The *I21* Initiative

Tom Herbert

CTO, Quantonium

tom@quantonium.net

Bringing the Internet into the 21st century

The Internet has become integrated into civilization and is an indispensable element of modern life. Even so, there are still far greater possibilities as to what it might become. The future Internet will provide sophisticated services and benefits that we can scarcely imagine. Unfortunately, there are significant headwinds against moving the Internet forward. The Internet has become ossified and is stuck in an architecture from 1995. Change and disruption needed to meet emerging requirements is becoming increasingly difficult to achieve.

I21 is the initiative to bring the Internet into the 21st century. The goal is to move the Internet back towards its original design principles that espoused simplicity, security, robustness, layered architecture, decentralization, and the end-to-end model. Achieving this goal requires a concerted, cross industry effort towards a long term vision. However, I21 is evolution not revolution. I21 will endeavour to identify steps that can provide incremental value, and will always be mindful to keep the business case for change at the forefront.

Background

The invention of the Internet is, by any account, one of the greatest achievements in human history. It's a world-wide communications platform for sharing ideas, reviving old friendships, being informed as well as being entertained. It's also a commercial success. The sum value created by the Internet is in the trillions of dollars and even some individual "Internet" companies have reached trillion dollar valuations. All is not well though. While the Internet is incredibly useful, it is becoming increasingly clear that evolving it is a major challenge.

Innovation is hard because the Internet has become *ossified*, meaning it is inflexible and resistant to change. This ossification implies that only a narrow set of protocols and protocol features are usable anywhere on the Internet with high probability. In particular, plain TCP over IPv4 packets without options is the most likely combination that will work on the Internet. Other protocols and features-- like IPv6, extension headers, QUIC, SCTP, fragmentation-- are only usable on different paths throughout the Internet with varying probabilities of success.

Ossification of the Internet did not come about by design. The original Internet architecture is generic, extensible, and well layered as enshrined in the *end-to-end* model and the *robustness principle*. Ossification came about through years of real implementation and deployment. Early on, developers filled in the gaps in standards to provide for security (firewalls), address sharing (Network Address Translation or NAT), and other product features. Assumptions were made, sometime non-conformant, out of practicality. For instance, some firewalls assumed that all communications are over TCP. At a time when TCP was the only transport protocol commonly in use, such an assumption expedited product development. However, there was a price to pay. In the long run, these ad hoc assumptions became baked in as de facto standards and subsequently ossified the Internet.

Network techniques of Deep Packet Inspection (DPI), proxies, and connection tracking have caused ossification in several ways:

- They only work with specific protocols. Attempts to use other protocols may fail. For instance, all NAT deployed solutions will support TCP, but very few can handle SCTP.
- They only work with the protocol features they are programmed for. For instance, a NAT device may fail to find a TCP header in an IPv6 packet if there is an extension header in the packet that the device is unprepared to deal with.
- When Transport protocols change there is a possibility that the change is incompatible with some anonymous intermediate node in the path thereby breaking communications.
- Devices that maintain transport state often make the implicit assumption that all packets of a connection traverse the same intermediate device in both directions. This forces a requirements on routing in the network and hinders multi-homing and multi-path.
- They can break security transport layer security mechanisms. For instance, a layer 4 TCP proxy would break the TCP authentication option.

Today's challenge

Fast forward to 2019. The Internet is no longer just about reading email and watching cat videos. There is an insatiable demand to connect applications of increasing complexity and criticality. These applications may have stringent latency requirements, operate on vast amounts of data, and have strict security and privacy requirements. We are seeing many possibilities in augmented and virtual reality (AR/VR), artificial intelligence and machine learning (AI/ML), real-time safety information provided to autonomous vehicles, machine to machine communications (MTM), applications in real-time robotics, and even remote surgery. With the emergence of IoT there is a projected explosion of numbers of connected devices on the Internet to possibly hit the trillion mark in the 2020s. And it's pretty much a given that the vast majority of devices will be mobile.

Underlying network technology is also advancing. Forward looking access technologies, such as 5G being developed by 3GPP, are in development and initial deployment. These provide the foundation for low latency, high throughput, and high number of users. Software Defined Networking, programmable network devices, such as those based on P4 programming language and eBPF bytecode, and the preponderance of open source are fundamentally changing the landscape of networking from one dominated by rigid and slow moving proprietary hardware design, to one based on flexible and fast moving software solutions.

