OSPF Monitoring: Architecture, Design and Deployment Experience

Aman Shaikh
AT&T Labs - Research
Florham Park, NJ 07932, USA
ashaikh@research.att.com

Albert Greenberg
AT&T Labs - Research
Florham Park, NJ 07932, USA
albert@research.att.com

Abstract

Improving IP control plane (routing) robustness is critical to the creation of reliable and stable IP services. Yet very few tools exist for effective IP route monitoring and management. We describe the architecture, design and deployment of a monitoring system for OSPF, an IP intra-domain routing protocol in wide use. The architecture has three components, separating the capture of raw LSAs (Link State Advertisements – OSPF updates), the real-time analysis of the LSA stream for problem detection, and the off-line analysis of OSPF behavior. By speaking "just enough" OSPF, the monitor gains full visibility of LSAs, while remaining totally passive and visible only at the point of attachment. We describe a methodology that allows efficient real-time detection of changes to the OSPF network topology, flapping network elements, LSA storms and anomalous behavior. The real-time analysis capabilities facilitate generation of alerts that operators can use to identify and troubleshoot problems. A flexible and efficient toolkit provides capabilities for off-line analysis of LSA archives. The toolkit enables post-mortem analysis of problems, what-if analysis that can aid in maintenance, planning, and deployment of new services, and overall understanding of OSPF behavior in large networks. We describe our experiences in deploying the OSPF monitor in a large operational ISP backbone and in a large enterprise network, as well as several examples that illustrate the effectiveness of the monitor in tracking changes to the network topology, equipment problems and routing anomalies.

1 Introduction

Effective management and operation of IP routing infrastructure requires sound monitoring systems. With the advent of applications that require a high degree of performance and stability, such as VoIP and distributed gaming, network operators are now paying considerable attention to the performance of the routing infrastructure – its convergence, stability, reliability and scalability properties. Yet very few monitoring tools exist for effective routing management and operation. In this paper we present a monitoring system for one of the widely used intra-domain routing protocols, OSPF [1] by providing its detailed architecture and design. The OSPF Monitor has been deployed in two operational networks: a large enterprise network and an ISP network. It has proved to be a valuable asset in both networks. We provide several examples illustrating different ways in which the monitor has been used, as well as the lessons learned through these experiences.

We designed the OSPF Monitor to meet the following objectives:

1. **Provide real-time tracking of OSPF behavior.** Such real-time tracking can be used for (a) identifying problems in the network and helping operators troubleshoot them, (b) validation of OSPF configuration changes made for maintenance or traffic engineering purposes, and (c) real-time presentation of accurate views of the OSPF network topology.
2. **Facilitate off-line, in-depth analysis of OSPF behavior.** Such off-line analysis can be used for (a) post-mortem analysis of recurring problems, (b) generating statistics and reports about network performance, (c) identifying anomaly signatures and using these signatures to predict impending problems, (d) tuning configurable parameters, and (e) improving maintenance procedures.

There are two basic approaches for monitoring OSPF: rely on SNMP [2] MIBs and traps, or listen to Link State Advertisements (LSAs) flooded

by OSPF to describe the network changes. Our prior work [3] has shown the superiority of the LSA-based approach, so we take the approach of passively listening to LSAs for our OSPF Monitor. The monitor directly attaches to the network, and speaks enough OSPF to receive LSAs. These LSAs are then analyzed in real-time to identify network problems and validate configuration changes. LSAs are also archived for a detailed off-line analysis, for example, for identification and diagnosis of recurring problems. The monitor uses a three-component architecture to provide a stable, scalable and flexible solution. The three components are:

1. *LSA Reflector (LSAR)* which collects LSAs from the network,
2. *LSA aGgregator (LSAG)* which analyzes LSA streams in real-time to identify problems, and
3. *OSPFScan* which provides off-line analysis capabilities on top of LSA archives.

The paper describes these three components in detail and the benefits offered by this three-component architecture. Since the LSAR and LSAG are key to real-time monitoring, their efficiency and scalability are of utmost importance. We demonstrate the efficiency and scalability of the LSAR and LSAG in terms of network size and LSA rate through lab experiments.

This paper is organized as follows. We discuss related work in Section 2. Section 3 provides an overview of OSPF. Section 4 discusses the three-component architecture of the OSPF Monitor. Sections 5, 6 and 7 provide detailed description of these three components. Section 8 presents the performance analysis of LSAR and LSAG through lab experiments. In Section 9, we describe salient aspects of our experiences with deploying the monitor in commercial networks. Finally, Section 10 presents conclusions.

2 Related Work

Monitoring and analyzing dynamics of routing protocols have become active areas of research of late. Route monitoring systems have started to appear in the market-place from networking startups, such as Packet Design [4] and Ipsum Networks [5]. However, the products offered by these companies have appeared in the market after our OSPF Monitor was designed. Moreover, details about the architecture and implementation of these products are not available in the public domain. The IP monitoring project at Sprint [6] consists of an IS-IS listener and a BGP listener that collects IS-IS and BGP data from the Sprint network. Although a number of studies have appeared based on the data collected by these listeners, the actual architecture of the monitoring system has not received attention. Our prior work [7] and Watson *et al.* [8] presented case studies of OSPF dynamics in real networks. Although [7] used the OSPF Monitor described in this paper to collect and analyze the OSPF data for the case study, the paper did not focus on the design and implementation of the monitor itself. Neither did [8] focus on the design of the monitor. Route-Views [9] and RIPE [10] collect and archive BGP updates from several vantage-points; a number of research studies have benefited from this data. However, both Route-Views and RIPE merely collect BGP updates; they do not provide software for monitoring or analyzing the updates.

Recall that one of the design goals of the OSPF Monitor is to track the OSPF topology. Several studies have dealt with the discovery and tracking of the network topology. For instance, our prior work [3] described SNMP and LSA-based approaches for designing an OSPF topology server, and evaluation of these approaches in terms of operational complexity, reliability and timeliness of information. The evaluation showed the superiority of the LSA-based approach in terms of reliability and robustness over the SNMP-based approach. This paper extends the LSA-based approach for monitoring OSPF. The Rocketfuel project [11, 12, 13] tackled the problem of inferring ISP topologies and weight settings through end-to-end measurements. Feldmann *et al.* [14] described the approach of periodically dumping router configuration files of routers. This approach provides a static view of the topology. One can make it more dynamic by increasing the dumping frequency, but it is hard to go beyond certain limits because of the size of IP networks today. Lakshman *et al.* [15] mentioned approaches for real-time discovery of topology in their work on the RATES System for MPLS traffic engineering. But topology discovery was just one of the modules of their system and they did not go into details. Siamwalla *et al.* [16] and Govindan [17] discussed topology discovery methods that do not require cooperation from the network service providers, 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 or monitoring, the topic of this paper.

