Internet Engineering Task Force
Internet Draft
Expires in May, 2001
draft-alaettinoglu-isis-convergence-00.txt

C. Alaettinoglu
V. Jacobson
H. Yu
Packet Design

November, 2000

Towards Milli-Second IGP Convergence

<draft-alaettinoglu-isis-convergence-00>

Abstract

It is possible for link-state routing protocols to converge in link propagation time scales, that is, in tens of milliseconds. However, current deployments of ISIS[ISIS-ISO][ISIS-IP], a link-state routing protocol, are not anywhere near this point. In this paper, we present some analyses of ISIS convergence by showing its behavior upon link/router failures and repairs, and its scaling properties to large networks, both in terms of number of nodes and links. We then explore changes needed in the ISIS specification and implementations to reach IGP convergence in milliseconds. Our results are based on experimentation done with ISIS, but some of the findings may apply to OSPF as well.

This paper contains color figures and graphs which are missing from the ASCII version of the paper. The postscript and pdf versions of this paper can be found in the Internet-Drafts repository.

1 Motivation

The theoretical limit for link-state routing protocols to re-route

is in link propagation time scales, that is in tens of milliseconds. However, the current ISIS re-route times are in tens of seconds. During the re-route period, a subset of destinations are either not-reachable or are reached through non-optimal routes.

It is important to close this gap between theory and implementation for several reasons. First, millisecond re-route times will lead to increased network reliability since the periods at which routes are not available/non-optimal will be shorter (one to three orders of magnitude shorter). Second, it enables multi-service traffic such as voice over IP since it decreases the number of packets dropped to tolerable levels for these services. And finally, it eliminates the need for more expensive and complex layer 2 protections schemes, such as SONET.

There are three steps to ISIS re-routing: detection, propagation and shortest path calculation. In the detection step, the topology change is locally detected. That is, a link failure/repair is detected by the routers the link is adjacent to, and a router failure/repair is detected by the router's neighbors. In the propagation step, a new link-state packet (LSP) reflecting the local topology after the change is flooded to all the other routers in the network. Finally, in the shortest path calculation step, each router having received the new LSP computes the new routes using a shortest path tree algorithm. Typically, Dijkstra's Shortest Path First (SPF) algorithm is used in this calculation. Only after these three steps, the re-route has completed, or equivalently we say that ISIS has converged.

It is prudent to understand where the tens of seconds of re-route times go before trying to fix ISIS re-routing. Hence, we have done extensive experiments to characterize the time spent at each of these steps. In our experiments, we tested ISIS running on Cisco 7200s running IOS 12.0S9 and IOS 12.1P and on Juniper M40s running JunOS 4.1. In Sections [detection], [propagation], and [spf], we present our findings on the detection, propagation, and shortest path calculation steps respectively. In Section [summary], we summarize our findings and suggest changes to ISIS protocol specification and on its implementations.

ISTalk Software

In many of our experiments, we emulated very large random networks.

To facilitate this, we developed a software tool called ISTalk. ISTalk can establish adjacencies with ISIS routers and inject them LSPs emulating any topology allowing us to vary both the number of nodes and the number of edges in the network. This is graphically illustrated in Figure [istalk].([fig] ISTalk Software emulates virtual topologies.<istalk>)

2 Detection<detection>

The topology change, or the link-state change, is either detected

and communicated to ISIS by the lower level protocols (i.e. link level protocols), or it is detected using the peer to peer ISIS hello protocol. The link level detection is fastest but seems to be inconsistently implemented by vendors. In our experiments, with certain medium we noticed that the device driver detected the failure but did not communicate it to the router's OS. The link level detection is also not always possible, for example in a switched environment where a router fails behind a switch. In this case, the switch gets the link layer notification, but has no mechanism to notify other routers in the network.

