Resilient Overlay Networks

David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris
MIT Laboratory for Computer Science
ron@nms.lcs.mit.edu
http://nms.lcs.mit.edu/ron/

Abstract

A Resilient Overlay Network (RON) is an architecture that allows distributed Internet applications to detect and recover from path outages and periods of degraded performance within several seconds, improving over today's wide-area routing protocols that take at least several minutes to recover. A RON is an application-layer overlay on top of the existing Internet routing substrate. The RON nodes monitor the functioning and quality of the Internet paths among themselves, and use this information to decide whether to route packets directly over the Internet or by way of other RON nodes, optimizing application-specific routing metrics.

Results from two sets of measurements of a working RON deployed at sites scattered across the Internet demonstrate the benefits of our architecture. For instance, over a 64-hour sampling period in March 2001 across a twelve-node RON, there were 32 significant outages lasting over thirty minutes each over the 132 measured paths. RON's routing mechanism was able to detect, recover, and route around *all* of them, in less than twenty seconds on average, showing that its methods for fault detection and recovery work well at discovering alternate paths in the Internet. Furthermore, RON was able to improve loss rate, latency, or throughput perceived by data transfers; for example, about 5% of the transfers doubled their TCP throughput and 5% of our transfers saw their loss probability reduced by 0.05. We found that forwarding packets via at most one intermediate RON node is sufficient to overcome faults and improve performance in most cases. These improvements, particularly in the area of fault detection and recovery, demonstrate the benefits of moving some of the control over routing into the hands of end-systems.

1. Introduction

The Internet is organized as independently operating autonomous systems (AS's) that peer together. In this architecture, detailed routing information is maintained only within a single AS and its constituent networks, usually operated by some network service provider. The information shared with other providers and AS's is heavily filtered and summarized using the Border Gateway Protocol (BGP-4) running at the border routers between AS's [21], which allows the Internet to scale to millions of networks.

This wide-area routing scalability comes at the cost of reduced fault-tolerance of end-to-end communication between Internet hosts. This cost arises because BGP hides many topological details in the interests of scalability and policy enforcement, has little information about traffic conditions, and damps routing updates when potential problems arise to prevent large-scale oscillations. As a result, BGP's fault recovery mechanisms sometimes take many minutes before routes converge to a consistent form [12], and there are times when path outages even lead to significant disruptions in communication lasting tens of minutes or more [3, 18, 19]. The result is that today's Internet is vulnerable to router and link faults, configuration errors, and malice—hardly a week goes by without some serious problem affecting the connectivity provided by one or more Internet Service Providers (ISPs) [15].

Resilient Overlay Networks (RONs) are a remedy for some of these problems. Distributed applications layer a "resilient overlay network" over the underlying Internet routing substrate. The nodes comprising a RON reside in a variety of routing domains, and cooperate with each other to forward data on behalf of any pair of communicating nodes in the RON. Because AS's are independently administrated and configured, and routing domains rarely share interior links, they generally fail independently of each other. As a result, if the underlying topology has physical path redundancy, RON can often find paths between its nodes, even when wide-area routing Internet protocols like BGP-4 cannot.

The main goal of RON is to enable a group of nodes to communicate with each other in the face of problems with the underlying Internet paths connecting them. RON detects problems by aggressively probing and monitoring the paths connecting its nodes. If the underlying Internet path is the best one, that path is used and no other RON node is involved in the forwarding path. If the Internet path is not the best one, the RON will forward the packet by way of other RON nodes. In practice, we have found that RON can route around most failures by using only one intermediate hop.

RON nodes exchange information about the quality of the paths among themselves via a routing protocol and build forwarding tables based on a variety of path metrics, including latency, packet loss rate, and available throughput. Each RON node obtains the path metrics using a combination of active probing experiments and passive observations of on-going data transfers. In our implementation, each RON is explicitly designed to be limited in size—between two and fifty nodes—to facilitate aggressive path maintenance via probing without excessive bandwidth overhead. This

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare Systems Center, San Diego, under contract N66001-00-1-8933.

Figure 1: The current sixteen-node RON deployment. Five sites are at universities in the USA, two are European universities (not shown), three are "broadband" home Internet hosts connected by Cable or DSL, one is located at a US ISP, and six are at corporations in the USA.

allows RON to recover from problems in the underlying Internet in several seconds rather than several minutes.

The second goal of RON is to integrate routing and path selection with distributed applications more tightly than is traditionally done. This integration includes the ability to consult application-specific metrics in selecting paths, and the ability to incorporate application-specific notions of what network conditions constitute a "fault." As a result, RONs can be used in a variety of ways. A multimedia conferencing program may link directly against the RON library, transparently forming an overlay between all participants in the conference, and using loss rates, delay jitter, or application-observed throughput as metrics on which to choose paths. An administrator may wish to use a RON-based router application to form an overlay network between multiple LANs as an "Overlay VPN." This idea can be extended further to develop an "Overlay ISP," formed by linking (via RON) points of presence in different traditional ISPs after buying bandwidth from them. Using RON's routing machinery, an Overlay ISP can provide more resilient and failure-resistant Internet service to its customers.

The third goal of RON is to provide a framework for the implementation of expressive routing policies, which govern the choice of paths in the network. For example, RON facilitates classifying packets into categories that could implement notions of acceptable use, or enforce forwarding rate controls.

This paper describes the design and implementation of RON, and presents several experiments that evaluate whether RON is a good idea. To conduct this evaluation and demonstrate the benefits of RON, we have deployed a working sixteen-node RON at sites sprinkled across the Internet (see Figure 1). The RON client we experiment with is a resilient IP forwarder, which allows us to compare connections between pairs of nodes running over a RON against running straight over the Internet.

We have collected a few weeks' worth of experimental results of path outages and performance failures and present a detailed analysis of two separate datasets: RON_1 with twelve nodes measured in March 2001 and RON_2 with sixteen nodes measured in May 2001. In both datasets, we found that RON was able to route around between 60% and 100% of all significant outages. Our implementation takes 18 seconds, on average, to detect and route around a path failure and is able to do so in the face of an active denial-of-service attack on a path. We also found that these benefits of quick fault detection and successful recovery are realized on the public Internet and do not depend on the existence of non-commercial or private networks (such as the Internet2 backbone that interconnects many educational institutions); our ability to determine this was enabled by RON's policy routing feature that allows the expression and implementation of sophisticated policies that determine how paths are selected for packets. A noteworthy finding from the experiments is that forwarding packets via at most one intermediate RON node is sufficient to overcome faults in most cases.

We also found that RON successfully routed around performance failures, doubling TCP throughput in 5% of all samples, improving the loss probability by at least 0.05 in 5% of the samples, and reducing end-to-end communication latency by 40ms in 11% of the samples. However, it did not change the median TCP throughput in our datasets. In addition, we found cases when RON's loss-, latency-, and throughput-optimizing path selection mechanisms all chose different paths between the same two nodes, suggesting that application-specific path selection techniques are likely to be useful in practice.

2. Related Work

To our knowledge, RON is the first wide-area network overlay system that can detect and recover from path outages and periods of degraded performance within several seconds. RON builds on previous studies that quantify end-to-end network reliability and performance, on IP-based routing techniques for fault-tolerance, and on overlay-based techniques to enhance performance.

2.1 Internet Performance Studies

Labovitz *et al.* [12] use a combination of measurement and analysis to show that inter-domain routers in the Internet may take tens of minutes to reach a consistent view of the network topology after a fault, primarily because of routing table oscillations during BGP's rather complicated path selection process. They find that during this period of "delayed convergence," end-to-end communication is adversely affected. In fact, outages on the order of minutes cause active TCP connections (i.e., connections in the `ESTABLISHED` state with outstanding data) to terminate when TCP does not receive an acknowledgment for its outstanding data. They also find that, while part of the convergence delays can be fixed with changes to the deployed BGP implementations, long delays and temporary oscillations are a fundamental consequence of the BGP path vector routing protocol.

Paxson's probe experiments show that routing pathologies prevent selected Internet hosts from communicating up to 3.3% of the time averaged over a long time period, and that this percentage has not improved with time [18]. Labovitz *et al.* find, by examining routing table logs at Internet backbones, that 10% of all considered routes were available less than 95% of the time, and that less than 35% of all routes were available more than 99.99% of the time [13]. Furthermore, they find that about 40% of all path outages take more than 30 minutes to repair and are heavy-tailed in their duration. More recently, Chandra *et al.* find using active probing that 5% of all detected failures last more than 10,000 seconds (2 hours, 45 minutes), and that failure durations are heavy-tailed and can last for as long as 100,000 seconds before being repaired [3]. These findings do not augur well for mission-critical services that require a higher degree of end-to-end communication availability.

The Detour measurement study made the observation, using Paxson's and their own data collected at various times between 1995 and 1999, that path selection in the wide-area Internet is sub-optimal from the standpoint of end-to-end latency, packet loss rate, and TCP throughput [23]. This study showed the potential long-term benefits of "detouring" packets via a third node by comparing

the long-term average properties of detoured paths against Internet-chosen paths.

2.2 Network-layer Techniques

Much work has been done on performance-based and fault-tolerant routing within a single routing domain, but practical mechanisms for wide-area Internet recovery from outages or badly performing paths are lacking.

Although today's wide-area BGP-4 routing is based largely on AS hop-counts, early ARPANET routing was more dynamic, responding to the current delay and utilization of the network. By 1989, the ARPANET evolved to using a delay- and congestion-based distributed shortest path routing algorithm [11]. However, the diversity and size of today's decentralized Internet necessitated the deployment of protocols that perform more aggregation and fewer updates. As a result, unlike some interior routing protocols within AS's, BGP-4 routing *between* AS's optimizes for scalable operation over all else.

By treating vast collections of subnetworks as a single entity for global routing purposes, BGP-4 is able to summarize and aggregate enormous amounts of routing information into a format that scales to hundreds of millions of hosts. To prevent costly route oscillations, BGP-4 explicitly damps changes in routes. Unfortunately, while aggregation and damping provide good scalability, they interfere with rapid detection and recovery when faults occur. RON handles this by leaving scalable operation to the underlying Internet substrate, moving fault detection and recovery to a higher layer overlay that is capable of faster response because it does not have to worry about scalability.