3 OSPF Overview

OSPF [1] is a link state routing protocol. With link state protocols, each router within the domain discovers and builds a complete and consistent view of the network topology as a directed graph. Each router represents a node in the graph, and each link between neighboring 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 defined in terms of link weights to their destinations. We will refer to the computation of the shortest path tree as an *SPF computation*, and the resultant tree as an *SPT*.

For scalability, an OSPF domain 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 area it is connected to. In general, a router does not learn the entire topology of remote areas (i.e., the areas in which the router does not have links), but instead learns the weight of the shortest paths from one or more border routers to each node in remote areas. In addition, the reachability of external IP prefixes (associated with nodes outside the OSPF domain) can be injected into OSPF. Roughly, reachability to an external prefix is determined as if the prefix were a node linked to the router that injects the prefix into OSPF.

Routers running OSPF describe their local connectivity in *Link State Advertisements (LSAs)*. These LSAs are *flooded* reliably to other routers in the network, which the routers use to build the consistent view of the topology described earlier. The set of LSAs in a router's memory is called the *link state database* and conceptually forms the topology graph for the router. A change in the network topology requires affected routers to originate and flood appropriate LSAs. For instance, when a link between two routers comes up, the two ends have to originate and flood LSAs describing the new link. Moreover, OSPF employs a periodic refresh of LSAs. The default value of the refresh-period is 30 minutes. So, even in the absence of any topological changes every router has to periodically flood self-originated LSAs. Due to the reliable flooding of LSAs, a router can receive multiple copies of a change or refresh triggered LSA. We term the first copy received at a router as *new* and subsequently received copies as *duplicates*.

Two routers are termed neighbor routers if they have interfaces to a common network (i.e, they have a 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.

4 Architecture

As mentioned earlier, the OSPF Monitor consists of three components:

1. **LSA Reflector (LSAR):** The LSAR captures LSAs from the network. Section 5 describes various modes used by the LSAR for network attachment. The LSAR sends the LSAs over a TCP connection to the real-time analysis component, and also archives them for off-line analysis.
2. **LSA aGgregator (LSAG):** The LSAG receives a stream of LSAs from one or more LSARs, and performs real-time analysis of the stream. The LSAG maintains and populates a model of the OSPF network topology (as described in Section 6), using the LSA stream.
3. **OSPFScan:** The OSPFScan is used for off-line analysis of the LSA archives. The OSPFScan implements a three step analysis method, described in Section 7.

Figure 1 depicts how the three components are deployed in an example OSPF network.

Separating real-time monitoring into the LSAR and LSAG components provides several benefits. Each function is simplified and can be replicated independently to increase the overall reliability. Another benefit is that the LSAG can selectively receive a subset of LSAs, for example, LSAs belonging to a given OSPF area. Furthermore, the LSAR has to reside close to the network to capture LSAs, and so must be very simple in order to achieve a high degree of reliability. Moreover, as shown in Figure 1, multiple LSAR boxes may be required to cover all

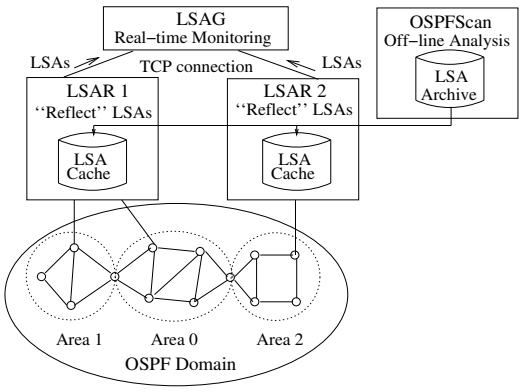

Figure 1: Three component architecture of the OSPF Monitor.

the areas since most LSAs only have an area-level flooding scope. Multiple LSAR boxes becomes almost a necessity 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 LSAG up and down without disturbing LSARs.

Separating the real-time analysis (LSAG) from the off-line analysis (OSPFScan) offers a number of benefits. The LSAG, being real-time, needs to be very reliable (24x7 availability) and efficient. This requires us to be extremely careful about what analysis capabilities are supported by the LSAG. The OSPFScan, on the other hand, is required to process a large volume of data as efficiently as possible and allow users to query the archives. However, it also has freedom in terms of what analysis capabilities it can support. Although, real-time and off-line analysis are implemented as separate components, they work hand in hand. Any analysis capability that is supported in real-time is also supported as an off-line playback.

5 LSAR

The LSAR captures LSAs from the network for real-time and off-line analysis. LSA traffic is reliably flooded by OSPF, not routed since LSAs need to be reliably communicated even when routing is impaired or broken. As a result, the LSAR has to be closely attached to the network it monitors. There are four choices for network attachment. Below, we describe these four choices along with their pros and cons:

1. **Wire-tap mode:** An obvious way of capturing LSAs from a network is to use a tap on a link of the network, either a physical tap or port forwarding on a layer-2 switch. We will generically refer to this way of capturing LSAs as the *wire-tap* mode. If done in the right manner, this allows one to capture LSAs in a completely passive manner. However, depending on the physical media or the layer-2 technology being used, wire-tapping may not be operationally feasible. As a result, the LSAR currently does not support wire-tap, though it should be relatively straight-forward to add a module that supports LSA capture via wire-tap.
2. **Host mode:** LSAs are exchanged via a multicast group *all-rtrs* on a broadcast network [1]. Thus, on broadcast networks, LSAs can be received by joining this group; we term this mode of network attachment as the *host mode*. In this mode, the LSAR does not have to establish any form of adjacency with operational routers in the network. As a result, the LSAR is completely invisible to the routers, which is the ideal situation for any passive monitoring system. However, there are a few disadvantages. First, the LSAR has to initialize the database as routers flood LSAs on the network during the refresh process. In the worst case, it can take one refresh cycle (30 minutes as per [1]) for the LSAR to receive the first copy of an LSA after it comes up. Second, OSPF's reliable flooding does not extend to the LSAR in the host mode. If the LSAR misses transmission of an LSA to the multicast group for any reason, the sending router will be unaware of this, and there will be no retransmission. Finally, the host mode can be used only on a broadcast capable media where LSAs are sent to the multicast group.
3. **Full adjacency mode:** On a point-to-point link where routers do not send LSAs to the multicast group, the LSAR has to establish an adjacency with a router to receive LSAs. We will refer to this mode as the *full adjacency mode*. In this mode, the LSAR cannot be completely invisible to the network. However, it is crucial to ensure that the LSAR has minimal impact on the network, and most importantly, other routers in the network never send data packets to the LSAR to be forwarded elsewhere. A natural line of defense is to use router configuration measures: assign high OSPF weights and install strict access control lists and route filters on the link to the LSAR. Another line of defense stems from the fact that the LSAR (by design) cannot send LSAs. As a result, the link from the LSAR to the

