One Tunnel is (Often) Enough

Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, Arvind Krishnamurthy
University of Washington
{simpeter, ujaved, qiao, dwoos, tom, arvind}@cs.washington.edu

ABSTRACT

A longstanding problem with the Internet is that it is vulnerable to outages, black holes, hijacking and denial of service. Although architectural solutions have been proposed to address many of these issues, they have had difficulty being adopted due to the need for widespread adoption before most users would see any benefit. This is especially relevant as the Internet is increasingly used for applications where correct and continuous operation is essential.

In this paper, we study whether a simple, easy to implement model is sufficient for addressing the aforementioned Internet vulnerabilities. Our model, called ARROW (Advertised Reliable Routing Over Waypoints), is designed to allow users to configure reliable and secure end to end paths through participating providers. With ARROW, a highly reliable ISP offers tunneled transit through its network, along with packet transformation at the ingress, 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, robustness, and security of mission critical transmissions. Unlike efforts to redesign the Internet from scratch, we show that ARROW can address a set of well-known Internet vulnerabilities, for most users, with the adoption of only a single transit ISP. To demonstrate ARROW, we have added it to a small-scale wide-area ISP we control. We evaluate its performance and failure recovery properties in both simulation and live settings.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet-switching networks; C.2.2 [**Network Protocols**]: Routing protocols; C.2.5 [**Local and Wide-Area Networks**]: Internet

Keywords

Internet; Source routing; Overlay networks; BGP; Reliability

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. Similarly, an economically 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.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2626318.

However, the present Internet is not up to the task. 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 up to days [14]. Often these outages are partial or asymmetric, indicating that a viable path exists but the protocols and configurations are unable to find it. Other triggers of 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. Recently, intra-US traffic was intentionally re-routed through Belarus [6], and earlier, traffic from the US Department of Defense was routed through China [35]. The Internet lacks any protocol mechanism to prevent this type of event from recurring.

Because of its scale, the Internet is of necessity multi-provider, and routes often involve multiple organizations. While a number of research projects have proposed tools to diagnose problems (e.g., [14, 16]), and fixes to specific issues, such as prefix hijacking [5, 17,21,23], route convergence [13], and denial-of-service [7,26,38], 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 the Internet to reliably deliver packets with reasonable latency? For example, Secure BGP addresses some of the vulnerabilities surrounding spoofed routes, but it doesn't address denial of service or route convergence. The commercial case for deploying a partial defense is weak.

We note that reliability is not equally important for all traffic. Our goal is to design a system that will provide highly available communication for selected customers as long as there is a policy compliant physical path, without diverting the traffic to non-trustworthy ISPs. This property should hold despite node and link failures, software upgrades, byzantine behavior by neighboring networks, and denial-of-service attacks by third parties.

In this paper, we propose, implement, and evaluate a system called *Advertised Reliable Routing Over Waypoints* (ARROW) 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 provision a tunnel across a remote ISP; packets entering the tunnel are authenticated by the ISP, delivered to a specific exit PoP, and slightly re-written, e.g., to modify the destination address. In providing ARROW, an ISP promises only what it can guarantee itself: a high quality tunnel across its own network. The customer is responsible for stitching together ARROW into an end-to-end solution. Like local transit, ARROW 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 cloud computing service.

ARROW shares similarities with Detour routing [29], the Internet Indirection Infrastructure [31], Nira [37], pathlet routing [10], and Platypus [27], among others. Unlike these earlier systems, the ARROW model takes into account resource exhaustion and byzantine attacks in addition to routing anomalies; experience has shown that these attacks are common. More importantly, a number of technology trends have converged to merit a fresh look at Detour-like systems for addressing Internet reliability:

- Large-scale ISPs have deployed sophisticated traffic and network management, making their own networks much more reliable. How can we best leverage this for end-to-end resilience?
- The Internet has become flatter and is more densely interconnected [20]. This has shortened BGP paths to the point that a single, well-placed tunnel is often sufficient to avoid a wide range of problems (as we show in §5).
- $5K PCs can cost-effectively process packets at Internet speeds (40Gbps), allowing for easy deployment of ARROW and the applications that build upon it.

In this paper, we present the design of ARROW, including how its main requirements, incremental deployability, high availability, and robustness, are achieved (§3). We present the ARROW API that can be used by ISPs and end-users to find, reserve, and establish paths on the Internet (§3.2). We present our implementation of ARROW and describe its deployment on a faux ISP that we control, including several applications we have built on top of this deployment (§4). Finally, we evaluate ARROW both in simulation and experimentally (§5). Our evaluation shows ARROW's resilience against transient routing problems, IP prefix-hijacking, inter-AS link failures, path performance problems, and ISP failures, with little overhead to Internet routing performance.

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 sending measurements to a data center, where they are analyzed in real-time. Anomalies are forwarded to alert human experts who can ensure that no medical problem has occurred. To support such applications 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; BGP will inform the upstream users only after the fact. 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 [32] enumerates other known security vulnerabilities of the Internet. A few examples include disruption of service by resource exhaustion attacks against network links and end hosts, prefix hijacks by malicious ISPs, and byzantine errors by neighboring ISPs (e.g., intentional disaggregation of addresses, causing router crashes).

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. Common 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 seems unlikely that 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. ARROW DESIGN

We would like to develop a simple system that can be used to provide highly available communication in addition to 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.

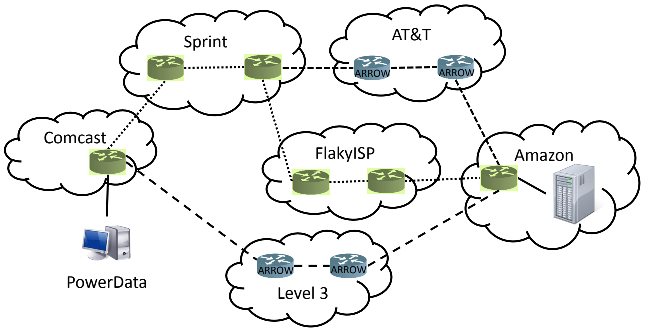

Figure 1: Three example ARROW paths from PowerData to Amazon: the dotted lines represent the BGP path. The two dashed lines are ARROW paths.

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 approach before describing the key components of our design.

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 ARROW 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) suffices to achieve the desired paths.

To ensure 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 ARROW 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 deployable by an ISP, with incremental incentives to that ISP. An ISP can provide ARROW even if none of its peer, customer or provider ISPs participate in the protocol. ARROW 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, ARROW is still useful to PowerData even if Sprint does not provide ARROW transit.

Second, ARROW 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. Redesigning services to work with ARROW requires minimal programmer effort, the costs of which should be outweighed by the benefits for mission-critical services. 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 ARROW.

In the rest of this section, we present the ARROW 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

An ISP offering ARROW 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 ARROW tunnels, and to request the tunnel) can be carried over the existing Internet, or in turn use ARROW 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 ARROW service. 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 ARROW service, including between which of its peers to provide the service, and whether to attach Service Level Agreements (SLAs) to it that might provide bandwidth and latency guarantees. Other value-added services might also be offered, such as DoS protection.

