability. With group-based subscriptions (e.g., RSS, over-the-top video 'broadcasts') and collaborative applications (e.g., teleconferencing, gaming) routine today, need for multi-point addressing is basic. Under this are multicast, multihoming, and also anycast, a 'one-of-many' abstraction important for a variety of reliability and load-sharing uses. Today's host-centric Internet essentially supports a point-to-point abstraction. IP-multicast is really an extension that's enabled by special interpretation of a small subset of destination addresses. Concerns of scalability means that multicast is commonly left disabled on network elements. Internet applications therefore regularly resort to multiple unicasts to address groups. In wireless environments, there is a desire to take advantage of inherent multicast/broadcast medium to enable efficient point to multi-point delivery services. Multi-homing, where entities have two or more active network attachments, deserves similar support with a few independent considerations that allow flexibility on how each attachment can be used separately or collectively for performance, reliability or other metrics.
NetArch implications: First, the network must have support for creating and managing groups of member identities. These must be dynamically available to the routing

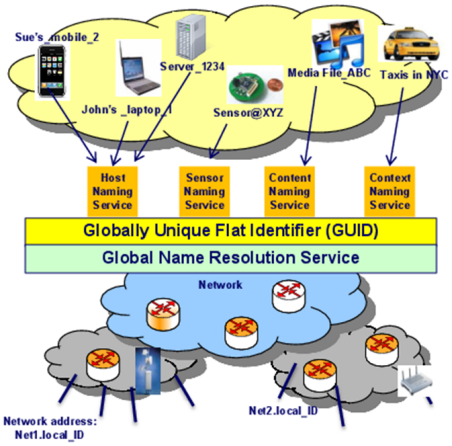

Figure 1: MobilityFirst architecture.

fabric dynamically. Be it as receiver driven multicast trees or some other means, a basic consideration would be limit per group state within network elements to minimize set-up procedures for easier and sustained adoption. Since names are flat and uniform, the service abstraction must provide a means for end-hosts to specify the requested delivery service - multicast, anycast, multihoming, broadcast, etc. - that's interpreted by the network elements. While network could participate in path choice and scheduling for multihoming, transport implementations may also be conceived at the end-host [19].

E. En-Route Storage and Compute. The traditional notion of keeping all data processing at end-points with the network as pipes with routing alone is beginning to dissolve. For instance, non-ISPs such as Akamai have independently placed compute/storage resources at the edges of the network to reduce access latency. It is not inconceivable, therefore, for ISPs to open up PoPs for placement of storage and compute services as well [6]. Routers in the request path may service content requests, or co-located compute resources may shoulder mobile offloaded security functions such as data encryption. In-network and en-route compute opportunity may hold particular appeal for mobile devices that are often on a limited resource budget [12] and may require non-trivial customization of data delivered to them.
NetArch implications: While certain services may be embedded transparently into the network [9] or be co-located without requiring a tight coupling with the routing fabric, network architectures with a extensible data plane in the form of a pluggable computer plane, would benefit from flexible service extensibility in the future.

3. BACKGROUND: MOBILITYFIRST NETWORK ARCHITECTURE

With the above stated network service abstractions in consideration, we have designed and prototyped a network architecture that addresses the principal goals of supporting at-scale and seamless *mobility*, along with *trustworthiness* in the future Internet. Figure 1 shows the main components of the MobilityFirst architecture which centers around the concept of self-certifying, Globally Unique IDentifiers (GUIDs) as names for all network principals. Below we present key details of the architecture that address the requirements outlined in the previous section.

Naming and Dynamic Resolution. At the crux of the MobilityFirst architecture is a new name-based service layer which serves as the 'narrow waist' of the protocol stack. The name-based service layer uses flat GUIDs for all principals or network-attached objects including hosts, content, and services, making each a first-class network object. Unlike IP addresses which conflate identity and location, addresses of objects in MobilityFirst are dynamically resolved using the object's GUID. This resolution is enabled by a globally accessible name resolution service (GNRS), which is used by objects to both announce their latest location/address and lookup end points they wish to communicate with. While a variety of incarnations of the GNRS are possible, we have validated 2 alternate designs that both meet our low resolution latency goals of less than 100ms on average for lookup operations [23, 22].

