

An OSPF Topology Server: Design and Evaluation

Aman Shaikh, Mukul Goyal, Albert Greenberg, Raju Rajan, K.K. Ramakrishnan

Abstract— In large scale, operational IP networks, creating timely, accurate and network-wide views of the intradomain topology is a fundamental problem. Typical network backbones consist of hundreds of routers, which establish routing adjacencies with one another through static configuration and dynamic routing protocols, such as OSPF. In this paper, we describe the design of an OSPF topology server which tracks intradomain topology, by passively and safely listening into OSPF’s reliable flooding mechanism, or by pushing and pulling information from the routers via SNMP. We provide a detailed evaluation and comparison of the two approaches in terms of operational issues, reliability, and timeliness of information.

Keywords— Network Management, Routing, Topology Discovery, OSPF, SNMP

I. INTRODUCTION

IP networks are becoming the single ubiquitous network for all our communication needs. As users depend increasingly on a common infrastructure for mission-critical needs, they require that the network provide increased reliability and support for new services. Value-added services (e.g., Virtual Private Networks (VPNs), Virtual Leased Lines (VLLs) or Voice over IP (VoIP)) require high levels of visibility, predictability and reliability in the network’s routing topology. IP service providers need systems that track traffic loads and paths through the network taken by those loads.

In particular, a fundamental building block needed for next generation network management systems is a *topology server*. In this paper, we consider the design and evaluation of topology servers for a single Autonomous System (AS), or administrative domain of authority. Within an AS, routing is controlled by an interior gateway protocol, with OSPF (Open Shortest Path Forwarding) [1], [2] and IS-IS (Intermediate System to Intermediate System) [3] the prevailing choices for IP networks today [4]. We focus here on OSPF.

An OSPF topology server would provide a real-time, accurate view of the components of the IP network and of paths traversed by different flows across the network. Its design is made challenging by dynamic changes in the network: link or router interface failures (permanent or intermittent) occur frequently in an IP network, caused by hardware and software problems. In addition, routine maintenance and provisioning events cause routers and in-

terfaces to go up and down. It is worth mentioning that MPLS and its extensions (e.g., MPLambdaS or GMPLS) for control of label-switched or optical network connections, employ interior gateway protocols such as OSPF for topology discovery. That is, MPLS uses OSPF to capture network topology. An OSPF topology server then extends to MPLS, if augmented to track additional link attributes.

Our principal contribution in this paper is a detailed design of an *OSPF topology server*, which tracks the network-level topology controlled by OSPF. With the tremendous investment in SNMP (Simple Network Management Protocol) based network management systems today [5], the obvious and natural approach to building an OSPF topology server is to utilize SNMP to gather topology information. We term this the *management plane* approach. An alternative, *control plane* approach, works by speaking enough of the OSPF protocol to take advantage of the messages (Link State Advertisements) that OSPF naturally floods throughout the network to ensure the routers arrive at a common view of the network topology. We describe design of a topology server using both the approaches. We have implemented both these designs, and verified their correctness in a lab testbed. We also describe the advantages and disadvantages of the two approaches in terms of qualitative complexity and overhead, and in terms of quantitative measures of reliability and information timeliness.

The paper is organized as follows. Section II describes the related work. Section III provides an overview of OSPF. In Section IV, we describe design of the topology server using both management plane and control plane approaches. The steps to ensure the passivity and safety of the control plane approach are discussed in Section IV-B. In Section V, we evaluate the two approaches. Conclusions are drawn in Section VI.

II. RELATED WORK

Feldmann et al. describe methods for network and topology discovery obtained from periodic dumps of router configuration files [6]. Similar tools periodically dump OSPF link state databases. This provides a static view of the topology. One can make this more dynamic by increasing the dumping frequency, but it is hard to go beyond a certain limit because of network and data collection overheads. Another approach to topology discovery might be to monitor updates from systems that provision or reconfigure the network. Unfortunately, IP networks are essentially distributed and it is hard to design a topology server that sees all changes to the network’s configuration. Moreover, this approach misses unintended changes (faults). With protocols such as OSPF and IS-IS that use flooding for distributing topology information, one might deploy a protocol monitor on a LAN between two production routers,

Aman Shaikh is at the University of California, Santa Cruz, CA 95064. E-mail: aman@soe.ucsc.edu

Mukul Goyal is at the Ohio State University, Columbus, OH 43210. E-mail: mukul@cis.ohio-state.edu

Albert Greenberg is with AT&T Research, Florham Park, NJ 07932. E-mail: albert@research.att.com

Raju Rajan is with Ipsilon Networks, Philadelphia, PA 19123, E-mail: raju@ipsumnetworks.com

K.K. Ramakrishnan is with TeraOptic Networks, Inc., Sunnyvale, CA 94085. E-mail: kk@teraoptic.com

and snoop on the protocol messages describing topology messages. However, sifting reliably through all the messages for the protocol messages describing the topology is a difficult task. Moreover, it is hard to obtain the initial view of the topology with this approach; the topology view is formed gradually as packets are captured. Lakshman et al. mention approaches to real-time discovery of topology in their work on the RATES System developed for MPLS traffic engineering [7]. But topology discovery is just one of the modules of their system and their discussion does not go into detail. Baccelli and Rajan discuss an OSPF monitoring architecture, tailored to a policy based networking framework [8].

Siamwalla et al. [9] and Govindan [10] discuss topology discovery methods that do not require cooperation from the network service provider, relying on a variety of probes, including pings and traceroutes. Such methods provide indications of interface up/down status and router connectivity. However, these methods do not deal directly with OSPF topology tracking, the topic of this paper. A serious disadvantage of any such method (common to ping-based network management schemes) is the inability to infer causes of missing pings. For example, traffic congestion will sometimes cause loss of OSPF connectivity and sometimes loss of pings, but the two are not directly correlated. Also, in the midst of a topology change, pings directed to interfaces that happen to be up may be lost owing to transient routing loops.

III. OSPF OVERVIEW

As noted in Section I, OSPF is widely used to control routing within an Autonomous System (AS). OSPF is a link state routing protocol, meaning that each router within the AS discovers and builds an entire view of the network topology. This topology view is conceptually a directed graph. Each router represents a vertex in this topology graph, and each link between neighbor routers represents a unidirectional edge. Each link also has an associated weight that is administratively assigned in the configuration file of the router. Using the weighted topology graph, each router computes a shortest path tree with itself as the root, and applies the results to build its forwarding table. This assures that packets are forwarded along the shortest paths in terms of link weights to their destinations [2]. We will refer to the computation of the shortest path tree as an *SPF computation*, and the resultant tree as an *SPF tree*.

As illustrated in Figure 1, the OSPF topology may be divided into areas determining a two level hierarchy. Area 0, known as the *backbone area*, resides at the top level of the hierarchy and provides connectivity to the non-backbone areas (numbered 1, 2, ...). OSPF assigns each link to exactly one area. The routers that have links to multiple areas are called *border routers*. Every router maintains a separate copy of the topology graph for each areas it is connected to. The router performs SPF computation on each such topology graph and thereby knows how to reach nodes in all the areas it is connected to. As for remote

areas, the router does not get to know about the entire topology of the area, but knows about the total weight of the path from one or more border routers of its areas to each node in remote areas. Thus, after computing SPF tree for each area, the router learns which border router to use as an intermediate node for reaching each remote node.