Introducing I21

I21 is the initiative to bring the Internet into the 21st century. There is a bit of irony self-contained in this goal since much of the effort would be to steer the Internet back to its original design principles of simplicity, robustness, end-to-end model, protocol layering, and distributed architecture. I21 discourages the use of DPI and other mechanisms that break the end-to-end model and otherwise ossify the Internet. Alternatively, it promotes the use of standards compliant methods that leverage existing features to achieve desired functionality.

Before presenting the specifics of I21, we should mention a fundamental premise to be taken into account. The Internet is a decentralized entity. There is no overall authority, there are no protocol police, and compliance with standards is entirely voluntary. Neither is there an on/off switch, a reset button, or "flag days" for the Internet. What is implemented and deployed is typically motivated by the business case. Often the business case aligns nicely with user requirements, but sometimes it doesn't. This capitalist model of Internet innovation works well to satisfy short term needs, but things becomes more complicated when considering investments in longer term initiatives (the history of IPv6 deployment can attest to that). I21 needs to take this into account. While there is a long term vision, evolving the Internet towards that vision will be most effective if there is incremental value to be had along the way. Therefore, it's prudent to always have a business case in mind to help justify the cost of change.

Goals and themes

The goals of I21 are:

- **An open, extensible, scalable, and secure Internet**
- Promote net neutrality
- Provide alternatives to Deep Packet Inspection
- Stop ossification of the Internet
- Restore the end-to-end model
- Take a holistic view, solve problems in an ecosystem
- Cultivate cooperation amongst players without requiring implicit trust
- Highlight business case and incentive for change
- Monetization in a fair and enabling manner

In considering the goals and ways to achieve them, several design trends seem to emerge:

- **Don't "boil the ocean": Adapt what exists instead of forcing a new architecture.**
- "Less is more", simple solutions are better than complex ones
- Protocol conformance is important, as are first principles of networking protocols
- Ubiquitous encryption is a *good* thing, solutions should assume that
- Real cooperation between end hosts and networks is needed at some point
- Assume minimal trust, smart edge/dumb network
- Software Defined Networking and network device programmability are important

The business case

As mentioned in the fundamental premise of I21, the business case for change needs to be articulated. We consider the business case for I21 for three "players" in the Internet: network operators and providers, hardware and software vendors, application and content providers. The best business case is one that is able to align all of three of these, the needs of users, and potential for innovation.

Player	**Business case for I21**
Network operators and providers	Monetize network services like slices in 5G. "Simpler" means lower cost, less vendor lock-in, more competition, flexible billing.
Hardware and software vendors	Software centric implementations allow fast paced development and innovation. Open standards assure interoperability. Differentiation can be done in software and services.
Application and content providers	Enable new applications that leverage new networking technology. Net neutrality spurs competition and innovation. Shrink support matrix of application/provider/protocol to reduce development costs.

Manifestation and scope

I21 is composed of a number of sub-projects that address sub-problems. The sub-projects are aligned with the high level goals of I21. Sub-problems tend to fall in three categories: 1) well known problems for years, 2) known problems that only recently have gotten attention, 3) problems in trying to use forward looking features.

Problems that have been well known for years-- such as difficulties in using a protocol other than TCP-- receive much discussion in IETF circles. Many workarounds for these types of problems have been developed, but in the long run such workarounds end up contributing to ossification.

Problems related to privacy and security often fall into the bucket of problems that are known, but until recently haven't risen high enough in priority that they get attention. Such problems pop up when the threat and risk they pose becomes significant and obvious. For instance, over the past ten years compute and data analysis capabilities have increased for everyone, including would be attackers. So, the risks to security and privacy have also increased, and thus the threat posed becomes higher priority to address.