router it attaches to does not exist in the OSPF topology graph. Since OSPF uses a link for data forwarding only if both of its unidirectional edges exist in the graph [1], this ensures that the LSAR-router link cannot be used for data forwarding. However, the LSAR might still have an impact on the network since the router advertises a link to the monitor in its router LSA. If the LSAR or its link with the router is flapping (going up and down), the associated adjacency can start flapping, triggering SPF calculations around the network.

4. **Partial adjacency mode:** To prevent network-wide SPF calculations when the adjacency between the LSAR and the router is flapping in the full adjacency mode, we can keep the adjacency in an "intermediate" state at the router, so that the router does not include a link to the LSAR but still sends LSAs to it. We refer to this mode as the *partial adjacency mode*. To keep the LSAR-router adjacency in the intermediate state, the LSAR describes a "fake" LSA to the router during the link state database synchronization process but never actually sends it out to the router. As a result, the database is never synchronized, the adjacency stays in OSPF's *loading* state, and is never fully established. Note that this is permissible under the OSPF specification [1]. With the partial adjacency, instability at the LSAR does not impact other routers in the network, which makes for an attractive choice. However, there are two potential issues. First, with an adjacency in the intermediate state, the router cannot delete LSAs from its link state database [1]. In the worst case, this might lead to memory exhaustion on the router. We deal with this problem by periodically dropping the partial adjacency (so that the router has a chance of garbage collecting the link state database) and then re-establishing the partial adjacency after a short time interval. In our implementation, the LSAR drops its adjacency once every 24 hours for a five second period. While there is a possibility for the LSAR to lose data during this five second period, we believe that chances of this happening are rare. Second, the use of the partial adjacency is a deviation from the normal behavior of OSPF. Keeping an adjacency in the *loading* state on a router for a long time might generate alarms, or might cause the router to drop the adjacency. However, we have not observed this problem with the commercial routers we have tested.

6 LSAG

As mentioned in Section 4, the LSAG receives a stream of LSAs from the LSAR for real-time analysis. The LSAG prints messages on the console when it detects changes to the network topology or a behavior that does not conform to the OSPF standards. These messages allow operators to identify problems in the network. Section 6.1 provides more details about various types of messages generated by the LSAG. Under the hood, the LSAG maintains and updates a snapshot of the network topology to identify topological changes and anomalous behavior. Section 6.2 provides a detailed description of the model and how it helps LSAG print console messages. This topology model also allows the LSAG to dump topology snapshots periodically and upon topological changes. These snapshots in turn can be used by applications that might benefit from them.

6.1 Classification of Real-time Messages

(1) Topology Change Messages

- **(a) Change messages insides an area**
  - **(i) Messages about a router**
    - RTR UP
    - RTR DOWN
    - BECAME BORDER RTR
    - NO LONGER BORDER RTR
    - BECAME ASBR
    - NO LONGER ASBR
  - **(ii) Messages about an interface on a router**
    - INTF DOWN
    - INTF MASK CHANGE
  - **(iii) Messages about an adjacency**
    - ADJACENCY UP
    - ADJACENCY DOWN
    - ADJACENCY COST CHANGE
    - ALL ADJACENCIES DOWN
  - **(iv) Messages about a host−route on a router**
    - STUB LINK UP
    - STUB LINK DOWN
    - STUB LINK COST CHANGE
  - **(v) Messages about DR on a broadcast network**
    - NEW DR
    - NO LONGER DR
    - DR CHANGE
    - MASK CHANGE
- **(b) Change messages for remote areas**
  - **(i) Messages about a prefix in a remote area**
    - TYPE−3 ROUTE ANNOUNCED
    - TYPE−3 ROUTE WITHDRAWN
    - TYPE−3 ROUTE COST CHANGE
  - **(ii) Messages about an ASBR in a remote area**
    - TYPE−4 ROUTE ANNOUNCED
    - TYPE−4 ROUTE WITHDRAWN
    - TYPE−4 ROUTE COST CHANGE
- **(c) Change messages for external routes**
  - TYPE−5 ROUTE ANNOUNCED
  - TYPE−5 ROUTE WITHDRAWN
  - TYPE−5 ROUTE COST CHANGE
  - TYPE−5 ROUTE COST_TYPE CHANGE
  - TYPE−5 ROUTE FORW_ADDR CHANGE

(2) Flap Messages

- RTR FLAP
- INTF FLAP
- ADJACENCY FLAP
- STUB LINK FLAP
- TYPE−3 ROUTE FLAP
- TYPE−4 ROUTE FLAP
- TYPE−5 ROUTE FLAP

(3) Messages related to Anomalous Behavior

- NON−BORDER RTR YET TO WITHDRAW TYPE−3/4 ROUTES
- NON−ASBR YET TO WITHDRAW TYPE−5 ROUTES
- DUPLICATE ADJACENCY
- DUPLICATE STUB LINK
- TYPE−3 ROUTE FROM NON−BORDER RTR
- TYPE−4 ROUTE FROM NON−BORDER RTR
- TYPE−5 ROUTE FROM NON−ASBR

(4) LSA Storm Messages

- LSA STORM

Figure 2: Classification of the LSAG messages.

Figure 2 shows classification of various message types supported by the LSAG. Each message contains a time-stamp, and a message type along with the attributes. The message type is used for identifying the kind of event or problem in the network. For example, a "RTR DOWN" message is issued when OSPF process on a routes dies. The attributes provide more detail about the message. For example, if the message type is "RTR DOWN", the attributes include the router-id of the associated router.

