Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.829 Fall 2002 | Overlay Routing in the Internet | November 12, 2002

1 Introduction

In this lecture, we explore the capabilities of overlay networks for both general communication and data distribution. We have seen how the IP substrate facilitates point-to-point communication between end hosts. However, this communication infrastructure is not without its own set of problems. In particular, we have witnessed that the wide-area routing protocol, BGP, is slow to converge and react to failures. Second, the hierarchical addressing proposed by IP becomes a problem when hosts move, since end-hosts are traditionally named by IP addresses. Other applications, such as multicast, could benefit from general communication abstraction.

In this lecture, we'll talk about the use of overlay routing in the Internet for two specific purposes. The first is a communication overlay (of which one example is Resilient Overlay Networks), and the second is an overlay for data dissemination (e.g., I3). Figure 1 summarizes this taxonomy. We'll discuss the problems solved by each of these systems, as well as the problems presented by each of them. Although our discussion is motivated by these two specific systems, we'll try to keep the discussion general.

One way to think about overlay routing is in terms of the communication substrate that lies beneath it at the IP layer. At the most basic level, IP enables a conversation between two end hosts by routing data through links and routers. Overlay routing, on the other hand, considers a conversation between two IP end hosts to be one link in the overlay path.[1]

In many of the cases we will look at (e.g., multicast, mobility, routing, etc.) there are probably reasonably good solutions that could be proposed at the IP layer. The advantage of using overlays to design new systems, however, is that there is a lower barrier to innovation—no changes are required to the existing IP substrate.

2 Communication Overlays

In this section, we'll discuss communication overlays, such as RON, and the problems they try to solve, as well as various problems introduced by these communication overlays.

2.1 Overview

While the IP infrastructure provides the ability for hosts to communication reasonably reliably, and BGP is a dynamic routing protocol that scales to tens of thousands of networks, the infrastructure

[1] Actually, an overlay does not need to use end hosts; IP routers could conceivably be nodes as well. The essential point is that one "hop" in an overlay network corresponds to several hops in the underlying IP network. For simplicity, we may call each node in these overlays "end hosts".

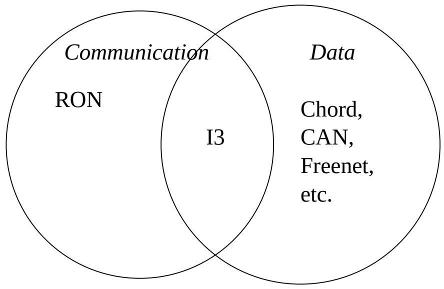

Figure 1: Overlays can be classified into two main areas: communication overlays, like RON, which help to optimize routing, and data overlays, like Chord, which focus on data dissemination by naming content. The Internet Indirection Infrastructure incorporates aspects of both of these.

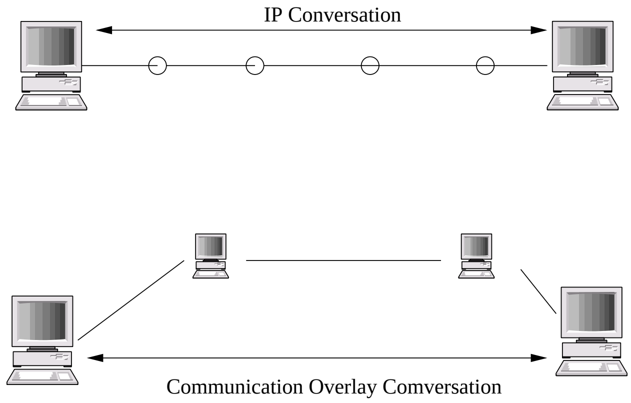

Figure 2: Overlay routing typically uses IP routing as a hop-to-hop abstraction. Whereas an IP conversation consists of data passed between two end hosts over a path of many IP-level hops (i.e., routers), an overlay conversation uses a path that treats the IP conversation as an abstraction, as one hop in the overlay conversation.

does not react very quickly to recover from link and router faults, etc. Previous researchers such as Labovitz have shown that, in the face of a fault, BGP can take up to 15 minutes to converge [1]. BGP is also not designed to react to application-sensitive metrics, such as high loss or latency, or to route around congested links.

Due to these performance shortcomings, end hosts can see potentially quite poor performance along various Internet paths, such as periods of high congestion or loss, and even extended outages, where two end hosts cannot communicate at all. In response, we can construct a communication overlay, where a path between two hosts may consist of several "IP conversation hops", as shown in Figure 2.

