Transit as a Service

Simon Peter Umar Javed Thomas Anderson Arvind Krishnamurthy

Abstract

Increasingly, the Internet is being used for services, such as home health monitoring, where correct and continuous operation is essential. Yet the current Internet is not up to the task. There are numerous issues that should make anyone pause before trusting the Internet with the timely delivery of something that truly mattered.

We propose a novel approach, called Transit as a Service (TaaS), that is designed to allow for enterprises and governments to configure reliable and secure end to end paths through participating providers. Unlike efforts to redesign the Internet from scratch, TaaS provides ISPs incremental incentives to adopt. A highly reliable ISP can offer transit through its network as a service to remote paying customers. Those customers can stitch together reliable end to end paths through a combination of participating and non-participating ISPs in order to improve the fault-tolerance and robustness of mission critical transmissions. We provide an implementation of TaaS, evaluate its performance in testbed settings, and demonstrate using simulations its ability to provide improved reliability and security.

1 Introduction

Increasingly, the Internet is being used for services where correct and continuous operation is essential: home health monitoring, active management of power sources on the electrical grid, 911 service, and disaster response are just a few examples. In these and other cases, outages are not just an inconvenience, they are potentially life threatening. A less life critical, but economically very important case is presented by the outsourcing of enterprise IT infrastructure to the cloud – connectivity outages to cloud servers can imply high costs due to disruptions of day-to-day business activities.

In summary, the Internet has become a necessary part of the world's economic infrastructure. However, the present Internet is not up to the task. Operational experience has uncovered numerous issues that would make anyone pause before trusting the Internet with the timely delivery of something that truly mattered. The list of known causes of outages is long. For example, router and link failures can trigger convergence delays in the Border Gateway Protocol (BGP). When combined with configuration errors on backup paths, outages can last for hours and even days. Often these outages are partial or asymmetric, indicating that a viable path exists but the protocols and configurations are unable to find it. Other problems that can and have triggered outages: required maintenance tasks such as software upgrades and policy reconfiguration, router misconfiguration, massive botnet denial-of-service attacks, router software bugs, ambiguities in complex protocols, and malicious behavior by competing ISPs. Even if the traffic is delivered, there are other vulnerabilities. For example, traffic from the US Department of Defense was recently routed through China. While it is unclear whether the problem was inadvertent or intentional, the Internet lacks any protocol mechanism from preventing this type of event from recurring.

Because of its scale, the Internet is of necessity multi-provider, and end-to-end routes often involve multiple organizations. While a number of research projects have proposed tools to diagnose problems (e.g., [12, 13]), and fixes to specific issues, such as prefix hijacking [4, 14, 17, 19], route convergence [11], and denial-of-service [5, 26, 34], there has been little progress towards deployment except in a few cases. Part of the problem is incentives. Many of the proposed solutions are only truly valuable if every ISP adopts; no one who adopts first will gain any advantage.

Another part of the problem is completeness. Is there a *set* of fixes that together would mean we could trust time critical communication to the Internet? Most existing proposals are only partial solutions. For example, Secure BGP addresses some of the vulnerabilities surrounding spoofed routes, but it doesn't address denial of service or route convergence. The resulting commercial case for deployment is weak.

We attempt to answer a simpler question: what are the minimal changes to the Internet needed to support mission critical data? We note that reliability is not equally important for all traffic. Our requirement is to design a system that will provide highly available communication for selected customers as long as there is a policy compliant physical path, traversing only trustworthy ISPs, and without diverting the traffic to non-trustworthy ISPs. This property should hold despite node and link failures, software upgrades, operator error or byzantine behavior by neighboring networks, and denial-of-service attacks by third parties. We assume ISPs and cloud

providers have a strong incentive to make their own networks highly available and robust against failures and attacks. How can we best leverage their work for end-to-end resilience?

In this paper, we propose a system called *Transit as a Service (TaaS)* that allows ISPs to *sell* reliability and security *as a service*, without widespread adoption happening first. End users can obtain this service from any ISP offering it, including ISPs that do not face end-users and primarily serve the backbone of the Internet. At the core of our system is a protocol to secure a provisioned path across a remote ISP. The remote ISP promises only what it can guarantee itself: a high quality path across its own network. The end host (or data center or enterprise or local ISP or government) is responsible for stitching together TaaS into an end-to-end solution. Like local transit, TaaS is paid for by the requestor, arranged over the web in much the same way as one would purchase computing cycles in Amazon's EC2 data center.

We make the following contributions:

- We present the design of TaaS, including how its main requirements, incremental deployability, high availability, and robustness, are achieved.
- We present the TaaS API that can be used by ISPs and end-users to find, reserve, and establish paths on the Internet.
- We present two different implementations of TaaS, based on the Click software router and the Serval [25] network protocol stack, respectively. We demonstrate a deployment of TaaS on the Internet using the Serval-based implementation and how it can be used to establish a path despite blocked Internet links.
- We evaluate TaaS's benefits both in simulation and experimentally. Our evaluation shows TaaS's resilience against IP prefix-hijacking, link failures, path performance problems, and byzantine ISP failures, with little overhead to Internet routing performance.

Next, we sketch the reasons that mission critical services should not rely on the Internet today. We then outline our approach in Section 3, describe several ways we have implemented TaaS in Section 4, evaluate TaaS in Section 4, and discuss related work in Section 5.

2 Motivation

Consider the following example scenarios.

Example 1: Imagine a healthcare monitoring application that operates over the Internet. The patient wears a monitoring device, and the measurements are sent to a data center or to the doctor's location. These measurements are analyzed in real-time, and anomalies are forwarded to alert human experts who can ensure that no medical problem has occurred (for example, side-effects from a concurrent therapy) or might then use interactive video streams to perform further diagnosis. The challenges of supporting such applications are substantial. The network must provide high availability because the network may be part of a life-critical medical feedback loop with timeliness constraints. It must also provide desired levels of quality of service, i.e., provide high bandwidth streams with low loss rates. These services should not be disrupted by transient changes in underlying paths either due to cross-traffic or due to BGP dynamics.

Example 2: A large enterprise that is physically distributed across multiple sites, such as a Fortune 500 company, needs to use the Internet for inter-site communications, serving its customers, and accessing outsourced IT services in the cloud. It might have multiple requirements for its communications: traffic should be communicated reliably even in the presence of outages, there should be no information leakage due to traffic analysis, and traffic should be robust to security attacks such as prefix hijacking. To address these concerns, it wants to ensure that its traffic only traverses a set of pre-approved, trustworthy providers or a predictable set of ISPs that satisfy certain geographical/jurisdictional requirements. This is impossible to guarantee today. Near the source, an ISP can select BGP routes to a specific destination that obey certain restrictions. However, those routes can be changed by the downstream ISPs without pre-approval or prior notice. Only after the fact will BGP inform the upstream users of the path of a change. Near the destination, the ISP has no standard way to signal that it should only be reached through pre-approved paths or through a predictable set of trusted ISPs.

The previous examples highlight just a few problems of Internet use for mission-critical services. A recent survey by Trimintzios et al. [29] enumerates other known security vulnerabilities of the Internet. A few examples include disruption of service by resource exhaustion attacks by botnets against network links and end hosts, prefix hijacks by malicious ISPs, and byzantine errors by neighboring ISPs (e.g., intentional disaggregation of addresses, causing routers to crash).

Even without vulnerabilities to malicious attack, the Internet protocols are operationally fragile: Internet paths are often disrupted for short periods of time as BGP paths converge. Operational changes, such as reboots or rewiring, and divergence between the control and data plane can also reduce availability. With today's protocols, an endpoint has no recourse in this case but to patiently wait for the problem to be repaired.