Let us describe the four top-level categories shown in Figure 2:

1. **Topology Change Messages:** These messages are generated for changes in the network topology. As Figure 2 shows, majority of message types fall into this category. These messages help operators identify problems in the network, as well as help them validate maintenance activities.
2. **Flap Messages:** These messages are generated for network elements (e.g., router, link etc.) that go up and down repeatedly. These messages are always preceded by topology change messages. For example, a "RTR FLAP" message is preceded by several "RTR DOWN" and "RTR UP" messages. These messages are useful for getting an operator's attention. They also act as early warning signs to the network stability.
3. **Messages related to Anomalous Behavior:** These messages are generated when the observed behavior deviates from the expected behavior of OSPF. An example of an anomalous behavior is a non-border router originating summary LSAs (the corresponding message type is "TYPE-3 ROUTE FROM NON-BORDER RTR"). Often these messages indicate configuration errors or bugs in the vendor software.
4. **LSA Storm Messages:** These messages are generated when too many refresh instances of an LSA are observed by the LSAG. Often these messages indicate bugs in the vendor software. They also act as early warning signs to the network stability.

6.2 Topology Model

The LSAG uses a topology model to generate messages described in the previous section. Apart from generating these messages, the LSAG is also required to efficiently support real-time queries about the network topology, such as "how many routers does area X have?" or "how many interfaces does router X have in area Y?". This requires a model that can be updated and searched efficiently, and can be scaled to networks consisting of hundreds of routers and thousands of links.

We have designed and implemented a model meeting these goals, which consists of the following six classes:

1. *Area*: represents an OSPF area.
2. *Rtr*: represents a router.
3. *AreaRtr*: represents area-specific parameters of a router. Recall that OSPF does not assign a router to an area. It assigns each interface of a router to an area. Thus, a router can have some interfaces in each of several areas. An *AreaRtr* object contains the set of interfaces a router has in a given area.
4. *Ntw*: represents a subnet or a prefix.
5. *Intf*: represents an interface of a router.
6. *Link*: represents a unidirectional weighted edge in the topology. The monitor classifies links into three types: intra-area links that represent link between a pair of routers or a router-subnet pair, inter-area links to represent summary routes injected by border routers, and external links to represent external routes redistributed into OSPF. The local and remote ends of each link are objects of one of the above mentioned five classes.

Figure 3 shows the model in terms of "containment" relationship between objects of these six classes. For example, at the highest level, the model consists of a set of *Area* objects representing the areas of the OSPF network. Each *Area* object in turn contains a set of *Ntw* objects representing all the subnets of the associated area. *AreaRtr* objects are special. Since each *AreaRtr* object represents area-specific parameters of a router, it is contained in two objects: a *Rtr* object, and an *Area* object. In our implementation, search, add and delete operations on the objects are implemented via hash tables, enabling scaling to large networks.

Upon receiving an LSA, the LSAG updates the relevant part of the topology model. As an example, consider what happens when the LSAG receives a router LSA. The LSAG has to update the *AreaRtr* object that represents the router LSA. This may result in addition or deletion of interfaces, or a change in the administrative weight of interfaces. Router LSA can also result in addition

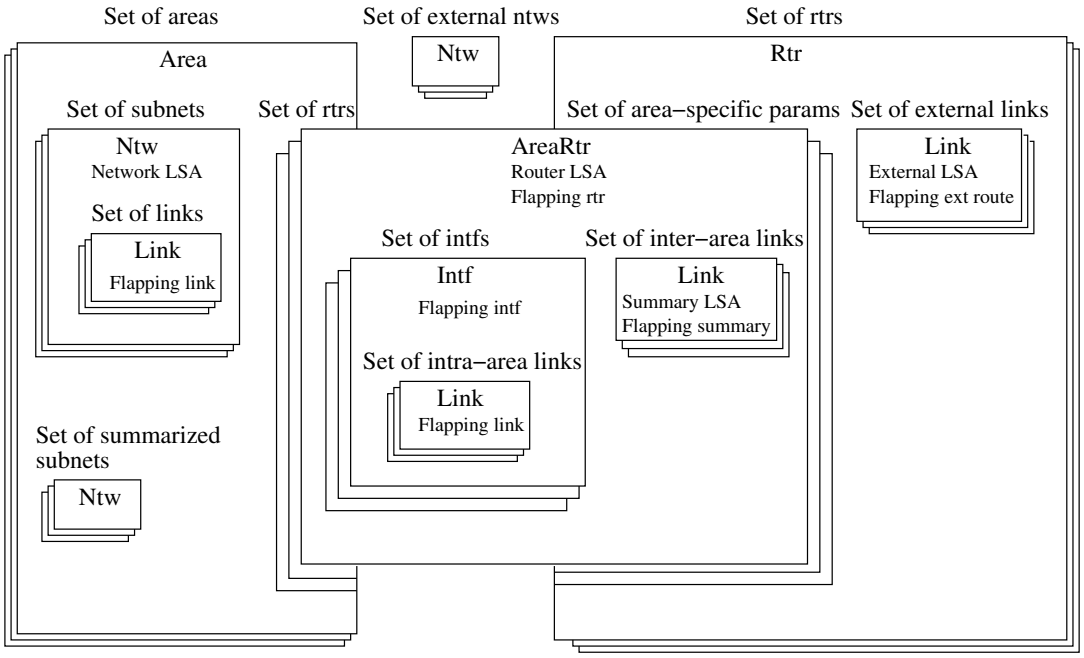

Figure 3: Topology model of the LSAG.

or deletion of an *AreaRtr* object.

The topology model conveniently allows the identification of a flapping node and generation of the associated flap message (see Section 6.1). The LSAG considers a node to be flapping if the node goes down and comes up n times within a t second time frame, where both n and t are configurable parameters. To identify flaps, each node of the OSPF topology is mapped to a specific object of the topology model. This mapping is shown in Figure 3 by statements of the form "flapping xxx" under appropriate objects. For example, an external route is mapped to a *Link* object in the set of external links in the *Rtr* object representing the advertising router.

The topology model is also used for identifying LSA storms. As described in Section 6.1, the LSAG issues an LSA storm message if many new copies of a refresh LSA are received within a short time period. Note that LSAs that indicate change(s) to the model are not considered a part of an LSA storm. More precisely, the LSAG issues a warning about too many LSA copies if it receives n refresh copies of an LSA within t seconds. To identify LSA storms, each LSA is mapped to a specific object of the topology model. The mapping is intuitive in the sense that the LSA essentially describes the status of the object it is mapped to. For example, each router LSA is mapped to an *AreaRtr* object. Similarly, each external LSA is mapped to a *Link* object in the set of external links of a *Rtr* object.