Problems that arise when trying to use forward looking features are tricky because deployment of a networking protocol often precedes attempts to use some of its "advanced features". An example of this extension headers in IPv6. In 1998, IPv6 was first standardized by IETF RFC2460. Extension headers provide extensibility in IPv6, and are a very powerful mechanism that could be used in many ways to innovate using the IPv6 protocol. But, extension headers were defined as a forward looking feature that was not critical to the normal operation of the IPv6 protocol (unlike the case with TCP options). As a result, many router vendors made a pragmatic decision to not invest in making extension headers work. Years after the fact, when interesting use cases for extension headers have emerged, they're not viable across the whole the Internet and hence inhibit deployment of features that depend on them.

I21 sub-projects

Here is a sampling ofI21 sub-projects:

- Community Happy Eyeballs and the Grand Map of the Internet
- Firewall and Service Tickets (FAST)
- Address Mapping System (AMS)
- Privacy in Internet addressing
- Identifier Locator Addressing (ILA)
- Sub-path transport layer
- Load balancing with VIPs (DNS BBBB records)

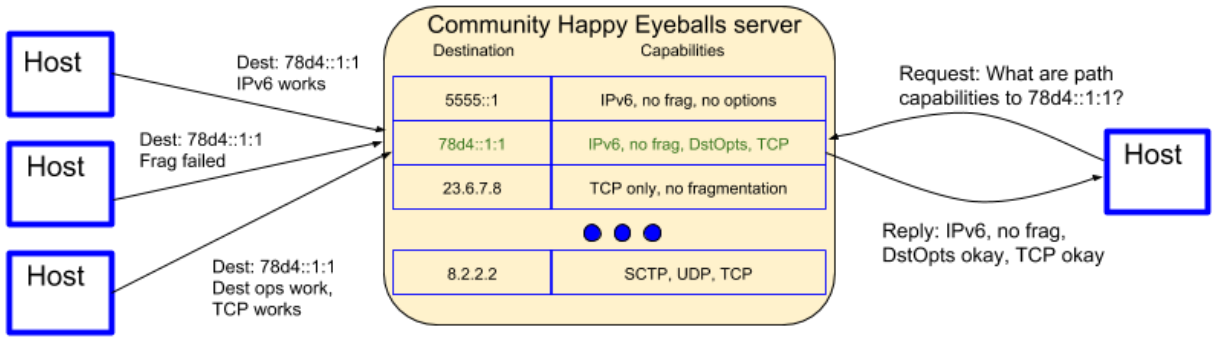
Community Happy Eyeballs and the Grand Map of the Internet

With the possible exception of TCP over plain IPv4, there are no protocols or protocol features that are guaranteed to work on the Internet. The reality is that *some* features and protocols work over *some* paths. This is not only caused by protocol ossification, but also the fact that new protocols take a long time to deploy (IPv6 for example). Given that support for standard features isn't ubiquitous, the question arises as to how to productively use features that might not be available on all paths.

To address the non-ubiquity of IPv6 in the Internet, *Happy Eyeballs* was created. The concept is fairly straightforward: try to connect to a peer on the Internet using IPv6. If the connection succeeds, then IPv6 is usable. If the connection fails, then just back-off and try using IPv4. While Happy Eyeballs contains considerably more detail to make the process efficient and robust, the basic concept of trying to use a feature or protocol and then backing off to a more rudimentary mechanism when that fails could be applied to other features. We call generalization of the approach *Community Happy Eyeballs*.

The idea of Community Happy Eyeballs is to apply knowledge gained through previous experience in communicating over the Internet. This can be implemented by creating a global and public map of capabilities of paths on the Internet (the *Grand Map of the Internet*). Given a source and destination address, the map could be queried to determine the expected protocols and features that will likely work in communications. The map is populated by the community of end nodes that individually probe and discover the state of various capabilities. Note the analogy between this and community based vehicle traffic maps like Waze.

The concept of Community Happy Eyeballs is diagrammed below.