In our work, a key observation is that the amount of traffic for mission-critical applications can be quite small, especially compared to normal everyday Internet

use. Yet this traffic is often very high value. Our proposal targets just these low-volume, high value applications. Most users find most of their Internet traffic works well enough most of the time, because much of the traffic on the Internet is for content delivery from nearby cached copies. For this type of traffic, the most critical factor is the reliability of the local ISP. Internet reliability is of course still an issue for many users, but it is hard to argue this part of the problem requires an architectural fix beyond designing better tools for network operators to diagnose their own networks.

Our focus is thus on developing a system that can enhance the reliability and performance of mission-critical traffic using solutions that are incrementally deployable and provide benefits even when it is deployed by a small number of ISPs. Further, re-architecting the Internet from ground up seems overkill for such a small amount of traffic, no matter how important in human or commercial terms. Given the large number of known problems, it is unlikely that even a well-designed set of changes would fix every problem, and a massive change to the Internet protocol suite would run the risk of having unintended side effects.

3 TaaS Design

We would like to develop a simple primitive that could be used to provide highly available communication in addition the the Internet's normal uses, as long as there is a usable and policy compliant physical path between a pair of endpoints. To this end, the key requirements of our solution are:

Incremental Deployability: In today's Internet, a provider ISP (or ISPs) mediates Internet service. This poses a chicken and egg problem; an ISP can't promise or charge for a new type of service unless all, or almost all, other ISPs already provide the service. We want to make it possible for end users, enterprises, and governments to leverage reliable intradomain paths made available by remote ISPs, *without* requiring global adoption of new protocols.

High Availability: We want endpoints to be able to establish one or more high quality paths across the Internet, provided a physical path exists through ISPs willing to be paid for the service. For availability, endpoints need the ability to route around persistent reachability problems, as well as to establish multiple paths to minimize disruptions due to transient routing loops and blackholes.

Robustness: Because security attacks against the Internet are a real threat, we need to provide endpoints the means to defend their routes, both by *proactive* installation of desirable paths and filters and *reactive* rerouting of traffic in response to degradation in packet delivery.

In this section, we provide a brief overview of our proposed approach before describing the key components of our design.

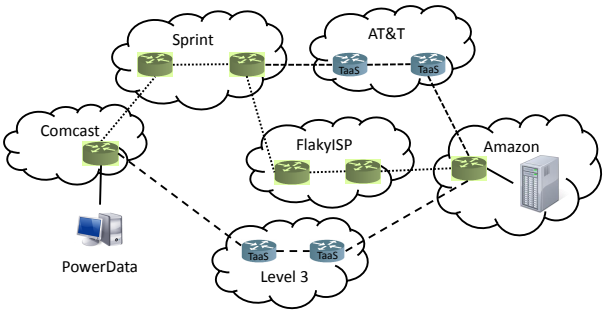

Figure 1: Three example TaaS paths from PowerData to Amazon: the dotted lines represent the BGP path. The two dashed lines are TaaS paths.

3.1 Overview

We provide an overview of our approach using a simple example shown in Figure 1. A company called PowerData is using Amazon cloud services for its day-to-day data storage. Using BGP, traffic to Amazon would be routed via Comcast (PowerData's upstream ISP), Sprint, and either FlakyISP or AT&T. However, FlakyISP often drops packets and has caused PowerData's service to be slow whenever the path through FlakyISP is chosen by Sprint and Comcast. Note that, while PowerData can find out about the problem using various available Internet measurement technologies, it has limited or no control over the paths selected by Sprint (a remote ISP) and Comcast (the local transit provider).

To remedy this, PowerData buys TaaS transit from AT&T, which involves provisioning a path through AT&T and establishing the appropriate packet forwarding rules to transmit PowerData packets along to Amazon and received responses back to PowerData. This ensures that PowerData packets to and from Amazon are routed around FlakyISP since it does not appear on any of the paths between Comcast and AT&T nor does it appear on the paths between AT&T and Amazon. Note that PowerData does not have to provision paths across every ISP on its path to/from Amazon in order to avoid FlakyISP. Rather, a limited amount of route control at a remote ISP (AT&T in this example) might suffice to achieve the desired paths.

To make sure that reconfigurations and temporary outages (for example, due to routing loops or misconfigurations) at Sprint and AT&T do not impact PowerData's service, PowerData also buys TaaS transit from Level 3 and can fail-over to this path in case of problems with the original path.

The example illustrates several properties of our proposed approach. First, the system is incrementally de-

ployable by an ISP, with incremental incentives to that ISP. An ISP can provide TaaS even if none of its peer, customer or provider ISPs participate in the protocol. TaaS benefits from a network effect, but it still provides value to enterprises and data centers needing to control routes even if only a few ISPs have adopted the approach. In the example in Figure 1, TaaS is still useful to PowerData even if Sprint does not provide TaaS transit.

Second, TaaS aims to require only modest changes to the existing Internet infrastructure to facilitate deployment. We assume no changes to normal traffic, but we do require that mission critical traffic be specially encoded to simplify packet processing at the router. Most mission critical services are new, so requiring a slightly modified protocol stack is less of a concern. Alternately, we explain how a local ISP could offer an end to end service to its clients, by rebundling their mission critical traffic to use TaaS.

In the rest of this section, we present the TaaS design and outline the key components of our proposal including:

- the management interface for setting up transit through a remote ISP,
- the data plane operations required for supporting remote transit,
- the issues in setting up end-to-end paths, monitoring them, and responding to changes in path quality, and
- business considerations that affect the adoption of the proposed scheme.

3.2 Setting up Remote Transit

Today, ISPs provide transit only to their immediate customers. The key idea with TaaS is to generalize the notion of transit, to allow an ISP to offer its transit as a service to anyone on the Internet. TaaS is optional: as with paid transit today, ISPs can choose to offer the service, or not, at whatever price point they like.

An ISP offering TaaS advertises its willingness to provide its transit, for a fee, via SSL, much as is currently done for cloud providers offering computer time. The control traffic (to find out about advertised TaaS tunnels, and to request the tunnel) can be carried over the existing Internet, at least at first. For one, ISPs already have an incentive to ensure that their own addresses can reach the rest of the Internet. But to the extent that an ISP finds its routes to its TaaS customers unreliable, it can use TaaS mechanisms to bootstrap more reliable routes that can be used for the contol traffic.

The ISP operates a portal that provides interested users with an interface for obtaining information regarding its TaaS service. We have implemented an RPC-based query interface, shown in Table 1. Users must authenticate with the service before calling any of its functions. If transit is granted for a fee, registering with an ISP's service would typically involve an exchange of the customer's credit card information.

An ISP can exercise fine-grained control over its TaaS service; it can offer it between all, some, or none of its peers, direct customers, or providers. Further, the transit provided can either be bandwidth-provisioned or best effort. For instance, it can offer strict transit SLAs (e.g., constant bit rate pipe with a maximum latency bound on its intra-ISP paths), guarantee protection across DoS attacks, or simply provide best-effort guarantees. Likewise, the pricing can be on any mutually agreeable terms, e.g., with a bandwidth cap or not, priced based on total bytes transferred or based on burst bandwidth, etc.

We envision most TaaS connections will be established as redundant fixed bandwidth pipes, ensuring that, say, home health monitoring data will be delivered regardless of failures, denial of service attacks, or Byzantine behavior by unrelated ISPs. Because TaaS tunnels are set up in advance, links within an ISP might run out of excess capacity, but that only prevents future TaaS tunnels from being set up; existing agreements can stay in place. Market prices can then signal a need for more capacity.