The portal should provide at least the following:

- All offered transit paths and services from a specific ingress ISP (and optionally, ingress link) to a specific egress ISP (and optionally, egress link), including backup ingress router IP addresses in case of failures,
- the pricing model of the offered transit paths and services, and
- the SLAs provided, such as bandwidth, latency, and maximum packet loss rate guarantees.

In addition, the portal needs to provide the ability to obtain and relinquish a path by returning or accepting a corresponding identification token, which we call an *authenticator*. The authenticator is used in actual ARROW traffic to prove that the endpoint originating the packet is authorized to use the transit path. The router must check the authenticator and drop the packet if it does not match. If the authenticator is compromised, the only penalty is that the customer of the service is charged for extra unrelated traffic traversing the pipe. Techniques to safeguard the authenticator against eavesdropping, such as encoding it with the hash of the checksum of the packet, may be used. Attackers can then only replay entire packets, but they cannot use snooped authenticators for other packets.

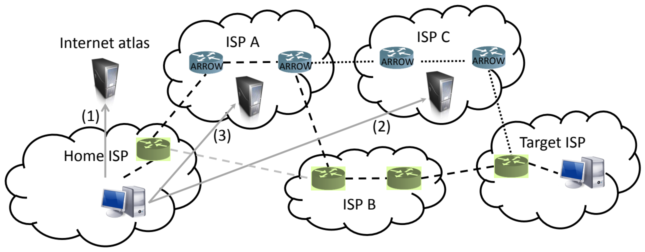

Figure 2: Example ARROW transit setup process. Blue routers in ISPs A and C are ARROW-compatible, gray routers in ISP B are not. Dashed lines show the path chosen in case (a). Dotted lines show the detour taken via another ARROW ISP in case (b). The BGP path in this example is Home-B-Target.

Finally, an RPC interface should be provided that allows clients to stitch together transit segments of multiple providers into an end-to-end path. The call may look like this:

```
chain_path(auth, nextHopAddr, nextHopAuth)
```

It causes the egress router of an existing ARROW tunnel segment (identified by its authenticator `auth`) to route tunnel traffic to the IP address `nextHopAddr` and set the ARROW authenticator of the next hop to `nextHopAuth`.

An end host contacts one or more of the ARROW ISPs on the route via this interface and requests provisioned paths through the individual networks. The ARROW 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. We note that links within an ISP might run out of excess capacity, but that only prevents future ARROW tunnels from being set up; existing agreements can stay in place. Market prices can then signal a need for more capacity.

From the ISP-provided lists of ingress and egress points, endpoints are able to compile an atlas, which they can use to determine a path to a destination. Any shortest path discovery algorithm can be used on the atlas to determine which ISPs to use to create an end-to-end circuit. We envision that, eventually, another Internet webservice maintains the atlas and provides a path query interface, returning paths according to any of a number of these algorithms. Most of the information we require ISPs to advertise to potential customers is already advertised to their direct peers, and much of it is already publically available.

Figure 2 shows an example of an Internet endpoint arranging an ARROW 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 ARROW and a circuit is created via this ISP. Other traffic is routed via BGP through a non-ARROW supporting **ISP B** to the final destination. The path in this case is **Home-A-B-Target**.

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

In both cases, the endpoint first optionally contacts an Internet atlas service to determine which ISPs to contract for ARROW service (1). Then, the home endpoint contacts the portals of the appropriate ARROW ISPs on the circuit to create ARROW SLAs. This typically has to happen in reverse order (2 and then 3), from destination endpoint to source endpoint, so that next-hop information can be given.

ARROW paths are unidirectional. The reverse path can be provisioned either by the peer, or by the originator of the traffic. This

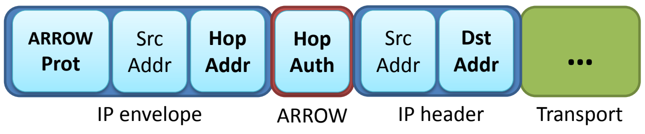

Figure 3: Relevant fields of an ARROW packet (in bold). ***Dst Addr*** **is the IP address of the final destination. Note that the source endpoint IP address and other IP header fields are duplicated by the envelope IP header.**

might depend upon the relationship of the peers. If the source party is a customer of a 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 ARROW, the source (or its proxy) encapsulates every IP data packet in a separate IP envelope, as shown in Figure 3. The envelope also contains the authenticator in a special field (Hop Auth). The envelope's destination is set to the next ARROW hop's IP address (Hop Addr) and its next-level protocol field is set to a value identifying ARROW (ARROW Prot).

The source then sends the packet normally through its local network. If the customer has a chain of ARROW providers, then each modifies the packet with the address and authenticator of the next hop, according to its local ARROW forwarding table; the last ISP in the chain removes the IP header encapsulation before forwarding. This model requires each ARROW router to have an appropriate entry per ARROW tunnel in its forwarding table, contributing to the growth of forwarding tables. Modern forwarding and table storage techniques [28], however, make routing table scalability less of a concern.

We do require some level of hardware support in routers, but it is minimal and similar to the hardware already in place: in many ISPs, ingress 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 ARROW: the ingress router must be able to demux on the ARROW address (and if necessary the ARROW authenticator), route the packet using MPLS or other means, and then modify the header to insert the next hop address and authenticator. Alternatively, ISPs can operate high-speed software routers (e.g., RouteBricks [8], PacketShader [11]) at ingress/egress PoPs to perform the necessary tasks for ARROW traffic.

3.4 Failures and Performance Regressions

Failures. Most failures along an ARROW circuit can be handled by ISPs directly. Some failures require endpoints to cooperate, however. Specifically, in an ARROW circuit, failure can occur at several different levels:

- failure of a router internal to an ISP,
- failure of a router at the edge of an ISP, or
- failure of a whole ISP.

If an internal router fails, the ISP is responsible for detecting this failure and routing around it. ISPs today typically 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. 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 [30].

Since edge routers correspond to boundaries between ISPs, edge routers belonging to different ISPs need to cooperatively handle failover. Thus, we have designed a failover protocol which must be implemented by each participating ISP. Our implementation does not perform failure detection, since we believe that in general the endpoint will be able to do a better job of determining whether its traffic is actually arriving at the server. Therefore, failover is an endpoint-driven process in this case.

When the endpoint detects a failure along an ARROW path (i.e., it detects that its traffic is not successfully arriving at the destination), it sends a special probe packet along the same path. Each ISP's router is required to respond to **the previous hop** of the probe packet and also forward it normally along the path. This is similar to what happens when an IP packet has reached its end of life, except that the packet is also forwarded and the response goes to the previous hop instead of the source of the packet.