7 OSPFScan

As mentioned in Section 4, the OSPFScan is used for off-line analysis of LSA archives. At present, the OSPFScan provides the following functionalities:

1. **Classification of LSA traffic.** The OSPFScan allows various ways of "slicing-and-dicing" of LSA archives. For example, it allows isolating LSAs indicating changes from the background refresh traffic. As another example, it also allows classification of LSAs (both change and refresh) into new and duplicate instances. We have used this capability of the OSPFScan to analyze one month worth of LSA traffic as a case study for the enterprise network [7].
2. **Modeling topology changes.** Recall that OSPF represents the network topology as a graph. Therefore, the OSPFScan allows modeling of OSPF dynamics as a sequence of changes to the underlying graph where a change represents addition/deletion of vertices/edges to this graph. Furthermore, the OSPFScan allows a user to analyze these changes by saving each change as a single topology change record. Each such record contains information about the topological element (vertex/edge) that changed along with the nature of the change. For example, a router is treated as a vertex, and the record contains the

OSPF router-id to identify it. As another example, a link between a pair of routers is treated as an edge, and the corresponding record uses router-ids of the two ends to identify the link. We have used change records for a detailed analysis of router/link availability as we will see in Section 9.1.2.

3. **Emulation of OSPF routing.** The OSPFScan allows a user to reconstruct the routing table of any given set of routers at a given point of time based on the LSA archives. For a sequence of topology changes, the OSPFScan also allows the user to determine changes to these routing tables. Together, these capabilities allow the user to determine an end-to-end path through the OSPF domain at a given time, and see how this path changed in response to network events over a period of time.
4. **Statistics and reports.** The OSPFScan allows generation of statistics and reports on specific OSPF dynamics and anomalies over given time intervals. A simple example is the ability to count the number of change, new and duplicate LSAs over a given time period.
5. **Correlation with other data sources.** The functionalities provided by the OSPFScan form a basis for correlating OSPF data with other data sources such as usage data (e.g., SNMP statistics and Cisco netflow statistics), fault data (e.g., SNMP traps and syslogs), network inventory and topology data (e.g., router configuration files), other dynamic routing data (e.g., BGP updates), and maintenance data (workflow logs). For example, the routing table entries generated by the OSPFScan have been used by Teixeira *et al.* [18] to analyze the impact of OSPF changes on BGP routing.

The OSPFScan implements a three-step procedure to analyze each LSA record. These three steps include parsing the LSA, testing the LSA against a query expression, and analyzing the LSA if it satisfies the query. The OSPFScan allows a user to specify the query expression and the kind of analysis to be carried out with the LSAs.

The parsing step converts each LSA record of the archive into a *canonical form*. The query expression is applied to the canonical form, and not to the raw LSA record. The use of a canonical form makes it easy to adapt OSPFScan's functionality to support LSA archive formats other than the native format used by the LSAR. Adaptation only requires addition of a routine to parse the new format into the canonical form. The query language supported by the OSPFScan has a C-style expression syntax. An example query expression is "areaid == '0.0.0.0"' which selects all the LSAs belonging to area 0. The OSPFScan uses an internally developed data stream scan library which allows efficient processing of arbitrary data, described via a canonical form for each data type. The OSPFScan also allows further analysis of the information derived from the LSA archives such as topology changes and routing entries by implementing a similar three-step procedure.

8 Performance Evaluation

In this section, we characterize the performance of the monitor through lab experiments. We focus on the LSAR and LSAG which are central to the real-time monitoring, and analyze how these two components scale with the LSA-rate and the network size.

8.1 Methodology

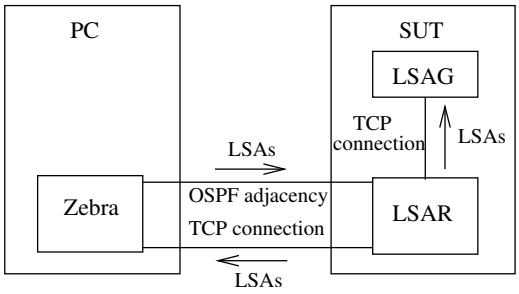

Figure 4: Experimental setup for measuring the LSAR and LSAG performance.

Our experimental setup consists of two hosts as shown in Figure 4. The host denoted by *SUT (System Under Test)* runs the LSAR and LSAG. The other host runs a modified version of Zebra [19]. The modifications include the ability to emulate a desired OSPF topology and changes to it by sending appropriate LSAs over an OSPF adjacency, and the ability to form an LSAG session with the LSAR to receive LSAs.

With this setup, we start an experiment by loading the desired topology into the LSAR running on the SUT. We use a fully connected graph having n nodes as the emulated topology. Once the desired topology is loaded at the LSAR, Zebra sends out a burst of back-to-back LSAs to the LSAR; we will denote the number of LSAs in a burst by l. These bursts are repeated such that there is a gap of inter-burst time (i) between the

beginning of successive bursts. Thus, every experiment instance consists of four input parameters: the number of nodes (n) in the fully connected graph, the number of LSAs in a burst (l), inter-burst time (i), and the number of bursts (b).

Each LSA in a burst results in changing the status of *all* n adjacencies of a router from up to down or from down to up. During a burst, we cycle through routers while sending the LSAs out. For example, if $n = 2$ and $l = 4$, the four LSAs sent out would result in the following events: (i) bring down all adjacencies of router 1, (ii) bring down all adjacencies of router 2, (iii) bring up all adjacencies of router 1, and (iv) bring up all adjacencies of router 2. We believe that using a fully connected graph, flapping adjacencies of routers, and sending out bursts of LSAs stresses the LSAR and LSAG most in terms of resources.

To characterize the LSAR performance, we measure how quickly it can send out an LSA received over an OSPF adjacency to the LSAG. Recall that our modified Zebra is capable of forming an LSAG session with the LSAR. This allows us to record the necessary time-stamps within Zebra itself thereby obviating the need for running a separate LSAG process on the Zebra PC. For each LSA, Zebra records the time when it sends the LSA over the adjacency, and the time when it receives the LSA back over the LSAG session. We will denote the mean of the difference between the send-time and receive-time for an LSA by T_{lsar}. For the LSAG, we measure how long it takes the LSAG to process every LSA. To measure this, we instrumented the LSAG code to record the time before and after every LSA is processed. We will denote the mean LSA processing time at the LSAG by T_{lsag}.