If the TaaS tunnel is best effort, this promise is nothing other than an enhanced version of what it already provides its (direct) customers: transit for specific packets across its network, from a specific ingress ISP (and optionally, ingress link) to a specific egress ISP (and optionally, egress link). Even with this small extension, TaaS customers can construct end-to-end paths that they normally wouldn't be able to use and could thus achieve improved resilience and route control for their communications.

An endpoint desiring route control sets up TaaS circuits through remote ISPs in order to ensure that its packets traverse pre-determined paths. The endpoint contacts one or more of the TaaS ISPs on the routing path and requests provisioned paths through the individual networks. The TaaS customer then arranges for the routing of the packet by associating with each hop the address for each subsequent hop that needs to be traversed.

Endpoints need a way to determine a path to their desired destination. From the ISP-provided lists of ingress and egress PoPs, endpoints are able to compile an atlas, where TaaS-providing ISPs can be marked. Any shortest path discovery algorithm can be used on the atlas to determine which ISPs to use to create an end-to-end circuit. It is realistic to assume that another Internet webservice maintains the atlas and provides a path query interface, returning paths according to any of a number of these

`[(pop_ingress, pop_egress), ...] = get_pops()`	Returns a list of ingress and egress PoP IP addresses, through which the ISP can be transited.
`string = query_sla(pop_ingress, pop_egress)`	Returns the service level agreement (SLA) that the ISP is willing to provide for an inter-PoP segment (from `pop_ingress` to `pop_egress`) as a human-readable string. The SLA includes possible performance guarantees and pricing.
`(pop_ingress_ip, pop_egress_ip, authenticator) = acquire_sla(pop_ingress, pop_egress)`	Acquires an SLA for transit on an inter-PoP segment (from `pop_ingress` to `pop_egress`) and returns the ingress and egress PoP IP addresses, as well as an authenticator. As we will discuss later, the primary purpose of the TaaS authenticator is to identify the TaaS tunnel corresponding to the arriving packet and to prove that the endpoint originating the packet is authorized to use the transit.
`relinquish_sla(authenticator)`	Relinquishes a previously acquired SLA, by accepting an authenticator returned from `acquire_sla()`.
`chain_path(auth, next_hop_addr, next_hop_auth)`	To allow TaaS tunnels to be chained, this call can optionally configure an existing TaaS tunnel (identified by its authenticator) with the TaaS address and TaaS authenticator of the next (TaaS) hop. Routers responsible for this TaaS tunnel are updated dynamically by the ISP upon calling this function.

Table 1: Path query interface, provided by TaaS-supporting ISPs.

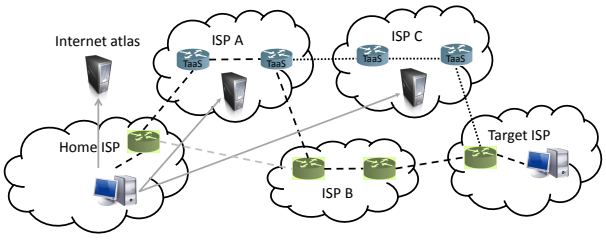

Figure 2: Example TaaS transit setup process. Blue routers marked TaaS are TaaS-compatible, gray routers are not. Dashed lines show the path chosen in case (a). Dotted lines show the detour taken via another TaaS ISP in case (b). The BGP path in this example is **Home-B-Target**.

algorithms.

Figure 2 shows an example of an Internet endpoint arranging a TaaS circuit with a target endpoint, registering with a number of ISPs. It also shows how the circuit is maintained by each of the ISPs. Two cases are considered:

(a) A tier 1 **ISP A** that is the provider for our home ISP supports TaaS and a circuit is created via this ISP. Other traffic is routed via BGP through a non-TaaS supporting **ISP B** to the final destination. The path in this case is **Home-A-B-Target**.

(b) An additional TaaS-supporting **ISP C** is configured to avoid the non-TaaS **ISP B**. In this case, we configure **ISP A** to forward packets to **ISP C**, via TaaS. The path in this case is **Home-A-C-Target**.

In both cases, the endpoint first contacts an Internet atlas service to determine which ISPs to contract for TaaS service. Then, the home endpoint contacts servers of interesting TaaS ISPs on the circuit to create TaaS SLAs, providing necessary next-hop information. A possible sequence of calls for both cases is demonstrated in Figure 3. In the figure, we leave out the step that determines the "best" TaaS ISP and instead select a low-latency PoP of **ISP A**. In the figure, `isp_a` and `isp_b` are pre-initialized RPC objects to the TaaS ISPs of Figure 2 and `atlas_query()` queries the Internet atlas service for paths with a latency below 300 milliseconds between a source IP address and a number of destination IP addresses. The function accepts the source IP address, followed by an array of destination IP addresses and returns an array of paths that match the latency requirement. We will cover in Section 4 how such a function might be implemented. Other functions from Table 1 are used to setup the routers of both ISPs to create the circuit.

TaaS paths are unidirectional. The reverse path can be provisioned either by the peer party, or by the originator of the traffic. This might depend upon the relationship of the peers. If the source party is a customer of some cloud service, it makes sense for the source to provision both paths. A peer-to-peer relationship can be provisioned by either peer individually.

3.3 Data Forwarding

To route traffic via TaaS, the source (or its proxy) encapsulates the IP data packet in a separate IP envelope, with the destination set to the first hop TaaS address (Hop Addr), the next-level protocol field set to a value identi-

```
# Case (a)
# Get all advertised TaaS PoPs of ISP A
t1_pops = isp_a.get_pops()
# Keep paths with latency <300ms to the ingress PoPs
src_to_t1 = atlas_query(src_ip, t1_pops.ingress())
# Take 1st path and get corresponding egress PoP
t1_egress = t1_pops.egress(src_to_t1[0][-1])
# Establish SLA
(t1_in, t1_out, t1_auth) =
  acquire_sla(src_to_t1[0][-1], t1_egress)

# Case (b)
# The same between PoPs of ISPs A and B
t2_pops = isp_b.get_pops()
t1_to_t2 = atlas_query(t1_egress, t2_pops.ingress())
t2_egress = t2_pops.egress(t1_to_t2[0][-1])
(t2_in, t2_out, t2_auth) =
  acquire_sla(t1_to_t2[0][-1], t2_egress)
# Chain TaaS PoPs of ISP A and B together
isp_a.chain_path(t1_auth, t2_in, t2_auth)
```

Figure 3: Call sequence to setup the two TaaS paths of Figure 2.

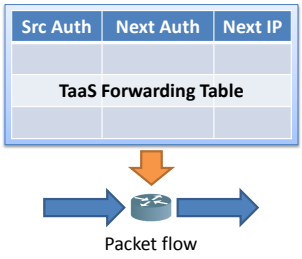

Figure 4: TaaS routing.

fying TaaS (TaaS Prot), and the first hop TaaS authenticator inside the packet (Hop Auth). Figure 5 shows a diagram of all relevant fields of such a TaaS packet.

The source sends its packets normally through their local network to their ISP. If the customer has a chain of TaaS providers, then each is set up with the address and authenticator of the next hop; the last ISP in the chain removes the IP header encapsulation before forwarding. Figure 4 shows this process. If some ISPs support the protocol and others do not, normal Internet routing can be used between the participating hops. While this does not completely prevent all BGP problems from affecting the traffic, it reduces the scope by reducing the length of the BGP path. For example, a tier 1 ISP at the core of the Internet is only one or two BGP hops away from most Internet addresses. Further, because tier 1 ISPs control a large portion of IP networks, BGP routes to/from tier 1s are more reliable than arbitrary Internet paths because of BGP filtering at lower tier ISPs.