An oft-cited "solution" to achieving fault-tolerant network connectivity for a small- or medium-sized customer to *multi-home*, advertising a customer network through multiple ISPs. The idea is that an outage in one ISP would leave the customer connected via the other. However, this solution does not generally achieve fault detection and recovery within several seconds because of the degree of aggregation used to achieve wide-area routing scalability. To limit the size of their routing tables, many ISPs will not accept routing announcements for fewer than 8192 contiguous addresses (a "/19" netblock). Small companies, regardless of their fault-tolerance needs, do not often require such a large address block, and cannot effectively multi-home. One alternative may be "provider-based addressing," where an organization gets addresses from multiple providers, but this requires handling two distinct sets of addresses on its hosts. It is unclear how on-going connections on one address set can seamlessly switch on a failure in this model.

2.3 Overlay-based Techniques

Overlay networks are an old idea; in fact, the Internet itself was developed as an overlay on the telephone network. Several Internet overlays have been designed in the past for various purposes, including providing OSI network-layer connectivity [10], easing IP multicast deployment using the MBone [6], and providing IPv6 connectivity using the 6-Bone [9]. The X-Bone is a recent infrastructure project designed to speed the deployment of IP-based overlay networks [26]. It provides management functions and mechanisms to insert packets into the overlay, but does not yet support fault-tolerant operation or application-controlled path selection.

Few overlay networks have been designed for efficient fault detection and recovery, although some have been designed for better end-to-end performance. The Detour framework [5, 22] was motivated by the potential long-term performance benefits of indirect routing [23]. It is an in-kernel packet encapsulation and routing architecture designed to support alternate-hop routing, with an emphasis on high performance packet classification and routing. It uses IP-in-IP encapsulation to send packets along alternate paths.

While RON shares with Detour the idea of routing via other nodes, it differs from Detour in two significant ways. First, RON seeks to prevent disruptions in end-to-end communication in the face of failures. RON takes advantage of underlying Internet path redundancy on time-scales of a few seconds, reacting responsively to path outages and performance failures. Second, RON is designed as an application-controlled routing overlay; because each RON is more closely tied to the application using it, RON more readily uses application-specific path metrics and path selection policies.

An alternative design to RON would be to use a generic overlay infrastructure like the X-Bone and port a standard network routing protocol (like OSPF or RIP) with low timer values. However, this by itself will not improve the resilience of Internet communications for two reasons. First, a reliable and low-overhead outage detection module is required, to distinguish between packet losses caused by congestion or error-prone links from legitimate problems with a path. Second, generic network-level routing protocols do not utilize application-specific definitions of faults.

Various Content Delivery Networks (CDNs) use overlay techniques and caching to improve the performance of content delivery for specific applications such as HTTP and streaming video. The functionality provided by RON may ease future CDN development by providing some routing components required by these services.

3. Design Goals

The design of RON seeks to meet three main design goals: (i) failure detection and recovery in less than 20 seconds; (ii) tighter integration of routing and path selection with the application; and (iii) expressive policy routing.

3.1 Fast Failure Detection and Recovery

Today's wide-area Internet routing system based on BGP-4 does not handle failures well. From a network perspective, we define two kinds of failures. *Link failures* occur when a router or a link connecting two routers fails because of a software error, hardware problem, or link disconnection. *Path failures* occur for a variety of reasons, including denial-of-service attacks or other bursts of traffic that cause a high degree of packet loss or high, variable latencies.

Applications perceive all failures in one of two ways: *outages* or *performance failures*. Link failures and extreme path failures cause outages, when the average packet loss rate over a sustained period of several minutes is high (about 30% or higher), causing most protocols including TCP to degrade by several orders of magnitude. Performance failures are less extreme; for example, throughput, latency, or loss-rates might degrade by a factor of two or three.

BGP-4 takes a long time, on the order of several minutes, to converge to a new valid route after a link failure causes an outage [12]. In contrast, RON's goal is to detect and recover from outages and performance failures within several seconds. Compounding this problem, IP-layer protocols like BGP-4 cannot detect problems such as packet floods and persistent congestion on links or paths that greatly degrade end-to-end performance. As long as a link is deemed "live" (i.e., the BGP session is still alive), BGP's AS-path-based routing will continue to route packets down the faulty path; unfortunately, such a path may not provide adequate performance for an application using it.

3.2 Tighter Integration with Applications

Failures and faults are application-specific notions: network conditions that are fatal for one application may be acceptable for another, more adaptive one. For instance, a UDP-based Internet audio

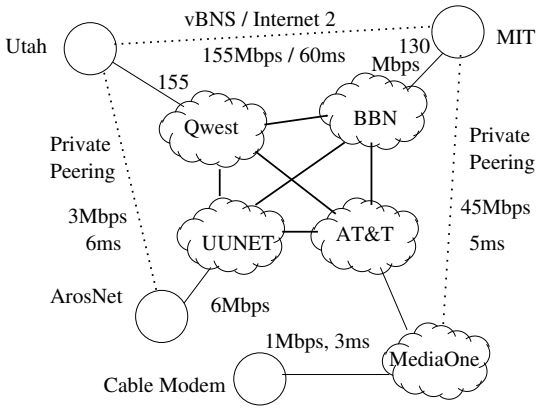

Figure 2: Internet interconnections are often complex. The dotted links are *private* and are not announced globally.

application not using good packet-level error correction may not work at all at loss rates larger than 10%. At this loss rate, a bulk transfer application using TCP will continue to work because of TCP's adaptation mechanisms, albeit at lower performance. However, at loss rates of 30% or more, TCP becomes essentially unusable because it times out for most packets [16]. RON allows applications to independently define and react to failures.

In addition, applications may prioritize some metrics over others (e.g., latency over throughput, or low loss over latency) in their path selection. They may also construct their own metrics to select paths. A routing system may not be able to optimize all of these metrics simultaneously; for example, a path with a one-second latency may appear to be the best throughput path, but this degree of latency may be unacceptable to an interactive application. Currently, RON's goal is to allow applications to influence the choice of paths using a single metric. We plan to explore multi-criteria path selection in the future.

3.3 Expressive Policy Routing

Despite the apparent need for policy routing and enforcement of acceptable use and other policies, today's approaches are primitive and cumbersome. For instance, BGP-4 is incapable of expressing fine-grained policies aimed at users or remote hosts. This lack of fine-grained policies not only reduces the set of paths available in the case of a failure, but also inhibits innovation in the use of carefully targeted policies, such as end-to-end per-user rate controls or enforcement of acceptable use policies (AUPs) based on packet classification. Because RONs will typically run on comparatively powerful end-points, we believe they are well-suited to providing fine-grained policy routing.

Figure 2 shows the AS-level network connectivity between a subset of our measurement hosts; the full graph for (only) 12 hosts traverses 36 different autonomous systems. The figure gives a hint of the considerable underlying path redundancy available in the Internet—the reason RON works—and shows situations where BGP's blunt policy expression inhibits fail-over. For example, if the Aros-UUNET connection failed, users at Aros would be unable to reach MIT even if they were authorized to use Utah's network resources to get there. This is because it impossible to announce a BGP route only to particular users, so the Utah-MIT link is kept completely private.

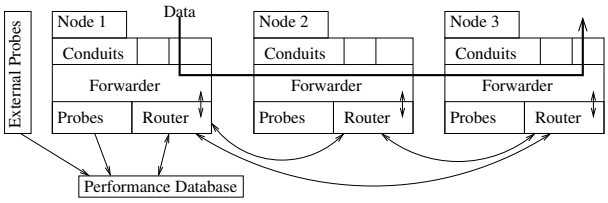

Figure 3: The RON system architecture. Data enters the RON from RON clients via a *conduit* at an *entry node*. At each node, the *RON forwarder* consults with its *router* to determine the best path for the packet, and sends it to the next node. Path selection is done at the entry node, which also tags the packet, simplifying the forwarding path at other nodes. When the packet reaches the RON *exit node*, the forwarder there hands it to the appropriate output conduit, which passes the data to the client. To choose paths, RON nodes monitor the quality of their virtual links using active probing and passive observation. RON nodes use a link-state routing protocol to disseminate the topology and virtual-link quality of the overlay network.

4. Design

The conceptual design of RON, shown in Figure 3, is quite simple. RON nodes, deployed at various locations on the Internet, form an application-layer overlay to cooperatively route packets for each other. Each RON node monitors the quality of the Internet paths between it and the other nodes, and uses this information to intelligently select paths for packets. Each Internet path between two nodes is called a *virtual link*. To discover the topology of the overlay network and obtain information about all virtual links in the topology, every RON node participates in a routing protocol to exchange information about a variety of quality metrics. Most of RON's design supports routing through multiple intermediate nodes, but our results (Section 6) show that using at most one intermediate RON node is sufficient most of the time. Therefore, parts of our design focus on finding better paths via a single intermediate RON node.

4.1 Software Architecture

Each program that communicates with the RON software on a node is a *RON client*. The overlay network is defined by a single group of clients that collaborate to provide a distributed service or application. This group of clients can use service-specific routing metrics when deciding how to forward packets in the group. Our design accommodates a variety of RON clients, ranging from a generic IP packet forwarder that improves the reliability of IP packet delivery, to a multi-party conferencing application that incorporates application-specific metrics in its route selection.

A RON client interacts with RON across an API called a *conduit*, which the client uses to send and receive packets. On the data path, the first node that receives a packet (via the conduit) classifies it to determine the type of path it should use (e.g., low-latency, high-throughput, etc.). This node is called the *entry node*: it determines a path from its topology table, encapsulates the packet into a RON header, tags it with some information that simplifies forwarding by downstream RON nodes, and forwards it on. Each subsequent RON node simply determines the next forwarding hop based on the destination address and the tag. The final RON node that delivers the packet to the RON application is called the *exit node*.

The conduits access RON via two functions:

1. `send(pkt, dst, via_ron)` allows a node to forward a packet to a destination RON node either along the RON or

using the direct Internet path. RON's delivery, like UDP, is best-effort and unreliable.

2. `recv(pkt, via_ron)` is a callback function that is called when a packet arrives for the client program. This callback is invoked after the RON conduit matches the type of the packet in the RON header to the set of types pre-registered by the client when it joins the RON. The RON packet type is a demultiplexing field for incoming packets.

The basic RON functionality is provided by the `forwarder` object, which implements the above functions. It also provides a timer registration and callback mechanism to perform periodic operations, and a similar service for network socket data availability.

Each client must instantiate a forwarder and hand to it two modules: a *RON router* and a *RON membership manager*. The RON router implements a routing protocol. The RON membership manager implements a protocol to maintain the list of members of a RON. By default, RON provides a few different RON router and membership manager modules for clients to use.

RON routers and membership managers exchange packets using RON as their forwarding service, rather than over direct IP paths. This feature of our system is beneficial because it allows these messages to be forwarded even when some underlying IP paths fail.