Trusted Communication. A GUID assigned to a network object by one of several name certification services (NCS), is self-certifiable, i.e., end-points claiming a GUID can authenticate each other without the need for third-party certification. When the GUID is derived as a cryptographic hash of the object's public key, the authentication requires a simple, bilateral challenge response procedure to be executed between the communicating end-points. For content, the GUID may optionally be derived as a cryptographic hash of the bits of the content. For full non-repudiation, i.e., to verify origin of content, a signature may accompany the content which could authenticate the principal that originated the content.

Storage-Informed Segmented Transport, Edge-Aware Routing. In contrast to end-to-end transports which perform poorly in wireless conditions [10, 8], MobilityFirst employs a segmented transport to reliably progress data hop-by-hop. Data is segmented into large blocks that are cached at each hop, if storage is available, to enable in-network retransmission under losses. Experiments under a variety of wireless conditions have shown significantly better fairness, throughput, latency and robustness under the hop-by-hop transport, including an order of magnitude gains in median throughput [14, 11]. Within a domain, a generalized storage-aware routing (GSTAR) combines link-state routing with DTN elements, and flexibly expands connectivity across wired and wireless segments, as also occasionally connected partitions [15]. Conditions at the wireless-edge are further heeded by adopting an edge-aware inter-domain routing (EIR) approach that scalably gathers (using *telescoping* or aggregation of updates) and utilizes capacity and load conditions at edge networks to instrument effective multi-path and multi-home delivery.

Extensible In-Network Services. MobilityFirst proposes a network fabric with native support for multi-point and multi-path delivery services, with little to no set up control signaling between network elements and minimal state within them. For example, group memberships for multi-point delivery are maintained within the GNRS and retrieved dynamically by source hosts or en-route network elements to determine forwarding paths. The requested delivery type is specified by end-hosts and encoded as the *service identifier (SID)* field within the routing header, and includes multicast, anycast, multi-path, and multi-home delivery. To support future extensibility of network function, MobilityFirst proposes a pluggable 'compute plane' for the

Table 1: MobilityFirst API

Basic	Content Centric	Service Centric
open(profile, [profile-opts], [src-GUID])	get(content-GUID, request, buffer, [svc-opts])	exec(svc-GUID, request, buffer, [svc-opts])
send(dst-GUID, data, [svc-opts])	get_handle(handle, dst-GUID, request)	exec_handle(handle, dst-GUID, request)
recv(src-GUID, buffer, [GUID-set])	get_response(handle, data, [svc-opts])	exec_response(handle, data, [svc-opts])
attach(GUID-set)	post(dst-GUID, data, buffer, [svc-opts])	
detach(GUID-set)	post_handle(handle, dst-GUID, data)	
close()	send_response(handle, response, [svc-opts])	

network fabric. Compute services traditionally implemented by end-hosts may be plugged into the network at strategic points to provide en-route or local services such as content caching, encryption, VPN, or video transcoding. These instances register their name-address mappings at the GNRS, and require that end-hosts request their invocation by specifying a compute-plane SID and the GUID of the particular service. Furthermore, multiple SIDs may be specified simultaneously within the header invoking any sensible combination of services on a packet.

4. MOBILITYFIRST API

In this section we discuss the specific API we are developing with the goal of supporting the abstractions presented in Section 2. Table 1 shows the complete set of API designed to take advantage of the architecture. The parameters therein are a loose depiction mainly to simplify the presentation. We have divided the operations into three major groups: basic, content-centric and service-centric operations. By doing so we want to be able to take advantage of the inherent characteristics of the communication patterns of these three groups of abstractions.

4.1 Basic Operations

Basic operations are in charge of: create end points with default stack operations customized for applications, support name based message delivery and manage network presence for the set of application GUIDs.

Endpoint/Socket creation and customization. At the beginning of each communication session, an application initializes a MobilityFirst socket by invoking the *open* operation. During this initialization, the application provides the API layer the informations about the profile of the communication that will occur. With the *profile* parameter we allow user to specify the set of elements that characterize the session such as: communication patterns, resources needed and services required. Additional extensions to the profile can be provided through a set of optional profile options. The final parameter is also optional and represents the GUID that the application want to use as its default identity within the network.