We do require some level of hardware support in routers, but it is minimal and similar to the hardware already in place on most routers. In many ISPs, ingress

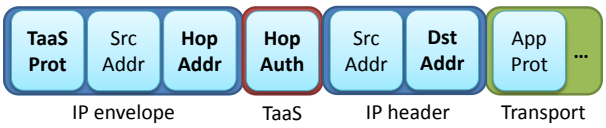

Figure 5: Relevant fields of a TaaS packet (in bold). *Dst Addr* is the IP address of the final destination. Note that the source endpoint IP address and other IP header fields are duplicated by the envelope IP header.

routers demux incoming traffic based on the destination address to a specific MPLS tunnel to route the traffic across their network. We can leverage similar hardware support in TaaS: the ingress router must be able to demux on the TaaS address (and if necessary the TaaS authenticator), route the packet using MPLS or other means, and then modify the IP header to insert the next hop address and authenticator. Alternately, ISPs can operate high-speed software routers (e.g., RouteBricks [6], PacketShader [10]) at ingress/egress PoPs to perform the necessary tasks for TaaS traffic.

The TaaS authenticator is simply a 64-bit sequence generated by the ISP for each TaaS circuit and included in every packet sent by the endpoint. The router must check the authenticator and drop the packet if it does not match. We consider the authenticator a hint. Provided that the intermediate ISPs do not eavesdrop on the data stream, the authenticator uniquely identifies the sender. Even if the authenticator is compromised, the only penalty is that the customer of the service is charged for extra unrelated traffic traversing the pipe. Since packets transmitted through a TaaS circuit are sent to a specific destination (or a subsequent TaaS circuit), an authenticator cannot be used with arbitrary flows and thus has limited utility even if compromised.

We note that it is relatively straightforward to safeguard against eavesdropping at a slightly increased cost to packet handling. Instead of transmitting the ISP-provided authenticator, the endpoint can include in each packet the hash of the checksum of the packet and the authenticator; misbehaving ISPs can then only replay entire packets, but they cannot use snooped authenticators for other packets.

3.4 Security and Redundancy

The impact of DoS attacks on TaaS PoPs is mitigated by dropping traffic at the ingress point of TaaS tunnels if the packets sent to it contain an invalid authenticator. Furthermore, TaaS paths can be setup to resemble swarms of packet forwarders [5]. This can be used to increase path availability in the face of DoS attacks and Figure 6 demonstrates how this can be achieved: multiple TaaS segments are configured within one TaaS supporting ISP to mitigate the effects of DoS attacks to TaaS ingress

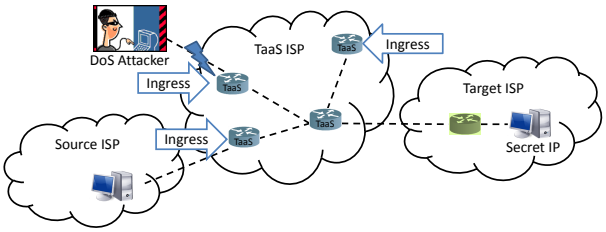

Figure 6: DoS attack prevention using TaaS.

points. Since TaaS clients can migrate their traffic to another ingress PoP if their PoP is overloaded, attackers have to overwhelm all provided ingress points simultaneously in order to stop traffic to the destination. If the destination endpoint's IP address is kept secret, it does not matter whether the TaaS supporting ISP is the provider of the endpoint or a random ISP on the Internet. Otherwise, if the provider of the destination endpoint provides TaaS, the endpoint's ISP can drop all non-TaaS traffic and protect the endpoint in this way.

TaaS ISPs will likely have multiple redundant paths between the ingress and egress PoPs, and can thus use MPLS mechanisms to configure backup paths and switch the intradomain paths in a seamless manner to handle failures inside a circuit. For instance, MPLS Fast Reroute allows routers inside the ISP to redirect traffic onto a predetermined backup path when they detect failures in upstream routers [27]. Failover will be the rare case, however. Routers are being constructed that continue to offer service despite component failures and even during software upgrades [3]. Of course, these existing solutions do not work across ISPs.

What happens when an intermediate ISP on a path does not support TaaS? The packet has to be routed via BGP to the next TaaS hop. If this is the case, it is important to know how many hops are in-between the TaaS hops to gauge the vulnerability of the connection to attacks and failures. To figure out the number of intermediate hops, we can initiate a traceroute to originate from the last TaaS supporting hop of the source end of the path, directed at the first TaaS hop of the destination end of the path. This can be done by sending a traceroute via TaaS from the source endpoint to the first TaaS hop of the destination end through the existing partial circuit. To support this and other network debugging tasks, TaaS routers need to be able to respond to ICMP echo requests.

3.5 Route Control

We next discuss a few examples of how TaaS can be used to setup end-to-end paths with the appropriate properties desired by the application, e.g., improved fault-tolerance, provisioned service, and security.

Fault-tolerant routes: Resilience is obtained through pre-configured backup paths, established by the endpoint and used in the case of failures. An endpoint can establish one or more TaaS paths to the destination and use them in conjunction with the direct Internet path. Endpoints can use the query interface provided by TaaS ISPs to choose efficient alternate TaaS routes. For instance, it can contact multiple tier-1 ISPs providing TaaS service, query them to compute the end-to-end performance of TaaS paths traversing the ISPs, and establish TaaS circuits through those ISPs that provide good performance. Note that the endpoint can also use compact Internet maps (such as *iPlane Nano* [20]) to predict which ISPs are likely to provide good routes and thus minimize the number of ISPs they have to query for performance data.

The transport layer requires a small change at the endpoint to ensure that the switch from one path to the other is transparent to the endhost application. This can be done using systems such as Serval [25]. The endpoint monitors the communication flow and fails over to the backup path in the event of disruptions or degraded performance.

Securing routes: An endpoint can setup TaaS circuits through each intermediate ISP in an end-to-end path to ensure that its packets traverse only trusted ISPs. If some of the intermediate ISPs don't provide TaaS support, endpoints have to resort to normal Internet routing between the TaaS ISPs. This means that those hops are vulnerable to BGP effects such as prefix hijacking and rerouting of packets through untrusted ISPs. However, if a TaaS provider is also a provider of the non-participating ISP (e.g., a tier 1 ISP), then it is unlikely that those effects will be problematic. To limit the scope of prefix hijacking, most ISPs in practice are configured to filter competing advertisements for addresses originating in their direct customers/providers. For example, if a small ISP advertises it is UUNet, other ISPs can be configured to ignore it. If so, even if a route is announced by multiple peers, the correct TaaS route will continue to be used. It is worth noting that if all we need is alternating compliant ISPs, the average number of TaaS hops we would need for an end to end path in today's Internet is very small, typically one or two. An endpoint can also constrain that all communications sent to it should be through TaaS tunnels by providing other endpoints with a TaaS address as opposed to its actual IP address. This provides the endpoint with a simple DoS protection mechanisms, as the attack traffic can then be filtered out at a large TaaS ISP.

3.6 Business issues

An ISP has an incentive to ensure correct forwarding of TaaS traffic across its network, because it is receiving revenue in addition to the price it is receiving for carrying the packet from its immediate neighbor. Also, since end-

points have the ability to switch over to pre-configured backup paths, it is in the ISP's interest to perform local fault recovery quickly if it wants to retain the traffic from the TaaS customers. Further, since TaaS would allow ISPs to attract traffic that they normally wouldn't receive, there is an incentive for ISPs to implement TaaS even when other ISPs don't.