4.2 Routing and Path Selection

Routing is the process of building up the forwarding tables that are used to choose paths for packets. In RON, the entry node has more control over subsequent path selection than in traditional datagram networks. This node *tags* the packet's RON header with an identifier that identifies the flow to which the packet belongs; subsequent routers attempt to keep a flow ID on the same path it first used, barring significant link changes. Tagging, like the IPv6 flow ID, helps support multi-hop routing by speeding up the forwarding path at intermediate nodes. It also helps tie a packet flow to a chosen path, making performance more predictable, and provides a basis for future support of multi-path routing in RON. By tagging at the entry node, the application is given maximum control over what the network considers a "flow."

The small size of a RON relative to the Internet allows it to maintain information about multiple alternate routes and selects the path that best suits the RON client according to a client-specified routing metric. By default, it maintains information about three specific metrics for each virtual link: (i) latency, (ii) packet loss rate, and (iii) throughput, as might be obtained by a bulk-transfer TCP connection between the end-points of the virtual link. RON clients can override these defaults with their own metrics, and the RON library constructs the appropriate forwarding table to pick good paths. The router builds up forwarding tables for each combination of policy routing and chosen routing metric.

4.2.1 Link-State Dissemination

The default RON router uses a link-state routing protocol to disseminate topology information between routers, which in turn is used to build the forwarding tables. Each node in an N-node RON has $N-1$ virtual links. Each node's router periodically requests summary information of the different performance metrics to the $N-1$ other nodes from its local performance database and disseminates its view to the others.

This information is sent via the RON forwarding mesh itself, to ensure that routing information is propagated in the event of path outages and heavy loss periods. Thus, the RON routing protocol is itself a RON client, with a well-defined RON packet type. This leads to an attractive property: The only time a RON router has incomplete information about any other one is when *all* paths in the RON from the other RON nodes to it are unavailable.

4.2.2 Path Evaluation and Selection

The RON routers need an algorithm to determine if a path is still alive, and a set of algorithms with which to evaluate potential paths. The responsibility of these *metric evaluators* is to provide a number quantifying how "good" a path is according to that metric. These numbers are *relative*, and are only compared to other numbers from the same evaluator. The two important aspects of path evaluation are the mechanism by which the data for two links are combined into a single path, and the formula used to evaluate the path.

Every RON router implements *outage detection*, which it uses to determine if the virtual link between it and another node is still working. It uses an active probing mechanism for this—if the last `OUTAGE_THRESH` (set to 4 in our implementation) probe packets were all lost, then an outage is flagged. Paths experiencing outages are rated on their packet loss rate history; a path having an outage will always lose to a path not experiencing an outage. The `OUTAGE_THRESH` and the frequency of probing (`PROBE_INTERVAL`) permit a trade-off between overhead time and the bandwidth consumed by the probing. With N nodes, we send $N/$`PROBE_INTERVAL` probe packets per second, and (on average) detect an outage in $\frac{1}{2}$`PROBE_INTERVAL` $+ 4 \times$ `PROBE_TIMEOUT` seconds. This trade-off is discussed further in Section 6.2.

By default, every RON router implements three different routing metrics: the latency-minimizer; the loss-minimizer, and the throughput-optimizer. The latency-minimizer forwarding table is computed by computing an exponential weighted moving average (EWMA) of round-trip latency samples with parameter α. For any link l, its latency estimate lat_l is updated as:

$$lat_l \leftarrow \alpha \cdot lat_l + (1-\alpha) \cdot new_sample_l \qquad (1)$$

We use $\alpha = 0.9$, which means that 10% of the current latency estimate is based on the most recent sample. This number is similar to the values suggested for TCP's round-trip time estimator [20]. For a RON path, the overall latency is the sum of the individual virtual link latencies: $lat_{path} = \sum_{l \in path} lat_l$.

To estimate loss rates, RON uses the average of the last $k = 100$ probe samples as the current average. Like Floyd *et al.* [7], we found this to be a better estimator than EWMA, which retains some memory of samples obtained in the distant past as well. It might be possible to further improve our estimator by unequally weighting some of the k samples [7].

Loss metrics are multiplicative on a path: if we assume that losses are independent, the probability of success on the entire path is roughly equal to the probability of surviving all hops individually: $lossrate_{path} = 1 - \Pi_{l \in path}(1 - lossrate_l)$.

RON does not attempt to find optimal throughput paths, but strives to avoid paths of low throughput when good alternatives are available. Given the time-varying and somewhat unpredictable nature of available bandwidth on Internet paths [2, 19], we believe this is an appropriate goal. From the standpoint of improving the reliability of path selection in the face of performance failures, avoiding bad paths is more important than optimizing to eliminate small throughput differences between paths. While a characterization of the utility received by programs at different available bandwidths may help determine a good path selection threshold, we believe that more than a 50% bandwidth reduction is likely to reduce the utility of many programs. This threshold also falls outside the typical variation observed on a given path over time-scales of tens of minutes [2]. We therefore concentrate on avoiding throughput faults on this order of magnitude.

Throughput intensive applications typically use TCP or TCP-like congestion control, so the throughput optimizer focuses on this type of traffic. The performance of a bulk TCP transfer is a function of the round-trip latency and the packet loss rate it observes. Throughput optimization combines the latency and loss metrics using a simplified version of the TCP throughput equation [16], which provides an upper-bound on TCP throughput. Our granularity of loss rate detection is 1%, and the throughput equation is more sensitive at lower loss rates. We set a minimum packet loss rate of 2% to prevent an infinite bandwidth and to prevent large route oscillations from single packet losses. The formula used is:

$$score = \frac{\sqrt{1.5}}{rtt \cdot \sqrt{p}} \quad (2)$$

where p is the *one-way* packet loss probability and rtt is the end-to-end round-trip time estimated from the hop-by-hop samples as described above.

The non-linear combination of loss and throughput makes this scoring difficult to optimize with standard shortest-path algorithms; for instance, the TCP throughput between nodes A and C via a node B is not usually the smaller of the TCP throughputs between A and B and B and C. While more complicated search algorithms can be employed to handle this, RON routing currently takes advantage of the resulting simplicity when only single-intermediate paths are considered to obtain throughput-optimized paths.

Our probe samples give us the *two-way* packet loss probability, from which we estimate the one-way loss probability by (unrealistically, for some paths) assuming symmetric loss rates and solving the resulting quadratic equation. Assuming that losses are independent on the two paths $A \rightarrow B$ and $B \rightarrow A$, then the maximum absolute error in the one-way loss rate estimate occurs when all of the loss is in only one direction, resulting in an error equal to $\frac{loss_{two-way}}{2}$. Assuming a minimum loss rate of 2% ensures that, when choosing between two high-quality links, our loss rate estimate is within 50% of the true value. At large loss rates, highly asymmetric loss patterns cause this method to disregard a potentially good path because the reverse direction of that path has a high loss rate. However, the benefits of avoiding the high loss rate path seem to outweigh the cost of missing one good link, but we intend to remedy this problem soon by explicitly estimating one-way loss rates.

This method of throughput comparison using Equation 2 is not without faults. For instance, a slow but relatively unused bottleneck link on a path would almost never observe packet losses, and none of our probes would be lost. The equation we use would predict an unreasonably high bandwidth estimate. One way of tackling this would be to send pairs of probe packets to estimate an upper-bound on link bandwidth,be to use non-invasive probes to predict throughput [8].

Oscillating rapidly between multiple paths is harmful to applications sensitive to packet reordering or delay jitter. While each of the evaluation metrics applies some smoothing, this is not enough to avoid "flapping" between two nearly equal routes: RON routers therefore employ hysteresis. Based on an analysis of 5000 snapshots from a RON node's link-state table, we chose to apply a simple 5% hysteresis bonus to the "last good" route for the three metrics. This simple method appears to provide a reasonable trade-off between responsiveness to changes in the underlying paths and unnecessary route flapping.

4.2.3 Performance Database

To make good routing decisions RON needs to have detailed performance information. It is impractical to send large performance

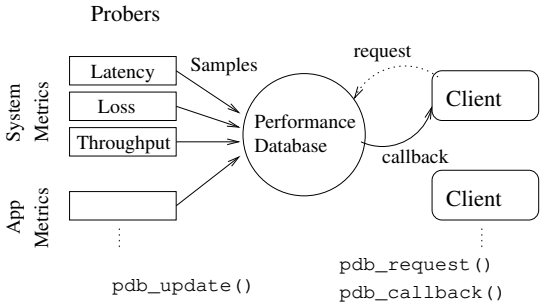

Figure 4: The RON performance database.

histories to all participants in the RON. A reliable performance repository must handle participants that crash, reboot, or rejoin the RON. Measurement data is often noisy, and different clients may have different ways in which they will use that data; for instance, an outage detector may want to know if any packets were successfully sent in the last 15 seconds, but a throughput improver may be interested in a longer-term packet loss average. Therefore, the system needs a flexible *summarization* mechanism.

To support these requirements, each RON node or local group of nodes uses a separate *performance database* to store samples (Figure 4). The database is a generalization of SPAND [24], supporting different data types and summarization mechanisms. Sources of information and clients that use it agree on a *class name* for the data and the meaning of the values inserted into the database. The probes insert data into the database by calling `pdb_update(class, src, dst, type, value)`. The `src` and `dst` fields contain the IP addresses of the sampled data.

4.3 Policy Routing

RON allows users or administrators to define the types of traffic allowed on particular network links. In traditional Internet policy routing, "type" is typically defined only by the packet's source and destination addresses [4]; RON generalizes this notion to include other information about the packet. RON separates policy routing into two components: *classification* and *routing table formation*.

When a packet enters the RON, it is classified and given a policy tag; this tag is used to perform lookups in the appropriate set of routing tables at each RON router. A separate set of routing tables is constructed for each policy by re-running the routing computation, removing the links disallowed by the corresponding policy. The routing computation computes the shortest path from the RON node to all other nodes, for any given metric. This construction applies to multi-hop or single-hop indirection; because a standard shortest-paths algorithm may not work for all metrics (specifically, TCP throughput), our implementation, described in Section 5.2, is specific to single-hop indirection. The result of running the table construction is the multi-level routing tables shown in Figure 5.