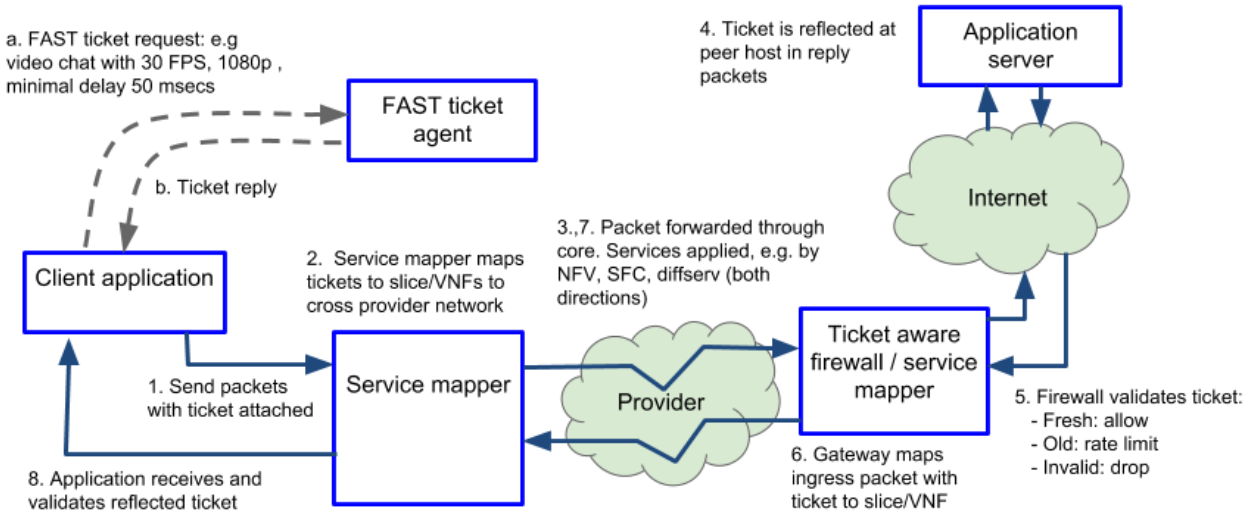
1. Different hosts probe destination for various features and report status to server

2. Host connecting to destination queries server for path capabilities

Firewall and Service Tickets

Firewall and Service Tickets (*FAST*) is a technique to allow an application to signal to the network requests for admission and services for a flow. A ticket is data that is attached to a packet by the source that is inspected and validated by intermediate nodes in a network. Tickets express a grant or right for packets to traverse a network or have services applied to them. FAST allows networks and hosts to work together to provide services to users without divulging the details of end user communications or resorting to DPI in the network.

An application requests tickets for admission or services from a ticket agent in their local network. A ticket request is an abstract description of traffic characteristics. The agent issues tickets to the application which in turn attaches them to its packets. In the forwarding path, intermediate network nodes interpret tickets and apply requested services on packets. Tickets have a number of security properties that prevent forgery or Denial of Service Attack (DOS)-- they are revocable, can be encrypted, have an expiration time, and are only understood by the network that issues them. To apply services to inbound packets for a communication, remote peers reflect received tickets in reply packets they send without interpreting them. An example use of tickets to provide service for a video chat is demonstrated below.