An ISP might intentionally disrupt traffic to a TaaS provider, e.g., if it sees a packet for the special address, it might drop it. On the other hand, the ISP is receiving revenue for the packet as a normal Internet service, so it would need to do traffic inspection and violate network neutrality to do so. If this were a problem, the packets could be encrypted when traversing non-cooperative ISPs, so that they appear to be normal SSL traffic. Further, note that such a disruption would cause the endpoint to fail over to an alternate path that traverses a different set of ISPs, thus providing a disincentive for the filtering ISP that stands to lose revenue due to its actions.

Although TaaS will allow enterprises to contract for exactly the amount of route control, resilience, and DoS protection that they need, ISPs may also find it useful to leverage TaaS services on behalf of their customers. That is, a customer-facing ISP would arrange tunnels to important data services, and this would be (nearly) transparent to the ISP's customers, except that they would find their Internet service through the ISP to be highly reliable. This aggregation will be particularly valuable for thin devices that lack the ability to monitor routes and perform route control on their own behalf.

4 Implementation

In this section, we describe two implementations of TaaS, as well as two TaaS deployments. The first deployment is on a local cluster to measure overheads incurred to ISPs due to TaaS' additional routing requirements. The second deployment is on several geographically distributed nodes on the Internet and serves to demonstrate that our system is practical and can be used on the Internet today.

4.1 GRE Implementation

Our first implementation of TaaS is based on Generic Routing Encapsulation (GRE) [8] tunnels, which we craft to resemble TaaS packets. We use the key field extension [7] to GRE, which we set to the TaaS authenticator value. The GRE protocol type stands in for the TaaS protocol type in the envelope IP header. We use the stock Linux kernel GRE implementation, which we setup such that generated GRE packets will duplicate the source IP address and all other IP header fields among the envelope and internal IP headers.

We exclude all other (optional) GRE fields, such that the only additional overhead from using GRE, compared

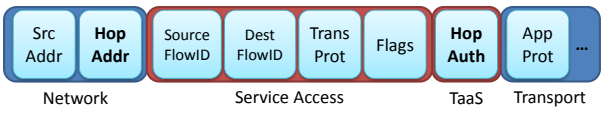

Figure 7: Relevant fields of a Serval packet with TaaS extension (in bold).

to using the TaaS packet format, comes from the mandatory 32-bit GRE header, which appears in front of the TaaS authenticator (GRE key field). This header is ignored by our router implementation.

To route packets, we have implemented a module for the Click [15] modular router version 2.1, which is running as a Linux kernel module. The routing module reads packets directly from the network interface and classifies them based on GRE header. If the packet has a GRE header, the module tries to resolve the authenticator, taken from the GRE key field, in its local TaaS forwarding table and, if found, replaces the IP envelope destination address and GRE key with the next-hop address and authenticator, respectively. If the next-hop authenticator is zero, the IP envelope and GRE headers are removed instead. In either case, the packet is fed to the host operating system routing mechanism, where its further fate is determined. Finally, if the TaaS forwarding table lookup fails, the packet is dropped.

4.2 Serval Implementation

We have also integrated TaaS support into the Serval [25] protocol stack to demonstrate how robustness is achieved using this solution. To extend Serval, we have added a new packet header extension, which contains the TaaS authenticator. This extension is included on any data packet. If the source endpoint's service access table contains a forward rule with a TaaS authenticator annotation, this header extension will be generated with the corresponding authenticator and all packets forwarded to the specified next-hop service router. The service routers detect the TaaS extension and match it in a special TaaS authenticator table to the next-hop IP address. Each packet's destination IP address is rewritten according to this table. Figure 7 shows the layout of a Serval packet with the TaaS extension.

4.3 Internet Atlas

To provide the Internet atlas service, we use a combination of the iPlane [22] database, which we augment with information about hypothetical TaaS providers.

iPlane is available as an Internet XMLRPC and SunRPC service that can be queried dynamically for metrics, such as reachability, latency and throughput performance, between any given two IP addresses. It is kept up-to-date with live traceroute information from Internet vantage points. As such, it can be used to determine

```
require 'iplane'

egress = ARGV[0]
prefixes = Array.new
(1..ARGV.length-1).each { |i|
    prefixes.push(ARGV[i])
}

iplane = IPlane.new
prefixes.each{ |p|
    iplane.addPath(egress, p)
}
responses = iplane.queryPendingPaths
responses.each{ |r|
    if (r.latency < 300) # 300ms
        puts(r.path.join(" "))
    end
}
```

Figure 8: A Ruby program that returns all TaaS paths with latency below 300 ms from a given egress PoP to a number of given IP addresses. Hops on a path are separated by spaces, paths are separated by newlines.

the performance between any two PoPs, as well as the performance to any Internet prefix from an egress PoP. This is especially useful when choosing among multiple TaaS-offering ISPs.

To reduce the amount of data transfered when multiple paths with a certain characteristic are requested, iPlane's SunRPC interface expects a Ruby program on its input and provides the output of that program as its result. The iPlane-specific objects and their methods are described in [21]. Figure 8 demonstrates a program that returns all paths with a latency below 300ms from a given egress PoP to a number of IP prefixes, given as a list of IP addresses living within each prefix, respectively.

4.4 Cluster Deployment

We have deployed the Click and Serval implementations of TaaS on a 6-node cluster. All cluster systems run Linux 3.2.0 on Intel Xeon E5-2430 processors, clocked at 2.2 GHz, with 15 Mbytes total cache, 4 Gbytes memory, and Intel X520 dual-port 10 Gigabit Ethernet adapters, connected to a 10 Gigabit Ethernet switch. Figure 9 shows this setup.

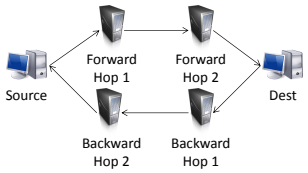

Figure 9: TaaS 6-node cluster deployment.

In the cluster, one node acts as the source endpoint of a route and another one as the target. The other nodes are used as TaaS routers. The deployment is symmetrical: Both forward and reverse TaaS paths are established between source and target, over distinct nodes in the cluster. We can construct up to 2 TaaS hops in this symmetrical fashion. We will use this deployment to measure TaaS overheads in Section 5.1.

4.5 TaaS Internet Backplane

We have deployed TaaS software routers at two locations[1], in Europe and the USA. Using the Serval implementation, we configure **Europe** to forward incoming TaaS traffic to **USA**. **USA** is configured to decapsulate incoming TaaS packets from **Europe** before forwarding to its final destination. The setup is symmetrical. On the reverse path, we configure **USA** to automatically add a TaaS header with a preconfigured authenticator to incoming traffic from, e.g., the New York Times, and forward to **Europe**, which in turn is configured to forward TaaS traffic from **USA** to **China**, where it is decapsulated. Figure 10 shows this setup.

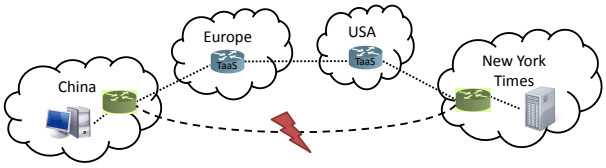

Figure 10: TaaS Internet deployment to the New York Times. Dotted lines show the configured TaaS path. The dashed line shows the firewalled BGP path to the New York Times.