What sorts of problems can you imagine solving with such a communication overlay? Given that the overlay typically consists of a small "clique" of participating hosts, we could imagine the design of an overlay that achieves better failure resilience characteristics than those provided by the existing IP infrastructure (i.e., that which BGP provides).

Additionally, different applications have different performance metrics. For example, some applications, such as Internet audio, may be delay sensitive, and may not work above a certain loss rate. Others, such as video, may seek to minimize the observed jitter. Bulk transfer applications have different still definitions of acceptable loss and latency. A communication overlay should be integrated with the application in the sense that it allows the application to define what metrics are most meaningful for its purposes.

There are also inherent shortcomings with the granularity of policies that can be expressed in BGP. The example in the RON paper, for example, supposes that a select set of users at one ISP be able to use a private peering connection to access Internet2 resources upon a failure of the primary path. Allowing select users access to a path upon failover is a policy that is not easily and flexibly expressed in BGP. Communication overlays can potentially make these policy expressions easier.

Resilient Overlay Networks (RON) is a particular instantiation of a communication overlay. The implementation allows communication between a relatively small set of clients that disseminate link-state information about the performance seen on each of the pairwise paths in the overlay. Note that this information is disseminated over RON itself, to ensure that routing information can still be forwarded during periods of path failures. Policy can be implemented in RON by separating policy routing into two distinct phases: classification and routing table formation. Each packet that traverses the RON is given a particular policy tag that determines which routing tables can be used to forward that packet. Routing tables for each policy are constructed by considering a notion of shortest paths for the topology that only includes the links satisfied by that policy. All of this is implemented by means of the "policy classifier". This allows the overlay to implement policies such as cliques (e.g., non-Internet2 traffic stays off of Internet2, etc.).

The paper shows a few key results. Namely, the overlay can route around complete failures about 60% of the time, and TCP-observed failures (i.e., $\geq 30\%$ loss rate) about 53% of the time. Occasionally, RON was able to improve latency between two hosts by 40 ms or more. Hysteresis can prevent route flapping, although the implications of this technique on responsiveness to path failures is not entirely clear from the paper. It's conceivable that simple outage detection was separated from simple loss and latency optimization for the purposes of hysteresis, though.

2.2 Issues

There are several imminent issues with constructing a communication overlay such as RON. First of all, it's not intuitively clear why such a system should work in the first place. For example, depending on where failures were actually occurring (as well as the composition of the overlay itself), paths may not fail independently. In the worst case, a particular path failure could knock out all alternate paths. How often does this occur?

Second, what happens when a RON grows particularly large? Can a RON find stable paths, or is the advantage gained by RON merely a result of the fact that it is a small subset of the total Internet traffic and hosts? Could a large RON alter the business relationships between ISPs by skewing traffic volumes? What happens when a large number or RONs interact?

2.2.1 When do overlays work, and why?

The results presented in the RON paper show the ability to circumvent path failures and periods of high loss for "single-hop" indirection paths. Why does this method work, though? One might expect that, if failures were occurring near the end hosts themselves, paths would not fail independently. Where do failures occur, and how does this influence an overlay network's ability (or that of any reactive routing scheme, for that matter) to route around failures? Additionally, do paths always fail in the same place? One might expect that, if this were the case, one could use some static mechanism to route around failures with the knowledge that they were always occurring in the same place.

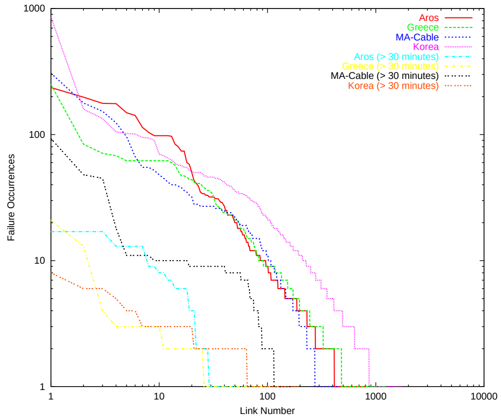

Figure 3: Failures appear in many different places, regardless of failure duration.

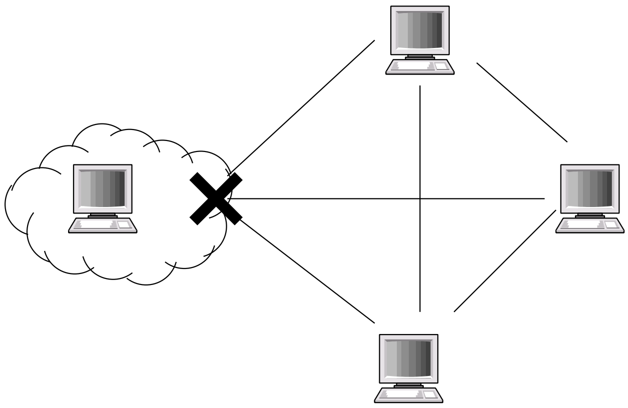

Figure 4: The location of a failure has a significant effect on whether alternate paths fail independently.