Addressing Mapping System

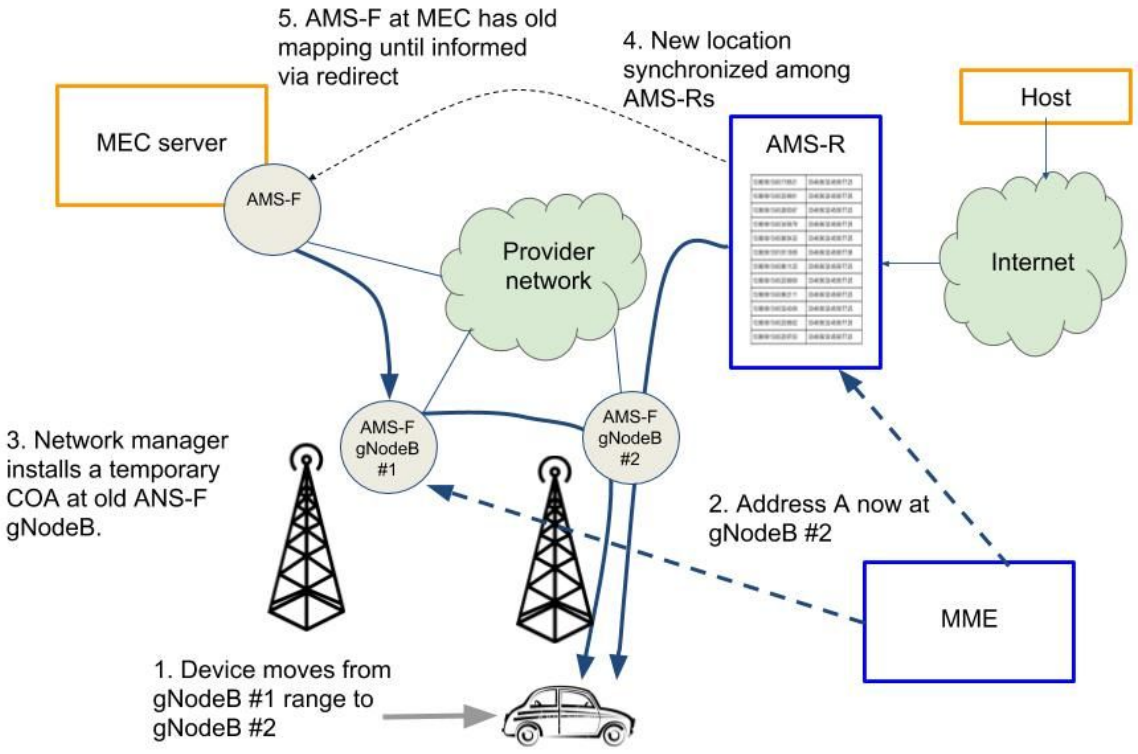
Address Mapping System (AMS) is a system that maps network addresses to other network addresses. The canonical use case is to map "identifiers" to "locators" (applying identifier-locator split terminology). Identifiers are logical addresses that identify a node, and locators are addresses that indicate the current location of a node. Identifiers are mapped to locators at points in the data path to facilitate device mobility or network virtualization.

The address mapping system may be queried on a per packet basis in the data path. For instance, an encapsulating tunnel ingress node for virtualization would perform a lookup on each destination virtual address to discover the address of the physical node to which a packet should be forwarded. It follows that access to the mapping system is expected to be tightly coupled with nodes that query the system to perform packet forwarding.

The mapping system contains a database or table of all the address mappings for a mapping domain. The database may be distributed across some number of nodes, sharded for scalability, and caches may be used to optimize communications. The mappings in a mapping system may be very dynamic, for instance end user devices in a mobile network may change location within the network at a high rate (e.g. a mobile device in a fast moving automobile may frequently connect to different cells). Protocols are defined to synchronize mapping information across devices that participate in the address mapping system.

The mapping information is managed by *AMS routers* (*AMS-R*) and *AMS forwarders* (*AMS-F*) AMS routers maintain and share the set of mappings for an AMS domain and are considered authoritative for a mapping domain. AMS forwarders are placed close to the end hosts and can maintain a working set cache of mappings as an optimization for communications.

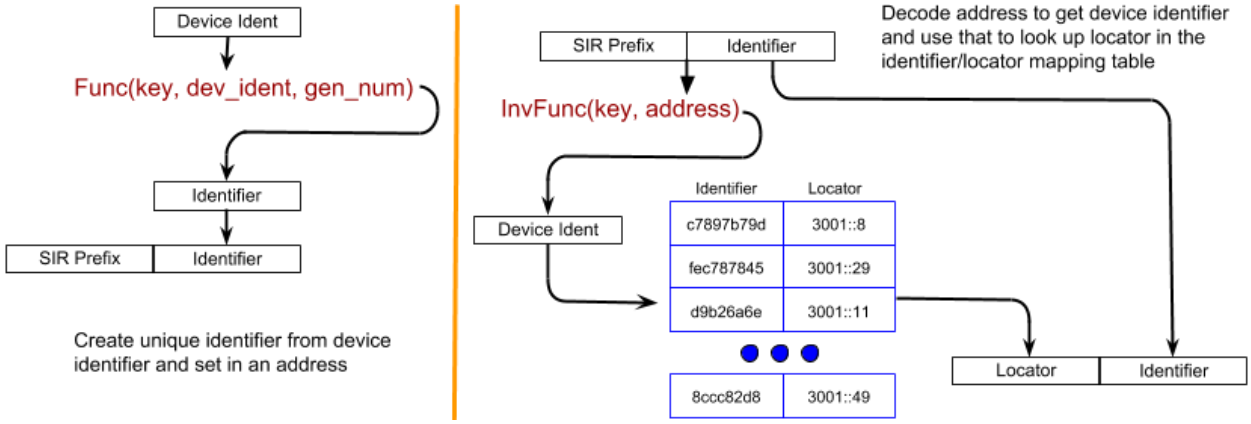
The diagram below illustrates the use of AMS in a mobile network.