If the endpoint does not receive an acknowledgment from the first ISP, it concludes that the first ISP is where the failure exists and fails over to another edge router in the same ISP. If the first ISP does receive the endpoint's probe, it first acknowledges it so that the endpoint doesn't assume it has failed, then forwards the probe to the second ISP. Thus, the probe is initiated by the endpoint but is actually performed at each non-failing ISP node in the circuit.

For example, assume a circuit has been established between an endpoint E, ISP A, ISP B, and a server S. Each ISP has three routing nodes; we call them A_1, A_2, A_3 and B_1, B_2, B_3. When the circuit is first established, it goes from E to A_1 to B_1 to S. At some later time, B_1 fails. E detects that S isn't receiving its messages and initiates the failover process by sending a probe packet to A_1. A_1 receives and acknowledges the probe, then forwards it to B_1. A_1 does not receive an acknowledgment within a link-latency-determined timeout, and concludes (correctly) that B_1 has failed. A_1 then forwards the probe to B_2, which acknowledges it. In response, A_1 updates its local state so that packets on the circuit go to B_2 rather than B_1. Subsequent traffic is now routed through the new circuit and E detects that the failover has succeeded.

If the ISP as a whole fails, the endpoint will need to provision an alternate route. An endpoint can establish multiple ARROW paths to the destination simultaneously and use them in conjunction with the original path for fast failover.

Performance regressions. To help endpoints investigate circuit performance, ISPs should provide a ping service at egress PoPs to arbitrary IP addresses and ARROW routers need to be able to respond to ICMP requests. Endpoints then have the ability to independently gather all required information to analyze the performance of each circuit segment.

3.5 Security

The design presented thus far has a few security issues: ARROW paths and their endpoints might be DoS-attacked, the availability of trusted ARROW ISPs might be spotty and traffic has to traverse non-ARROW ISPs, and ARROW users themselves might be malicious. We now address these issues.

DoS attacks. ARROW paths can be set up to resemble swarms of packet forwarders [7] in order to increase path availability in the face of DoS attacks. Figure 4 demonstrates how this can be achieved: multiple ARROW segments are configured within one ARROW supporting ISP to mitigate the effects of DoS attacks to ARROW ingress points. Since ARROW 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

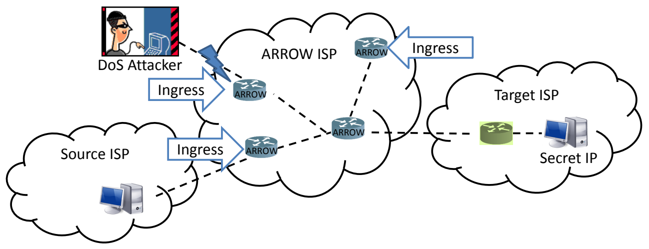

Figure 4: DoS attack prevention using ARROW.

IP address is kept secret, it does not matter whether the ARROW supporting ISP is the endpoint's local ISP or an arbitrary remote ISP. Otherwise, if the provider of the destination endpoint provides ARROW, the endpoint's ISP can drop all non-ARROW traffic and protect the endpoint in this way.

Securing spotty ARROW routes. To ensure that its packets traverse only trusted ISPs, an endpoint can set up ARROW circuits through each intermediate ISP in an end-to-end path. If some of the intermediate ISPs don't provide ARROW support, endpoints have to resort to normal Internet routing between the ARROW ISPs. This means that those hops are vulnerable to BGP effects such as prefix hijacking and rerouting of packets through untrusted ISPs. However, if the AS path lengths of the non-ARROW segments are small, then only those ASes that are within the local neighborhood of these segments would be able to mount an effective prefix hijacking attack. Additionally, in order to disrupt an ARROW route, the hijacking entity has to pollute the routing entry at a specific egress PoP as opposed to any arbitrary PoP inside the ARROW supporting ISP. Further, if an ARROW ISP is a provider of the non-participating ISP (e.g., the ARROW ISP is a tier 1 ISP), then also it is unlikely that prefix hijacking is effective—most ISPs in practice are configured to filter competing advertisements for addresses originating in their direct customers/providers, so the ARROW ISP would effectively filter out most forged announcements.

It is worth noting that if all we need is alternating compliant ISPs, the average number of ARROW 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 ARROW tunnels by providing other endpoints with an ARROW address as opposed to its actual IP address.

In any case, it is important to know how many BGP hops are between the ARROW hops to gauge the vulnerability of the connection to attacks and failures. To figure out the number of intermediate hops, sources can initiate a traceroute to originate from the last ARROW supporting hop of the source end of the path, directed at the first ARROW hop of the destination end of the path. This can be done by sending a traceroute via ARROW from the source endpoint to the first ARROW hop of the destination end through the existing partial circuit.

Misbehaving ARROW users. We have thus far assumed that ARROW clients are well-behaved. A malicious or faulty client could repeatedly send false failover notices, or set up an ARROW circuit with a loop. An ARROW-supporting ISP would need to detect these scenarios (this is feasible because clients require strong identities to buy ARROW service) and ignore or rate-limit failover traffic from misbehaving customers.

3.6 Business issues

An ISP has an incentive to ensure correct forwarding of ARROW 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 endpoints 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 ARROW customers. Further, since ARROW would allow ISPs to attract traffic that they normally wouldn't receive, there is an incentive for ISPs to implement ARROW even when other ISPs don't.

An ISP might intentionally disrupt traffic to an ARROW provider, e.g., if it sees a packet destined for an ARROW address, it might drop it. While a sufficient number of such misbehaving ISPs will cause a problem regardless of the technology used, ARROW would allow customers to route around them, and provide further incentive (in the form of more potential customers) for other ISPs to adopt ARROW. As another option, ARROW destined packets could be encrypted when traversing non-cooperative ISPs, so that they appear to be SSL traffic.

Although ARROW 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 ARROW 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.

4. IMPLEMENTATION

In this section, we describe our implementation of ARROW and its deployment. The deployment is a step toward building and running an ISP supporting ARROW, such that any Internet user will be able to try out ARROW for themselves. To this end we leverage the BGP Transit Portal [33] and PlanetLab VICCI clusters [34]. We have also built several services designed to enhance end-to-end performance and security on this deployment. These services include a DoS prevention system akin to the one described in §3.5, and a simple CDN.

4.1 Serval Implementation

We have integrated ARROW support into the Serval [25] protocol stack. Serval is a great fit for ARROW as it already supports client-side failover for connection failures and we can extend this support to failover among several ARROW paths. To extend Serval to support ARROW, we have added a new packet header extension (header extensions are a Serval feature), which contains the ARROW authenticator. This extension is included on every data packet. If the source endpoint's service access table contains a forward rule with an ARROW authenticator annotation, this header extension will be generated with the corresponding authenticator and all data packets forwarded to the specified next-hop service router. The service routers detect the ARROW extension and match it in a special ARROW authenticator table to the next-hop IP address, possibly with another ARROW annotation. Each packet's destination IP address and potentially ARROW authenticator is rewritten according to this table.

