# The Case for Resilient Overlay Networks

*David G. Andersen*
*Hari Balakrishnan, M. Frans Kaashoek, Robert Morris*

**Slide 0**

## MIT Laboratory for Computer Science

May 2001

`http://nms.lcs.mit.edu/ron/`

Hope everyone's not about to miss a plane to listen to me talk today, but thanks for sticking around.

Today, I'm going talk a bit about the reliability of the Internet. I'll show you some reasons and data that show that it's got some problems, and I'll present an architecture we've designed to overcome some of those problems. This system is called Resilient Overlay Networks, and it's work that I've been doing with Hari Balakrishnan, Frans Kaashoek, and Robert Morris.

**Slide 1**

## Idea

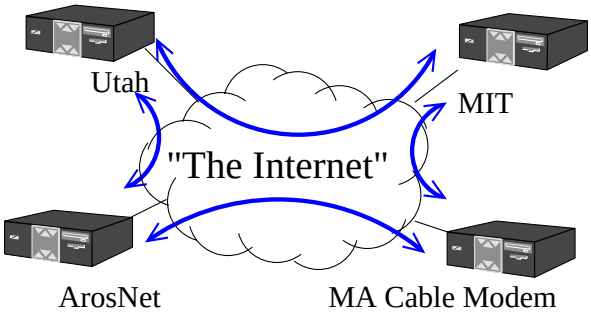
- Scalability may get in the way of deploying services and protocols that may not scale
- So do cool things in small overlays
  - More aggressive
  - Things that're less efficient

The Internet is huge. By definition, it has to scale, but this need can get in the way of deploying network services and protocols that could do very cool things, but might not scale too well.

Overlay networks have been used for years to test out new protocols. I think that a limited-size overlay is also a great venue in which to deploy less scalable protocols. For the rest of the talk, I'm only going to think about networks of, say, 50 nodes or less, and see some things we can do with them.

## Routing around Internet Failures

**Slide 2**

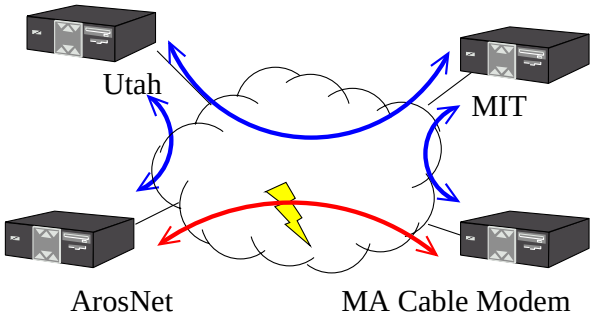



People expect all-to-all communication...

At a conceptual level, the Internet is a simple beast. Everyone connects to the “Great Internet Cloud.” We put packets in one end, and they pop out the other. This expectation is a good thing; it provides a nice, simple abstraction for using the Internet.

## Routing around Internet Failures

**Slide 3**




Which the Internet can't always provide.

Of course, this is a big lie, and I think we've all used the Internet enough to know that it's not available 24x7.

## Internet Failures

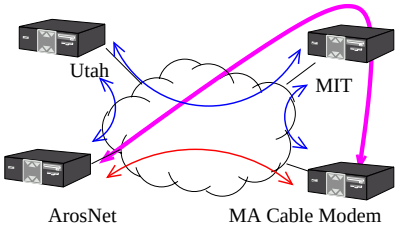
**Slide 4**

- Physical link failures (backhoes)
- Excess Traffic (14-year-olds...)
- Router misconfiguration
- The list goes on...

There are a lot of things that can cause a failure between two hosts on the Internet, of course. A backhoe operator in Ohio with a grudge against the east coast can cut off MIT from the rest of the world. Denial of Service attacks are easy to launch and can take out even substantial links. Of course, operators *never* misconfigure routers, and the list of things that can go wrong is really only limited by our imagination, and the complexity of the systems we design.

## Routing around Internet Failures

**Slide 5**




But we think cooperating hosts can do better...

Multiparty videoconferences

Overlay Internet Service, Companies with VPNs, etc.

➔Do we need this?

But we believe that *hosts* can cooperate to route traffic through each other, thus giving themselves better service than the Internet could.

Now, I've been presenting all this doom and gloom, and by this point, you might be scratching your head and thinking, "But Dave, the Internet is designed to route around failures."

Of course, you're completely right. It is, and it does. But it can't route around everything - like denial of service attacks - and you might be surprised at how slowly it routes around some things.

## The Internet Recovers Slowly

**Slide 6**

[Labovitz 00]:
"Internet ... routing convergence is an order of magnitude slower than previously thought."