Name-based Messaging. Once the session is initialized, *send* and *recv* are used for the exchange of data messages. While the baseline profile is common for the entire session, a per-message characterization is possible through the use of service options. These service options (*svc-opts*) are used to define the set of network services offered by the architecture. This set of features spans from the ability of exploiting the computing layer located at routers, to different delivery systems and security options. Additionally, GUIDs can be used to express intentional data receipt through the use of the optional *GUID-set* parameter in the *recv* operation. Messages from source GUIDs other than those specified are not delivered to the local end-point.

Management of network presence. To modify the set of GUIDs that an application wants to be responsible for *attach* and *detach* operations can be used. Through these operations network reachability for specified GUIDs can be announced if not already established. The utility of these operations can be easily identified in a content delivery scenario where availability of particular content may be in continuous flux (e.g., changing news items) and their availability needs to be updated both locally and remotely on the GNRS.

4.2 Content and Service Extensions to the API

As introduced before, GUIDs can represent any network object. While we do not present here details of how human readable names would get translated into GUIDs, we assume that from this transaction the semantic relationship between the nature of the GUID and the network object that it identifies is easily tracked by the programmer.
Starting from this principle that additional information about the nature of the GUID could be known a-priori, we extended the API to support the following three specific operations as related to content and service entities: requests for transferring contents from a remote location (*get*), transfer of content to a remote location (*post*) and requests for a service to be invoked at a remote location (*exec*). The use of type specific operations translate into different advantages; on the host side this enables the network stack to select the best transport protocol and allocate in advance the adequate amount of resources; moreover it allows the user to choose between handling content transactions asynchronously or as atomic operations and use per operation security and data delivery options. On the network side this enables the usage of specific header SIDs to provide the network components additional information about the type of data flowing.

All three communication patterns are characterized by a three way transaction represented as: request (i.e., *get, post, exec*), handling of the request by the receiving host(i.e. *xxx_handle*) and final response (i.e., *xxx_response*). *Handle* objects are used to identify specific requests and characterize the responses that follow them. Additional informations for the end points involved in the communication can be passed through the use of the *request* and *response* parameters.

5. IMPLEMENTATION

To gain some experience with our design, we are developing a proof-of-concept prototype of MobilityFirst's API and end host protocol stack. The API is implemented as a system library and can be interfaced both from C/C++ and from JAVA; the stack implementation takes the form of a standalone, multi-threaded user-level process that uses *pcap* for low-level packet capture and injection. As all com-

Table 2: Latency in ms on 54Mbps WiFi link

	MobilityFirst	IP
64B	5.62 ± 3.69	0.55 ± 0.42
256B	4.71 ± 0.75	0.65 ± 0.80
1KB	5.68 ± 0.88	0.81 ± 0.35
4KB	8.93 ± 3.75	3.31 ± 1.00
16KB	20.44 ± 1.78	12.72 ± 1.14

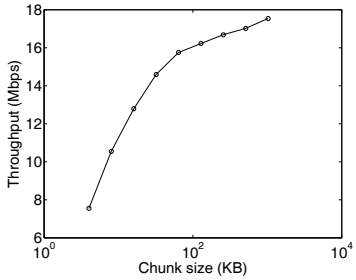

Figure 2: Throughput on a 54Mbps WiFi link

ponents are developed in C/C++ or JAVA with no major library dependencies they can be easily utilized on x86/ARM platforms running Linux or Android.

5.1 Micro-benchmarks

To ensure a reasonable implementation of the API and the protocol stack, we ran basic latency and throughput experiments to establish the combined API library and stack performance.

In Table 2 and Figure 2 are shown respectively the round trip times and maximum throughput achieved by two machines with Intel i7 K875 processors and 8GB of memory directly connected through an Intel 54 Mbps Wi-Fi interface. From the table is possible to notice that even though the software implementation generates additional computing overhead, the behavior of the average RTTs with increasing packet size follows the one obtained using ICMP control messages. The graph provides an idea of the difference of performances achieved by transferring a large file dividing it in chunks of different sizes. While the performance is not yet comparable to what we get from using *iperf* with UDP (21 Mbps on average) and TCP (17.1 Mbps on average), the behavior of the graph is consistent with what would be expected from the application of the wireless transport protocol Hop [14] where with additional protocol overhead better performances are achieved with big chunk sizes.

6. USE CASES

We now show several examples of how the defined API would support: common Internet applications, difficult usage scenarios and MobilityFirst specific scenarios.