Long duration LSA bursts can cause the LSAR to lose LSA instances occasionally. Despite OSPF's reliable flooding, most of these losses are irrecoverable if the lost instance is "overwritten" by a new instance of the LSA before the retransmission timer expires. Therefore, we measure the number of LSAs lost during each experiment by calculating the fraction of LSAs that were sent by Zebra to the LSAR but never received back. We will denote this quantity by L_{lsar}.

8.2 Results

We used PCs each having a 550 MHz AMD-K6 CPU, 64 MB of RAM and RedHat Linux 6.2 as the SUT and for running Zebra. We varied the number of nodes (n) in the topology in the range $50, 60, \ldots, 100$; and varied the number of LSAs per burst (l) in the range $50, 100, \ldots, 500$. We set the inter-burst gap (i) to one second, and sent 100 bursts (b) in each experiment. For every value of (n, l, b, i) quadruplet, we carried out the experiment three times.

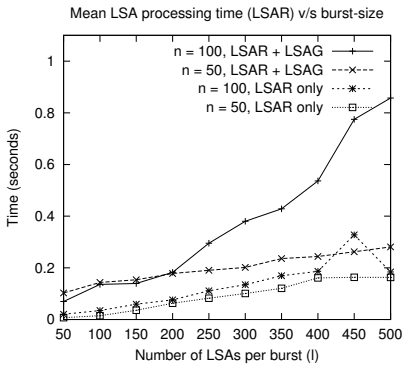

Figure 5: LSAR performance (T_{lsar}) versus the number of LSAs per burst for 50 and 100 nodes in the topology.

Figure 5 shows how T_{lsar} varies as a function of the burst-size (l) for two values of n. Apart from running both LSAR and LSAG on the SUT, we also repeated the same set of experiments with only LSAR running on the SUT. As Figure 5 shows, T_{lsar} increases as the burst-size increases under all circumstances. This can be explained as follows. The inter-departure time while sending a burst of LSAs out at Zebra is less than the inter-arrival time of receiving them back since the inter-arrival time includes the two-way propagation delay and the processing time per LSA. As a result, the turn-around time (the difference between receive-time and send-time) is lowest for the first LSA within a burst, and gradually increases for subsequent LSAs within the same burst. Moreover, this disparity in the turn-around time across LSAs widens as the burst-size increases, ultimately resulting in a higher value of T_{lsar} for higher burst-sizes. For the "LSAR only" case, the number of nodes in the topology does not have much impact on T_{lsar} since the LSAR merely deals with passing LSAs to LSAG (Zebra in this case) and archiving. On the other hand, T_{lsar} is higher for the "LSAR + LSAG" case than the "LSAR only" case and is sensitive to the number of nodes in the topology. Both these observations can be attributed to the LSAG contending for CPU and memory with the LSAR on the SUT.

Figure 6 shows how T_{lsag} varies as a function of the number of nodes (n) for two values of the burst-size. As expected, T_{lsag} increases linearly

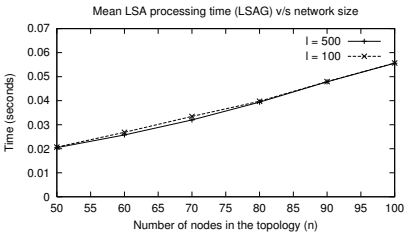

Figure 6: LSAG performance (T_{lsag}) versus the number of nodes in the network for the burst-size equal to 100 and 500 LSAs.

with the number of nodes in the topology. However, it is insensitive to the burst-size mainly because of flow control imposed by TCP.

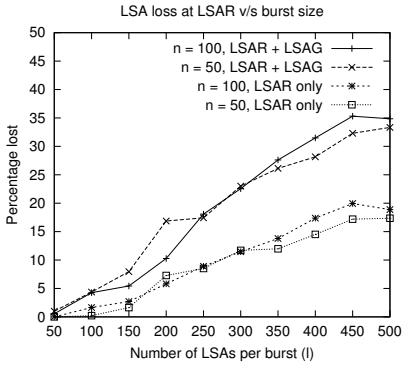

Figure 7: Loss-rate at the LSAR (L_{lsar}) versus the number of LSAs per burst for 50 and 100 nodes in the topology.

Figure 7 shows how L_{lsar} varies as a function of the burst-size for two values of n. As expected, L_{lsar} increases as the burst-size increases, and L_{lsar} is higher for the "LSAR + LSAG" case than the "LSAR only" case. However, L_{lsar} is insensitive to the topology size. For all the experiments, we checked that the LSAs not received by Zebra did not exist in the LSAR archive implying that the LSAR never received them due to a memory exhaustion in the IP stack on the SUT. Unlike LSAR, the LSAG does not lose LSAs due to the use of TCP. In fact, if given enough time at the end of the experiment, the LSAG ultimately is able to process all the LSAs sent by the LSAR.

These results show that the LSAR and LSAG are capable of handling large networks and high LSA-rate even on a low-end PC. Furthermore, their performance degrades gracefully as the load increases beyond their processing capability. We believe that at high LSA-rates and excessive flapping as was the case here the routers are more likely to melt-down before the LSAR and LSAG processes especially if a high-end server is used for running these processes. This is because route processors used on even high-end routers do not tend to be as powerful as the ones used on servers. Furthermore, route processors typically run processes other than OSPF, whereas the LSAR and LSAG can run on dedicated servers.

9 Deployment Experience

We have deployed the monitor in two commercial networks: a large enterprise network and a large ISP network. Table 1 provides some facts about these two networks and the deployment of the monitor. More details about the enterprise network architecture can also be found in [7]. It is important to note how diverse the two networks are in terms of their layer-2 architecture, and their use of OSPF. The layer-2 architecture dictated which LSAR mode we used for network attachment, whereas the use of OSPF affected the number of LSAs received per day. All servers running LSAR and LSAG processes are NTP-synchronized in both the networks.

Operators of both the networks were very cautious about deploying the LSAR and LSAG in their networks. As a result, before the deployment, both LSAR and LSAG underwent extensive testing and review to convince the operators about passiveness of the LSAR and robustness of both the components. During testing, we also ensured that there were no memory leaks in either of the components. To further enhance security, we turned off all unnecessary services, and put in measures to tightly control access to the server running the LSAR. To increase the accuracy of the time-stamping of the LSAs being archived at the LSAR, we took care of minimizing the I/O that might result from activities such as writing of syslog messages. In addition, to keep the measurements as clean as possible, we separated the interfaces used for remote management from those used to capture LSAs. This minimized the competition between measurement and management traffic.