Failure Location. It turns out that paths fail all over the place. Figure 3 shows the occurrences of failures on paths involving various hosts in the RON testbed. The x-axis represents canonical link numbers (where "link" is in the IP sense of the word), and the y-axis represents the number of times a failure between that host and some other host on the overlay appeared on that particular link. The main thing to notice is that path failures seem to be a general property of the Internet and aren't really isolated to one or two perpetually bad locations. This is true, even for prolonged path failures. This argues that using reactive routing techniques can in fact help, since averting path failures isn't just about staying away from persistent trouble spots. Note that it's also the case that a virtual link that sees bad performance over short time scales is not necessarily going to perform relatively worse over longer timescales.

It's also the case that most failures occur within one or two IP-layer hops from end hosts. If this is the case, can RON really be effective? Doesn't a failure then imply that most paths will not fail independently? We find that, because of this characteristic, having multiconnected endhosts is a sufficient (though not necessary) condition for providing RON with significant alternate path options to route around the failure. It also turns out to help if the path on which a failure occurs contains a high-degree autonomous system through which other indirect paths may allow the destination to be reached. Note that the "RON_1" measurements and claims about being to route around

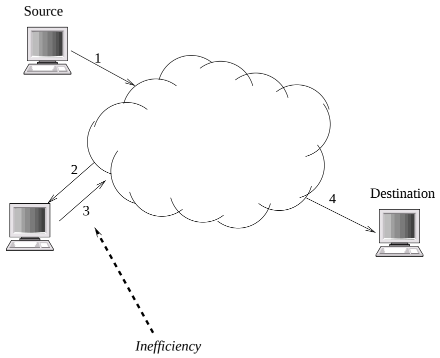

Figure 5: In RON, traffic is often carried on the same links twice, resulting in some bandwidth inefficiency.

100% of path failures, as well as the claims about edge failures, is simply bogus. The existence of an edge failure is not fatal if the site happens to be multiconnected.

Failure Duration It so happens that the median path failure length is about 3 minutes, and over 90% of path failures last less than 15 minutes. In general, RON can route around about 60% of the path failures. This assumes, however, that it (or any reactive routing scheme) can quickly detect the existence of a failure and route around it (e.g., before it is over). However, for the most part, detecting a failure is trivial and probably doesn't even require active probing. Passive monitoring techniques are also useful here. For example, products such as RouteScience use a passive monitoring trick by serving a small HTML object for the server's Web page.

2.2.2 Scalability

The particular implementation of RON brings to light several scaling issues that may or may not be fundamental to communication overlays. The first is that actively probing all paths to quickly react to failures and choose an alternate path may not scale as the number of hosts in the overlay grows large. This is probably not a fundamental limitation of communication overlays, and rather is just an issue with the particular implementation of the overlay presented in the paper. More intelligent, conservative probing mechanisms, potentially combined with n-hop indirection, could potentially avert this scaling issue. Probing is also a delayed picture of network state, and may introduce congestion itself, if not done carefully.

Second, and more fundamentally, RON may be exploiting a commons effect, and the benefit provided by one resilient overlay may diminish as a growing number of RONs begin to compete for the same shared resources. This may not necessarily be the case, but it is an important question and is currently an unsolved problem. That is, are communication overlays potentially candidates for a "tragedy of the commons"-style problem? Could they be a victim of their own success?

An obvious inefficiency is that, due to how the overlay is constructed, traffic is often carried on the same links twice. For example, in a single-hop indirection scheme, traffic flows on the link into and out of the intermediate host. If a communication overlay were to grow to the size of the Internet, every packet could be sent twice! This is shown in Figure 5.

Something that's not a scaling issue, per se, but something to think about, is how a RON that transits a potentially large amount of traffic could alter ISP relationships. If the path between two hosts were constantly failing, such that one host were always taking a certain indirect hop, the ISP that provides transit for the host serving as the indirect hop will start sending more traffic toward the destination's AS than it previously did (maybe these two AS's didn't even exchange traffic before). This could result in the AS of the intermediate host having to reconsider its business relationships, etc. This "problem", however, does not really mean that an overlay network is doing a "bad thing"; after all, the overlay nodes are customers of their respective ISPs, and the overlay is simply an application of sorts. P2P applications, in effect, can pose the same problem.