The ISIS hello protocol is more robust, but slower. In this case, the adjacent routers send periodic hello packets to each other (the adjacency is also established through the use of hello packets, but not described here). If a router misses a fixed number of hello packets from an adjacent router, it declares the adjacency to that router down. By default, the routers send each other hello packets every 10 seconds, and declare the adjacency down after missing
3 hello packets. Hence, with default parameters, the detection
can take up to 30 seconds. The specification limits the granularity of hello interval to seconds, hence making 3 seconds the fastest possible detection based on the hello protocol. Several ISPs use
2 second hello intervals, but declare an adjacency down after missing
5 hello packets, resulting in a detection time of 10 seconds.([fig] Hello
Experiment Setup<hello-setup>)

To study the effect of hello interval, we configured 3 routers in a triangle topology and connected a workstation running ISTalk to router 1 as shown in Figure [hello-setup]. The routers are connected to each other using switched ethernet interfaces. Hence, when we take down one of the routers, the other two have to rely on the hello protocol to detect the change. The ISTalk establishes an ISIS adjacency with router 1 so that it receives the LSPs flooded in the network. The workstation pings the router 3 using router 3's interface address towards router 2 so that the pings follow

the path shown on the left hand side of Figure [hello-setup]. We
then bring down router 2 and after re-route the pings follow the
path shown on the right hand side of Figure [hello-setup]. The
pings are sent once every 100 milliseconds. ([fig] Packets received
by the workstation during the Hello Experiment<hello-received-packets>)

Figure [hello-received-packets] shows the relevant packets received
by the workstation. The short spikes are the ICMP ECHO REPLY packets
received; they are only received if there is a valid route to the
ping's destination and back. As seen in the figure, there is a
33 second period where there are no ICMP ECHO REPLY packets received.
This is the re-route time, during which router 1 does not have
a route to the destination. The two tall spikes are LSPs reflecting
the topology change as detected by routers 1 and 3. Note that the
two LSPs are separated by 1 second. This is because the two routers
detect the change at different times due to their 3 hello packet
intervals ending at different times. This separation can be anywhere
from 0 second to one full hello interval, in this case 10 seconds.

^L

Why then after receipt of LSPs there is still 5 seconds of re-route
time? This is because the routers delay the SPF calculation by
a number of seconds (default is 5 seconds) in the hope that they
may catch more LSPs, particularly this second LSP in the figure,
and do one SPF calculation instead of many (because the costly
SPF calculation overwhelmed routers in the past). We refer to this
delay as the SPF delay. For millisecond convergence, the SPF delay
should be in milliseconds as well, and should ideally be zero.

The argument against a small hello interval is the bandwidth consumed
by the hello packets and the hello packets getting queued behind
data packets and the routers not receiving them in time. The most
desirable solution to the latter problem is to treat hello packets
preferentially over the data packets. In case this solution is
not available, we studied how the hello protocol is affected by
different hello intervals by simulating a link and varied both
the data load (using heavy tail traffic) and the hello interval
(as a fraction of the link's bandwidth). Figure [effect of hello interval]
shows the results. As seen from the figure, the hello interval
does not play a significant role, i.e. the hello packets do not
miss their deadlines, until the hello packets become a dominant
bandwidth consumer of the link's capacity.([fig] Effect of hello
interval and load on the Hello Protocol.<effect of hello interval>)

Since the protocol's ultimate limit on the hello interval is set
by the bandwidth used by hello packets, extending the specification
to allow sub-second intervals would allow sub-second detection

on almost all links. This does not mean that the detection time can be made arbitrarily small, only that detection should be limited by the physical constraints of the link (its bandwidth, propagation delay, etc), not by arbitrary clock granularities set by the protocol designers.

Detection: stability and damping

For either event triggered (via the link layer notification) or hello driven detection, there are network-wide stability issues if routing tries to follow rapid link transients (i.e., a link that goes down and up several times a second). Making hello interval smaller may seem like allowing more instability into the network. The usual way of dealing with this is to treat "bad news" differently from the "good news" so routing is quick to find an alternate path on any failure but slow to switch back when the link comes up. The current ISIS specification treats bad news and good news the same way but it should be trivial to change the detection specification to allow different filtering constants for "down" and "up" state changes.