The OSPF Monitor has handled the load imposed by these networks with ease, demonstrating and further confirming the scalability of the system. Furthermore, the LSAR and LSAG implementations have proved to be extremely reliable and robust. We have observed the LSAR crash only once during these two years, and have not observed the LSAG crash at all. The LSAR

Parameter	Enterprise network	ISP network
Layer-2 architecture	Ethernet-based LAN	Point-to-point links
Customer reachability information	Use of EIGRP between network-edge routers and customer-premise routers. EIGRP routes are imported into OSPF.	Use of I-BGP to propagate customer reachability information. I-BGP routes are not imported into OSPF.
Scale of monitor deployment	15 areas; 500+ routers	Area 0; 100+ routers
LSAR attachment mode	Host mode	Partial adjacency mode
Deployment history	Deployment started in February, 2002 by connecting LSAR to two areas. Remaining areas were gradually covered in the next two months.	Deployment started in January, 2003 by connecting LSAR to area 0.
Size of the LSA archive	10 MB per day	8 MB per day

Table 1: Facts related to the deployment of the OSPF monitor in two commercial networks.

has been upgraded three times for bug fixes, and eight times for enhancements, whereas the LSAG has been upgraded three times for bug fixes and 26 times for enhancements.

9.1 Utility of the Monitor

The OSPF monitor is being used primarily in two ways. First, the LSAG is used for day-to-day network operation, continuously tracking the health of the network. Second, the OSPFScan is used for detailed and long term analysis of OSPF behavior.

9.1.1 LSAG in Day-to-day Operations

As mentioned earlier, the LSAG provides two data sources in real-time: messages related to the topology changes and anomalous behavior, and network topology snapshots. Both the sources provide valuable insight into the health of a network.

We have developed a web-site for viewing LSAG messages, interacting with network operators to make the site as simple and user-friendly as possible. The web-site allows the operators to query the LSAG message logs, generate statistics about the messages, and navigate past archives. The web-site makes use of a configuration management tool to map IP addresses into names. This web-site is now used extensively by network support and operations on a regular basis, and has proved invaluable during network maintenance to validate maintenance steps as well as to monitor the impact of maintenance on the network-wide behavior of OSPF.

Network operation groups also use the LSAG messages for generating alarms by feeding them into higher layer alerting systems. This in turn allows correlation and grouping with other monitoring tools. To prevent a deluge of alerts generated due to a high frequency of LSAG messages, we have taken two steps. First, we prioritize messages to help operators in the event of "too many flashing lights". For example, the alerting system assigns "RTR DOWN" message a higher priority than a "RTR UP" message. Second, we group multiple messages into a single alarm. For example, a fiber cut can bring down a number of adjacencies prompting the LSAG to generate several "ADJACENCY DOWN" messages. We group all these messages into a single alert to prevent a flurry of alerts for a single underlying event.

Network operators may change OSPF link weights from their design values to carry out maintenance tasks. We have designed a "link-audit" web-site that allows operators to keep track of such link weight changes. The web-site makes use of the topology snapshots to display the set of links whose weights differ from the design weights. This allows operators to validate the steps carried out for maintenance. At the end of the maintenance interval, the web-site also allows operators to verify that weights of the affected links are reverted back to their original values.

Below we describe a few specific cases where the LSAG served to identify network problems.

1. **Internal problem in a crucial router:** The LSAG identified an intermittent hardware problem in a crucial router in area 0 of the enterprise network [7]. This problem resulted in

episodes lasting a few minutes during which the problematic router would drop and re-establish adjacencies with other routers on the LAN. Each episode lasted only for a few minutes and there were only a few episodes each day. The data suggests that during the episodes the network was at the risk of partitioning or was in fact partitioned. During these episodes, a second router failure could have resulted in a catastrophic loss of connectivity. Fortunately, a flurry of "ADJACENCY UP" and "ADJACENCY DOWN" messages recorded by the LSAG during each episode helped operators identify the problem, and perform preventative maintenance. It is worth noting here that this problem did not manifest in other network management tools being used by the enterprise network.

2. **External link flaps:** The LSAG helped identify a flapping external link in the enterprise network [7]. One of the enterprise network routers (call it A) maintains a link to a customer premise router (call it B) over which it runs EIGRP. Router A imports EIGRP routes into OSPF as external LSAs. LSAG messages led to a closer inspection of network conditions, which revealed that the EIGRP session between A and B started flapping when the link between A and B became overloaded. This led to router A repeatedly announcing and withdrawing EIGRP prefixes via external LSAs. The flapping of the link between A and B persisted nearly every day for months between 9 PM and 3 AM. The LSAG messages ("TYPE-5 ROUTE ANNOUNCED" and "TYPE-5 ROUTE WITHDRAWN") helped network operators to identify and mitigate the problem, though they could not completely eliminate it as the operators did not have access to the customer-premise router.
3. **Router configuration problem:** In another case, the LSAG helped operators of the enterprise network identify a configuration problem: assignment of the same router-id to two routers. This error resulted in these routers repeatedly originating their router LSAs which showed up as a series of "ADJACENCY UP/DOWN" LSAG messages.
4. **Refresh LSA bug:** The LSAG helped identify a bug in the refresh algorithm of the routers from a particular vendor in the ISP network. The bug resulted in a much faster refresh of summary LSAs under certain circumstances than the RFC-mandated [1] rate of 30 minutes. The bug was identified due to the "LSA STORM" messages generated by the LSAG. At the time of writing this paper, the vendor is investigating the bug. It is worth noting that it would be impossible to catch such a bug with any other class of available network management tools.

9.1.2 Use of OSPFScan for Detailed Analysis

In this section, we touch on ways in which the OSPFScan has been used for analyzing long-term behavior of OSPF. For both the networks where the monitor is deployed, in addition to archiving all LSAs, we also archive topology snapshots and LSAG message logs. Furthermore, we use the OSPFScan to extract change LSAs, topology change records and to compute routing tables for each router, grouped by 24-hour intervals. All this data (raw and change LSAs, topology change records, routing tables, topology snapshots, and LSAG message logs) forms the data repository for the OSPFScan analysis. Although there is a redundancy (raw LSAs are sufficient to construct all other forms of data), we have found that keeping the derived data greatly assists interactive analysis of OSPF behavior. To illustrate, suppose a user is interested in analyzing how the path between two end-points evolved over time. It is much faster to automatically compute paths between two end-points using the routing table data than to construct the paths from raw LSAs.