Privacy in Addressing

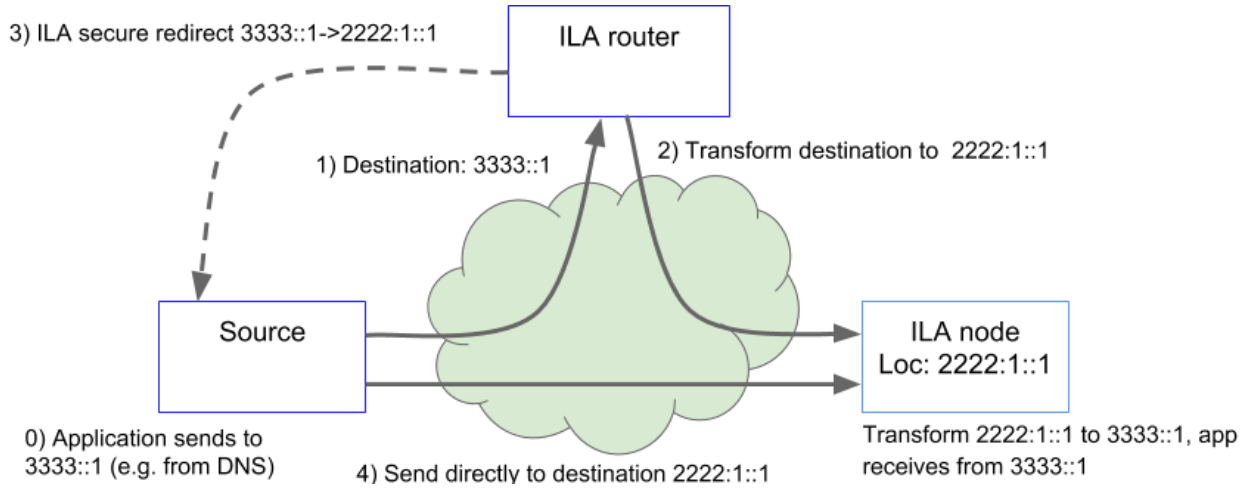
In the original IPv6 addressing model, subnets (links) were assigned a sixty-four bit prefix . Hosts in the subnet would then generate IIDs that are combined with the subnet prefix to create IPv6 addresses. This model was subsequently extended to assign network prefixes, such as /64s, to general purpose hosts. When a prefix is assigned to an end host, the prefix becomes an identifier for the host. So, if two such addresses have the same prefix (i.e. same upper sixty-four bits) then they can be assumed to refer to the same host. The IID portion of the addresses (lower sixty-four bits) are immaterial in this inference, so IID generation techniques don't affect the ability to make correlations. The fact that two addresses can be correlated to be from the same host is a privacy concern. If an attacker knows that a network provider assigns /64 prefixes to end hosts, as is common in mobile networks, then it can deduce that two addresses in the provider prefix sharing the same sixty-four bit prefix refer to the same host. With a little more information, an attacker may not only deduce two addresses refer to the same end host, but also may be able to discover the user's identities in communications.

Identifier/locator split can be used to address this privacy issue. Strong privacy in addressing can be achieved by using a different randomly generated identifier address for each flow. Conceptually, this would entail that the network creates and assigns a unique and untrackable source address to a host for every flow created by a host. Scaling this solution is a challenge since every flow would create a unique address that needs to be in the identifier/locator mapping table. This can be addressed by *hidden aggregation*, whereby addresses are aggregable using a method that is known by the network provider, but hidden to the rest of the world. In this method, a keyed function creates a unique identifier given a device identifier that represents the device node. An inverse function is used to decode an address and return the device identifier. Addresses creation and identifier lookup for addressing privacy with ILA are shown below.