Because we did not have access to ISP infrastructure, we deployed the routers at datacenter locations within the ISP's autonomous system (AS). Unfortunately, this prevents us from forwarding packets by rewriting only their destination address in most cases. The upstream ISPs filter packets with source IP addresses that do not originate within their allocated block. To work around this, we rewrite the source IP address to the router's IP address. This has the implication that BGP-based routes of the forwarded packets might end up being different if intermediate ISPs decide to route specially based on source IP address. We expect this discrepancy to go away if TaaS was deployed as part of an ISP's infrastructure.

To test our setup, we initiated a browsing session from a Planetlab node in China to the New York Times website[2]. When we send the request via the regular BGP route the access was blocked. When configuring the Planetlab node to generate TaaS traffic instead and for-

[1] Names not given for double-blind reviewing.
[2] http://www.nytimes.com/

ward to **Europe**, the request went through fine. Reverse traffic was sent TaaS-encapsulated back to us.

This demonstrates that TaaS can be deployed to change actual routes on the Internet to route around blocked links.

5 Evaluation

We evaluate TaaS both via simulation—to estimate effects on the large-scale Internet topology—and by measurement of an actual implementation deployed on a local cluster of machines, to gauge the overhead of TaaS on Internet traffic. Specifically, this section seeks to answer the following questions:

- How are throughput and latency of Internet traffic affected when a TaaS path (of various lengths) is used?
- How resilient are various TaaS deployments to Internet link failures?
- Can we achieve more reliable performance using TaaS?
- How effectively do various TaaS deployments prevent IP prefix-hijack attacks?
- Can we use TaaS to route around ISPs that behave in a Byzantine way?

5.1 Performance Overhead

TaaS should impose only minor overhead to latency and throughput of traffic when compared to standard routing on the Internet. We evaluate the latency and throughput overheads of the cluster-deployed version. We determine the latency along a path by measuring the average round-trip time (RTT) of 100 individual ICMP echo requests sent from source to target endpoint. Serval does not support ICMP. Hence, latency measurements on the Serval stack were carried out by sending 100 individual 64 byte UDP packets to an echo server, which sends them back unmodified. We measure the average throughput over 5 TCP transfers of a data stream over 10 seconds each, using the `iperf`[3] bandwidth measurement tool.

In the first iteration of our throughput measurement, we noticed that throughput fell sharply, from 9.1 to 2.7 Gbits/s, when encapsulating packets in GRE. This was due to TCP segmentation offload to the Ethernet network interface card. With the GRE headers in front of the TCP headers, hardware offloading is not possible and the operating system has to perform the segmentation, at a significant performance hit. Thus, to perform our measurements, we configured each network interface's MTU to the maximum supported by our switch (9198 bytes) instead of the default 1500 bytes. This eliminates the overhead, as the operating system is able to create larger TCP segments. Hardware routers typically employed on ISP infrastructure do not exert this problem. Also, even without large MTUs, the throughput is still good enough for most of our target, small-bandwidth applications.

[3] `http://iperf.sourceforge.net`

	Ping RTT [μs]	**Thruput [Gbits/s]**
Linux	44/96/107	9.05/9.36/9.68
GRE	96/105/131	7.93/9.03/9.87
Click	182/189/266	9.35/9.52/9.74
1 TaaS hop	265/272/289	9.37/9.55/9.85
2 TaaS hops	454/463/485	8.19/8.49/8.72
Serval	73/81.23/154	0.62/1.19/1.71
1 TaaS hop	113/131.96/290	0.89/0.97/1.04
2 TaaS hops	158/191.38/444	0.90/0.96/1.03

Table 2: TaaS overhead of different path lengths to packet latency and TCP throughput vs. Click and Serval. Numbers for GRE and Linux are also given. Min/avg/max are shown.

Table 2 shows the measurement results of different lengths of TaaS routes compared to Linux, the GRE protocol, the Click software router, and Serval when TaaS is not active. The Linux measurements measure direct bandwidth and latency between two endpoints, without going through any intermediate hops. The GRE measurement measures the overhead of using the GRE protocol on the same path. The Click measurement uses 1 Click hop, without any special TaaS processing, to forward the GRE packets to the target endpoint. The experiments using 1 TaaS hop do the same, but with the extra TaaS processing to determine the packets' fate. Finally, the 2 TaaS hop experiments involve one extra TaaS node in each direction that forwards packets to the next TaaS hop without decapsulating them, as shown in Figure 9.

In terms of latency, TaaS adds an overhead of 44% over the baseline Click software router implementation. A 2-hop TaaS path has an overhead of 76% over the latency of a 1-hop path. Throughput is not affected by adding TaaS to a 1-hop path. However, adding another TaaS hop impacts throughput by 10%. This might be due to our switch not being able to handle the bandwidth requirement.

The Serval measurements use the Serval protocol stack to forward packets instead of the GRE protocol and the Click router. We use the default MTU of 1500 bytes for the Serval experiments, hence the lower throughput rates. Our measurements are in-line with those done in the Serval paper [25] and show that average throughput drops by 18% on a 1-hop TaaS path, which is due to the additional packet processing and routing table lookup. Latency overheads in Serval are better than, but generally comparable to that of the Click implementation. This is due to the Serval implementation, which is tailored to the Linux packet processing code. Click instead compiles packet processing code from high-level processing

modules.

5.2 Simulation Dataset and Methodology

Next, we explore the reliability and performance properties of TaaS deployed at Internet-scale. For this purpose we simulate routing events on the Internet topology, based on measurements collected by iPlane.[4] The iPlane network atlas is built using traceroutes from over 200 PlanetLab sites to more than 140K prefixes, i.e., almost every routable prefix on the Internet. The iPlane dataset also provides IP-to-AS mapping, IP-to-PoP mapping (where each PoP is a set of routers from a single AS co-located at a given geographic location), and the RTTs of inter-PoP links. The resulting topology is a superset of that provided by the CAIDA AS-level graph [1] or the RouteViews BGP tables [2]. We use the most recent iPlane snapshot collected for February 2013. This has a total of 27,075 ASes and 106,621 unique AS-AS links. At the PoP-level, it has 183,131 PoPs and 1,540,466 PoP-level links.

5.3 Resilience to Link Failures

We start by evaluating the resilience provided by TaaS in case of link failures. To simulate all failures, we select each provider link L of each multi-homed stub AS A, successively. A multi-homed stub AS is an AS with more than one provider and no customers; our topology includes 16110 such ASes. We focus on these because the stub AS has a valid physical route to the rest of the Internet even if the provider link L fails. We arrive at a total number of 42605 failures, affecting 475 sources per victim AS. At the beginning of the experiment, we select a small number of tier-1 ASes as our TaaS -supporting ASes, ordered by the size of their customer tree [30]. For example, if we want k tier1s as TaaS ASes, we will select k tier1 ASes with the largest customer tree size. For each failure trial, we fail the link L, and see what fraction of the sources still have connectivity to the stub AS A through any of the TaaS path segments. Figure 11 is a CCDF plot of these failures showing the results of our experiment. Each curve represents a particular TaaS deployment scenario. The x-axis measures the disconnectivity seen in topology as the result of the failure, i.e., the fraction of sources unreachable from the victim AS. For each such fraction f on the x-axis we have the corresponding fraction of failures that resulted in at most f disconnectivity. We compare four TaaS deployments of various sizes with simple BGP routing.

All four deployments of TaaS provide significantly better reliability against failures than BGP. For example, with a TaaS deployment of size 1, more than 60% of failures result in only 20% or less disconnections, as opposed to just above 20% of failures using only BGP routing. This number goes up to nearly 90% when 8 TaaS ASes are deployed. In fact, more than 85% of failures in the 8-AS deployment case result in less than 1% disconnectivity.