4.2 Internet Atlas

To provide the Internet atlas service, each ARROW ISP continuously performs measurements to generate and maintain a map of Internet paths from each of its PoPs to each routable prefix, with each path annotated with a rich set of attributes corresponding to the latency, available bandwidth, and loss rate of the path.

These collected measurements are made available through a *network atlas* service (with XMLRPC and SunRPC interfaces) so that they can be queried dynamically for metrics, such as reachability, latency and throughput performance, between an ARROW PoP and an arbitrary IP address. It is kept up-to-date with live active probe measurements, which are performed at regular intervals and also selectively reprobed based on passive observation of BGP feeds. As such, it can be used to determine 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 ARROW-offering ISPs.

To determine whether keeping the network atlas up-to-date is feasible, we have evaluated the average daily rate with which Internet paths change at the PoP-level. To do so, we count daily PoP-level changes in downstream paths from all tier-1 ASes to all prefixes within the entire year 2012 of historical iPlane traceroute measurements (cf. §5.4). We count in two ways: 1. also counting paths with IP-level changes where we have no IP-to-PoP mapping, and 2. where we ignore these changes. We arrive at an average rate of 10% and 1%, respectively, of paths changing daily, which can be processed efficiently at the atlas service.

The network atlas service expects a Ruby program on its input and provides the output of that program as its result. The Ruby program can make repeated queries to the database of collected measurements (using the call queryPendingPaths), filter out unnecessary results, and report back to the endpoint a pruned list of paths that match application-specific criteria. Figure 5 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.

```
require 'atlas'

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

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

Figure 5: A Ruby program that returns all ARROW 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.

4.3 Wide Area Deployment

We have deployed ARROW nodes in the wide area using the Transit Portal (TP) BGP testbed [33] and the VICCI set of geographically distributed compute clusters [34]. TP lets us announce /24 prefixes using six US universities as our providers: University of Washington (UW), University of Wisconsin (WISC), Georgia Tech (GATech), Princeton University, Clemson University and University of Southern California (USC). VICCI clusters are co-located with most of the TP sites, except for another VICCI cluster site at Stanford University that is not co-located. We envision each TP/VICCI site as a PoP for our ARROW ISP. An ARROW client simply addresses packets to a specific IP address in the TP prefix.

The Portal software router then redirects these packets to an ARROW software router via a VPN tunnel. We deploy the ARROW routers on a co-located VICCI cluster.

For redundancy, ARROW routers have replicas that are kept consistent via a distributed coordination service [12]. In a real deployment, router-specific protocols would be used to keep standby routers consistent with their primaries. Software routers run a Serval routing component in user space (kernel modifications are not allowed on VICCI); this is the data plane. Each router also runs an ARROW circuit creation daemon, a replica manager, and a failover management daemon; the control plane—implemented in Python.

4.4 ARROW Services

We have built several prototype services on our ARROW deployment, to demonstrate the value of additional services deployed with ARROW. We briefly describe two of these services in this section: a DoS protection service and a content distribution network.

DoS Protection. We have implemented the Phalanx [7] DoS protection scheme and deployed it at ARROW software router nodes at each Transit Portal PoP. The service can be offered to anyone on the Internet wishing to obtain a DoS protection layer for their own services. Phalanx achieves DoS protection by requiring clients to interleave packets to the server through a predetermined sequence of bandwidth-limiting proxy nodes (called mailboxes), each of which has a different IP address. Since the attacker does not know this sequence, they can affect only a fraction of the traffic exchanged. The client and server initially agree on the random sequence of mailboxes (through a secret key) before traffic can flow. The communication proceeds by the client sending a packet to the appropriate mailbox and the server requesting the packet from that mailbox. The router at the TP node forwards client packets and server requests to the ARROW VICCI nodes where client packets are stored until the server requests them.

Content Distribution Network. We have also implemented a simple CDN offering a traffic cache at each Transit Portal PoP. The CDN consists of a controller and a group of geo-replicated cache nodes. The controller is responsible for collecting content requests and directs them to the cache replica with the lowest latency to the requesting client. Each ISP offering CDN service is in a good position to know this information for its PoPs. In our prototype deployment, each replica stores a copy of each file. After receiving a client request, the controller instructs the cache nodes to measure their RTT latencies to the client and picks the one with the lowest RTT to serve the content to the client.

5. EVALUATION

We evaluate ARROW via simulation on AS and PoP-level Internet topology, as well as by measurement of our implementation, both when deployed on a local cluster of machines, as well as within our large-scale deployment. Specifically, we seek to answer the following questions:

- How are throughput and latency of Internet traffic affected when an ARROW path (of various lengths) is used?
- How nimble are the individual failover mechanisms that are deployed along an ARROW path?
- How quickly is ARROW able to re-establish end-to-end connectivity in the event of a link failure on the Internet?
- How resilient are various ARROW deployments to (potentially cascading) Internet link and AS failures and can we use ARROW to avoid untrusted ISPs?

	RTT [μs]	Throughput [Gbits/s]
UDP/TCP	44/96/107	9.05/9.36/9.68
Serval	73/81.23/154	9.35/9.52/9.74
1 ARROW hop	113/131.96/290	9.37/9.55/9.85
2 ARROW hops	158/191.38/444	8.19/8.49/8.72

Table 1: RTT and throughput (min/avg/max) of different ARROW path lengths vs. UDP/TCP and Serval.

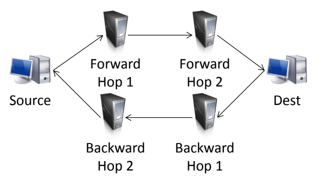

Figure 6: ARROW 6-node cluster deployment.

- How effectively do various ARROW deployments prevent IP prefix-hijacking attacks?
- Is the end-to-end latency of ARROW paths comparable to that of regular BGP paths?

We are especially concerned with evaluating whether a single tunnel (one ARROW hop) is enough to provide all of the benefits. Hence, with the exception of performance evaluation, we only evaluate one-hop tunnels within the experiments in this section (as opposed to the number of ARROW-supporting ASes, which we *do* vary).