Every router running OSPF is responsible for describing its local connectivity in a *Link State Advertisement (LSA)*. These LSAs are *flooded* reliably to other routers in the network, which allows them to build the view of the topology as mentioned earlier. The flooding is made reliable by mandating that a router acknowledge the receipt of every LSA it receives from every neighbor. The flooding is hop-by-hop and hence does not depend on routing. The set of LSAs in a router's memory is called a *link state database* and conceptually forms the topology graph for the router.

OSPF uses several types of LSAs for describing different parts of topology. Every router describes links to all its neighbor routers in a given area in a *Router LSA*. Router LSAs are flooded only within an area and thus are said to have an area-level flooding scope. Thus, a border router has to originate a separate Router LSA for every area it is connected to. Border routers also summarize information about one area into another by originating *Summary LSAs*. It is through Summary LSAs that other routers learn about nodes in the remote areas. Summary LSAs have an area-level flooding scope like Router LSAs. OSPF allows routing information to be imported from other routing protocols like BGP. The router that imports routing information from other protocols into OSPF is called an *AS Border Router (ASBR)*. An ASBR originates *External LSAs* to describe the external routing information. The External LSAs are flooded in the entire AS irrespective of area boundaries, and hence have an AS-level flooding scope.

Two routers are neighbor routers if they have interfaces to a common network (i.e, they have link level connectivity). Neighbor routers form an *adjacency* so that they can exchange routing information with each other. OSPF allows a link between the neighbor routers to be used for forwarding only if these routers have the same view of the topology, i.e., the same link state database. This ensures that forwarding data packets over the link does not create loops. Thus, two neighbor routers have to make sure that their link state databases are in sync, and they do so by exchanging parts of their link state databases when they establish an adjacency. The adjacency between a pair of routers is said to be *full* once they have synchronized their link state databases.

While sending LSAs to a neighbor, a router bundles them together into a *Link State Update* packet. OSPF also has other kinds of packets, which are listed in Table I.

IV. DESIGN

The topology server tracks the OSPF topology described in Section III. As mentioned earlier, we consider two basic approaches to the design of the topology server, a *management plane* approach (Section IV-A), which involves speak-

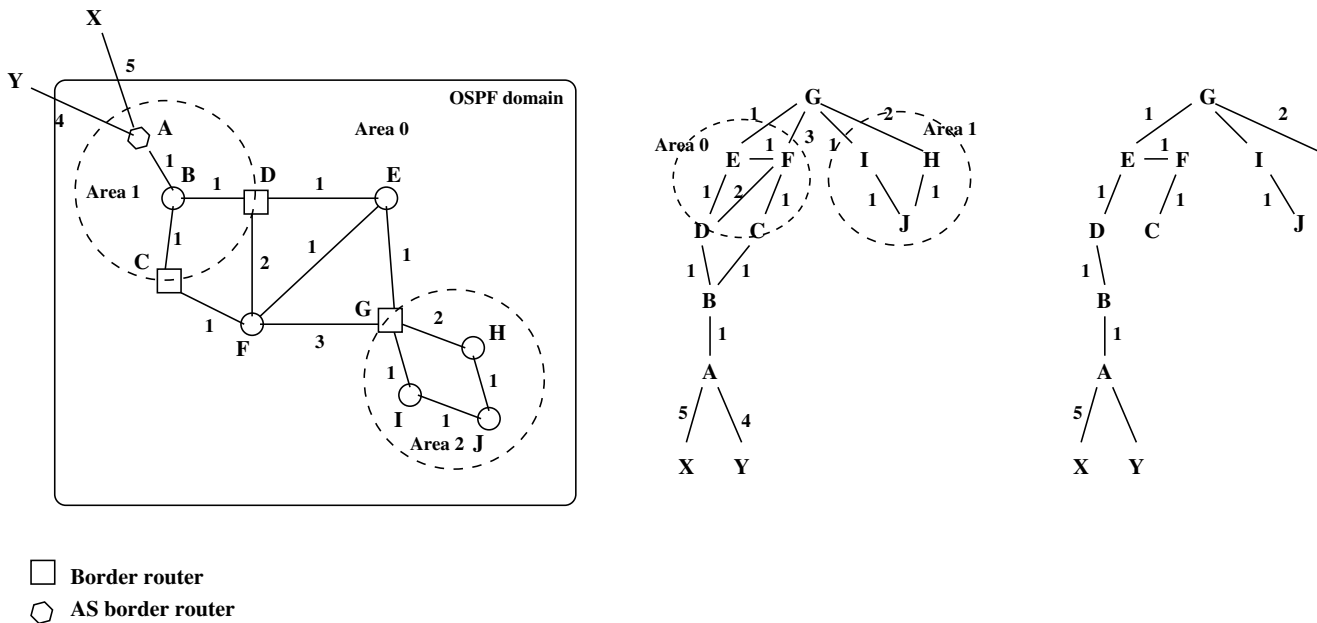


Fig. 1. From left to right, the figure depicts an example OSPF topology, the view of that topology from router G, and the shortest path tree calculated at G. (Though we show the OSPF topology as an undirected graph here for simplicity, in reality the graph is directed.)

Packet Type	Use
Hello	To periodically check status of adjacent routers
Database Description	To describe link state database for synchronization purposes during adjacency formation
Link State Request	To request missing or stale pieces of link state database during adjacency formation
Link State Update	To send one or more LSAs to adjacent routers as a part of the flooding protocol
Link State Acknowledgment	To send LSA acknowledgments

TABLE I
OSPF PACKET TYPES AND SEMANTICS.

ing to the routers via SNMP, and a *control plane* approach (Section IV-B), which involves some level of participation in OSPF.

The goals for the topology server design are:

- **Independent and Incremental Deployment.** No change should be needed to existing network routing or management protocols or systems.
- **Safety and Passivity.** There should not be any impact on packet routing, forwarding, and network reliability.
- **Accuracy and Timeliness.** The view of topology maintained in the topology server should be identical to that in the router databases. Following an event that changes the topology, the topology server should converge to the new view near the time that the routers themselves converge to the new view.
- **Reliability and Robustness.** Updates to the topology should be reliably detected by the topology server. The design of the topology server should be very simple and robust.

A. The Management Plane Approach

SNMP provides the most natural and convenient mechanism to monitor OSPF topology from the management plane. The OSPF SNMP MIB [11] defines variables that indicate whether a given OSPF adjacency is up or down. These MIBs are maintained on the routers and values of the variables can be extracted using SNMP GET or GET-NEXT methods. We use GET to acquire the initial view of the topology when the topology server starts.