3 Propagation<propagation>

After a topological change is detected, a new link state packet (LSP) is generated at the point of detection and then flooded unmodified (except for the "remaining life" field) through the network. Flooding

should propagate the LSP across the network at near the speed of light plus one store-and-forward delay per hop. Thus, in theory the LSP propagation should make a negligible contribution to the re-route time. Unfortunately, this is not what we observed.

In our experiment, we connected 3 routers in a line topology using switched ethernet as shown in Figure [LSP Experiment Setup]. The ISTalk established ISIS adjacencies with router 1 on its interface **0 and with router 3 on its interface 3. It also passively listened** on its interfaces 1 and 2. Our experiments simulated networks of varying sizes. In the steady state, that is after three routers think they are part of a large network, we injected an LSP at interface **0 and observed the delay until it appeared at other interfaces** of the workstation.

Since the SPF calculation can take a significant amount of time (see Section [spf]), commercial router implementations impose a limit on how frequently the SPF calculation can be done using the SPF delay parameter. In some implementations this is 5 seconds fixed. In some implementations, this is changeable but the granularity is in seconds. This limit essentially adds to the propagation time. In this experiment, to remove its affect we set the SPF delay to zero seconds.([fig] LSP Propagation Experiment Setup.<LSP Experiment Setup>)

Figure [LSP Propagation Results] shows the results for 500, 800 and 1000 node topologies (the average node degree is fixet at 5). The x-axis is the interface number and the y-axis is the time in seconds at which the LSP is seen. As seen from the figure, propagating an LSP across 3 routers takes close to one second and propagation time is related to the topology size. The reason for this delay is related to the topology size is that the routers are performing the SPF calculation before propagating the LSPs. ([fig] LSP Propagation Experiment Results.<LSP Propagation Results>)

In our experiment, we set SPF delay to zero seconds resulting in a race condition at the routers between performing an SPF calculation or propagating the LSP. The routers chose to do the SPF calculation first. This degrades the propagation time from near the speed of light to O(diameter\times SPFtime). To prevent this, the specification might be amended to explicitly state that the LSP flooding is higher priority than the SPF calculation.

4 Shortest Path Calculation<spf>

The final step in re-route is to compute new routes using a shortest path tree algorithm, typically Dijkstra's Shortest Path First algorithm. For sparse topologies, a binary heap implementation of Dijkstra's algorithm has O(n\log n) complexity, and a naive implementation has O(n^{2}) complexity. Figures [vary nodes] shows the SPF calculation time in milliseconds versus number of nodes in the network. The Figure [vary nodes log] shows the same but the y-axis is in log scale. In this experiment each node had a fixed number of neighbors, 5.([fig] SPF calculation time versus number of nodes.<vary nodes>)

([fig] SPF calculation time versus number of nodes (log scale).<vary nodes log>) The top 3 curves are the commercial router vendor implementations (collected at the Ciscos using "show isis spf log" and using "show isis spf-log" in Junipers). The bottom curve is an in-house implementation of Dijkstra's shortest path algorithm. All the time values are wall-clock times, and there is no other load on the routers. The SPF calculation is indeed CPU intensive, taking 100s of milliseconds. As can be seen from the log-scale graph, the bottom 3 curves have the same shape. As a matter of fact an n\log n curve fits perfectly to them, unfortunately the top curve, one of the commercial router implementations fits an n^{2} curve perfectly.([fig] SPF calculation time versus number of edges per node.<vary degree>) In Figure [vary degree], we kept the number of nodes at 300 but varied the average number of edges per node. Today, some ISPs have full-mesh topologies (using ATM/MPLS technologies) and run ISIS over it. With full-mesh, the SPF calculation is even more intense, almost about one second.