The construction of these routing tables and the production of packet tags is facilitated by the *policy classifier* component. The policy classifier produces the `permits` function that determines if a given policy is allowed to use a particular virtual link. It also provides a conduit-specific data classifier module that helps the RON node decide which policy is appropriate for the incoming packet. For example, if packets from a commercial site are not allowed to go across the Internet2 educational backbone, and we received a packet from such a site at an MIT RON node, the data classifier module would determine which policy to use and set the corresponding tag on the packet. Subsequent RON nodes examine the

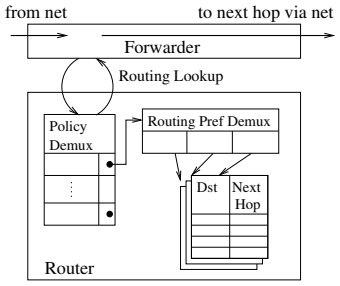

Figure 5: The forwarding control and data paths. The forwarder passes the packet header to the routing table, which performs three levels of lookups. The first level is based upon the policy type; the second is based upon the routing preference, and this lookup leads to a hash table of next hops indexed by destination.

Version	Hop Limit	Routing Flags
RON Source Address		
RON Destination Address		
Source port		Dest port
Flow ID		
Policy Tag		
Packet Type		

Figure 6: The RON packet header. The routing flags and flow ID are set by the input conduit. The packet type is a demux key indicating the appropriate conduit or protocol at the receiver.

policy tag instead of reclassifying the packet.

We have designed two policy mechanisms: *exclusive cliques* and *general policies*. In an exclusive clique, only data originating from and destined to other members of the clique may traverse inter-clique links. This mimics, for instance, the "educational only" policy on the Internet2 backbone. This policy classifier takes only a list of networks and subnet masks to match against.

Our general policy component is more powerful; it accepts a BPF-like packet matcher [14], and a list of links that are denied by this policy. It returns the first policy that matches a packet's fields to the stored BPF-based information. We do not currently address policy composition, although users may create composed versions of their own policies using the general policy component. In Section 7, we discuss possible abuse of AUPs and network transit policies.

4.4 Data Forwarding

The forwarder at each RON node examines every incoming packet to determine if it is destined for a local client or a remote destination. If it requires further delivery, the forwarder passes the RON packet header to the routing table, as shown in Figure 5.

The RON packet header is shown in Figure 6. It is inspired by the design of IPv6 [17]. Because RON needs to support more than simply IP-in-IP encapsulation, RON uses its own header. RON does not fragment packets, but does inform applications if they exceed the Maximum Transmission Unit (MTU). As in IPv6, applications must perform end-to-end path MTU discovery. RON also provides a policy tag that is interpreted by the forwarders to decide which network routing policies apply to the packet. If the packet is destined for the local node, the forwarder uses the packet type field to demultiplex the packet to the RON client.

If the packet's flow ID has a valid flow cache entry, the forwarder short-cuts the routing process with this entry. Otherwise, the routing table lookup occurs in three stages. The first stage examines the policy tag, and locates the proper routing preference table. There is one routing preference table for each known policy tag. Policy adherence supersedes all other routing preferences. Next, the lookup procedure examines the routing preference flags to find a compatible route selection metric for the packet. The flags are examined from the right, and the first flag understood by the router directs the packet to the next table. All RON routers understand the basic system metrics of latency, loss, and throughput; this provides a way for a shared RON environment to support users who may also have client-defined metrics, and still provide them with good default metrics. A lookup in the routing preference table leads to a hash table of next-hops based upon the destination RON node. The entry in the next-hop table is returned to the forwarder, which places the packet on the network destined for the next hop.

4.5 Bootstrap and Membership Management

In addition to allowing clients to define their own membership mechanisms, RON provides two system membership managers: a simple static membership mechanism that loads its peers from a file, and a dynamic announcement-based, soft-state membership protocol. To bootstrap the dynamic membership protocol, a new node needs to know the identity of at least one peer in the RON. The new node uses this neighbor to broadcast its existence using a *flooder*, which is a special RON client that implements a general-purpose, resilient flooding mechanism using a RON forwarder.

The main challenge in the dynamic membership protocol is to avoid confusing a path outage to a node from its having left the RON. Each node builds up and periodically (every five minutes on average in our implementation) floods to all other nodes its list of peer RON nodes. If a node has not heard about a peer in sixty minutes, it assumes that the peer is no longer participating in the RON. Observe that this mechanism allows two nodes in the same RON to have a non-functioning direct Internet path for long periods of time: as long as there is some path in the RON between the two nodes, neither will think that the other has left.

The overhead of broadcasting is minuscule compared to the traffic caused by active probing and routing updates, especially given the limited size of a RON. The resulting robustness is high: each node receives up to $N - 1$ copies of the peer list sent from a given other node. This redundancy causes a node to be deleted from another node's view of the RON only if the former node is genuinely partitioned for over an hour from *every other* node in the RON. A RON node will re-bootstrap from a well-known peer after a long partition.

5. Implementation

The RON system is implemented at user-level as a flexible set of C++ libraries to which user-level RON clients link. Each client can pick and choose the components that best suits its needs. To provide a specific example of a client using the RON library, we describe the implementation of *resilient IP forwarder*. This forwarder improves IP packet delivery without any modification to the transport protocols and applications running at end-nodes. The architecture of the resilient IP forwarder is shown in Figure 7.

RON uses UDP to forward data, since TCP's reliable byte-stream is mismatched to many RON clients, and IP-based encapsulation would restrict its application-specific forwarding capabilities. The RON core services run without special kernel support or elevated privileges. The conduit at the entry node is the client's gateway to

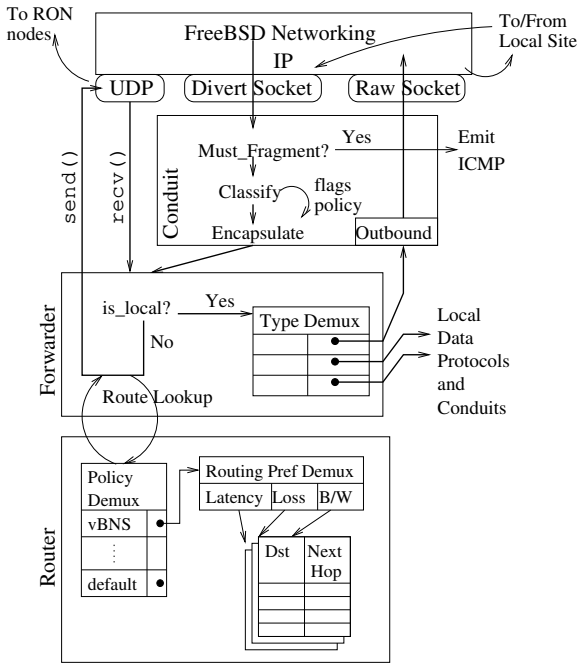

Figure 7: The resilient IP forwarder. Packets enter the system through FreeBSD's divert sockets. They are handed to the RON forwarder, which looks up the next hop. When the packet arrives at its destination, the conduit writes the packet out a local raw socket, where it re-enters IP processing.

the RON, and classifies every packet. This classification is client-specific; it labels the packet with information that decides what routing metric is later used to route the packet. As an example, a RON IP forwarder conduit might label DNS and HTTP traffic as "latency-sensitive" and FTP traffic as "throughput-intensive", which would cause the downstream RON forwarders to use the appropriate routing metric for each packet type. Non-entry RON nodes route the packet based only on the attached label and destination of the packet.

This design ensures that all client- and application-specific routing functions occur only at the entry and exit conduits, and that forwarding inside the RON is independent of client-specific logic. In addition to reducing the per-packet overhead at intermediate nodes, it also localizes the client-specific computation required on each packet. This means, for example, that a RON node implementing an IP forwarder may also participate in an overlay with a conferencing application, and help improve the reliability of data delivery for the conference. The conference node conduits implement the application-specific methods that label packets to tell the IP forwarder which routing metrics to use for conference packets.

5.1 The IP Forwarder

We implemented the resilient IP forwarder using FreeBSD's divert sockets to automatically send IP traffic over the RON, and emit it at the other end. The resilient IP forwarder provides classification, encapsulation, and decapsulation of IP packets through a special conduit called the `ip_conduit`. (Top of Figure 7).

5.2 Routers

RON routers implement the `router` virtual interface, which has only a single function call, `lookup(pkt *mypkt)`. The RON library provides a trivial static router, and a dynamic router that routes based upon different metric optimizations. The dynamic router is extensible by linking it with a different set of `metric descriptions`. Metric descriptions provide an evaluation function that returns the "score" of a link, and a list of metrics that the routing table needs to generate and propagate. The implementation's routing table creation is specific to single-hop indirection, which also eliminates the need for the flow cache. This algorithm fills in the multi-level routing table at the bottom of Figure 7:

```
MAKEROUTINGTABLE(POLICIES, METRICS, PEERS)
  foreach P in POLICIES
    foreach M in METRICS
      foreach Dest in PEERS
        foreach Hop in PEERS
          if P.permits(me, Hop)
          AND P.permits(Hop, Dest) {
            sc = M.eval(me, Hop, Dest);
            if sc > best_score {
              best_score = sc;
              next_hop = Hop;
            }
          }
        table[P][M][Dest] = next_hop;
```

We have implemented the clique classifier discussed in Section 4.3, and are implementing a general policy classifier. Both provide classifiers for the resilient IP forwarder.

5.3 Monitoring Virtual Links

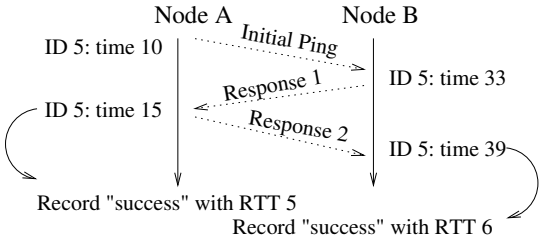

Figure 8: The active prober's probing mechanism. With three packets, both participants get an RTT sample, without requiring synchronized clocks.

Each RON node in an N-node RON monitors its $N-1$ virtual links using randomized periodic probes. The active prober component maintains a copy of a `peers` table with a `next_probe_time` field per peer. When this field expires, the prober sends a small UDP probe packet to the remote peer. Each probe packet has a random 64-bit ID. The process used by the prober is shown in Figure 8. When a node receives an initial probe request from a peer, it sends *response 1* to that peer, and resets its probe timer for that peer. When the originating node sees *response 1*, it sends *response 2* back to the peer, so that both sides get reachability and RTT information from 3 packets.

The probing protocol is implemented as a RON client, which communicates with performance database (implemented as a standalone application running on the Berkeley DB3 backend) using a simple UDP-based protocol.

6. Evaluation

RON reduced outages by a factor of 5 to 10, and routed around nearly all complete outages and nearly all periods of sustained high loss rates (30% or more).	6.2
RON takes 18 seconds, on average, to route around a failure and can do so in the face of a flooding attack.	6.2
Ron successfully routed around bad throughput failures, but did not substantially impact the median throughput, despite improving both loss and latency.	6.3
In 5% of sample periods, RON reduced the loss probability by 0.05 or more.	6.3
Single-hop route indirection captured the majority of benefits in our RON deployment.	6.4

Table 1: Major results from measurements of the RON testbed.