SNMP also defines TRAPS that are triggered upon changes in OSPF status such as interface up/down or neighbor up/down. These TRAPS can be enabled on a router. While enabling a TRAP, one also needs to specify the IP address(es) where the TRAP needs to be sent. Once enabled, TRAPS are sent to these addresses when the associated event occurs. Thus, by registering with a router for TRAPS related to OSPF status changes, the topology server can keep track of the OSPF topology. Of course, the topology server has to register with all the routers in the network to construct the entire topology view. In general,

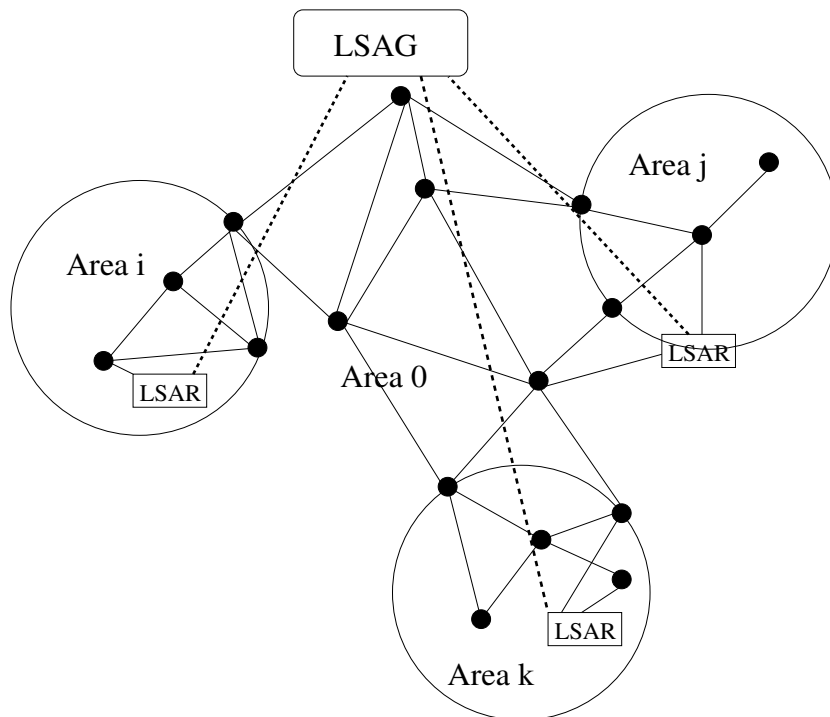


Fig. 2. Control plane approach for topology tracking: To monitor OSPF in a given area, an LSAR establishes direct adjacencies with one or more routers with at least one link in that area. An LSAR may monitor as many areas as it has interfaces to; e.g., the LSAR monitoring Area 0 and Area j. TCP transport (illustrated by dashed lines) is used to export LSAs collected at the LSARs to the LSAG.

TRAPs are not acknowledged, and so TRAP-based methods are not reliable. (Support for TRAP acknowledgment is beginning to become more widespread [5].)

Details

The management plane based server uses the OSPF SNMP MIB [11] tables and variables defined therein as well as the TRAPs. The key MIB variables used are those defined in *ospfIfTable*, *ospfNbrTable* and *ospfAreaTable*. The *ospfIfTable* provides OSPF specific information on interfaces of a router while the *ospfNbrTable* describes neighbors of the router. The *ospfAreaTable* contains information about the OSPF areas to which the router is attached. The server polls these MIB variables using GET and GETNEXT methods for topology discovery at the start time and for periodic updates to the topology view after that. Among the TRAPs defined in OSPF MIBs, the ones we rely on are *ospfIfStateChange* and *ospfNbrStateChange* TRAPs. The *ospfIfStateChange* TRAP is generated by a router when the OSPF state of one of its interface progresses or regresses from a terminal state. Similarly, the *ospfNbrStateChange* TRAP is generated by a router when the OSPF state of its neighbor progresses or regresses from a terminal state. Together, these two TRAPs allow the server to keep track of when adjacencies between routers get formed or are teared down. This, in turn, allows the server to keep track of the OSPF topology.

The topology server executes three main processes: network discovery, TRAP configuration and handling, and polling. The network discovery process is used for discov-

ering routers in the network. The process extracts the local OSPF variable set from the router's OSPF MIB: the administrative state, the interfaces, the attached areas and the OSPF adjacencies among others. It maintains two sets of routers: Set Ω for "yet-to-be-processed" routers, and set Θ for routers which are already processed. The discovery process starts with one or more "seed" routers in Ω . While Ω is not empty, the process removes a router from Ω , processes it, adds it to Θ , and adds all its adjacent routers that have not already been discovered to set Ω .

The TRAP configuration and handling process enables the TRAPs mentioned above on the router as they are discovered. At the time of enabling, the process adds the topology server as a recipient of these TRAPs. The process also handles the TRAPs sent by routers and updates the topology view.

The polling process implements a straightforward polling schedule. The process cycles through the set Θ at a configurable rate, and checks if these routers have any new neighbors. If the new routers are discovered, they are first added to Ω . These routers are then processed and moved to the set Θ . This way the topology server learns about new routers as they are added to the network. The server then enables TRAPs on these newly discovered routers.

B. The Control Plane Approach

The topology server consists of two principal functional components in our control plane approach: an LSA Reflector (LSAR) and an LSA Aggregator (LSAG). The LSAR establishes adjacencies with routers and collects LSAs that

are then passed along to one or more LSAGs over reliable TCP connections. Each LSAG implements a topology server, in the sense that it synthesizes a global view of the topology from the received LSAs. The LSAG also provides APIs to the applications that require topology views. Note that an LSAR acquires the initial view of the topology as the end-result of database synchronization procedure when it establishes adjacency with a router. It learns of subsequent updates to the topology via LSAs that the router floods over the adjacency. Figure 2 illustrates an implementation with one LSAG collecting LSAs from a number of LSARs, one in each OSPF area.

Separating the LSAR and LSAG functions provides a degree of fault isolation; each function can be simplified and replicated independently to increase overall reliability. Another benefit is that the LSAG may subscribe to a subset of the LSAs, for example, just the Router and Network LSAs for a given OSPF area. Furthermore, an LSAR has to reside close to the network since it establishes adjacency with a router, and so must be very simple in order to achieve a high level of reliability. Moreover, we may require more than one LSAR to cover all the areas if areas are geographically widespread. Finally, the LSAG, having to support applications, may require more complex processing and more frequent upgrades. Separating the LSAG from the LSAR allows us to bring the LSAG up and down without disturbing the LSARs.

Details

One of the main design goals for the LSAR is that it must be *passive*. Though the LSAR speaks OSPF it is not a real router, and it must not be involved in packet forwarding. No other router in the network should send data packets to the LSAR to be forwarded elsewhere. A natural line of defense against having the LSAR participate in normal forwarding is to use router configuration measures: assign effectively infinite OSPF weights to the links to the LSARs, and install on neighbor routers strict access control lists and route filters. A stronger line of defense is to exploit standardized features of the OSPF protocol that permit links to the LSAR to carry LSAs from the routers, but do not permit the routers to use these links to carry data traffic. We accomplish this by keeping the LSAR-router adjacency in an intermediate state such that the two ends start the database synchronization but never finish it. In order to achieve this, the LSAR originates an LSA L and informs the router that it has this LSA during the synchronization process but never actually sends it out to the router. This makes sure that the database is never synchronized and hence the adjacency is never fully established from the router's perspective. As a result of this, the adjacent router never gets to advertise a link to the LSAR in its Router LSA although it sends all the LSAs it receives to the LSAR as a part of the flooding procedure. A side benefit of this approach is that instability in the LSAR does not impact other routers in the network. It is important to mention that keeping the adjacency in an intermediate state indefinitely does not generate alarms on

the commercial routers we have tested.