5.1 Performance Overhead

To evaluate the latency and throughput overheads of our ARROW prototype, we have deployed it on a 6-node cluster. All nodes run Linux 3.2.0 on Intel Xeon E5-2430 processors at 2.2 GHz clock frequency, 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 6 shows this setup.

In the cluster, one node acts as the source endpoint of a route and another one as the destination. The other nodes are used as ARROW routers. The deployment is symmetrical: Both forward and reverse ARROW paths are established between source and target, over distinct nodes in the cluster. We can construct up to 2 ARROW hops in this symmetrical fashion. We determine the latency along a path by measuring the average round-trip time (RTT) of 100 individual 64 byte UDP packets to the destination which echoes them back unmodified. We measure the average throughput over 5 TCP transfers of a data stream over 10 seconds each, using the iperf[1] bandwidth measurement tool.

Table 1 shows the measurement results of different lengths of ARROW routes compared to UDP/TCP and unmodified Serval. The TCP, UDP and Serval measurements measure direct bandwidth and RTT between two endpoints, without going through any intermediate hops. The ARROW measurements forward packets according to the experimental setup shown in Figure 6.

In terms of latency, ARROW adds an RTT overhead of 62% over the baseline Serval implementation and 37% over UDP. A 2-hop ARROW path has an overhead of 45% over the RTT latency of a 1-hop path. Throughput is not affected by adding ARROW to a 1-hop path. However, adding another ARROW hop impacts throughput by 10%. This might be due to our switch not being able to handle the bandwidth requirement.

We conclude that the overheads imposed by ARROW are small enough to merit the deployment of a wide-area prototype. In a pro-

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

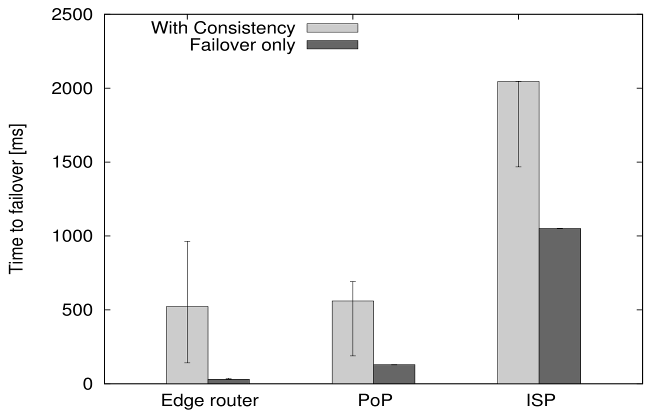

Figure 7: Time to failover for various component failures along a ARROW path. The plot shows the median (min/max in error bars) over 5 measurements, each with Zookeeper and without.

duction environment, purpose-built routers would likely be used. They already support the required features: MPLS tunnels can be used to forward packets among tunnel segments and Software Defined Networking (SDN) technology aids the setup and management of tunnel segments.

5.2 Path Failover Latency

For ARROW to provide significant benefit versus BGP, it is important that its own paths are not prone to long failover times itself. We examine the time taken to failover a complete path for each endpoint-observable failure along that path within our wide-area ARROW deployment.

The measured topology involves 3 PoPs (UW, Princeton, and Stanford), as well as 3 ASes: Home, Transit, and Dest. Home meets Transit at both UW and Princeton, and Transit meets Dest at Stanford. Using the VICCI cluster nodes, we “deploy” three redundant routers (one active, and two standby) of each AS at each PoP—six routers total at each PoP and 18 routers in the entire topology.

We then establish an ARROW path from Home via Transit to Dest. After the connection is established, we intentionally kill the software on a router node to simulate a failure. To simulate an edge router failure, we kill the active Transit ingress router at UW. For a PoP failure, we kill all routers at UW. Finally, for an ISP failure, we kill all Transit routers at both UW and Princeton.

Figure 7 shows the measured failover times over 5 measurements for each of the 3 failure cases, as observed from the endpoint, i.e. after the endpoint detected there is a failure until the connection is restored. This involves the execution of the entire failover protocol, as described in §3.4. Failover times increase with the scope of the failure. We also observe high failover times when the consistency service is used. While some consistency service is necessary, we believe that our choice of consistency service (Zookeeper [12]) could be optimized to remove some or all of this cost.

Edge routers exist within the same PoP and failure detection can be very aggressive. Hence, this is the shortest failover time of only 340ms (30ms without consistency service). If an entire PoP fails, none of the 3 edge router replicas are reachable and we have to re-route to another PoP, which can be far away. This is the case in our deployment (from UW to Princeton) and thus PoP failovers take slightly longer, roughly 414ms (128ms without consistency service). Finally, if an entire ISP fails, we have to probe several of its PoPs until we determine that the ISP is down. The endpoint then has to failover to a backup ARROW path. This takes on the order of 1.8 seconds (1 second without consistency service).

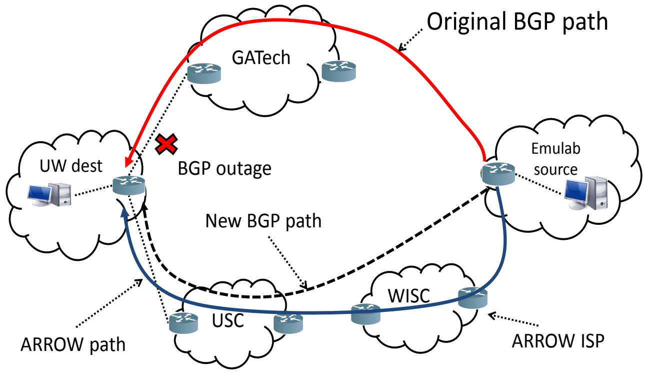

Figure 8: Setup of the BGP outage experiment. We cause a failure at UW’s upstream link with GATech. BGP will eventually failover to a path via USC. ARROW can use an alternative path via WISC.

We conclude that most failures along an ARROW path can be addressed in a timespan that is hardly noticeable to most users, especially when router consistency is optimized. Only whole ISP failures take longer and might be noticeable at the endpoints. Compared to the time it takes for BGP to converge after such failures, ARROW still compares favorably in all of the cases.

5.3 End-to-end Recovery After BGP Outage

To examine how quickly end-to-end connectivity can be restored using an ARROW backup path in the event of a BGP outage, we conduct an experiment on our wide area deployment where we forcibly take down a BGP link, similar to an experiment conducted in RON [3]. The experiment setup is shown in Figure 8, where we have a multi-homed destination at the UW, by announcing a prefix through TP sites located at GATech and USC simultaneously. Our source is an Emulab [9] node, and its original path to the UW destination passes through GATech. We also have an ARROW deployment at WISC that is reachable using a prefix announced only through WISC. This provides us with a one-hop ARROW path through WISC to UW. We then establish a regular UDP connection from the source node to a destination within the UW prefix and continuously measure its throughput. We conduct this experiment twice, once with a failover using ARROW and once with a regular BGP failover.

The measured throughput over time is shown for both cases in Figure 9. 7 seconds into the experiment, we shut down the VPN tunnel between UW and GATech, resulting in a BGP session failure and an instantaneous loss of all packets along the path. The failure eventually causes BGP to withdraw the route through GATech and converge to the new route through USC (red line). This outage lasts for nearly 90 seconds until convergence to the new route takes place. In contrast, the ARROW deployment is able to detect and recover from the failure within a few hundred milliseconds (blue line). The source in this case detects the outage through a sequence of unacknowledged packets and immediately starts forwarding packets through the pre-established ARROW path.