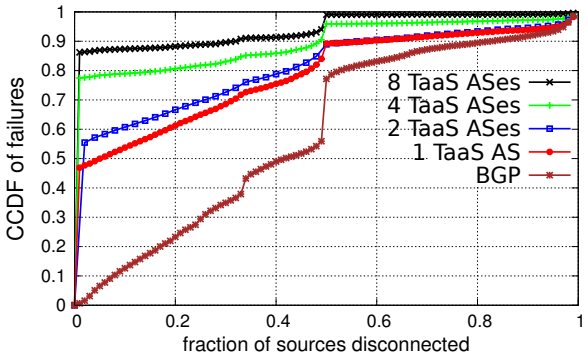

Figure 11: CCDF showing the fraction of failures resulting in a certain amount of disconnectivity, as measured by the fraction of sources unable to reach the target as a result of the failure.

As can be seen from the plot, increasing the number of tier-1s supporting TaaS provides additional resilience, but the gains are diminishing. The reliability provided by 8 TaaS ASes is not much better than that provided by 4 TaaS ASes. This is intuitive since most tier-1s have a significant global presence as well as peerings with other tier-1s. Therefore deployment on 2 or 3 tier-1s likely provides as rich a topology as that on 7 or 8 tier-1s. In fact, deployment on just one tier-1 already provides significant gains in reliability compared to BGP.

5.4 Resilience to Byzantine Failures

To reduce the risk of encountering any AS that behaves in a Byzantine manner we ask if we can build a TaaS path with complete or near-complete AS-level redundancy. We define a path q to be completely redundant to path p, if the set of AS-hops in q is disjoint from that of p, except for the source and destination ASes. The metric that we are interested in evaluating is the number of common hops between the original path p and the best TaaS path q as a fraction of p's length. Figure 12 plots this distribution over all paths in our dataset (around 5 million). The two curves represent the CDFs for TaaS deployments on 2 tier-1 and 4 tier-1 ASes respectively.

As can be seen in Figure 12, alternative TaaS paths ensure a high degree of redundancy between the old and new paths. Almost 40% of the paths for the 2-AS deployment, and 50% for the 4-AS deployment provide completely disjoint paths, respectively (disregarding the source and the destination). Almost 80% of the paths in both cases have less than half of the ASes from the old path still present in the new TaaS path. The significant redundancy with a small TaaS deployment can again be

[4] http://iplane.cs.washington.edu/data/data.html

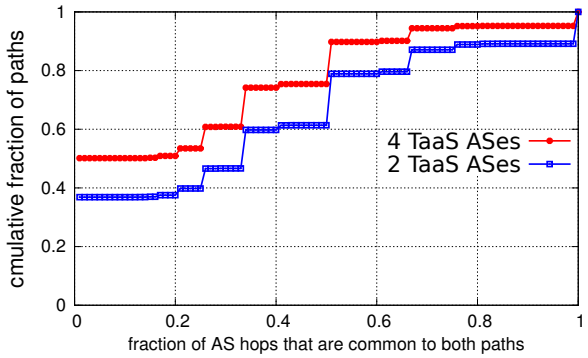

Figure 12: A TaaS deployment provides significant path redundancy, with almost half of the paths having completely disjoint TaaS paths at the AS-level.

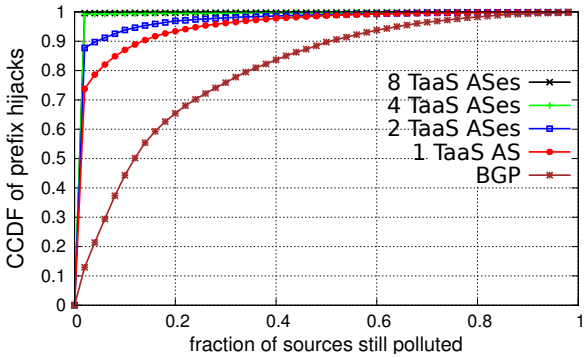

Figure 13: CCDF showing the fraction of prefix-hijack attacks resulting in a certain amount of pollution, as measured by the fraction of sources unable to reach the target as a result of the attack.

explained by the rich peering provided by a tier-1 ISP.

5.5 Protection against Prefix-hijacking

IP prefix hijacking is a serious challenge to the reliability and security of the Internet. Since the Internet lacks any authoritative information on the ownership of prefixes, IP prefix-hijacking is extremely hard to eliminate. TaaS can be used to mitigate the effects of prefix hijacking. We imagine a scenario where the prefix-hijacking has already been detected. Specifically, given a standard TaaS deployment on a small number of tier-1s, we ask what fraction of sources still remain polluted (i.e., paths going through any of the polluted ASes) for a particular prefix-hijacking attack.

To simulate prefix hijacks, we select a victim AS and an attacker AS, both stubs. We use all stubs in our topology as victims and average the results over a random selection of 20 attackers for each victim. This gives us a total of 16160 victim ASes. For each attack, we determine the set of polluted ASes as follows: an AS is polluted if its BGP path to the attacker is shorter than its path to the victim [36]. For each attack and a given TaaS deployment we see what fraction of the sources remain unpolluted, i.e., able to send traffic to the victim through any of the TaaS path segments. Figure 13 shows the CCDF of the hijack attacks. The x-axis measures the level of pollution, i.e., the fraction of sources remaining polluted as a result of the attack. For each such fraction p on the x-axis we have the corresponding fraction of attacks that resulted in at most f pollution. Again we compare four TaaS deployments of various sizes with simple BGP routing.

Again TaaS provides significant advantages to combat prefix-hijack attacks, even more so than failures as evaluated earlier in Section 5.3. All four deployments of TaaS provide significant protection against prefix-hijack attacks. For example, for a maximum number of polluted sources of 5%, TaaS deployments of size 1, 2, 4 and 8 cover 75%, 88%, 100% and 100% of the attacks, respectively. We need a TaaS deployment only on 4 tier-1s to eliminate most of the unreachability caused by prefix-hijack attacks.

5.6 Reliable Performance

We now evaluate the performance gains achievable from a TaaS deployment. Assuming that the destination AS is a TaaS client of all k TaaS -supporting ASes, we ask the question: *what is the fraction of sources that have an alternative TaaS path with an end-to-end latency that is at least X% lower than the original path?* For this purpose we use the PoP-level link latencies provided by *iPlane*. Figure 14 is the distribution of fractional improvement in the end-to-end latencies for a total of 1143652 PoP-level paths.

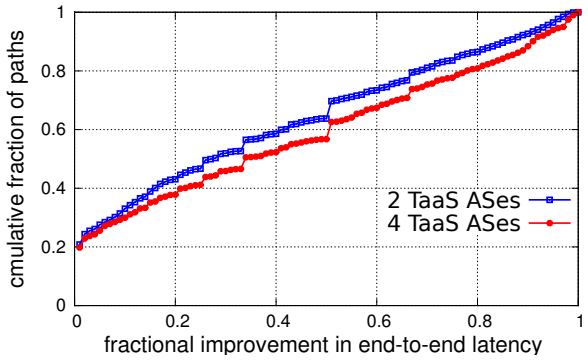

Figure 14: A TaaS deployment provides performance latency gains for more than 80% of the paths.

As can be seen from Figure 14, more than 80% of the source-destination pairs experience an improvement in the end-to-end latency while using a TaaS path. The distribution of improved latencies is pretty even for both deployment scenarios, with the gains slightly higher for a

deployment over four tier-1ASes.

6 Related Work

TaaS draws inspiration and builds upon a number of related proposals and systems targeted at other goals, including OpenFlow [23], ATM networks, MPLS, i3 [28], pathlet routing [9], denial of service defenses [5,34], and Telex [31]. We will discuss the most important in this section.