Content Retrieval. Among the services that would inherently benefit from name based routing there is content retrieval. The GUID abstraction nicely fits the needs of this context by allowing two different retrieval methods: by referencing contents with specific GUIDs or by contacting web servers through their GUIDs. Moreover delivery options could be exploited to achieve retrieval flexibility and to exploit in-network services.

As a supportive example of the second case we have implemented a small content retrieval scenario where two content servers are located at different distances, in terms of hops

Table 3: Latency in ms to content severs

	WIFI	ETH 10	ETH 100
Server 1	23.03 ± 10.52	14.74 ± 1.17	13.15 ± 1.02
Server 2	30.23 ± 8.66	17.58 ± 0.75	15.16 ± 0.81

count, from a client. Figure 3 (left) shows the structure of the network. The core network is based on Ethernet links at 100Mbps or 10Mbps, while we change link *L1* from either a 54 Mbps Wi-Fi link, a 10 Mbps Ethernet link or a 100 Mbps Ethernet link. Table 3 shows that the latencies in the three cases do not differ much between the two servers. In this scenario, the end user sequentially requests 20 contents (replicated at both servers) of size 24 MB sending a request message addressed to a GUID representing the web service and try to retrieve them using anycast delivery, as reflected in the following listing:

```
int downloadFile(char *myGUID, char *sGUID){
   /* b is the request, opts is "anycast" */
   ...
   hdl = open(sGUID, myGUID, NULL);
   ret = send(hdl, b, 32, &opts);
   while(remSize) {
      ret = recv(hdl, b, MAX_CH,NULL);
      /* process 'ret' bytes of content */
      remSize -= ret;
   }
   ...
}
```

After the 10th transfer is completed a failure occurs on the link connecting to *S2*. The retrieval times in Figure 3 depict clearly how the routing paradigms embedded in the GUID abstraction enable the handling of the link failure. In this case the 11th request is delayed due to the adjusting period of the network. The effect of the failure would be additionally reduced in case of retrieval of larger contents. Figure 3 (center) shows the throughput change after the failure occurs.

This same example could be also represented exploiting the other approach where GUIDs are used to reference contents; in this case contents would be retrieved through *get* requests providing inherent support to content location services that nowadays are achieved through DNS tricks that would not be desirable otherwise [17].

Multihoming receive. While the concept of multihoming is inherently supported in name based routing systems, how to handle multiple interfaces from a network interface point of view is still an open issue as multiple options are available. We argue that there should be a two level policy to determine the usage of the available interfaces: one based on application specific needs and one defined by the user, with the latter having higher priority. In the proposed API, the user level policy could be then expressed during the context creation provided by the open operation while the application requirements would be inherently defined from the context defined. These policies would be then applied to provide additional information to other network elements by the usage of header SIDs or entries in the GNRS. These concepts could be applied to the sample network of Figure 3. While we have analyzed the network behavior in case of failure, the use of multiple interfaces to the network could be exploited to increase performances by exploiting content presence on both servers.

In-network services usage. Extensible In-Network Services are a key feature of MobilityFirst's architecture that