Identity Locator Addressing

Identifier-Locator Addressing (*ILA*) is a networking protocol for mobility and network virtualization. ILA is an identifier-locator split protocol meaning that identity and location are distinguished in IP addresses. In ILA, part of an IPv6 address indicates the identifier of a node ("who") and part of the address indicates the node's location ("where"). To achieve communications, ILA address transformation is performed which sets the locator in an address appropriately to facilitate forwarding across a network underlay towards a mobile node.
The diagram below demonstrates the basic operation of ILA:



ILA can be used to optimize mobility in a mobile network. It would be used in conjunction with *anchorless-mobility* where packets take the most direct path to a mobile device. The control plane for anchorless mobility is facilitated by the Address Mapping System described above..

Sub-path Transport Layer

Different portions of a network path for a flow, *sub-paths*, may exhibit radically different characteristics. For instance, a radio network portion of a path may be much more susceptible to congestion and packet loss relative to the rest of a path for a flow. End-to-end mechanisms for congestion control and reliability don't take this into account, so transport layer end points are often far removed from problems. The result is that congestion control and reliability may be sub-optimal and have high response times to problems originating in a sub-path.

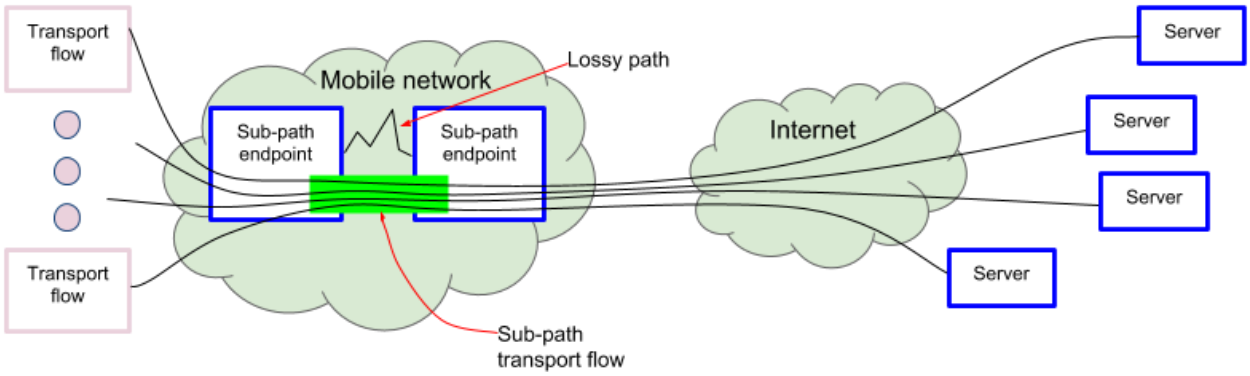
A solution is to run a congestion control and reliability protocol over relevant sub-paths in a network. This would constitute an in-network transport layer below the end-to-end transport (e.g. TCP). This has already been done in a couple of ways: some layer 2 protocols implement their own reliability and congestion control, and transport layer proxies have been used within

networks to overlay transport connections over the sub-path of interest. Neither of these techniques are generic. Layer 2 solutions are below IP layer and are technology specific; proxies only work with select transport protocols and also break the end-to-end model.

An important consideration in the design of a sub-path transport layer is how it interacts with a higher layer transport such a TCP. Accordingly, there are two levels of requirements for a sub-path transport layer:

1. Avoid conflict with higher layer transport protocols
2. Work in concert with higher layer transport protocols

An example use of sub-path transport layer is diagrammed below. In this picture four end user flows are being aggregated into one sub-path transport flow for transit across a lossy path in a mobile network. Retransmission, congestion control, and other transport layer protocol mechanisms can be done over the sub-path between sub-path endpoints.