- 3 minute average recovery time
  15 minute max for *simple* failures
- Our tests: Indirect routing had 5x-10x fewer outages [Sneak preview]

Surprisingly enough, it wasn't until last year that we had some concrete numbers about how long BGP really takes to route around very simple failures - little burps, not backhoe catastrophies. Labovitz found that (and I'm going to quote it because I love it), "Internet routing convergence is an order of magnitude slower than previously thought."

Putting this in concrete numbers, he found that the average recovery time for a simple failure is about three minutes, and can take as long as 15 minutes. And let's not even talk about what happens when routers get mis-configured or melt down...

We found in some later tests, that aren't reported in this paper, that we reduced the outages seen by about a factor of five to ten. That's just a teaser, of course, and I'll tell you how we plan to do it in a bit, but first, let's look at some of the reasons the Internet can't always ... deliver.

## Internet Trade-offs

**Slide 7**

- Scalability and heterogeneity:
  ➔Slow Recovery
  (Is this a fundamental trade-off?)
- RON takes a different approach:
  Fast recovery for small groups in an overlay
- Exploit redundancy in the Internet

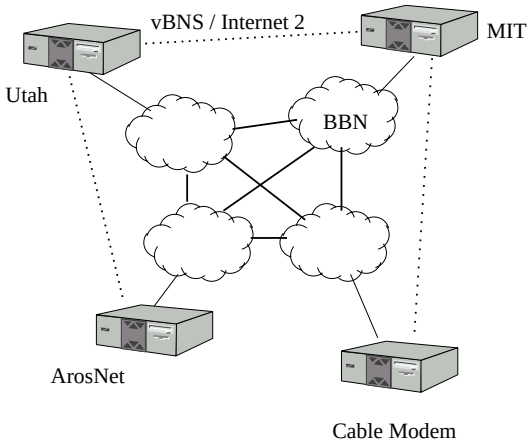
Why does the 'Net behave that way? I believe that at some fundamental level, we've made a decision to trade fast recovery for the ability to scale and support a huge, diverse mesh of heterogenous networks.

That's the Internet. We take a different tack: Fast recovery for a small group of communicating applications.

To do this, we exploit some of that richness in the Internet, in this case, the redundancy in the underlying Internet.

## A More Realistic Picture

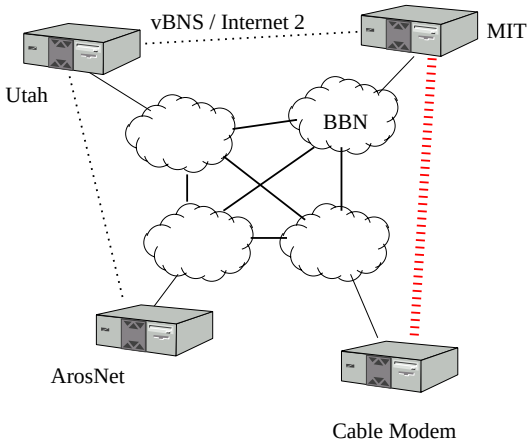
**Slide 8**




With that in mind, let's look at a more realistic picture of how sites connect to the Internet. It's not a matter of one connection to "the great cloud," but more a funky, interesting mesh, repelete with opportunities to be clever. Taking MIT as an example, we connect to the commercial Internet via a company called BBN, but we also have an Internet2 connection that hooks us up with Utah, Berkeley, and most other universities in the US. To top it off, we have a private link to the local cable modem provider, so our students can pretend to do their homework more quickly.

## Hidden Links

**Slide 9**




But that's a private link. Not all redundant links are, but this points out another fault with how Internet routing works these days: MIT isn't capable of expressing the policy, "Let MIT students visiting Utah (er, that would be me, when I visit my family) access their home machine through MIT if MediaOne goes on the fritz." This is a pretty reasonable policy, since I'm an MIT student, but it's astoundingly hard to actually make that happen. We'd like to enable more flexible policies, while we're at it.

## Policy and AUPs

- WAN routing policy expression is a sledgehammer
- But we need policy control (Internet2, etc)

**Slide 10**

✘ RON could violate AUPs

✔ But RON can provide flexible policies

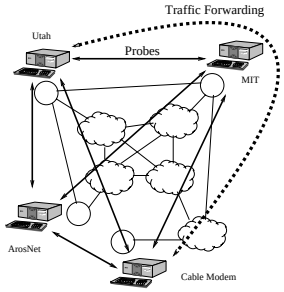
- More complex routing decisions
- Multiple routing tables
- Deeper packet inspection

Yesterday, rebooting was the sledgehammer. I'll continue with the theme: WAN routing policy expression is pretty much a sledgehammer. It's way too inflexible. However, operators definitely need control over routing policy—a lot of the mechanisms in BGP implementations deal with policy. RONs create the possibility of violating acceptable use policies; I've used my RON to shuttle music from ArosNet to MIT over the Internet2, which probably wasn't what people had in mind...

but at the same time, RON can provide far more flexible and powerful policies than BGP-based routing systems, because we can take the time to have multiple routing tables for different policies, look deeper inside packets to classify them for policies, and so on.