The goal of the RON system is to overcome path outages and performance failures, without introducing excessive overhead or worse new failure modes. In this section, we present an evaluation of how well RON meets these goals. We show that RONs are beneficial to applications, and that they can successfully route around path outages and performance failures. We evaluate the performance of the resilient IP forwarder, which uses RON for outage detection and loss, latency, and throughput optimization.

Our evaluation has three main parts to it. After describing our sixteen-site wide-area RON deployment and experimental methodology, we study RON's ability to detect outages and recover quickly from them. Second, we investigate performance failures and RON's ability to improve the loss rate, latency, and throughput of badly performing paths. Finally, we investigate two important aspects of RON's routing, showing the effectiveness of its one-intermediate-hop strategy compared to more general alternatives and the stability of RON-generated routes. Table 1 summarizes our key findings.

6.1 Methodology

Most of our results come from experiments with a wide-area RON deployed at several Internet sites. There are $N(N-1)$ different paths between the hosts in an N-site RON deployment. Table 2 shows the location of sites for our experiments. We analyze two distinct datasets—RON_1 with $N = 12$ nodes and 132 distinct paths, and RON_2 with $N = 16$ nodes and 240 distinct paths. In RON_1, `traceroute` data shows that 36 different AS's were traversed, with at least 74 distinct inter-AS links; for RON_2, 50 AS's and 118 inter-AS links were traversed. The $O(N^2)$ scaling of path diversity suggests that even small numbers of confederating hosts expose a large number of Internet paths to examination [18, 19]. We do not claim that our experiments and results are typical or representative of anything other than our deployment, but present and analyze them to demonstrate the kinds of gains one might realize with RONs.

Several of our host sites are Internet2-connected educational sites, and we found that this high-speed experimental network presents opportunities for path improvement not available in the public Internet. To demonstrate that our policy routing module works, and to make our measurements closer to what Internet hosts in general would observe, all the measurements reported herein were taken with a policy that prohibited sending traffic to or from commercial sites over the Internet2. Therefore, for instance, a packet could travel from Utah to Cornell to NYU, but not from Aros to Utah to NYU. This is consistent with the Acceptable Use

Name	Description
`Aros`	ISP in Salt Lake City, UT
`CCI`	.com in Salt Lake City, UT
`Cisco-MA`	.com in Waltham, MA
`* CMU`	Pittsburgh, PA
`* Cornell`	Ithaca, NY
`Lulea`	Lulea University, Sweden
`MA-Cable`	MediaOne Cable in Cambridge, MA
`* MIT`	Cambridge, MA
`CA-T1`	.com in Foster City, CA
`* NYU`	New York, NY
`OR-DSL`	DSL in Corvallis, OR
`* Utah`	Salt Lake City, UT
`VU-NL`	Vrije Univ., Amsterdam, Netherlands
Additional hosts used in dataset RON_2	
`NC-Cable`	MediaOne Cable in Durham, NC
`PDI`	.com in Palo Alto, CA
`Mazu`	.com in Boston, MA

Table 2: The hosts in our sixteen-node RON deployment, which we study in detail to determine the effectiveness of RON in practice. Asterisks indicate U.S. Universities on the Internet2 backbone. The two European universities, `Lulea` and `VU-NL` are classified as non-Internet2.

Policy of Internet2 that precludes commercial traffic. As a result, in our measurements, *all path improvements involving a commercial site came from only commercial links, never Internet2 links.*

The raw measurement data used in this paper consists of probe packets, throughput samples, and `traceroute` results. To probe, each RON node independently repeated the following steps:

1. Pick a random node, j.
2. Pick a probe-type from one of $\{direct, latency, loss\}$ using round-robin selection. Send a probe to node j.
3. Delay for a random interval.

This characterizes all $N(N-1)$ paths. For RON_1, we analyze 10.9 million packet departure and arrival times, collected over 64 hours between 21 Mar 2001 and 23 Mar 2001. This produces 2.6 million individual RTT, one-way loss, and jitter data samples, for which we calculated time-averaged samples averaged over a 30-minute duration [1] We also took 8855 throughput samples from 1 MByte bulk transfers, recording the time at which each power of two's worth of data was sent, and the duration of the transfer. RON_2 contains 85 hours from 7 May 2001 and 11 May 2001, consisting of 13.8 million arrival and departure times. Space constraints force us to present an analysis of only a subset of RON_2, but the general results from RON_2 are similar RON_1 [1].

Most of our hosts were Intel Celeron/733-based machines running FreeBSD, with 256MB RAM and 9GB of disk space. The processing capability of a host was never a bottleneck in any of our wide-area experiments. Our measurement data is available at `http://nms.lcs.mit.edu/ron/`.

6.2 Overcoming Path Outages

Precisely measuring a path outage is harder than one might think. One possibility is to define an outage as the length of time when no packets get through the path. As a yardstick, we could use ranges of time in which TCP implementations will time out and shut down a

[1] Some hosts came and went during the study, so the number of samples in the averages is not always as large as one might expect if all hosts were continually present. More details are available in the documentation accompanying our data.

connection; these vary from about 120 seconds (e.g., some Solaris versions) to 511 seconds (e.g., most BSDs). This gives us a metric for when batch applications will fail; from our own experience, we believe that most users have a far lower threshold for declaring their Internet connectivity "dead."

The problem with this small time-scale definition is that robustly measuring this and differentiating between a very high packet loss rate and true disconnections is hard. However, since distributed applications suffer in either case, we assert that it is operationally useful to define a path outage as the length of time over which the packet loss-rate is larger than some threshold. We define $outage(\tau, p) = 1$ if the observed packet loss rate averaged over an interval τ is larger than p on the path, and 0 otherwise. For values of τ on the order of several minutes, a measured value of p larger than 30% degrades TCP performance by orders of magnitude, by forcing TCP into frequent timeout-based retransmissions [16].

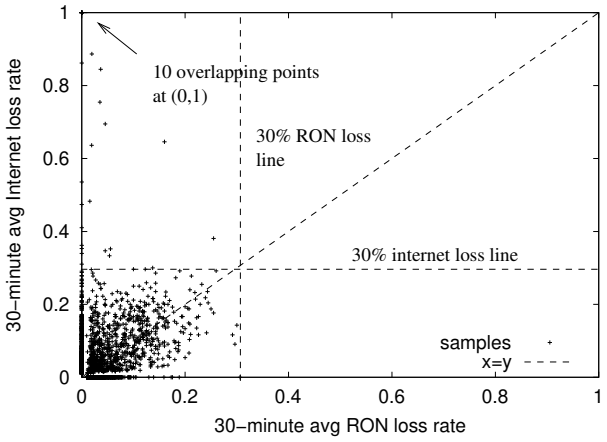

Figure 9: Packet loss rate averaged over 30-minute intervals for direct Internet paths vs. RON paths for the RON_1 dataset. There are 32 points above the $p = 0.3$ horizontal line, and 20 points above $p = 0.5$, including overlapping points. In contrast, RON's loss optimizing router avoided these failures and never experienced a 30-minute loss-rate larger than 30%.

How often is $outage(\tau, p) = 1$, and how often is RON able to route around these situations? Figure 9 shows a scatterplot of the improvement in loss-rate, averaged over $\tau = 1800$s achieved by RON. To identify outages consider the 32 points above the $p = 0.3$ line parallel to the horizontal axis, which signifies a condition bad enough to kill most applications. There are no points to the right of the $p = 0.3$ line parallel to the vertical axis. (The scatterplot conceals most of the data in the lower-left corner; see the CDF in Figure 11 too.)

The precise number of times $outage(\tau, p)$ was equal to 1 for $\tau = 1800$s is shown in Table 3. These are statistics obtained over 13,650 measured loss-rate averages obtained over 64 hours, spread over the 132 communicating paths in RON_1. We count a "RON Win" if the time-averaged loss rate on the Internet was $\geq p\%$ and the loss rate with RON was $< p\%$; "No Change" and "RON Loss" are analogously defined. We find that the number of complete communication outages was 10 in this dataset, which means that there were 10 instances when the sampled paths had a 100% loss rate. TCP would perceive significant problems at loss rates $\geq 30\%$, and there were several occurrences of these in our data. The numbers in this table are not the number of link or routing failures observed in the Internet across the sampled paths; the number of such failures could have been lower (e.g., multiple 30-minute averaged samples with a 100% loss rate may have been the result of the same problem) or higher (e.g., a time-averaged loss rate of 50% could have resulted from multiple link outages of a few minutes each).

We emphasize that Internet2 paths were never used to improve a non-Internet2 connection's performance. In fact, the vast majority of the problems corrected by RON involved only commercial Internet connections, as is shown (in brackets) by the number of outages when we remove *all* Internet2 paths from consideration.

Loss Rate	RON Win	No Change	RON Loss
10%	526 [517]	58 [51]	47 [45]
20%	142 [140]	4 [3]	15 [15]
30%	32 [32]	0	0
40%	23 [23]	0	0
50%	20 [20]	0	0
60%	19 [19]	0	0
70%	15 [15]	0	0
80%	14 [14]	0	0
90%	12 [12]	0	0
100%	10 [10]	0	0

Table 3: Outage data for RON_1. A "RON win" at $p\%$ means that the loss rate of the direct Internet path was $\geq p\%$ and the RON loss rate was $< p\%$. Numbers in brackets show the contribution to the total outage number after eliminating all the (typically more reliable) Internet2 paths, which reflects the public Internet better.

The numbers and percentage of outages for RON_2 (Table 4) are qualitatively similar to RON_1. RON_2 had 34,000 half-hour samples, about 2.5X more samples than RON_1.

Loss Rate	RON Win	No Change	Ron Loss
10%	557	165	113
20%	168	112	33
30%	131	84	18
40%	110	75	7
50%	106	69	7
60%	100	62	5
70%	93	57	1
80%	87	54	0
90%	85	48	2
100%	67	45	1

Table 4: Outage data for RON_2.

These results show that RON offers substantial improvements during a large fraction of outages, but is not infallible in picking the best path at lower outage rates when p is between 0 and 20%. However, in the RON_1 dataset, RON's outage detection and rerouting machinery was in fact able to successfully route around *all* the outage situations! This is especially revealing because it suggests that all the outages we encountered were not on "edge" links connecting the site to the Internet, but elsewhere where path diversity allowed RON to provide connectivity even when Internet routing could not. In RON_2, about 60% of the serious outage situations were overcome; our analysis showed that the ones that were not overcome were all cases where *no* other host in our RON had connectivity to the unreachable host.