Several proposals provide endpoints with greater control over Internet routing. In Icing [24], every entity on a communication path has to provide consent before packets can be transmitted over the path. Yang el al. [33] propose a solution that allows both senders and receivers to choose AS-level routes to the Internet core, with the end-to-end path the concatenation of the two segments. Routing as a Service [18] recognized the tussle between users who want control over end-to-end paths and ISPs who desire control over how their infrastructure is used. To resolve this tussle, the authors introduce a separate entity that contracts with both ASes and customers and establishes paths that are acceptable to all entities. These proposals are clean-slate redesigns of the routing protocol and provide limited incentives and opportunities for incremental deployment.

Pathlet routing [9] is a related proposal that allows for endpoints to perform *source routing* over a virtual topology. Endpoints can select any path within the topology and can take into account the needs of the application in doing so. i3 introduces a level of indirection in network communications, decoupling the act of sending from the act of receiving; as a consequence, it can efficiently support a wide variety of communication services (e.g., mobility, service composition, and multicast) [28]. TaaS shares the flexibility goals of these proposals, but strives to achieve them in the context of today's Internet without re-architecting it from the ground up.

MIRO [32] is a multi-path interdomain routing protocol that allows ISPs to negotiate alternate paths as needed. MIRO is designed to be an incrementally deployable extension to BGP. RBGP [16] proposes to use pre-computed backup paths to provide reliable delivery during periods where the network is adapting to failures. TaaS has similar goals, but obtains additional deployability benefits since it doesn't require changes to the interdomain routing protocol. A single ISP can unilaterally provide TaaS service and obtain revenues directly from end users who would benefit from the service.

There are two widely used solutions to improving Internet reliability that help a bit, but not enough: Multihoming and overlays. With multihoming, a customer arranges for multiple Internet providers, in case one fails. However, this does not provide a guarantee – if the paths through both provider autonomous systems (ASes) traverse a specific problem AS, then the endpoint will experience an outage, despite multihoming. Using a Detour overlay can avoid these problems, but measurements have shown that because of the unreliability of the underlying Internet, at best Detour routes improve reliability by a factor of two. Detour routes also generally do not protect end-to-end communication against denial-of-service attacks or byzantine behavior by some ISPs.

Because of the importance of Internet reliability and security, large ISPs have widely deployed MPLS to provide more reliable and more predictable routes within their own networks. IP fast reroute proposals have been developed by the IETF and others to improve recovery from intradomain faults. Within a data center, network topologies and routing protocols are increasingly being designed to be resilient to network device failures. We build on this work to provide a deployable end-to-end solution.

Our approach is complementary to clean-slate Internet re-designs and builds upon some of their ideas. For example, SCION [4, 35] introduces the notion of trust domains and endpoint-selected path preferences. TaaS builds upon both ideas to provide an incrementally-deployable routing architecture for mission-critical traffic on the existing Internet.

7 Conclusion

The Internet is increasingly being used for critical services, such as home health monitoring, management of the electrical grid, 911 IP service, and disaster response. Yet, there is increasing evidence that the current Internet is unable to meet the availability demands of these emerging and future uses. In this paper, we examine what are the minimal changes needed for the Internet to support such mission critical data transmissions.

Our proposal is to provide a mechanism that would enable end users, enterprises, and governments to stitch together reliable end to end paths by leveraging highly reliable intradomain path segments. At the core is a protocol called *Transit as a Service*, which allows users to provision a path across a remote ISP. We outline the design of TaaS, examine how it can be used to enhance the robustness and security of end-to-end paths, and describe an implementation of its key components. Our evaluations show that TaaS imposes only minor overheads and can provide significant resiliency benefits even when deployed by a limited number of ISPs.

References

[1] `http://www.caida.org/data/active/asrelationships/.`

[2] `http://www.routeviews.org.RouteViews.`

[3] A. Agapi, K. Birman, R. Broberg, C. Cotton, T. Kielmann, M. Millnert, R. Payne, R. Surton, and R. van Renesse. Routers for the Cloud: Can the Internet Achieve 5-Nines Availability? *Internet Computing, IEEE*, 15(5), 2011.

[4] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (aip). In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008.

[5] C. Dixon, T. Anderson, and A. Krishnamurthy. Phalanx: Withstanding multimillion-node botnets. In *NSDI*, 2008.

[6] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. Routebricks: exploiting parallelism to scale software routers. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, 2009.

[7] G. Dommety. RFC 2890: Key and sequence number extensions to GRE, Sept. 2000.

[8] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. RFC 2784: Generic routing encapsulation (GRE), Mar. 2000.

[9] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. In *SIGCOMM*, 2009.

[10] S. Han, K. Jang, K. Park, and S. Moon. Packetshader: a gpu-accelerated software router. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, 2010.

[11] J. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkataramani. Consensus routing: the Internet as a distributed system. In *Proc. of NSDI*, 2008.

[12] E. Katz-Bassett, H. Madhyastha, J. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying blackholes in the Internet with Hubble. In *Proc. of Networked Systems Design and Implementation*, 2008.

[13] E. Katz-Bassett, C. Scott, D. R. Choffnes, I. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy. Lifeguard: practical repair of persistent route failures. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012.

[14] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 2000.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, Aug. 2000.

[16] N. Kushman, S. Kandula, and D. Katabi. R-BGP: Staying Connected in a Connected World. In *NSDI*, 2007.

[17] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: a Prefix Hijack Alert System. In *USENIX Security Symposium*, August 2006.

[18] K. Lakshminarayanan, I. Stoica, S. Shenker, and J. Rexford. Routing as a service. Technical Report UCB/EECS-2006-19, UC Berkeley, 2006.

[19] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: secure and adoptable source authentication. In *NSDI*, 2008.

[20] H. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *Proc. of NSDI*, 2009.

[21] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: Measurements and query interface, June 2007. `http://iplane.cs.washington.edu/iplane_interface.pdf`.

[22] 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. of Operatings System Design and Implementation*, 2006.

[23] 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), 2008.

[24] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra. Verifying and enforcing network paths with icing. In *CoNEXT*, 2011.

[25] E. Nordstrm, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, and M. J. Freedman. Serval: An End-Host Stack for Service-Centric Networking. In *Proc. of NSDI*, 2012.

[26] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. In *SIGCOMM*, 2007.

[27] M. Shand and S. Bryant. IP Fast Reroute Framework. IETF Draft, 2007.

[28] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *SIGCOMM*, 2002.

[29] P. Trimintzios, C. Hall, R. Clayton, R. Anderson, and E. Ouzounis. Resilience of the Internet Interconnection Ecosystem. http://www.enisa.europa.eu/.

[30] UCLA Internet topology collection. `http://irl.cs.ucla.edu/topology/`.

[31] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the Network Infrastructure. In *Proc. of the USENIX Security Symposium*, 2011.

[32] W. Xu and J. Rexford. MIRO: multi-path interdomain routing. In *Proc. of SIGCOMM*, 2006.

[33] X. Yang, D. Clark, and A. W. Berger. NIRA: A New Inter-Domain Routing Architecture. *IEEE/ACM Transactions on Networking*, 2007.

[34] X. Yang, D. Wetherall, and T. Anderson. TVA: A DoS-limiting Network Architecture. *IEEE/ACM Transactions on Networking*, 2008.

[35] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, 2011.

[36] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: detecting IP prefix hijacking on my own. *IEEE/ACM Trans. Netw.*, 18(6):1815–1828, Dec. 2010.