We modified GateD routing software [12] to implement the LSAR. Modifications include steps to ensure:

1. The LSAR does not originate any LSAs other than its Router LSA which is used as LSA L as mentioned above. It does not send out any Link State Update packets, which makes sure that its Router LSA never actually goes out.
2. The LSAR does not perform any SPF computation. This is because we do not want it to act as a router.
3. The LSAR does not take part in flooding. So, even if it is connected to two routers and receives a new instance of an LSA from one of the routers, it does not flood the LSA out to the other router. This is consistent with the fact that the LSAR does not send out any Link State Update packets.
4. The LSAR does not refresh LSAs, although it ages the LSAs while they reside in the link state database like a normal OSPF router. Since the LSAR does not refresh LSAs, it never flushes an LSA out when its age reaches MaxAge (normally 1 hour).
5. The LSAR is not allowed to become the Designated Router (DR) or the Backup DR (BDR) on a LAN even if the configuration file specifies a non-zero priority for it.
6. The LSAR accepts any LSA it receives irrespective of whether the copy of the LSA is newer or older than the received copy.
7. The LSAR does not describe any LSAs in the Database Description packets other than its own Router LSA (described in database exchange but never transmitted over the link).

Our current implementation of the LSAG is in Java. It communicates over a TCP connection with the LSAR and fills a data model [13] using LSAs received from the LSAR. We are enhancing the LSAG to provide various management and monitoring functionalities.

V. EVALUATION

In this section we present a detailed evaluation of management and control plane approaches described in the previous section. We use three main criteria for evaluation: operational issues, reliability and timeliness. We use a simulation-based study for evaluating reliability and timeliness issues.

A. Operational Issues

We consider three operational issues: ease of deployment, overhead of the approach, and correctness in acquiring topology information.

First consider deployment issues. With the control plane approach, in order to monitor an OSPF area, the LSAR must establish an OSPF adjacency with at least one router in the area, which implies it must have a layer-2 or a physical adjacency with that router. As non-backbone areas are commonly deployed to cover different geographical regions, several LSARs would be needed to cover the entire AS. An LSAR's need for direct connectivity to an area it monitors may be overcome for area 0, through the use of *virtual links*

[2]. In general, it is worth exploring whether this requirement can be largely overcome by drawing on semi-static information from router configuration files and Summary LSAs. With the management plane approach, the topology server may be realized as one box, which need not be physically adjacent to routers in the areas being monitored. However, to control the amount of messaging and processing, one may use multiple boxes in a larger network.

Next consider the overheads of the approaches. With the control plane approach, every router that the LSAR establishes an adjacency with needs to maintain one extra adjacency. The router has to send all the LSAs to the LSAR over this adjacency. Moreover, the LSAR-LSAG communication introduces extra data traffic in the network. With the management plane approach, all the routers have to respond to SNMP queries. They also have to generate TRAPs when the topology changes. A basic engineering rule in router design is that forwarding and routing tasks have top priority, followed by network management tasks, such as SNMP processing [14]. Therefore, SNMP processing has little interference with protocol work or with packet forwarding on the router. SNMP queries and TRAPs also introduce additional traffic in the network. An additional overhead is the need for the server to register with all routers.

Finally, consider correctness. The correctness of the control plane approach depends directly on the correctness of the prevailing OSPF implementations in the network, which are essential to network and are as a result quite robust. The LSARs obtain the same OSPF information as the routers and arrive at the same view of the topology. In contrast, the management plane approach depends on complete and correct implementation of the SNMP MIB and TRAPs. Vendor support for SNMP is often non-standard, incomplete and inadequate. OSPF MIBs and TRAPs are not always implemented by all vendors. This may limit the ability of the management plane approach in acquiring correct and complete view of the topology.

B. Reliability and Timeliness

In this section, we compare the two approaches in terms of how fast and reliably the topology server is informed of a topology change. When a topology change happens, the management plane approach uses an SNMP TRAP to inform the server of the change as we saw in the previous section. The control plane approach on the other hand relies on an LSA for the same purpose. Therefore, we evaluate how TRAPs and LSAs compare in terms of reliability and speed in propagating a topology change. When we wish to generically refer to either of them, we will use the term *herald* in this section.

The process of informing the server about a change can be divided into (1) the interval of time between the change event and the generation of a herald message (LSA or TRAP), and (2) the propagation of the herald to the server.

B.1 Herald Generation

The generation of an LSA or a TRAP is affected by the event in question, the flow control mechanisms in place (like the *MinLSInterval* [1] and TRAP throttling [11]) and implementation dependent delays in LSA or TRAP generation. Flow control mechanisms place a limit on how frequently LSAs or TRAPs can be generated. The *MinLSInterval* determines how frequently a new instance of an LSA can be generated and is a fixed architectural constant having a value of 5 seconds [1]. Similarly, the TRAP throttling mechanism [11] allows only up to a certain number of TRAPs to be generated within a given time window. The recommended values for OSPF TRAPs are a maximum of 7 TRAPs within a window of 10 seconds. Further, the throttling is not applied to certain kinds of TRAPs [11]. Finally, implementation dependent delays also have a significant impact on herald generation time. Often, LSA or TRAP generation times are the major components of the total delay before the change reaches the server.

B.2 Herald Propagation

Once the herald is generated, its propagation depends on two factors: (1) the time taken to travel across one hop in the network, and (2) the path taken by the herald from the source of the change to the server.

First let us consider per-hop propagation time. Assuming that packet forwarding in routers takes minimal time, the TRAP undergoes propagation and queuing delays on each hop to the server. On the other hand, the LSA undergoes not only queuing and propagation delays but also some processing and pacing delays. In our measurements on Cisco routers, we found LSA processing delays to be in sub-millisecond range, however pacing delays can be substantial, often in the range of tens of milliseconds [15].

Next consider the path taken by a herald. Since LSAs are flooded, they can traverse multiple paths from their source to the server. This means that an LSA indicating a change takes the shortest possible path in terms of delay from the source to the server. In comparison, for the TRAP, each router on the path decides the next hop for the TRAP based on the minimum weight path from itself to the server. The hop count or delay may or may not constitute the metric determining the weight of a path. Thus, the minimum weight path might not be the one with minimum delay or minimum number of hops. Even if hop count or delay constitutes the metric for determining weight, because of routing transients, the actual path taken by the TRAP from source to the server might not be the globally minimum weight path.