We conclude that ARROW allows endpoints to failover an end-to-end path within a timespan of a typical TCP timeout, which is hardly noticeable at the endpoint and does not disrupt open connections, except for a short dip in throughput.

5.4 Simulation Dataset and Methodology

Next, we explore the potential reliability and performance properties of ARROW deployed at Internet-scale. For this purpose we examine BGP and ARROW routing on Internet topology. We do

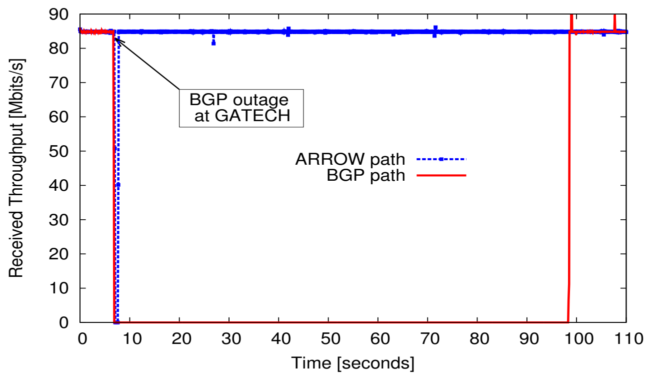

Figure 9: Time-series showing throughput for a BGP failover and a ARROW failover for the experiment in Figure 8. The link fails at 7 seconds. BGP takes 90 seconds before convergence to a new path. ARROW failover is on the order of hundreds of milliseconds.

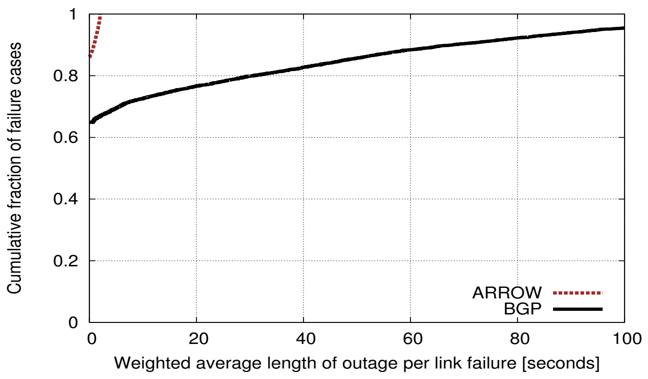

Figure 10: CDF showing the fraction of link failures resulting in a certain amount of disconnectivity, weighted by the average duration of the disconnectivity event.

this using two methodologies: 1. we simulate the effects of BGP routing decisions (§5.5 and §5.6) and 2. we use actual BGP routing measurements collected by iPlane[2] (§5.7, §5.8, and §5.9).

For the first method, we acquired the simulator used to evaluate Consensus Routing [13], which simulates routing decisions, BGP protocol control traffic, and link failures. We use the November 2013 CAIDA AS-level connectivity graph [1], gathered from RouteViews BGP tables [2], to simulate on a realistic Internet topology. This dataset has a total of 29,730 ASes and 159,049 unique AS-level links, annotated with the inferred business relationships of the linked ASes (customer-provider or peer-peer). The simulator uses standard route selection and advertisement policies of the Internet, such as "valley-free" export and "hot potato" routing. More detail about the simulator can be found in [13].

The iPlane dataset used in the second method is built using traceroutes from over 200 PlanetLab sites to more than 140,000 prefixes (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. We use the most recent iPlane snapshot collected in December 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.

5.5 Resilience to Link Failures

We start by evaluating the resilience provided by ARROW in case of link failures. For this evaluation, we choose only provider links of multi-homed stub ASes in the topology. A multi-homed stub AS is an AS with more than one provider and no customers; our topology includes 20,338 such ASes. We focus on these because the stub AS has a valid physical route to the rest of the Internet even if a provider link L fails and we argue that this is also the worst possible case, as the Internet topology features much less redundancy towards its leaves (the stub ASes). Link failures closer to the core of the Internet would simply affect a much smaller fraction of ASes for both ARROW and BGP.

We arrive at a total number of 47,652 unique failures (the number of parent links over all multi-homed stub ASes), affecting an average of 4,404 AS paths for every multihomed AS. We fail each link L of each multi-homed stub AS A, successively. For each failure trial, we fail a link L, and see what fraction of ASes (on the whole topology) are temporarily disconnected from the corresponding stub AS A, due to BGP convergence effects. In some cases, ASes become permanently disconnected, but we ignore these in our results (note that ARROW might still provide them with a valid route). We then also determine how long it takes, on average, for these ASes to become re-connected. For each failed link, we multiply the fraction of affected ASes by the length of the duration. This is the result we use for the BGP line in Figure 10.

To plot the ARROW line, we select one random tier-1 AS as our ARROW-supporting AS and fix it for all simulation runs. Then we conduct the same failure simulations, but in addition, we check for each affected AS whether it has a valid path to A via the chosen ARROW AS. For the failover duration we choose a fixed time of 2 seconds, which is a conservative estimate for ARROW, given the results in §5.3.

Figure 10 is a CDF plot of the results. The x-axis shows the fraction of disconnectivity in the topology as the result of the failure, weighted by the duration of the outage. For each such fraction f on the x-axis we have the corresponding fraction of failures that resulted in at most f disconnectivity on the y-axis. We crop the graph at 100 seconds. The tail of BGP outages goes on until a maximum of 418 seconds.

We can see that for 30% of link failures, BGP had an outage of at least 2 seconds, with a long tail: 12% of outages have a duration of at least a minute and 5% of outages last 100 seconds and longer. All outages are handled by a single ARROW tier-1 relay node within 2 seconds (and likely much shorter, according to §5.3). We conclude that a single tunnel is indeed enough to recover from link failure disconnectivities in a time-span significantly shorter than that of BGP in the worst possible case of a link failure to a parent of a multi-homed stub AS.

5.6 Resilience to AS Failures

The failure of an entire Autonomous System (AS) is a rare occurrence, but we would still like to gauge the effectiveness of ARROW routing around such catastrophic outages. To do this, we conduct a similar simulation to the previous one, but this time we simulate a simultaneous failure of all links of a particular AS. We picked 200 ASes at random from the set of all tier 2 and tier 3 transit ASes. Figure 11 is a CDF plot of the results. The x and y axes are the same as in Figure 10, except we crop the x-axis at 200 seconds.

We can see that around 60% of AS failures cause a BGP convergence period of at least 2 seconds. The tail is long again, with more than 50% of failures resulting in a convergence duration of at least 100 seconds. A single ARROW-supporting tier-1 AS provides instant failover for more than 55% of the failures. If ARROW were

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

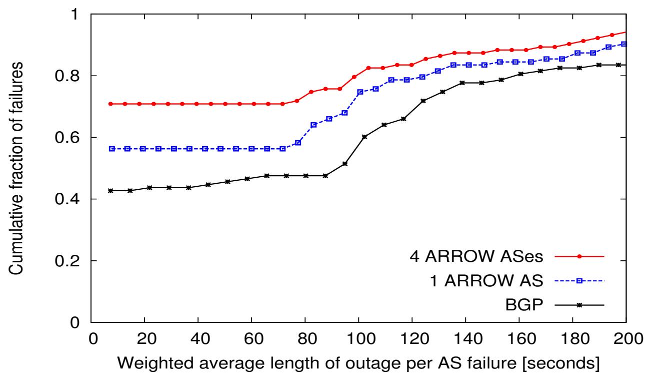

Figure 11: CDF showing the fraction of AS failures resulting in a certain amount of disconnectivity, weighted by the average duration of the disconnectivity event.

deployed at 4 tier-1 sites, a single tunnel can provide failover for more than 70% of the failures. While a large number of failures can be circumvented this way, the number is not as encouraging as in Figure 10. This is because the interconnection density of tier-2 and tier-3 ASes is much higher than that of stub ASes and they are also closer to an ARROW supporting tier-1.