VIP-less Front Ends and Load Balancers

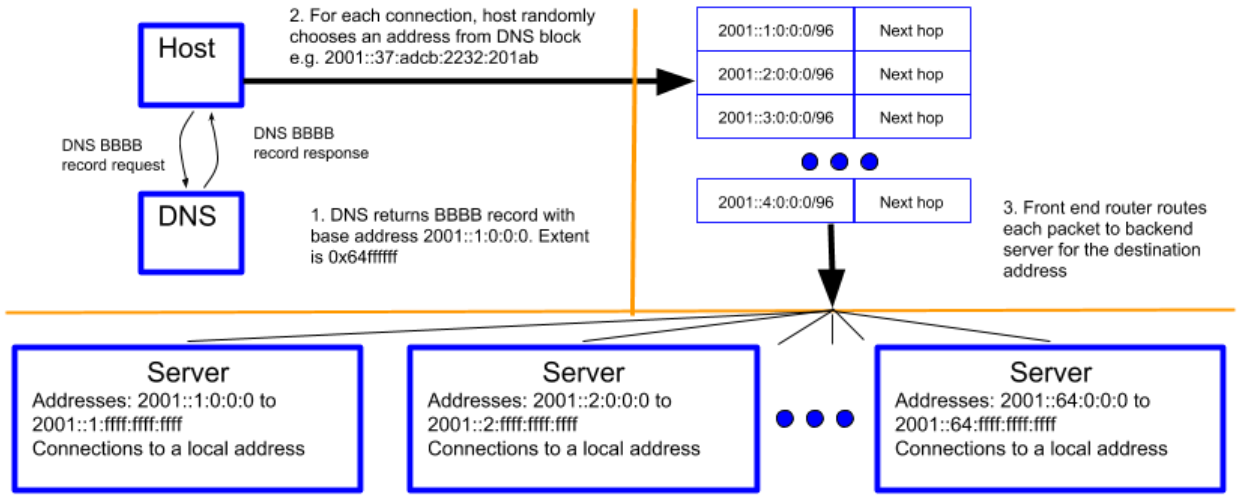
Layer 4 Load balancers are quite common as front ends to content service providers such as Google and Facebook. In this model, a service is reachable by a single IP address. The address is a *Virtual IP Address* (*VIP*), meaning that it is shared amongst some number of backend servers that terminate transport connections. For each stateful flow, only one backend server has stateful context (e.g. a TCP connection). A load balancer *must always* forward packets for a flow or connection to the same backend server. If a packet is forwarded to the wrong server, then that server won't have the flow context and hence would drop the packet.

The problem that load balancers must consistently forward packets with VIP addresses is well known. For instance, Google's Maglev load balancer using a combination of consistent hashing and connection tracking to load balance traffic to backend servers for a VIP address. Even with a finely tuned solution, there is still a non-zero percentage of flows that are terminated because

packets aren't forwarded to the correct server. The underlying problem is that one address for a service is being shared amongst multiple servers. We propose an alternate solution to use many addresses for the service where each address is unique to a server. In this way, VIPs can be eliminated and load balancing becomes a natural artifact of routing packets.

In order to eliminate VIPs, two things are required: 1) A means to assign multiple addressed and route them to individual servers 2) A means to set multiple addresses in a DNS record. For the first item, IPv6 fits the bill. IPv6 addresses allows multiple addresses to be assigned to each server-- not just a few addresses, but millions or billions of them. For the second item, we propose a new DNS record type, called BBBB, that provides blocks of IPv6 addresses as a base address and number of contiguous addresses in the block starting from the base address.

The concept of VIP-less load balancing with DNS BBBB records is shown below.



Conclusion

The Internet is undoubtedly an amazing success. However, like so many other technological achievements, if can't continue to evolve it will become a victim of its own success. The key is to promote innovation without breaking existing functionality. There are many avenues for innovation in security, services, privacy, and connectivity to be pursued. In order to explore these fully, the Internet needs to be a platform for innovation. But, the Internet is *useful* and *critical* infrastructure, hence the introduction of innovation must be measured and evolutionary. I21 is an initiative to facilitate innovation in the Internet without breaking it.

The premise of I21 is simple: apply the original principles of layering, end-to-end model, distributed architecture, and standard interoperable protocols. Additionally, we accept the fact that there are business interests at work. The Internet has created immense monetary value

and spawned whole industries-- the business case for “doing the right thing” needs to be articulated.

Given the premise and goals of the I21 initiative, we have elaborated on a number of sub-projects. A common thread amongst of these projects is that they are enablers for innovation. Not just innovation for the sake of innovating, but innovation that materially improves user experience, enables new applications, and allows fair monetization to get return on investment. The list of projects described here are in no way a complete list. I21 itself must be dynamic and evolving, so in time we expect more projects with similar themes to be proposed under the umbrella initiative.