Thus, a number of factors can contribute to the reliability and speed of herald propagation. To gain more insight, we performed a simulation-based study to understand some of these factors and how they affect herald propagation. In our simulations, we took the approach of fixing the location of a topology change and observing how the herald propagates to each node in the topology via two approaches. The idea was to observe the arrival of herald at every possible location of the topology server. In our experiments,

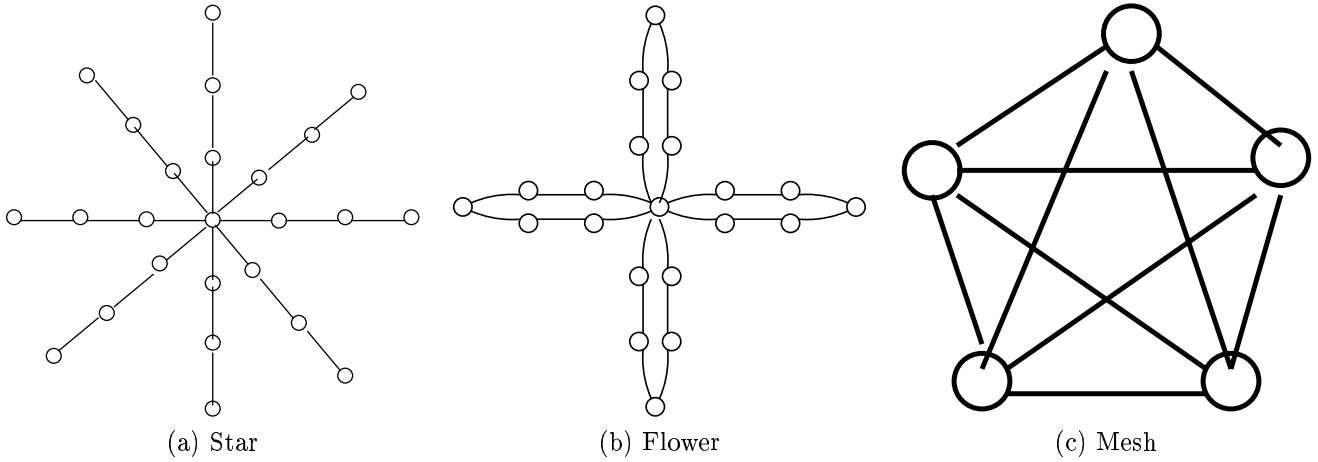


Fig. 3. Simulation Topologies

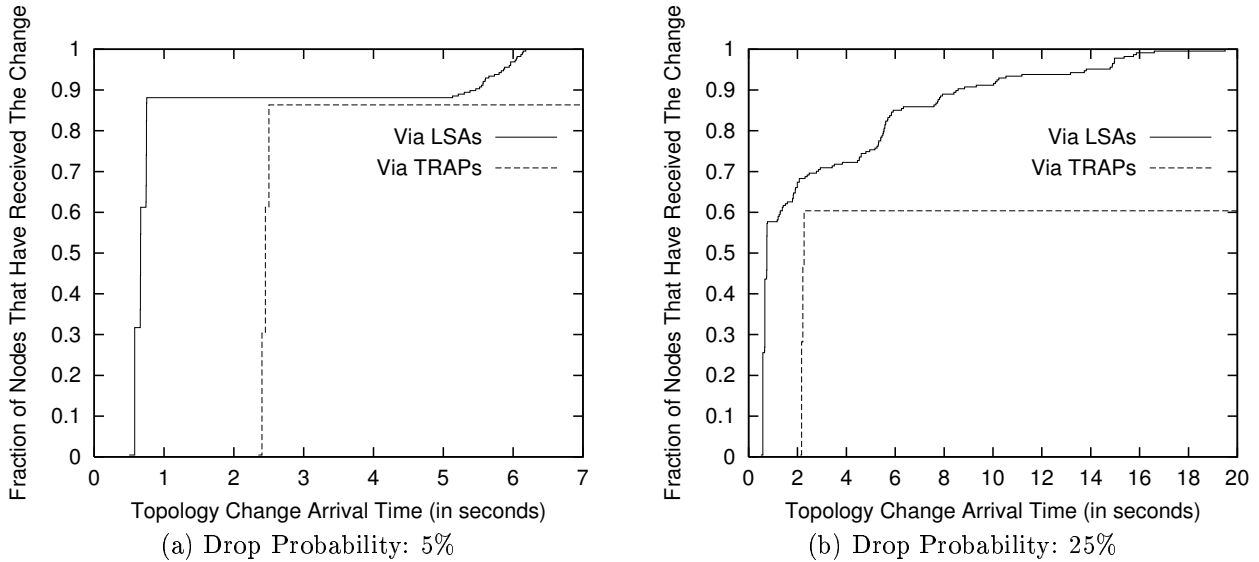


Fig. 4. Star topology with 75 rays and radius 3: Arrival of topology change via LSAs and via TRAPs.

we used an *adjacencyUp* event on an interface of a router as the topology change event. Once the adjacency came up, the router in question originated a Router LSA or a TRAP (depending on the approach being simulated) which served as the herald. We have observed delays of 0.5 seconds and 2.2 seconds for the generation of LSA and TRAP respectively after *adjacencyUp* event on Cisco routers [13]. We used these values for generation delays in our simulations.

We enhanced the NS2 simulator [16] to implement essential features of OSPF and SNMP. We used default values of all architectural and configurable constants for OSPF [1]. To determine realistic OSPF processing delays for tasks such as LSA processing and SPF calculation, we performed extensive measurements on Cisco 3660, 7513 and GSR routers [15]. The results were then used to parameterize the OSPF model. More details of our modifications to NS2 can be found in [13].

Parameter Space

In our simulations, we considered the effect of the following two factors on herald propagation:

1. **Topology:** The size and connectivity of the net-

work topology determine the path taken by a herald. For TRAPs, the larger the hop count from the point where the change takes place to the topology server, the smaller the probability of the TRAP making its way to the topology server. For LSAs, on the other hand, topology determines the number of paths from the source to the server and number of hops on each path.

In our simulations, we explored the influence of the topology by first choosing simple regular topologies (Figure 3) that allowed us to understand the impact of hop count and multiple paths on the propagation of LSAs and TRAPs. Subsequently, we used a more realistic topology similar to area 0 of an AT&T's IP network.

2. **Network Characteristics:** The load and characteristics of traffic on the network impact the per-hop propagation of a herald. First of all, they determine the queuing delay along each hop thereby affecting the speed at which a herald propagates. Additionally, they affect reliability since heralds can get lost due to link congestion and buffer overflows. In our simulations, we capture the impact of network traffic as a loss probability (p) for packets in the