As illustrated by these experiments, even in high-end platforms, the SPF calculation can take a long time (seconds) and has poor scaling properties (n\log n to n^{2}). This has a serious impact both on re-route times since the SPF calculation is in series with the LSP propagation and on overall network stability since the CPU on the routers will be saturated. To fix this, an algorithm that scales much better than Dijkstra's is needed. Dijkstra's algorithm re-computes all routes after a topology change regardless of whether they were affected or not, each time starting from scratch. More recent algorithms[Franciosa97:dijk][Frigioni94:dijk][Narvaez99:dijk] store the data structures from earlier calculations and only re-compute the affected routes. Their average case complexity is O(\log n).

We implemented one of these algorithms and results for 300 node topology with degree varied are shown in Figure [dynamic SPT]. Note that the y-axis, SPF time in milliseconds, is in log scale, and the new dynamic shortest path tree algorithm computes new routes in microseconds, that is 10,000 times faster than the commercial implementations of Dijkstra's algorithm. Note that with the dynamic shortest path tree algorithm, the calculation time first increases and then decreases. This is because as the average degree increases, the number of affected nodes by a topology change first increases and then decreases. In the case of a full-mesh topology, the number of affected nodes is often only one node.([fig] Dynamic SPT algorithm is 10,000 times faster.<dynamic SPT>)

5 Summary of Findings and Reccomendations<summary>

Stable, robust IP re-routing that works at the network's propagation time (the theoretical maximum for any re-routing scheme) is both possible and achievable. To get there, we have to make some minor changes to the ISIS specification in the following priority order:

* switch to a modern algorithm for the SPF calculation

* make the granularity of the Hello Interval in millisecond, rather than seconds

* allow different detection and damping filter constants for the

  link up and down events, i.e. differentiate good and bad news

* give higher priority to LSP propagation than SPF computation

* queue hello packets in front of data packets

What we did not see: During our experiments, we injected topologies with 90,000 edges, thousands of nodes, drove all the routers to 100% CPU utilization, randomly unplugged links and powered down routers. Although we were looking for it, we saw no evidence of routing instability and we observed several subtle things done to avoid getting into an unstable operating regime. For example, even under 100% CPU utilization, the routers have never missed sending out an hello packet. If they did, they could have dropped adjacencies, causing more link-state changes, which in turn causing more routing load, and which in turn causing more hello packet being missed and so on. It appears that the two vendors we looked at have learned a lot from a decade's worth or routing disasters and meltdowns and are currently shipping robust routing code.

What we did not look for: In full-mesh topologies, each newly generated LSP is received by a router n times, one from each of its neighboring routers. Even though the router only performs one SPF for this LSP, it needs to recognize that the other n-1 LSPs are duplicates. When a router goes down in a full-mesh topology, each router loses an adjacency and generates a new LSP. Hence, routers now need to recognize O(n^{2}) duplicate LSPs and perform up to O(n) SPF calculations (in practice LSPs received while performing an SPF calculation are bundled together for the next round of SPF calculation). We did not design experiments to study the time it takes to recognize O(n^{2}) duplicate LSPs since there are already proposals[mpls-unnum] addressing this issue.

6 Acknowledgments

Authors are grateful to the Global Crossing Production Testing Lab, Cisco Systems, and Juniper Networks for allowing us to use their facilities to perform some of these tests. Authors would also like to thank Kathie Nichols, Judy Estrin, Chia-Chee Kuan, and Doug Klein for their comments on this document.

7 References

8 Authors' Addresses

Cengiz Alaettinoglu
Packet Design

66 Willow Place
Menlo Park, CA 94025
USA
email: cengiz@packetdesign.com

Van Jacobson
Packet Design
66 Willow Place
Menlo Park, CA 94025
USA
email: van@packetdesign.com

Haobo Yu
Packet Design
66 Willow Place
Menlo Park, CA 94025
USA
email: haoboy@packetdesign.com