**Slide 11**

## The Approach

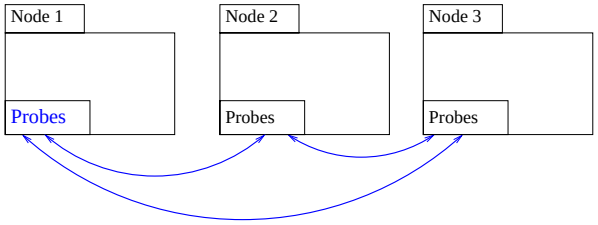



- Measure *all* inter-node paths
- Exchange routing information
- Route along best path

The general approach we advocate is shown here. We get measurements of some properties of the paths between all the nodes. The nodes then exchange this information with each other so that they can do some routing based upon this information. Once they've exchanged this information, they then route the data over this path, if it's better than the default Internet path. Let's peek at the details of how we do these things.

## Architecture

**Slide 12**

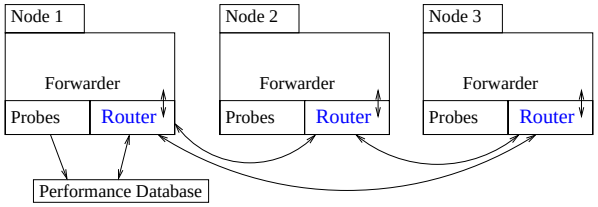



- Probe between nodes to establish best route
  - Active, application probing of $N^2$ paths
  - Passive measurements

To measure the paths between the nodes, we use *active* probing – we send packets back and forth, because we can't depend on there bein gactual traffic between these nodes (especially if the path stinks and we're avoiding it!). We send the probes between *all* of the hosts. This is another of those $N^2$ things that doesn't scale, but remember, for the next 10 minutes, we don't care about scalability. I'm a worse coder than Steve Gribble, so I don't stand a chance of having bug-free code; probing at the application layer lets me survive it when some other nodes core-dump or get rebooted. We look at loss, latency, and some day maybe throughput and application-derived or passive measurements.

## Architecture

**Slide 13**



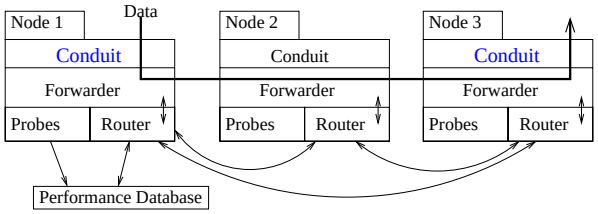

- Probe between nodes to establish best route
- Link-state routing protocol between nodes

We use a link-state routing protocol to let everyone know our estimates of the path properties. The routing runs constantly, makes sure that everyone knows the state of *all* of the links in the Network. Once we know this information, we can make sensible decisions about the paths over which our data should travel.

Of course, RON gives us some cool ways to route around network policies if we want to, and I'll confess that it's kind of convenient to be able to do so. But the small size of RON also lets us enforce really detailed policies, like "Only DNS traffic can go over the Internet2, everything else has to use the commercial Internet."

## Architecture

**Slide 14**




- Probe between nodes to establish best route
- Link-state routing protocol between nodes
- Data handled by application-specific conduit
  Forwarded in UDP

Finally, we want to be able to support a variety of applications with our infrastructure, but we don't want the basic services to be tied to any one service. Therefore, all application-specific processing is handled by the conduit, which is the term for any gateway between an application and the RON.

All RON packets use UDP, because we want to ensure that the system can be used for anything from a router-level application, to an unprivileged user program. TCP obviously wouldn't do, because its strict reliability semantics conflict with our needs.

## Conduits: Gateways into the RON

**Slide 15**

- IP off the wire conduit
  (Used for evaluation)
- Emulates `sendto` and `recvfrom`
- The application itself
- Interface: `send`, `register`, `callback`

For the slides I've got coming up, the conduit simply took IP packets off of the wire and stuffed them into RON packets, but it's equally valid to have a conduit that emulates the "sendto" and "recvfrom" system calls, or even an application that *is* a RON conduit itself. The conduits have a totally simple interface into RON, consisting of a "send" function, a "register for packets of a particular type" function, and a callback when packets arrive at the RON that are destined for the conduit.