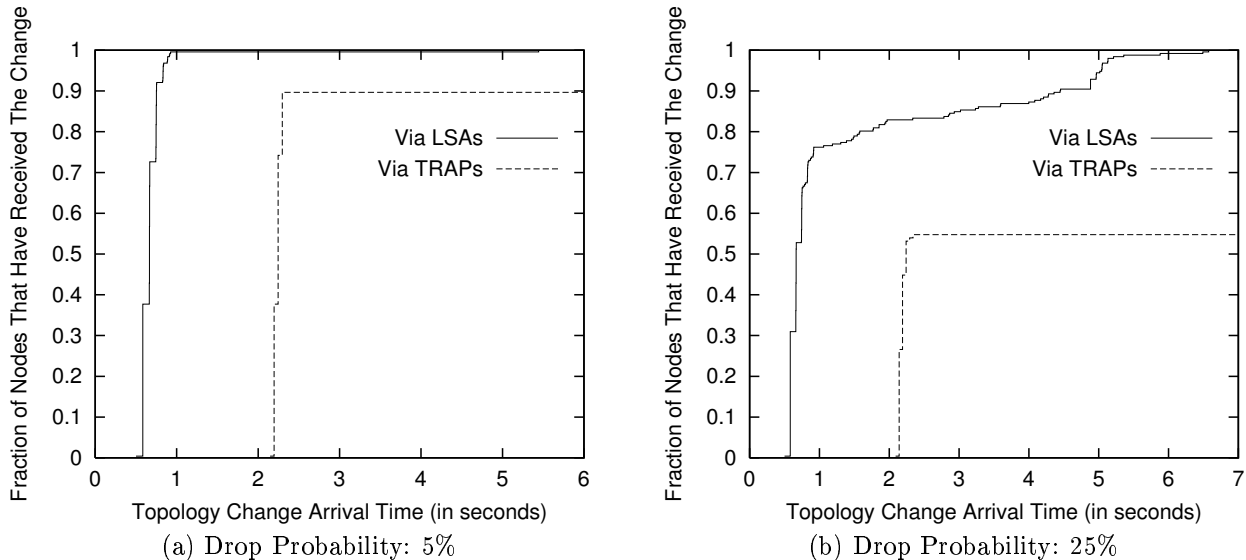


Fig. 5. A Flower with 50 petals and 3 hops in each petal: Arrival of topology change via LSAs and via TRAP.

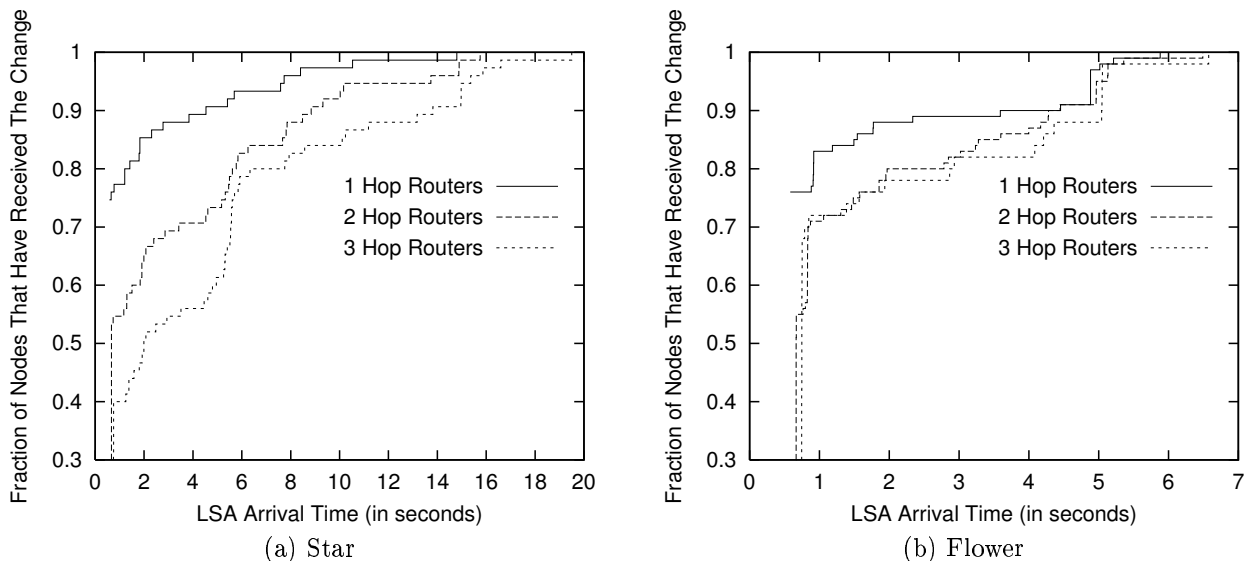


Fig. 6. Arrival of LSAs as hop count increases for star and flower topologies.

network. This loss model seems simplistic in that successive packet losses are assumed to be independent, as are losses on distinct links. However, such a model is not too far from reality since OSPF implementations for commercial routers cause successive packets containing LSAs to be paced using time delays that are large enough to prevent strong correlations in packet loss events for OSPF LSAs.

Let us describe simulation results now. The results show arrival of a herald at all the nodes of the topology after the *adjacencyUp* event. The herald arrival time also includes herald generation delay (0.5 second for LSA generation and 2.2 second for TRAP generation).

Single Path Characteristics

We start with the simplest topology, the star (Figure 3(a)), and make a few observations about herald propagation. Suppose the *adjacencyUp* event occurs at the center of the star. Thus, a herald is generated at the center of the star which is propagated to all the other nodes. Note that there is a single path from the center to every other node

in the topology.

We present results for a star with 75 rays and a radius of 3 (75 nodes each at distance 1, 2 and 3). Figure 4 shows the arrival of topology change via LSAs and TRAPs for all nodes. The fraction of nodes that get the LSA or the TRAP without loss is $1/3 \times ((1-p) + (1-p)^2 + (1-p)^3)$ as expected. Upon loss, a TRAP is not retransmitted, which means that a subset of nodes never learn about the topology change via TRAP. The LSA, if lost, is retransmitted as many times as required. We used the default value of 5 seconds as the retransmission interval in the simulations [1].

Notice that for the drop rate of 5%, nodes that missed the LSA on the first transmission get it after 5 seconds. However, for the drop rate of 25%, some of the nodes that missed the LSA on the first transmission, get the LSA earlier than the time the LSA would have been retransmitted. Many scenarios are responsible for this behavior. One such scenario arises because at such a high loss rate, when the topology change occurs, a number of adjacencies involving

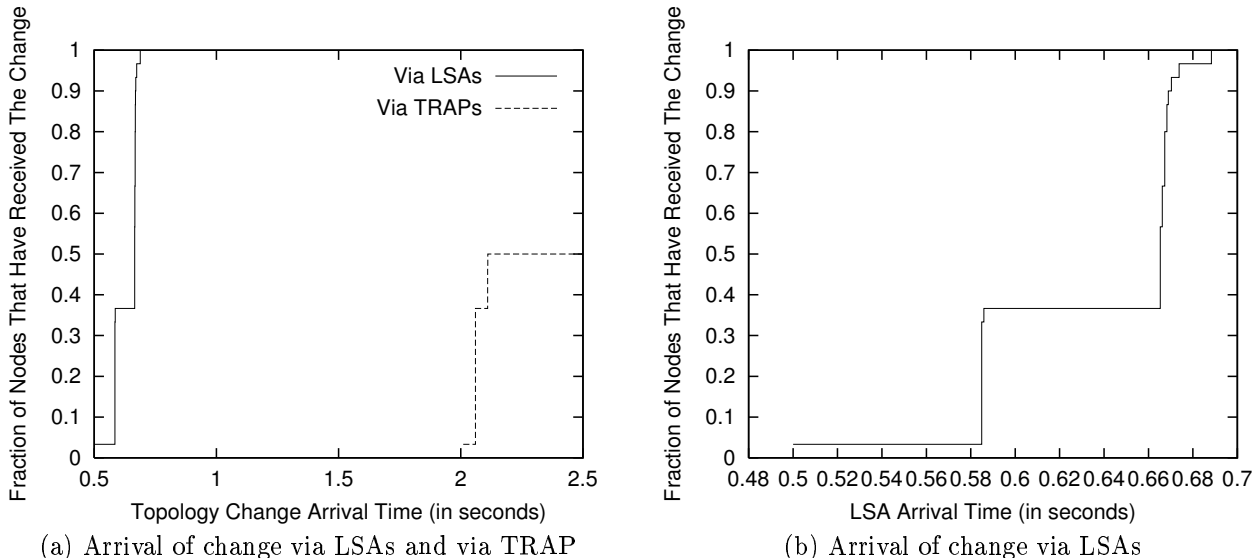


Fig. 7. Results for a mesh of 30 routers (drop probability: 50%)