One way to view our data is to observe that in RON_1 there are a total of 13,650/2 = 6,825 "path hours" represented. There were

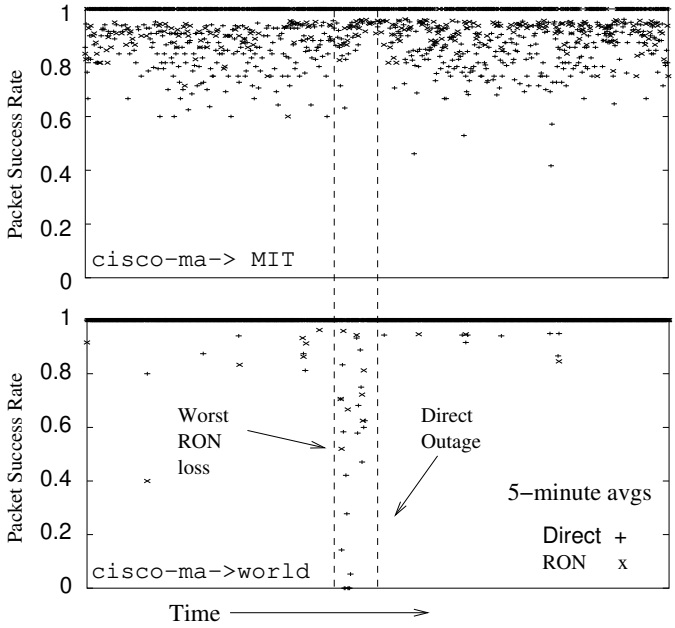

Figure 10: The lower figure shows a 10-minute outage from `Cisco-MA` to most places on the Internet (the few notches on the horizontal axis at the bottom). RON was able to route around the outage, because `Cisco-MA`'s packet loss rate to `MIT` was relatively unaffected during this outage.

5 "path-hours" of complete outage (100% loss rate); RON routed around all these. Similarly, RON_2 represents 17,000 path-hours with 56 path-hours of complete outage. RON was able to route around 33.5 path-hours of outage, but not in the remaining, mostly because no other paths through the RON reached a partitioned site.

We also encountered numerous outages of shorter duration (typically three or more minutes, consistent with BGP's detection and recovery time scales), and were able to recover around them faster. As one example of outage recovery, see Figure 10, which shows a 10-minute interval when no packets made it between `Cisco-MA` and most of the Internet (the few notches on the horizontal axis of the lower figure are important to note). In fact, among our RON sites, the only site to which it had any connectivity at this time was `MIT`. RON was able to detect this and successfully route packets between `Cisco-MA` and the commercial RON sites.

6.2.1 Overhead and Outage Detection Time

The implementation of the resilient IP forwarder adds about 220 microseconds of latency to packet delivery, and increases the memory bandwidth required to forward data. This overhead is primarily due to the use of divert sockets to obtain data. We do not envision our prototype being used on high-speed links, although it is capable of forwarding at up to 90 Mbps [1].

The frequency of routing and probing updates leads to a tradeoff between overhead and responsiveness to a path failure. Using the protocol shown in Figure 8 RON probes every other node every `PROBE_INTERVAL` seconds, plus a random jitter of up to $\frac{1}{3}$`PROBE_INTERVAL` extra seconds. Thus, the average time between two probes is $\frac{7}{6}$`PROBE_INTERVAL` seconds. If a probe is not returned after `PROBE_TIMEOUT` seconds, we consider it lost. A RON node sends a routing update to every other RON node every `ROUTING_INTERVAL` seconds on average. Our implementation uses

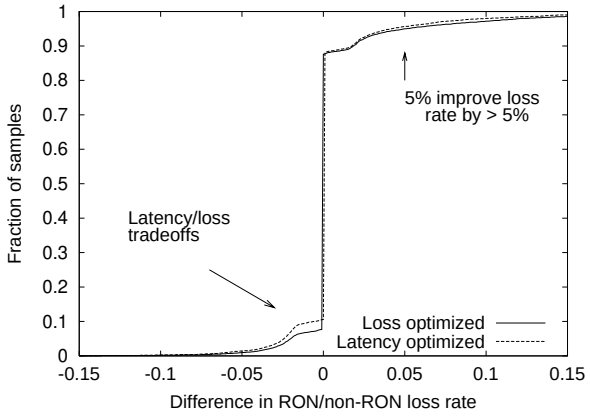

Figure 11: The cumulative distribution function (CDF) of the improvement in loss rate achieved by RON. The samples detect unidirectional loss, and are averaged over $\tau = 1800$s intervals.

the following values:

`PROBE_INTERVAL`	12 seconds
`PROBE_TIMEOUT`	3 seconds
`ROUTING_INTERVAL`	14 seconds

When a packet loss occurs, the next probe packet is sent immediately, up to a maximum of 4 "quick" probes. After 4 consecutive losses, we consider the path down. Our implementation detects failures in 12–25 seconds, with an average of 18 seconds.

Sending packets through another node introduces a potential new failure coupling between the communicating nodes and the indirect node that is not present in ordinary Internet communications. To avoid inducing outages through nodes crashing or going offline, RON must be responsive in detecting a failed peer. Recovering from a *remote* virtual link failure between the indirect host and the destination requires that the indirect host detect the virtual link failure and send a routing update. Assuming small transmission delays, a remote virtual link failure adds between 0 and `ROUTING_INTERVAL` seconds to the recovery times. In our implementation, a remote virtual link failure takes about 25 seconds to detect, during which time the intermediate node may or may not be able to re-route the packet. Packet losses can increase this duration; we require about $\frac{\texttt{ROUTING_INTERVAL}}{2(\texttt{success rate})}$ when packet losses occur. On extremely variable networks, the underlying network may heal before RON is able to. However, if some virtual links are good, RON's loss adaptation mechanisms will have already routed the connection around the bad ones.

The time to detect a failed path suggests that passive monitoring of in-use links will improve the single-virtual-link failure recovery case considerably, since the traffic flowing down the virtual link can be treated as "probes." Although recovery from multiple virtual link failures will still take on the order of twenty seconds, it is still significantly faster than recovery with wide-area inter-domain Internet routing.

RON probe packets are $S = 69$ bytes long. The probe traffic (see Figure 8) received at each node in an N-node RON is about $\frac{2S\times(N-1)}{\frac{7}{6}\texttt{PROBE_INTERVAL}}$ bytes/sec. RON routing traffic has $H = 60$ bytes of header information, plus $P = 20$ bytes of information to describe the path to each peer. Thus, each node sees $\frac{(N-1)\times(H+P(N-1))}{\texttt{ROUTING_INTERVAL}}$ bytes/sec of routing traffic. The traffic for several sizes of RONs, with our default intervals, is shown below in kilobits per second.

10 nodes	20 nodes	30 nodes	40 nodes	50 nodes
2.2Kbps	6.6Kbps	13.32Kbps	22.25Kbps	33Kbps

For a RON of $N = 50$ nodes, about 30 Kbps of active probing overhead allows recovery between 12 and 25 seconds. Combining probing and routing traffic, delaying updates to consolidate routing announcements, and sending updates only when virtual link properties change past some threshhold could all be used to reduce the amount of overhead traffic. However, the $O(N^2)$ growth in total traffic is caused by the need to guarantee that all $O(N^2)$ virtual links in the RON are monitored.

We believe that this overhead is reasonable for several classes of applications that require recovery from failures within several seconds. 30 Kbps is typically less than 10% of the bandwidth of today's "broadband" Internet links, and is the cost of achieving the benefits of fault recovery in RON. However, we are currently developing techniques that will preserve these recovery times without consuming as much bandwidth.[2]

6.2.2 Handling Packet Floods

To evaluate the effectiveness of RON in routing around a flood-induced outage, we conducted tests on the Utah Network Emulation Testbed, which has Intel PIII/600MHz machines on a quiescent 100Mbps switched Ethernet with Intel Etherexpress Pro/100 interfaces. The network topology emulated three hosts connected in a triangle, with 256 Kbps, 30 ms latency links between each pair. Indirect routing was possible through the third node, but the latencies made it less preferable.

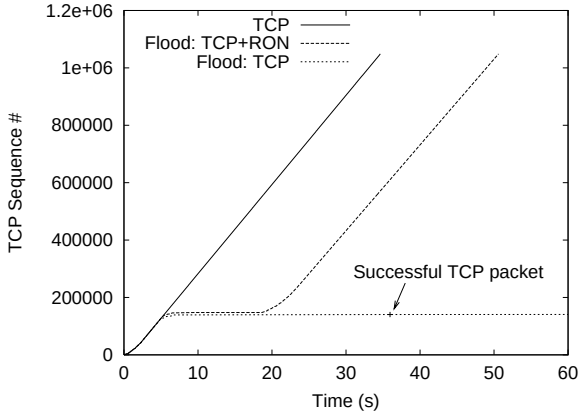

Figure 12: A TCP sequence trace at the receiver of a connection re-routed by RON during a flooding attack on its primary link. The 18 second recovery time is a combination of RON's link detection time and TCP's retransmission timers. Without RON, the connection is unable to get packets through at more than a crawl, though the link is still technically "working."

Figure 12 shows the TCP sequence traces of three bulk transfers taken at the receiver. The leftmost trace is an uninterrupted TCP transfer, which finishes in about 35 seconds. The middle trace shows the transfer running over RON, with a flooding attack beginning at 5 seconds. RON recovers relatively quickly, taking about 20 seconds to re-route the connection through the third node after which the connection proceeds normally. The rightmost trace (the horizontal dots) show the non-RON TCP connection in the presence of the flooding attack. Because TCP traffic was still getting through at a very slow rate, BGP would not have marked this link as down. Had it been able to do so, an analysis of the stability of BGP in congested networks suggests that BGP recovery times are at least an order of magnitude larger than RON's [25]—assuming that a BGP route change would not end up carrying the flooding traffic along the new links.

The flooding was only in one direction (the direction of the forward data traffic). RON still routed the returning ACK traffic along the flooded link—if BGP had declared the link "dead," it would have eliminated a perfectly usable link. This experiment also shows an advantage of the independent, client-specific nature of RON: By dealing only with its own "friendly" traffic, RON route changes can avoid re-routing flooding traffic in a way that a general BGP route change may not be able to.

6.3 Overcoming Performance Failures

In this section we analyze RON_1 for the improvements in loss-rate, latency, and throughput provided by RON.