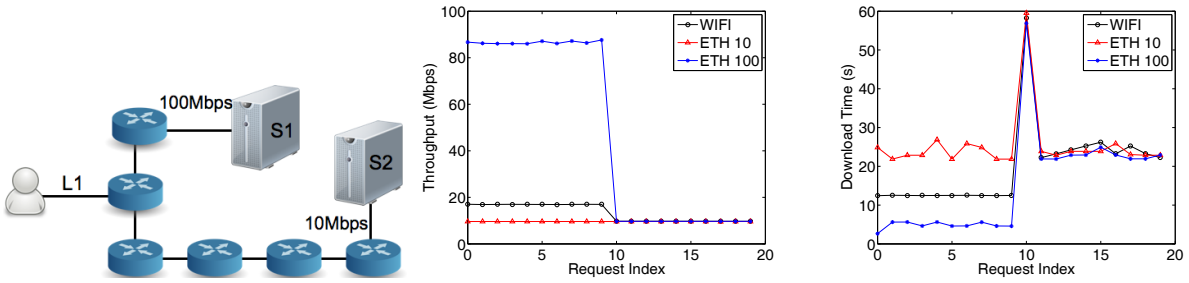

Figure 3: Anycast content retrieval experiment

enables en-route or local services such as content caching, encryption, VPN, or video transcoding. While we do not plan to provide full access to network services at the application level, we think that a set of content and service specific features should be addressable through the API. With the proposed API, these services could be activated on a per message level, allowing enough flexibility to the developer. As an example, we could consider the on route caching of contents. When servers or network caches respond to a content request (i.e., get (GUID, ...)) they set a content response SID in the header with cache options. Other routers in the path can then decide to cache the response for serving future requests.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented the basic network service abstractions that MobilityFirst, as a potential Future Internet Architecture, plans to support. We then presented the design of the API we are developing along with other network components that will be used to evaluate the proposed concepts. These interfaces provide complete support to the presented abstractions using a base group of network operations in conjunction with content and service specific extensions. Finally we showed through a series of use cases how this API could be exploited to offer the set of features that MobilityFirst aims to provide.

We plan to further explore how effective the API can be and how well MobilityFirst would perform in scenarios that are difficult to realize in today's Internet architecture.

8. ACKNOWLEDGMENTS

The authors would like to thank Chunhui Zhang and Guanling Chen of University of Massachusetts Lowell for their contributions to the design and implementation of the Android/Linux protocol stack for the MobilityFirst network.

9. REFERENCES

[1] Host identity protocol. `http://tools.ietf.org/html/rfc5201`.

[2] Ip mobility support for ipv4. `http://tools.ietf.org/html/rfc3344`.

[3] The locator/id separation protocol (lisp). `http://tools.ietf.org/html/rfc6830`.

[4] Mobility support in ipv6. `http://www.ietf.org/rfc/rfc3775.txt`.

[5] MobilityFirst. `http://mobilityfirst.winlab.rutgers.edu/`.

[6] A. V. Abhigyan Sharma and R. Sitaraman. Distributing content simplifies isp traffic engineering. In *Proc. of ACM SIGMETRICS*, 2013.

[7] D. G. Andersen et al. Accountable Internet Protocol (AIP). In *Proc. ACM SIGCOMM*, August 2008.

[8] M. C. Chan and R. Ramjee. Tcp/ip performance over 3g wireless links with rate and delay variation. *Wireless Networks*, pages 81–97, 2005.

[9] J. Erman et al. Network-aware forward caching. In *Proceedings of the 18th international conference on World wide web*, pages 291–300. ACM, 2009.

[10] S. Farrell et al. When tcp breaks: Delay- and disruption-tolerant networking. *IEEE Internet Computing*, 10(4):72–78, 2006.

[11] S. Gopinath, S. Jain, S. Makharia, and D. Raychaudhuri. An experimental study of the cache-and-forward network architecture in multi-hop wireless scenarios. In *Proc. of LANMAN*, 2010.

[12] M. S. Gordon et al. Comet: Code offload by migrating execution transparently. In *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, pages 93–106. USENIX Association, 2012.

[13] B. Y. Kimura and H. C. Guardia. Tips: wrapping the sockets api for seamless ip mobility. In *Proc. of Applied computing*. ACM, 2008.

[14] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: a new paradigm for wireless transport. In *Proc. of NSDI*, 2009.

[15] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet. In *Proc. of MobiArch*, pages 19–24. ACM, 2011.

[16] E. Nordstrom et al. Serval: An end-host stack for service-centric networking. *Proc. 9th USENIX NSDI*, 2012.

[17] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[18] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 155–166. ACM, 2000.

[19] R. Stewart and C. Metz. Sctp: new transport protocol for tcp/ip. *Internet Computing, IEEE*, 5(6):64–69, 2001.

[20] J. Su et al. Haggle: Seamless networking for mobile applications. In *UbiComp 2007: Ubiquitous Computing*, pages 391–408. Springer, 2007.

[21] A. G. Valkó. Cellular ip: a new approach to internet host mobility. *ACM SIGCOMM Computer Communication Review*, 29(1):50–65, 1999.

[22] A. Venkataramani et al. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *Communication Systems and Networks and Workshops, 2013. COMSNETS 2013*. IEEE, 2013.

[23] T. Vu et al. Dmap: a shared hosting scheme for dynamic identifier to locator mappings in the global internet. In *Distributed Computing Systems (ICDCS), 2012*, pages 698–707. IEEE, 2012.