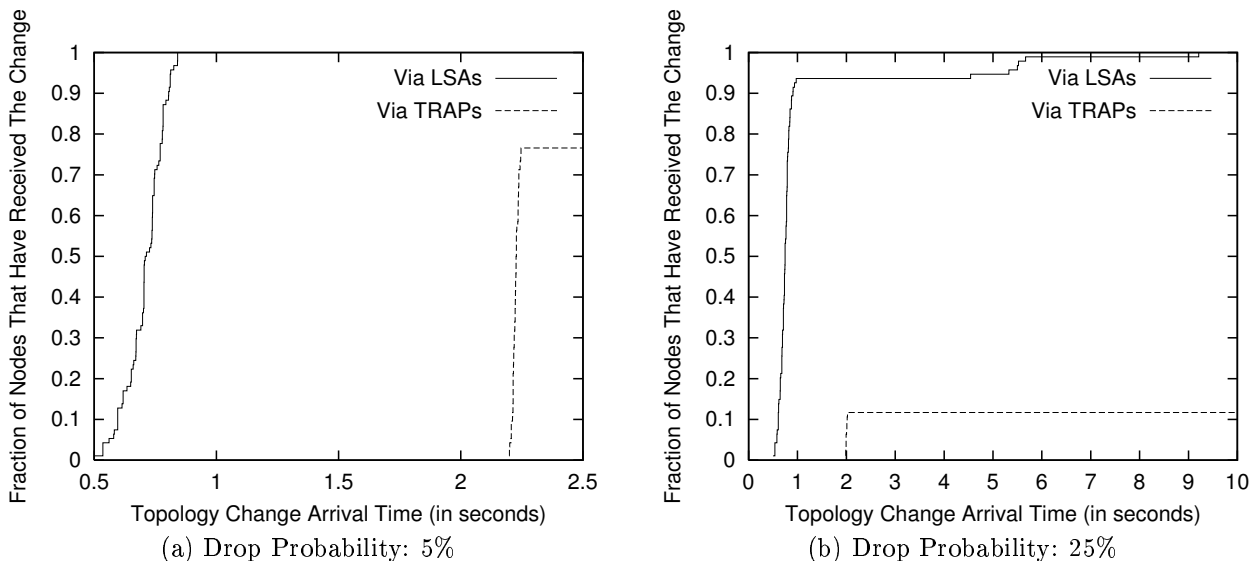


Fig. 8. Simulated OSPF backbone: Arrival of topology change via LSAs and via TRAP.

the central node can be at intermediate stages of establishment. As soon as such adjacencies are fully established, the central node re-originates its Router LSA which propagates the earlier topology change we are tracking.

Multi-path Characteristics

In the case of the star topology, each node has only one path from the center which is taken by TRAPs as well as LSAs and hence the fraction of nodes receiving the TRAP and those receiving the LSA without any retransmission is the same. However, if there are multiple paths for the LSA to reach a node, as with the flower and mesh topologies, the packet loss probability has less of an impact on the arrival of a topology change via LSAs. This effect is evident from Figure 5 which shows herald propagation for a flower topology with 50 petals and 3 hops per petal (see Figure 3(b)). For the flower topology, each node can receive an LSA from the center (where the change takes place) on two separate paths. As the packet drop rate increases from 5% to 25% (Figure 5), the fraction of nodes receiving the

LSA without retransmission becomes much larger than the corresponding fraction receiving the TRAP.

To illustrate how having an alternate path benefits LSA propagation, Figure 6 compares how LSA arrival time varies at nodes 1, 2 and 3 hops away from the center for star and flower topologies. Note that for the flower topology, the fraction of nodes getting the LSA without retransmission is much higher than those for the star topology. The benefit is greatest for nodes 3 hops away from the center. For these nodes, there are two independent 3 hop paths for the LSA to take from the center in the flower topology as opposed to a single path in the star topology. For nodes at distance 1 or 2 from the center, the longer path from the center involves 4 and 5 hops respectively, and hence becomes less useful as the packet drop rate increases.

Thus, having even one alternate path benefits LSAs considerably. In the extreme case of a fully connected mesh (Figure 3(c)), LSAs have a large number of paths available to them. Therefore, even with a huge drop rate of 50%, all

nodes are likely to get the LSA without any retransmission, as can be seen from Figure 7. Note that only 40% of the nodes obtain the LSA directly from the source of change as can be seen from Figure 7(b). However, this population of 40% in turn floods the LSA to all the other nodes in the network (except the source). Thus, all the nodes receive the LSA roughly within the delay involved in LSA traversal of two hops without any need of retransmission.

Simulations with a Realistic Topology

After having understood the factors affecting the propagation of heralds in simple regular topologies, we now present results for simulations done using an example OSPF backbone topology similar to the AT&T's IP network. The topology used had about 100 nodes and 200 edges. We selected a node in one corner of the topology for the *adjacencyUp* event.

Figure 8 shows the simulation results for this topology. The average hop count of nodes from the source of the change was about 6. This explains why the fraction of nodes receiving the TRAP drops drastically as the drop rate increases from 5% to 25%. For LSAs, very few nodes require retransmissions even when drop rate is 25%. Thus, even though the topology is much less richly connected than a mesh, LSA propagation behavior is very similar to the mesh.

B.3 Discussion

The simulations very clearly bring out the reliability and robustness of the control plane approach even in the face of drastic network conditions. With the management plane approach, the lack of reliability of SNMP TRAPs is a serious issue. SNMPv2 provides some safeguards against losses that take place due to buffer overflows at a router through a notification called INFORM [5]. However, both TRAPs and INFORMs could be lost due to routing loops or in the absence of a path from source to server during transient periods when routing has not yet converged. Moreover, out of order INFORMs can cause inconsistencies in the topology view of the server [13]. On the other hand, LSAs are flooded reliably which ensures LSAs always reach the server (LSAR to be precise) unless the network is partitioned. Flooding also provides multiple paths over which LSAs can reach the server which increases the robustness of the approach. Moreover, since LSA flooding does not depend on the routing, LSAs reach the server even during routing transients during which normal forwarding might fail provided the network remains connected.

VI. CONCLUSIONS

In this paper we addressed the problem of tracking the intra-domain routing topology in real-time. Such topology tracking is essential in building any kind of value-added services to the network. In particular, we identified various design choices, and evaluated them by comparing advantages and disadvantages of these choices.

We described two main approaches for building a topology server: control plane and management plane. The control plane approach requires listening to the OSPF mes-

sages (LSAs), each of which describes a part of the topology and is flooded reliably. The main issue with this approach is how to make it passive and non-intrusive. On the other hand, the management plane approach relies on SNMP and involves using TRAPs for notification of topology changes and GETs for polling. We provided a detailed description of our implementation of a topology server based on each of these approaches. We also compared these two approaches in terms of operational issues, reliability, and timeliness of a topology change information.