2.3 Other Approaches

The main problems that overlay routing tries to solve is that routing in the IP substrate is slow to react to failures, and that overlay routing can allow routing according to application-specified metrics. In fact, "intelligent routing" schemes try to do the same thing, in some sense, by observing path properties (either actively or passively), and adjusting BGP import policies to select an alternate path.

Given that RON seems to work best between nodes that are multiconnected, the advantage that an overlay provides over these reactive routing schemes is not entirely clear. In both cases, the ability to probe and quickly detect failures (i.e., more quickly than BGP) is a clear benefit over the IP-level host communication. However, the benefit that the overlay itself provides over simply adjusting import policies is not immediately clear (except for the fact that it provides *different* alternate paths than being multiconnected provides).

3 Data Overlays

We now have an idea about the types of problems that a communication overlay can be designed to solve. Using the abstraction of a path between two hosts consisting of one or more "virtual links" in the overlay, we can also imagine games we might play to place conversations on top of this overlay. For example, what if we considered the path between two hosts in the overlay as some sort of abstraction itself? What sort of services could this allow us to compose?

3.1 Overview

For one, we could implement systems such as Chord in this fashion. Consider that Chord retrieves data by locating the node in its finger table that is the closest successor, and then routing the data in this fashion. What this essentially gives you is the ability to do content-addressable networking, where A host can name an end-host according to the content that it wants to retrieve. A conversation, in this sense, involves $O(\log n)$ *overlay* hops. Of course, this may lead to particularly bad performance in the underlying IP substrate if the overlay is not constructed with good knowledge about the underlying IP network. You could do something like a nearest server selection, but this would only guarantee you hop-by-hop performance, not a globally optimal path in the underlying network.

Distributed hash table systems thus allow for an data-centric abstraction based on "put" and "get" operations, since data can be passed to a node in a manner that is independent of the IP-layer naming of a particular end-host. One can think about extending this idea further to allow two end hosts to communicate along paths in the overlay. One might imagine doing this by having one host issue a PUT for some data, while a second host issues a GET on the associated data. This allows for a level of indirection whereby a path between two hosts may contain multiple paths in the overlay itself!

The I3 architecture uses this level of indirection to tackle several problems, including:

- mobility
- load balance
- multicast
- service composition

This level of indirection can make it easier to provide services such as mobility, since we can abstract the notion of named endpoints away. That is, adding a level of indirection such as that provided by a data overlay allows hosts to effectively rendezvous to communicate. In this fashion, communication between two hosts can exist without either end having to name the end hosts themselves (as in necessary in IP). As a result, mobility becomes an easier problem: an end host can change its "name" (i.e., its IP address) in the IP substrate, but those hosts that are sending data need not know of a change in IP address, since the communication abstraction is based on a rendezvous point that does not change. The I3 paper describes that, for optimization, the hosts might perform point-to-point communication after the initial discovery has taken place (although this admittedly breaks an abstraction barrier).

This naming abstraction has been used in several other contexts to achieve mobility. Consider two alternate approaches; first, the Migrate [2] architecture, which uses DNS updates to rebind the name of a host to a new address, thus making the DNS name for a host the endpoint through which a host can communicate. Second, there is the mobile IP approach, which uses the concept of a foreign agent to implement mobility. Correspondent hosts communicate with a host's home agent, which is responsible for knowing the location of the host itself. Both of these approaches are specific to the mobility problem, however.

The architecture proposed by using DHTs as a naming abstraction is more general, because it treats mobility as just one of the many problems that can be solved by establishing conversations while abstracting the identity of the communicating end hosts. Because data overlays can route packets according to identifiers, communication no longer needs to be point-to-point. Many end hosts can be on the other side of an identifier (multicast), or the end-host corresponding to the longest matching trigger can change (load-balancing). It's also possible to do things like specify application-specific operations, such as transcoding, by sending packets on a particular path through the overlay (e.g., various bitrates for video); better yet, it can even be receiver specified, since the receiver can insert information that specifies other intermediate triggers (i.e., operations).

3.2 Issues

In this section, we overview some of the issues that arise with DHT-type overlays. In particular, the nature of overlays is such that the hosts storing data may be transient. In these cases, how can we guarantee high availability? It might be the case that $\log n$ hops is too many hops to provide suitable performance for some applications. Finally, we'll look at some of the lingering (and fostering) security problems introduced by DHTs, as well as the proposed Internet indirection infrastructure.