6.3.1 Loss Rate

Figure 11 summarizes the observed loss rate results as a CDF: in RON_1, RON improved the loss rate by more than 0.05 a little more than 5% of the time. A 5% improvement in loss rate (in absolute terms) is substantial for applications that use TCP. Upon closer analysis, we found that the *outage detection* component of RON routing was instrumental in detecting bad situations promptly and in triggering a new path.

This figure also shows that RON routing does not always improve performance. There is a tiny, but noticeable, portion of the CDF to the left of the -0.05 region, showing that RON can make loss rates worse too. There are two reasons for this: First, RON uses a longer-term average of packet loss rate to determine its low-loss routes, and it may mispredict for a period after link loss rates change. Second, and more importantly, the RON router uses *bi-directional* information to optimize uni-directional loss rates.

This combination of substantial successes but observable failures of RON's loss optimizing router is shown in detail for one path in Figure 13. This figure examines three days of loss behavior between `MIT` and `CCI` averaged over 30-minute intervals. The top curve shows the packet loss rate across the Internet and RON for the `MIT` sender, while the bottom curve shows the same for the `CCI` sender. RON is a tremendous improvement in the former case, but not in the latter. In fact, in the latter, there are infrequent occurrences where RON makes things substantially worse. This problem arises because our current prober returns bi-directional loss estimates, which the optimizer uses to infer a uni-directional loss rate by assuming symmetric loss rates. The `MIT-CCI` Internet path appears to have a highly asymmetric loss rate. We did not think it would be too important to worry about loss-rate asymmetry in path selection, but this data clearly indicates that we should.

6.3.2 Latency

RON reduces communication latency in many cases, despite the additional hop and user-level encapsulation, as shown in Figure 14.[3] Figure 15 shows the same data as a scatterplot of the round-trip latencies achieved via RON against those found directly

[2] Our opinion is that this overhead is not excessive. Many of the packets on today's Internet are TCP acknowledgments, typically sent for every other TCP data segment. These "overhead" packets are necessary for reliability and congestion control; similarly, RON's active probes may be viewed as "overhead" that help achieve rapid recovery from failures.

[3] During outages, RON may find paths that we cannot compare to the direct path; we eliminated 113 out of 39683 samples due to long outages.

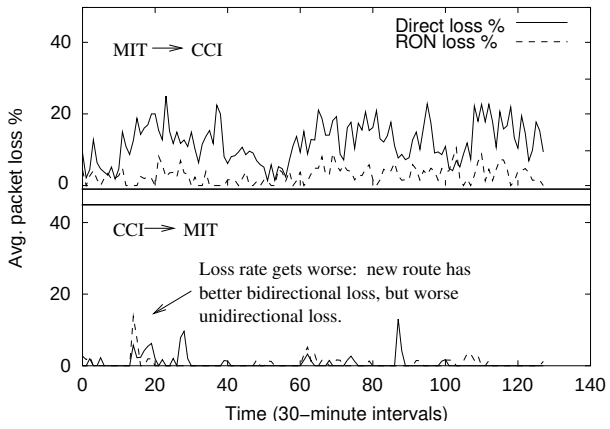

Figure 13: The benefits and limitations of bidirectional loss probes. The `MIT-CCI` link is heavily congested and plagued by outages. RON finds a path with much better *bi-directional* loss characteristics—but the direct path from `CCI` to `MIT` was sometimes better *in one direction* than the new RON path, even though the new RON path was superior overall. Note that the Y-axis goes only to about the 40% mark, not 100%.

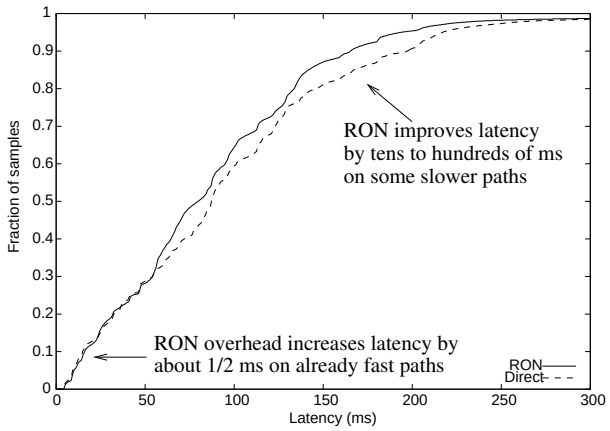

Figure 14: 5-minute average latencies over the direct Internet path and over RON, shown as a CDF.

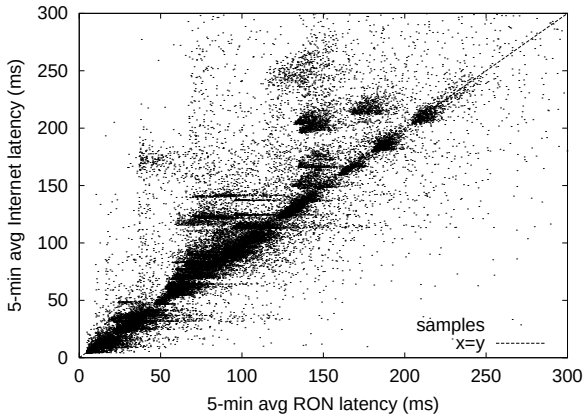

Figure 15: The same data as figure 14. Dots above the x=y line signify cases where the RON latency was lower than the direct Internet latency. The clustering and banding of the samples shows that the latency improvements shown in Figure 14 come from specific host-host pairs at different times.

on the Internet, averaged in 5-minute blocks. We also find that the the points in the scatterplot appear in clustered bands, showing the improvements achieved on different node pairs at different times.

6.3.3 TCP Throughput

RON also improves TCP throughput between communicating nodes in many cases. RON's throughput-optimizing router does not attempt to detect or change routes to obtain small changes in throughput, since underlying Internet throughput is not particularly stable on most paths; rather, it seeks to obtain at least a 50% improvement in throughput on a re-routed path.

To compare the RON path used by throughput-optimizing RON routers to the direct Internet path, we took four sequential throughput samples—two with RON and two without—and compared the ratio of the average throughput achieved by RON to the average throughput achieved directly over the Internet. Figure 16 shows the distribution of these ratios. Out of 2035 paired quartets of throughput samples, only 1% received less than 50% of the throughput with RON, while 5% of the samples doubled their throughput. In fact, 2% of the samples increased their throughput by more than a factor of five, and 9 samples improved by a factor of 10 during periods of intermittent Internet connectivity failures.

6.4 RON Routing Behavior

We instrumented a RON node to output its link-state routing table roughly every 14 seconds, with a randomized jitter to avoid periodic effects. We analyzed a 16-hour trace of the table containing 5616 individual snapshots. The trace showed the routing information used to determine 876,096 different point-to-point routes over the period of the capture.

6.4.1 RON Path Lengths

Our outage results show that RON's single-hop indirection worked well for avoiding problematic paths. If there is a problem with the direct Internet path between two nodes s and t, it is often because some link $l(s,t)$ between them is highly congested or is not working. As long as there is even one RON node, u, such that the Internet path between s and u, and the Internet path between u and t, do not go through $l(s,t)$, then RON's single-hop indirection will suffice. Of course, if all paths from s to the other RON nodes traverse $l(s,t)$, or if the paths from every RON node to t traverse $l(s,t)$, then the RON cannot overcome the outage of $l(s,t)$. However, if the intersection of the set of nodes in the RON reachable from s without traversing $l(s,t)$, and the set of nodes in the RON from which t can be reached without traversing $l(s,t)$, is not null, then single-hop indirection will overcome this failure.

Single-hop indirection suffices for a latency-optimizing RON too. We found this by comparing single-hop indirection with a general shortest-paths algorithm on the link-state trace[4]. The direct Internet path provided the best average latency about 48.8% of the time. In addition, the remaining 51.2% of the time when RON's overlay routing was involved, the shortest path involved only one intermediate node essentially all the time: about 98%.

The following simple (and idealized) model provides an expla-

[4]We did not analyze the effects of the no-Internet2 policy here, and only considered latency optimization *without* outage avoidance.

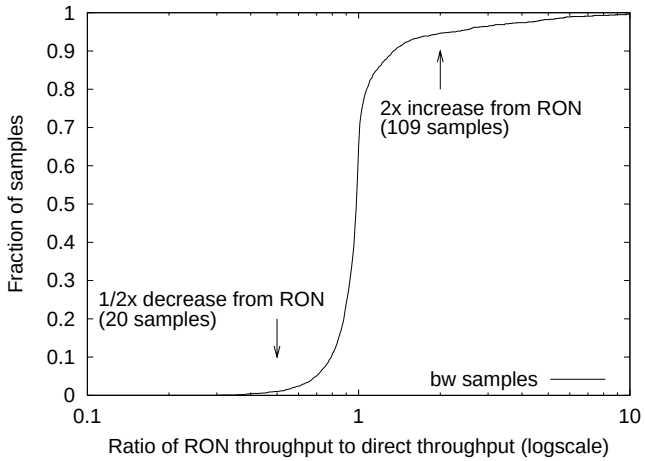

Figure 16: CDF of the *ratio* of throughput achieved via RON to that achieved directly via the Internet. 2035 ratios shown. RON markedly improved throughput in near-outage conditions, but did not change the median throughput.

nation. Consider a source node s and a destination node t in a RON with R other nodes. Denote by p_s the probability that the lowest-latency path between node s and another node in the RON is the direct Internet path between them. Similarly, denote by p_i the probability that the lowest-latency path between node $i(\neq s)$ in the RON and t is the direct link connecting them. We show that even small values of these probabilities, under independence assumptions (which are justufiable if RON nodes are in different AS's), lead to at most one intermediate hop providing lowest-latency paths most of the time.

The probability that a single-intermediate RON path is optimal (for latency) given that the direct path is not optimal is given by $1 - \Pi_{i=1}^{R}(1 - p_s p_i)$, since this can only happen if *none* of the other R nodes has a direct-link shortest path from s and to t. This implies that the probability that either the direct path, or a single-hop intermediate path is the optimal path between s and t is given by $1 - (1 - p_s)\Pi_{i=1}^{R}(1 - p_s p_i)$. In our case, p_s and p_i are both around 0.5, and R is 10 or more, making this probability close to 1. In fact, even relatively small values of p_s and p_i cause this to happen; if $p_s \approx p_i = p \approx c/\sqrt{R}$, then the optimal path has either the direct path or has one intermediate RON hop with probability at least $1 - (1 - p^2)^{R+1} = 1 - (1 - c/R)^{R+1} \approx 1 - e^{-c}$, which can be made arbitrarily close to 1 for suitable c.