We conclude that ARROW enables us to route around a large number of transit AS failures. While a great improvement over BGP, a multi-hop ARROW path might be required to increase the number of preventable failures even further. We also note that these results can serve as an indicator of how likely it is that ARROW can be used to route around a transit AS that is otherwise not delivering traffic in the way we expect or that we simply do not trust, provided that it is not malicious. We will address one case of malicious ASes (prefix hijacking) in §5.8.

5.7 Path Redundancy

It is clear that ARROW can be used to route around single defective links and misbehaving ISPs. To reduce the risk of encountering a string of such misbehaviors, we ask if we can build an ARROW path with (near-)complete AS-level redundancy. We define a path q to be completely redundant to path p if the AS-hops in q is disjoint from that of p, except for the source and destination ASes. We are interested in the number of common hops (disregarding source and destination) between the original path p and the ARROW path q with the highest disjointedness. Figure 12 shows this distribution over all paths in the iPlane dataset (approximately 5 million) for ARROW deployments on 2 and 4 tier-1 and tier-2 ASes.

In the tier-1 case, almost 40% of the paths for the 2-AS deployment, and 50% for the 4-AS deployment provide completely disjoint paths. Almost 80% of the paths in both cases have less than half of the ASes from the old path still present in the new ARROW path. The significant redundancy with a small ARROW deployment can be explained by the rich peering provided by tier-1 ISPs and the resulting high redundancy in the core of the Internet. This is confirmed by comparing with the tier-2 case, where significantly less redundancy is present.

We conclude that alternative ARROW paths ensure a high degree of redundancy between the old and new paths, making ARROW resilient even in the rare case of multiple cascading link and AS-level misbehaviors.

5.8 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 ex-

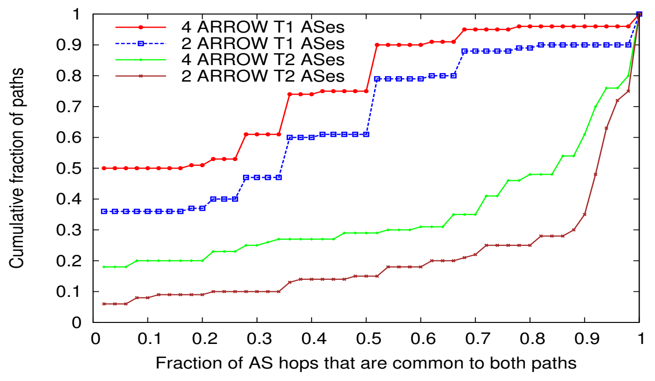

Figure 12: CDF showing the number of common AS hops as a fraction of p's length (x-axis) versus the cumulative fraction of paths with that amount of commonality (y-axis).

tremely hard to eliminate. ARROW 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 ARROW 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 iPlane topology as victims and average the results over a random selection of 20 attackers for each victim. This gives us a total of 16,160 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 [39]. For each attack and a given ARROW deployment we see what fraction of the sources remain unpolluted, i.e., able to send traffic to the victim through any of the ARROW path segments. Figure 13 shows the CCDF of the hijack attacks. The x-axis shows the fraction of sources remaining polluted as a result of the attack versus the corresponding fraction of attacks that resulted in at most p pollution on the y-axis. We compare ARROW deployments of various sizes with BGP routing.

We observe that all three deployments of ARROW provide significant protection against prefix hijacks: For 75% of the attacks, a single ARROW AS cuts down pollution to 5% or less. Without ARROW the same scenario results in up to 30% pollution. An ARROW deployment on only four tier-1s eliminates almost all of the unreachability caused by prefix hijacks. We conclude that even a small ARROW deployment provides an effective means to combat prefix hijack attacks.

5.9 Reliable Performance

For redundant ARROW paths to be useful, it is important that the performance of these paths is not significantly worse than the original BGP path. We evaluate the performance of paths available in an ARROW deployment compared to the respective BGP path. For this purpose we use the PoP-level link latencies provided by the iPlane dataset. Using the same methodology as in previous subsections, we ask the question: what is the fraction of sources that have an alternative ARROW path with an end-to-end latency that is at most X% that of the original path?

We assume no overhead for ARROW routing in this simulation and solely look at the measured path latencies from the iPlane dataset. Hence, these results are slightly skewed towards ARROW and we expect an experimental evaluation on a real deployment to look slightly less encouraging for ARROW. Figure 14 shows a CDF of the ratio of the end-to-end latency achieved by an ARROW

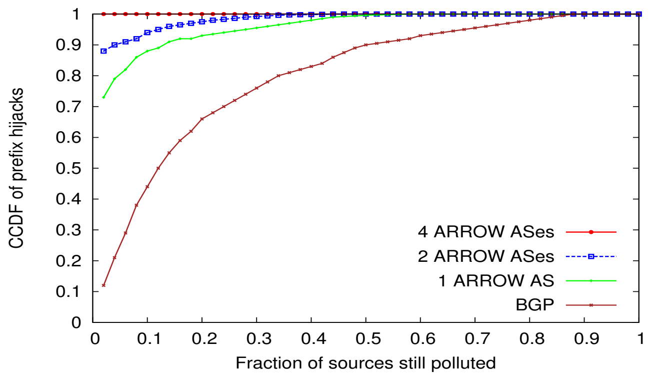

Figure 13: CCDF showing the fraction of prefix hijack attacks resulting in a certain amount of pollution, as measured by the fraction of sources unable to reach the target as a result of the attack.

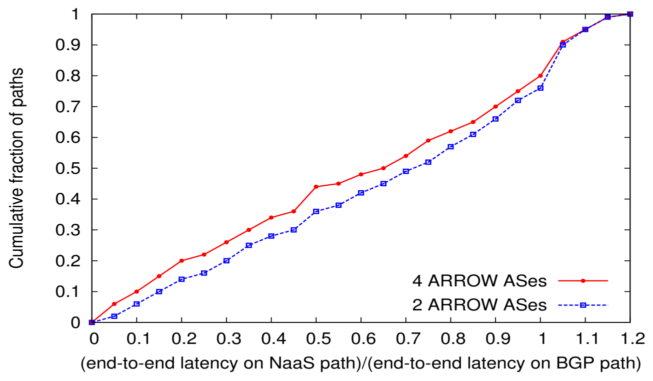

Figure 14: CDF showing the ratio of end-to-end latency of an ARROW path between a source and destination PoP and the latency of the equivalent BGP path versus the cumulative fraction of paths with up to that ratio. Two ARROW deployments are shown.

path and the latency of the equivalent BGP path between the same source and destination PoP for a total of 1,143,652 PoP-level paths.

We observe that in a deployment with 2 and 4 ARROW ASes, only 24% and 20% of ARROW paths have a latency slightly (up to 20%) worse than that of their equivalent BGP path, respectively. With 2 ARROW ASes, more than 76% of the sources actually experience an *improvement* in the end-to-end latency while using an ARROW path. As expected, the gains are even slightly higher for a deployment of 4 ARROW ASes.

We conclude that ARROW does not degrade the end-to-end latency for the vast majority of paths and, in fact, has the potential to *improve* their performance. This can serve as an important factor for ISPs in attracting traffic from the greater Internet. It also shows that using ARROW does not have to come at a performance tradeoff for most users.

6. RELATED WORK

Our goal in this work is to quantify the extent that a simple remote tunneling model such as ARROW can address a range of Internet vulnerabilities, particularly given the recent trend towards a flatter Internet topology [20] and very high-bandwidth servers.

Fifteen years ago, Detour [3, 29] observed that well-chosen indirect paths were often more reliable and higher performance than direct paths through the Internet, although it stopped short of suggesting this as a universal solution because processing costs would have been prohibitive at the time. The success of various Detour-like systems built by companies such as Akamai SureRoute and others have shown that better Internet reliability is commercially viable. Likewise the widespread move towards multihomed enterprises is proof of the value of improved reliability to end users.

Our work goes beyond earlier studies of Detour and multihoming to consider the range of attacks, both inadvertent and intentional, against the Internet. At an architectural level, a key distinction with Detour is that ARROW routes are composable. A sequence of ARROW circuits through participating providers will deliver packets at a given rate along the path, barring failure. Detour routes provide no such guarantee.