3.2.1 Stabilization and Availability

As seen in previous lectures, DHT-based systems such as Chord must achieve stabilization in the face of hosts that join and leave. They should also provide communication, even when hosts fail. That is, these trigger-based mechanisms such as I3 should work, even when hosts responsible for storing these triggers leave the network.

Additionally, there may be applications for which the $O(\log n)$ overlay hops is not sufficient to send data with low enough latency to be useful. In these cases, there should be a better way to route packets to their destination. I3, for example, suggests that, after the initial lookup, packets be routed along a point-to-point path in IP. This, however, suffers from several shortcomings, the least of which is hampering the mobility itself by breaking the indirection abstraction. Similarly, this seems to preclude the ability to perform the function composition, as described in the paper. Clearly there is some tradeoff between the ability to quickly route to an end host and the flexibility and abstraction provided by the DHT. This brings us against a performance vs. flexibility tradeoff that, at present, is probably the most significant tradeoff for data overlays. We discuss this further later.

It has been suggested that the sender and receiver choose some indirection point that is "close" to it in the underlying topology. This seems like a tricky proposition, for a couple of reasons. First, it may be difficult for the sender and receiver to agree on a trigger that is close to both of them. Second, this might make certain types of trigger-guessing attacks easier by narrowing the search space.

To solve the availability problem, it's possible that triggers can be replicated on k successors following the immediate successor, such that if a successor dies, communication can still continue. Additionally, such a system should take care to balance load across all indirection points for that data channel, so that one indirection point does not become overloaded. The I3 paper suggests caching as a potential solution to this problem, but doesn't really go into great deal describing it.

3.2.2 Security Issues

The abstraction introduced by data overlays introduce a number of key security problems. Most of them stem from the assumption that the additional layer of abstraction makes it more likely for certain attacks, such as man-in-the-middle attacks, to be mounted.

Note that the I3 paper claims that eavesdropping requires hijacking a key out of a fairly large keyspace, this runs counter to the suggestion that a trigger be selected such that it be associated with an end host that is reasonably close in the network space to the host that is either sending data or inserting a triggers. In this sense, these data-type overlays are again making a tradeoff, as the goals of performance and security seem to run counter to flexibility.

While preventing eavesdropping might, to some degree, be prevented by the public key approach presented in the paper, the protocol remains vulnerable to an attack whereby the server that is responsible for maintaining the trigger for the sender A can simply replace id_a with its own trigger, it can trick B into thinking that it is talking with A by replacing the private trigger for A with it's own choice for a private trigger. It can then intercept B's message in reverse, and substitute B's choice for a private trigger with its own choice. With this approach, a man in the middle has the ability to insert arbitrary messages. This type of attack can be easily fixed by making different assumptions, or by using signatures, but it's not entirely clear that the authors gave this much thought. Additionally, the claim about anonymity doesn't really stand up to sophisticated traffic analysis attacks.

The loop detection proposed in the I3 paper has to assume that the nodes holding the triggers are not malicious themselves.

Various routing, data, and storage attacks are possible on distributed hash table-based systems such as Chord. A bootstrapping node, for example, could be fooled into joining the a malicious, parallel network, etc.

4 Discussion and Tradeoffs

Communication overlays provide a nice abstraction for a conversation between two hosts by creating these virtual links that are effectively several hops in the underlying IP network. On a small scale, such overlays can react more quickly to adverse network conditions than traditional routing protocols. In exchange, these networks incur some inefficiencies. Bandwidth requirements increase, various monitoring techniques won't scale to overlay networks of very large sizes, and many interacting overlays may essentially act as sheep grazing on the common. Nevertheless, communication overlays serve to improve application performance due to their ability to react quickly and exploit alternate paths (note the similarity with intelligent routing techniques).

From a different perspective, data overlays like Chord separate content from the hosts that are responsible for hosting that content. Point-to-point communication is pushed aside in favor of naming content, rather than end-hosts themselves. Issues here include stabilization, and maintaining high availability as hosts come and go.

Moving back toward the communication-style overlays, we have systems like I3, which exploit content-addressable network framework to establish an indirection/rendezvous point through which two endpoints can communicate. Fundamentally, this hybrid-style overlay suffers from a performance vs. flexibility tradeoff that is (as yet) unresolved. In particular, nearly every performance optimization proposed (e.g., server selection, better trigger selection, caching, using the overlay only for discovery, etc.) seems to require dipping into the IP layer and breaking the abstraction barrier proposed by the overlay (as well as limiting flexibility).

References

[1] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet Routing Convergence. *IEEE/ACM Transactions on Networking*, 9(3):293–306, June 2001.

[2] Alex C. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *Proc. 6th ACM/IEEE MOBICOM*, August 2000.