6.4.2 RON Route Stability

As a dynamic, measurement-based routing system, RON creates the potential for instability or route flapping. We simulated RON's path selection algorithms on the link-state trace to investigate route stability. The "Changes" column of Table 5 shows the number of path changes that occurred as a function of the hysteresis before triggering a path change. The other columns show the *persistence* of each RON route, obtained by calculating the number of consecutive samples over which the route remained unchanged. The average time between samples was 14 seconds.

The median run-length with the 5% hysteresis we implemented was 5, about 70 seconds. The frequency of our table traces does not preclude undetected interim changes, especially with shorter run-lengths, but the longer run-lengths indicate reasonable stability

Method	# Changes	Avg	Med	Max
No Hyst	26205	19.3	3	4607
Hyst-5%	21253	24	5	3438
Hyst-10%	9436	49	10	4607
Hyst-25%	4557	94	17	5136
Hyst-50%	2446	138	25	4703
Random process	260,000	2	1	<16

Table 5: Number of path changes and run-lengths of routing persistence for different hysteresis values.

with good probability. 5% and 10% hysteresis values appear to provide a good trade-off between stability and responsiveness. The "random process" row shows the expected number of route changes if the routes were flapping at random.

6.4.3 Application-specific Path Selection

There are situations where RON's latency-, loss-, and throughput-optimizing routers pick different paths. As an example, RON was able to find a lower-latency path between `CCI` and `MA-Cable` that had nearly three times the loss rate:

	`CCI` → `MA-Cable`	`MIT` → `Cisco-MA`
Direct	112ms, 0.77% loss	38ms, 12.1% loss
Lat-Opt	100ms, 2.9% loss	43ms, 10.5% loss
Loss-Opt	114ms, 0.6% loss	189ms, 3.62%

In contrast, between `MIT` and `Cisco-MA`, RON's latency optimizer made the latency *worse* because the outage detector was triggered frequently. The loss-optimizing router reduced the loss rate significantly, but at the cost of a five-fold increase in latency. The existence of these trade-offs (although we do not know how frequently they occur in the global Internet), and the lack of a single, obvious, optimal path reinforces our belief that a flexible, application-informed routing system can benefit applications.

7. Discussion

This section discusses three common criticisms of RONs relating to routing policy, scalability, and operation across Network Address Translators (NATs) and firewalls.

RON creates the possibility of misuse or violation of AUPs and BGP transit policies. RON provides a flexible policy mechanism that allows users to implement *better* network policies than those permitted by BGP. Because RONs are deployed between small groups of cooperating entities, they cannot be used to find "backdoors" into networks without the permission of an authorized user of that network. Users who violate network policy today are dealt with at a human level, and that should remain unchanged with RON deployment. On the positive side, if an Overlay ISP buys bandwidth from traditional ISPs and routes data using a RON, it can express and enforce sophisticated policy routes.

Preventing misuse of an established RON requires relatively straightforward cryptographic authentication and access controls that we have not yet implemented. However, participating in a RON implies trust that the other participants will forward your data correctly and will not abuse your forwarding. We believe that when RONs are small enough that the consequences of forming a RON with a malicious entity can be resolved at an administrative (rather than technological) level, but the widespread deployment of RONs may require other mechanisms to detect misbehaving RON peers.

A potential criticism of RON is that while they may work well in limited settings, they may not when they start to become widely

popular. We have shown in this paper that the RON design trades scalability for improved reliability because it enables the aggressive maintenance and exploration of alternate paths, facilitating prompt outage detection and recovery. If RONs become popular, we do not expect this fundamental trade-off to change, but we expect to see many RONs co-existing and competing on Internet paths. Understanding the interactions between them and investigating routing stability in an Internet with many RONs is an area for future work.

NATs cause two problems for RON. The first problem relates to naming—a host behind a NAT does not usually have a globally reachable IP address or DNS name. One way to solve this problem is to cache a "reply to" address/port pair for the host behind a NAT; when a RON node receives a packet from another whose IP address (not globally visible) is A, but whose RON address says it comes from the globally visible address B and port p, the receiving node creates an entry that says "To reach RON node A, send to B at port p." The RON probe traffic will suffice to keep the NAT's port-mapping entry alive. The second problem posed by NATs is that if two hosts are both behind NATs, they may not be able to communicate directly. RON will perceive this as an outage, and attempt to route around it. This will establish connectivity between the said hosts, but may result in sub-optimal routing.

8. Conclusion

In this paper, we have shown that a Resilient Overlay Network (RON) can greatly improve the reliability of Internet packet delivery by detecting and recovering from outages and path failures quicker than current inter-domain routing protocols. A RON works by deploying nodes in different Internet routing domains, which cooperatively route packets for each other. Each RON is an application-layer overlay; nodes in a RON monitor the quality of the underlying Internet between themselves and use this information to route packets according to application-specified routing metrics either via a direct Internet path or via other RON nodes.

We studied the benefits of RON by evaluating two datasets collected from our wide-area deployment. We found that RON was able to overcome between 60% and 100% of the several hundred significant observed outages. Our implementation takes 18 seconds, on average, to detect and recover from a fault, significantly better than the several minutes taken by BGP-4. RONs also improve the loss rate, latency, and TCP throughput of badly performing Internet paths. A noteworthy finding from our experiments is that forwarding packets via at most one intermediate RON node is sufficient to recover from faults and also obtain low-latency paths.

These results suggest that RON is a good platform on which a variety of resilient Internet applications may be developed. In addition, the RON platform will allow researchers developing routing protocols to evaluate their schemes under real-world conditions.

Acknowledgments

We are grateful to all the people and institutions who hosted RON sites: M. Biesele, K. Bot, M. Chang, M. Cutler, J. Finley, J. Jung, L. Larzon, D. Mazieres, N. Miller, C. North, C. Pohlabel, R. van Renesse, M. Sanders, S. Shira, and E. Sit. We thank Jay Lepreau and his group for the Utah Network Emulation Testbed. We also thank David Karger for several useful discussions on RON. John Jannotti, David Tennenhouse (our shepherd), and the SOSP reviewers provided useful comments that improved this paper.

9. References

[1] ANDERSEN, D. G. Resilient Overlay Networks. Master's thesis, Massachusetts Institute of Technology, May 2001.

[2] BALAKRISHNAN, H., SESHAN, S., STEMM, M., AND KATZ, R. Analyzing Stability in Wide-Area Network Performance. In *Proc. ACM SIGMETRICS* (Seattle, WA, June 1997), pp. 2–12.

[3] CHANDRA, B., DAHLIN, M., GAO, L., AND NAYATE, A. End-to-end WAN Service Availability. In *Proc. 3rd USITS* (San Francisco, CA, 2001), pp. 97–108.

[4] CLARK, D. *Policy Routing in Internet Protocols*. Internet Engineering Task Force, May 1989. RFC 1102.

[5] COLLINS, A. The Detour Framework for Packet Rerouting. Master's thesis, University of Washington, Oct. 1998.

[6] ERIKSSON, H. Mbone: The Multicast Backbone. *Communications of the ACM 37*, 8 (1994), 54–60.

[7] FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. Equation-Based Congestion Control for Unicast Applications. In *Proc. ACM SIGCOMM* (Stockholm, Sweden, Sept. 2000), pp. 43–54.

[8] GOYAL, M., GUERIN, R., AND RAJAN, R. Predicting TCP Throughput From Non-invasive Data. Unpublished manuscript `http://www.seas.upenn.edu:8080/~guerin/publications/TCP_model.pdf`.

[9] GUARDINI, I., FASANO, P., AND GIRARDI, G. IPv6 Operational Experience within the 6bone. In *Proc. Internet Society (INET) Conf.* (Yokohama, Japan, July 2000). `http://www.isoc.org/inet2000/cdproceedings/1e/1e_1.htm`.

[10] HAGENS, R., HALL, N., AND ROSE, M. *Use of the Internet as a Subnetwork for Experimentation with the OSI Network Layer*. Internet Engineering Task Force, Feb 1989. RFC 1070.

[11] KHANNA, A., AND ZINKY, J. The Revised ARPANET Routing Metric. In *Proc. ACM SIGCOMM* (Austin, TX, Sept. 1989), pp. 45–56.

[12] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed Internet Routing Convergence. In *Proc. ACM SIGCOMM* (Stockholm, Sweden, September 2000), pp. 175–187.

[13] LABOVITZ, C., AHUJA, A., AND JAHANIAN, F. Experimental study of Internet stability and wide-area network failures. In *Proc. FTCS* (Madison, WI, June 1999).

[14] MCCANNE, S., AND JACOBSON, V. The BSD Packet Filter: A New Architecture for User-Level Packet Capture. In *Proc. Winter '93 USENIX Conference* (San Diego, CA, Jan. 1993).

[15] The North American Network Operators' Group mailing list archive. `http://www.cctec.com/maillists/nanog/`.

[16] PADHYE, J., FIROIU, V., TOWSLEY, D., AND KUROSE, J. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM* (Vancouver, Canada, September 1998), pp. 303–323.

[17] PARTRIDGE, C. *Using the Flow Label Field in IPv6*. Internet Engineering Task Force, 1995. RFC 1809.

[18] PAXSON, V. End-to-End Routing Behavior in the Internet. In *Proc. ACM SIGCOMM '96* (Stanford, CA, Aug. 1996), pp. 25–38.

[19] PAXSON, V. End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM* (Cannes, France, Sept. 1997), pp. 139–152.

[20] POSTEL, J. B. *Transmission Control Protocol*. Internet Engineering Task Force, September 1981. RFC 793.

[21] REKHTER, Y., AND LI, T. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, 1995. RFC 1771.

[22] SAVAGE, S., ANDERSON, T., ET AL. Detour: A Case for Informed Internet Routing and Transport. *IEEE Micro 19*, 1 (Jan. 1999), 50–59.

[23] SAVAGE, S., COLLINS, A., HOFFMAN, E., SNELL, J., AND ANDERSON, T. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM* (Boston, MA, 1999), pp. 289–299.

[24] SESHAN, S., STEMM, M., AND KATZ, R. H. SPAND: Shared Passive Network Performance Discovery. In *Proc. 1st USITS* (Monterey, CA, December 1997), pp. 135–146.

[25] SHAIKH, A., KALAMPOUKAS, L., VARMA, A., AND DUBE, R. Routing Stability in Congested Networks: Experimentation and Analysis. In *Proc. ACM SIGCOMM* (Stockholm, Sweden, 2000), pp. 163–174.

[26] TOUCH, J., AND HOTZ, S. The X-Bone. In *Proc. 3rd Global Internet Mini-Conference in conjunction with IEEE Globecom* (Sydney, Australia, Nov. 1998), pp. 75–83.