We performed simulations to provide some quantitative idea as to how a topology change propagates via the control plane and the management plane approaches. The results clearly demonstrated the superiority of the control plane approach over the management plane approach in terms of reliability and robustness. The results brought out three main reasons for this. First, LSAs are flooded, so there are multiple paths over which the information regarding a topology change traverses to the topology server whereas TRAPs have only one path to get to the topology server. Second, LSAs are flooded reliably whereas TRAPs are not reliable. Third, LSAs are flooded hop-by-hop and hence do not depend on routing state whereas TRAPs do.

At first blush, it might seem that with the tremendous investment in SNMP based network management systems today, the obvious and natural approach to building an OSPF topology server is the management plane approach. However, as pointed out in the paper, significant operational and reliability issues loom large with this approach. The management plane approach can at best compliment the control plane approach because of its ability to provide early warning. Some applications might leverage this early warnings to do some useful precomputation.

To summarize, the paper shows that the control plane approach can be realized in a simple, extremely effective way to track the topology for link state protocols such as OSPF and IS-IS, the dominant intra-domain routing protocols. Our realization of the control plane approach achieves simplicity and reliability by separating LSA receipt from topology construction and higher level applications.

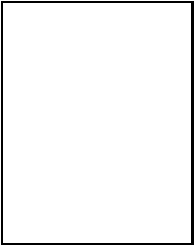
In the future, we plan to explore other systems needed to construct an IP network bandwidth broker, including a resource monitor and a policy controller. We also plan to investigate how varying various OSPF timer intervals can affect the detection of a topology change. Another direction for the future work is extending the topology server to track status "below" and "above" intra-domain routing like tracking physical level assets and inter-domain reachability information. Finally, there is the question of how to optimize placement of topology servers in the network. Often, the choices will be severely limited by operational constraints. However, when possible, one would like to place a control plane based topology server at the center of the network where it has multiple, small hop count paths for LSAs to arrive on from each router in the topology.

ACKNOWLEDGMENTS

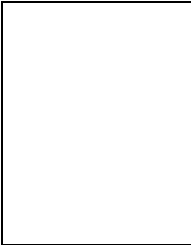
We are grateful to colleagues in AT&T Research for several enlightening conversations about this work: Brian Colbry, Bill Fenner, Joel Gottlieb, Tim Griffin, Jean-Philippe Martin-Flatin, and Jennifer Rexford. We would like to thank David Katz of Juniper Networks for answering several questions related to prevailing OSPF implementations.

REFERENCES

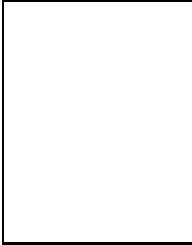
- [1] John Moy, "OSPF Version 2," Request for Comments 2328, April 1998.
- [2] J. T. Moy, *OSPF: Anatomy of an Internet Routing Protocol*, Addison-Wesley, January 1998.
- [3] R. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," RFC1195, December 1990.
- [4] C. Huitema, *Routing in the Internet*, Prentice Hall, 2000.
- [5] William Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*, Addison-Wesley, 1999.
- [6] Anja Feldmann and Jennifer Rexford, "IP network configuration for traffic engineering," Tech. Rep. 000526-02, AT&T Labs - Research, May 2000.
- [7] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter, "RATES: A server for MPLS traffic engineering," *IEEE Network*, pp. 34-41, March/April 2000.
- [8] E. Baccelli and R. Rajan, "Monitoring ospf routing," in *IM 2001: IFIP/IEEE International Symposium on Integrated Network Management*, May 2001.
- [9] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering internet topology," in *Proc. IEEE INFOCOM*, July 1998.
- [10] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. IEEE INFOCOM*, March 2000.
- [11] F. Baker and R. Coltun, "OSPF Version 2 Management Information Base," Request for Comments 1850, November 1995.
- [12] "The GateDaemon (GateD) Project," <http://www.gated.org>.
- [13] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K.K. Ramakrishnan, "An OPSF Topology Server: Design and Experience," Tech. Rep., AT&T Research, March 2001, full version.
- [14] "Cisco web site," <http://www.cisco.com>.
- [15] Aman Shaikh and Albert Greenberg, "Experience in Black-box OSPF Measurement," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2001.
- [16] "NS2 Simulator," <http://www.isi.edu/nsnam/ns/>.



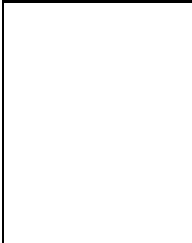
Aman Shaikh is a Ph.D. candidate in Computer Engineering Department at the University of California, Santa Cruz. He obtained an M.S. in computer Engineering from the same department in 2000, and a B.E. (HONS) in Computer Science and an M.Sc. (HONS) in Mathematics from the Birla Institute of Technology and Science, Pilani, India in 1998. His current research interests include routing and network management.



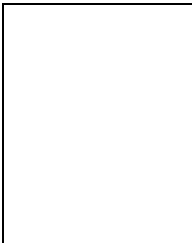
Mukul Goyal is currently a Ph.D. candidate in the CIS Department at the Ohio State University. He obtained an M.S. from the same department in 1999 and a B.Tech. in Electronics and Telecommunications from the Regional Engineering College, Kurukshetra, India in 1995. From 1995 to 1997, he worked as a Software Engineer at Siemens Communication Software Ltd., Bangalore, India.



Albert Greenberg received the B.A. degree in Mathematics from Dartmouth College in 1978, and the M.S. and Ph.D. degrees in Computer Science from the University of Washington in Seattle in 1981 and 1983, respectively. He joined AT&T Bell Labs Mathematics Research Center in 1983, and became a Dept. Head in the Network Services Research Center in 1995. In 1996, he moved to AT&T Labs-Research, where he heads the IP Network Management and Performance Dept., a group which conducts research in networking, with an emphasis on large scale IP networks and systems, and emerging Internet technologies. His research interests include Internet traffic measurement, modeling and engineering, policy based networking, and optical networking (IP/WDM). In collaboration with with several others in AT&T Labs, he is developing a unified toolkit to manage IP networks.



Dr. K. K. Ramakrishnan is a Founder and Vice President at TeraOptic Networks, Inc. Until recently, he was a Technology Leader at AT&T Labs Research. He holds a B.S. in Electrical Engineering from Bangalore University in India in 1976, an M.S. in Automation from the Indian Institute of Science in 1978 and a Ph.D. in Computer Science from the University of Maryland in 1983. From 1983 to 1994, he was with Digital Equipment Corporation working on Network Architecture and Performance. He was an editor for the IEEE/ACM Transactions on Networking and IEEE Network. His research interests are in design and performance of computer and communication network protocols and algorithms.



Raju Rajan is broadly interested in the areas of queuing theory, data networking and network management. Prior to his current position as CTO of Ipsum Networks, Dr. Rajan held research positions at AT&T Labs in the IP Network Management and Performance Department, and at IBM T.J. Watson Research Center in the IP Networking Department. Raju Rajan holds an M.S. in Statistics from Indian Statistical Institute, New Delhi, an M.S. in Mathematics and a Ph.D. in Electrical Engineering from University of Wisconsin, Madison.