Source control over routing is an old idea. Loose source routing in the Internet is widely but not universally blocked [15] due to concerns about its possible use in denial-of-service attacks. This has led to a set of efforts that provide joint control over routes between the ISPs along the path and the endpoint. The most flexible of these is Icing [24]. Icing attempts to head off subversion of routing advertisements by asking every entity along the path to approve the use of the entire path; no traffic flows unless everyone agrees. Icing does not, however, attempt to address incremental or partial adoption. Yang el al. [37] 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 [22] recognized the conflict between users who want control over end-to-end paths and ISPs who desire control over how their infrastructure is used. To resolve this conflict, the authors introduce a separate entity that contracts with both ASes and customers and establishes paths that are acceptable to all entities.

Although ARROW is in many ways a simpler model, our results should apply to these other models as well. A goal of our work is to help answer whether a simpler model is sufficient for addressing known malicious and non-malicious errors, as well as to point a way towards incremental adoption of one of these more far-reaching solutions.

We also borrow a number of ideas from other proposals. Dynamic resolution of circuit ID's was introduced by ATM. Our topology announcements are similar to those in pathlet routing [10], although we assume users will want to bind traffic PoP to PoP rather than only ISP to ISP. By handing topology announcements out of band (that is, not via BGP), we make it easier for ISPs to add information such as price and expected performance to various prefixes. Using indirection as a method for addressing denial of service originated with the Internet Indirection Infrastructure (i3) [31]; packet authenticators for governing access to a tunnel was introduced by SOS and Mayday [4, 18].

Considerable effort has gone into incremental changes to BGP to address specific vulnerabilities. For example, MIRO [36] 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 [19] proposes to use pre-computed backup paths to provide reliable delivery during periods where the network is adapting to failures. ARROW has similar goals, but obtains additional deployability benefits since it doesn't require changes to the inter-domain routing protocol. A single ISP can unilaterally provide ARROW service and obtain revenues directly from end users who would benefit from the service.

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. The current Internet, however, is unable to meet the availability demands of these emerging and future uses. In this paper, we attempt to identify the minimal changes needed for the Internet to support such mission critical data transmissions.

This paper presents a mechanism to 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 *Advertised Reliable Routing Over Waypoints*, which allows users to provision a path across a remote ISP. We outlined the design of ARROW, examined how it can be used to enhance the robustness and security of end-to-end paths, and described an implementation of its key components. Our evaluations show that ARROW imposes only minor overheads to Internet routing infrastructure and can provide significant resiliency and performance benefits even when deployed by only a limited number of ISPs. Finally, ARROW provides the fabric for additional services that can be deployed on top of it. To show that this is feasible, we presented two of our own implementations: a DoS protection service and a content distribution network.

Acknowledgments

We would like to thank AMS-IX and the universities that generously host the Transit Portal testbed. This work was funded by Cisco and NSF Grant CNS-1040663. Qiao Zhang was supported by a Wilma Bradley Endowed Fellowship.

8. REFERENCES

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

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

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.

[4] D. G. Andersen. Mayday: Distributed filtering for internet services. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[5] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (aip). In *SIGCOMM*, 2008.

[6] J. Cowie. The new threat: Targeted internet traffic misdirection. `http://www.renesys.com/2013/11/mitm-internet-hijacking/`. Retrieved 2014-05-20.

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

[8] 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 *SOSP*, 2009.

[9] Emulab: A network emulation testbed. `http://www.emulab.net`.

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

[11] S. Han, K. Jang, K. Park, and S. Moon. Packetshader: a gpu-accelerated software router. In *SIGCOMM*, 2010.

[12] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX Annual Technical Conference*, 2010.

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

[14] E. Katz-Bassett, H. Madhyastha, J. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying blackholes in the Internet with Hubble. In *NSDI*, 2008.

[15] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy. Reverse traceroute. In *NSDI*, 2010.

[16] 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 *SIGCOMM*, 2012.

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

[18] A. Keromytis, V. Misra, and D. Rubenstein. SOS: An architecture for mitigating DDoS attacks. *IEEE Journal on Selected Areas in Communications*, 2003.

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

[20] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *SIGCOMM*, 2010.

[21] 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.

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

[23] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: secure and adoptable source authentication. In *NSDI*, 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. Nordstrom, 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 *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] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. In *SIGCOMM*, 2004.

[28] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *SIGCOMM*, 2013.

[29] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *SIGCOMM*, 1999.

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

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

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

[33] V. Valancius, N. Feamster, J. Rexford, and A. Nakao. Wide-area route control for distributed services. In *USENIX Annual Technical Conference*, 2010.

[34] VICCI: A programmable cloud-computing research testbed. `http://www.vicci.org`.

[35] S. Waterman. Internet traffic was routed via chinese servers. `http://www.washingtontimes.com/news/2010/nov/15/internet-traffic-was-routed-via-chinese-servers/`. Retrieved 2014-05-20.

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

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

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

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