Note that data is both inserted by the conduit, and decapsulated at the end.

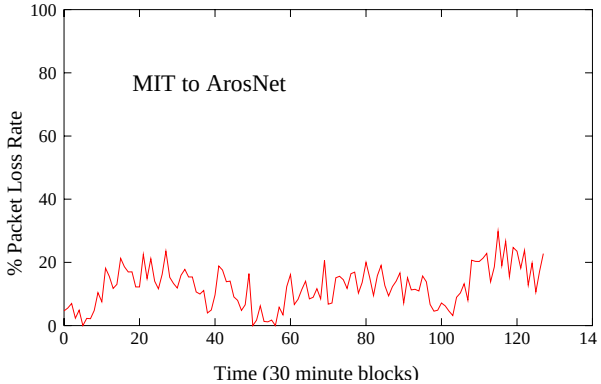
## *Preliminary* Investigation

**Slide 16**

- Tested between 4 hosts
- 70 hours of ping-style measurements
- Looks promising, but we suspected it would.

# From MIT to ArosNet on the Internet

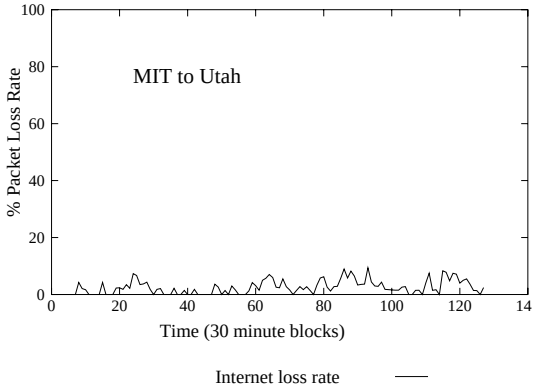
**Slide 17**




The direct Internet path is quite bad.

# From MIT to Utah on the Internet

**Slide 18**

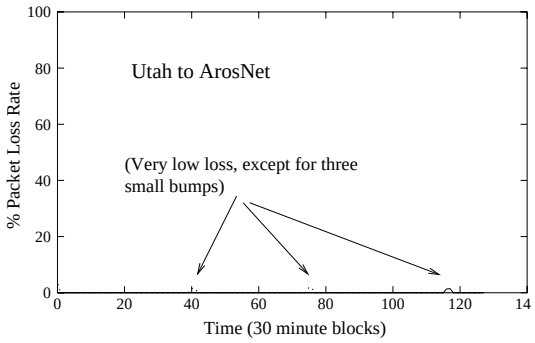



Internet loss rate

But the path from MIT to Utah looks good...

# From Utah to ArosNet on the Internet

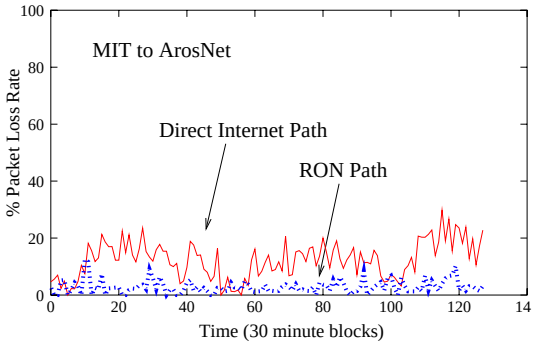
**Slide 19**




As does the path from Utah to ArosNet...

## From MIT to ArosNet with RON

**Slide 20**




Putting them together...

## Other results

**Slide 21**

- Big latency reduction between MIT - ArosNet
- Big latency reduction between Cable Modem - Utah
- Real results are hiding in a thesis

## Some Research Questions

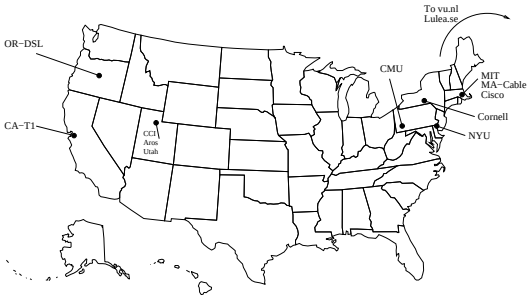
Slide 22

- Is this a stupid idea?
- How many intermediate hops?
- How do we best choose routes?
- How frequently do we probe?
- What routing policies can we express?
- **How do RONs interact?**

`http://nms.lcs.mit.edu/ron/`

# Status

**Slide 23**




http://nms.lcs.mit.edu/ron/