Specific illustrations of the OSPFScan usage include:

1. **Duplicate LSA analysis:** The LSA traffic analysis in the enterprise network by the OSPFScan [7] revealed excessive duplicate LSA traffic. For some OSPF areas, the duplicate LSA traffic formed 33% of the overall LSA traffic. Subsequent analysis led to the root-cause of the excessive traffic and preventative measures, details of which can be found in [7].
2. **Change LSA statistics:** The SPF calculation on Cisco routers is paced by two timers [20]: (i) *spf-delay*, which specifies how long OSPF waits between receiving a topology change and starting an SPF computation; and (ii) *spf-holdtime*, which determines the lag between two successive SPF computations. In order to reduce OSPF convergence time, it is desirable to decrease these timers to small values; however, reducing these values too much

can lock the routers into performing excessive SPF calculations, possibly destabilizing the network. Analysis of the inter-arrival time of change LSAs in the network can help administrators configure these timers to "good" values. The network administrators of the ISP network have done precisely this. To facilitate the process, we built a web-site on top of the change LSA repository, providing statistics such as minimum, maximum, mean, standard deviation and empirical CDF of inter-arrival times of change LSAs over a given time period and for a given LSA type.

3. **Availability analysis:** Assessing reliability and availability of intra-domain routing is crucial for deploying new services and associated service assurances into the network. OSPF monitor data has proved very useful in answering questions such as: what is the mean down-time and mean service-time for links and routers in the network at the IP level? Again, we created a web-site to answer such questions for the ISP network. The site relies on the topology change records stored in the repository.
4. **Use of OSPF routing tables:** For each router, the routing table archive contains the entire history of routing tables across the measurement interval (e.g., several months or longer). This data is being used by the ISP network engineering teams to determine and analyze end-to-end paths within the network at any instance of time, to correlate OSPF routing changes with I-BGP updates seen in the network [18], and to analyze how OSPF events impact the traffic flow within the network by correlating this data with active probing.

9.2 Lessons Learned

In this section, we point out some of the lessons learned during and after the deployment of the system. The points may help in design, development and deployment of other route monitoring systems.

- **New tools reveal new failure modes.** The LSAG has allowed us to find and fix several problems in a pro-active fashion. Some of these problems would have been impossible to find with other network management tools (e.g., the refresh LSA bug).
- **Real-time alerting and off-line analysis are complementary.** Some problems such as router-bug were caught by real-time messages, whereas some other problems such as excessive duplicate LSA traffic were caught because of off-line analysis of LSAs stored over long time intervals. Finally, problems such as refresh LSA bug were identified using LSAG messages in real-time, but they also required a more detailed off-line analysis.
- **OSPF exhibits significant amount of activity.** Based on our experience, we have noticed that both the networks monitored exhibit significant amount of OSPF activity. This activity is due to maintenance tasks as well as network problems. Efficient and scalable design of the system has helped us tackle this high level of activity with relative ease.
- **Add functionality incrementally.** We have added new functionality and improved the system by close interaction with network operators. At one level, this pertained to the user interfaces. For example, it took several iterations until the operators were satisfied with LSAG message formats and could make sense of associated logs at a glance. At another level, it was important to customize and enhance value by building custom reports that reflected operational practices.
- **Archive all the LSAs.** The analysis of excessive duplicate LSAs and refresh LSA bug required archiving all the LSAs captured from the network, not just those that indicated topology changes. The volume of all OSPF LSAs is not onerous. As seen in Table 1, the volume of raw LSAs collected from each of the two networks is in the order of 10 MB per day. This makes it fairly easy to collect all the LSAs from the network, store these for a long period, as well as transfer and replicate the archives as needed.

10 Conclusion

In this paper, we have described the architecture and design of an OSPF monitor, which passively listens to LSAs flooded in the network, provides real-time alerting and reporting on network events, and allows off-line investigation and post-mortem analysis. The three main components of the system are:

- The LSAR (LSA Reflector), which captures LSAs from the network. The LSAR supports three modes by which it can be attached safely and passively to the network.
- The LSAG (LSA aGgregator), which receives "reflected" LSAs from one or more LSARs,

monitors the network in real-time for operationally meaningful events. The LSAG populates a network topology model using the LSA stream, tracking changes to the topology, and issuing associated alerts and console messages. These messages allow network operators to quickly localize and troubleshoot problems. The topology model also provides timely and accurate views of the OSPF topology to other network management applications.

- The OSPFScan, which allows efficient post-mortem analysis of LSA archives via streamlined parse, select and analyze capabilities. The OSPFScan includes a large set of libraries, implementing, in particular, playback and isolation of topology change events, generation of statistics, and construction and evolution of routing tables and end-to-end paths for every topology change event.

We demonstrated that the LSAR and LSAG can scale to large networks and large LSA rates through lab experiments. Overall, the monitor cleanly separates instrumentation, real-time processing and off-line analysis. The monitor has been successfully deployed in two commercial networks, where it has run without glitches for months. We provided several examples illustrating how operators are using the monitor to manage these networks, as well as some lessons learned from this experience.

In future, we plan to work on improving the LSAG by incorporating more intelligent grouping and prioritization of messages. We also plan to focus on correlation of OSPF data with other data sources both for better root-cause analysis of network problems and for better understanding of network-wide interaction of various protocols.

Acknowledgments

We are grateful to the operators of the ISP and the enterprise network for allowing us to deploy the OSPF Monitor. Their subsequent feedback improved the functionality of the monitor immensely. We thank Jennifer Rexford, Jay Borkenhagen, anonymous reviewers and our shepherd, Stephan Savage for their comments which helped us a lot in improving the paper. Finally, we thank Kiranmaye Sirigineni for modifications to Zebra that enabled the OSPF topology emulation.

References

[1] J. Moy, "OSPF Version 2." Request for Comments 2328, April 1998.

[2] W. Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison-Wesley, 1999.

[3] A. Shaikh *et al.*, "An OPSF Topology Server: Design and Evaluation," *IEEE J. Selected Areas in Communications*, vol. 20, May 2002.

[4] "Route Explorer." `http://www.route-explorer.com/`.

[5] "Route Dynamics." `http://www.ipsumnetworks.com`.

[6] "IP Monitoring Project." `http://ipmon.sprint.com/`.

[7] A. Shaikh *et al.*, "A Case Study of OSPF Behavior in a Large Enterprise Network," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, November 2002.

[8] D. Watson *et al.*, "Experiences with Monitoring OSPF on a Regional Service Provider Network ," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[9] "University of Oregon Route Views Project." `http://www.routeviews.org/`.

[10] "RIPE Network Coordination Center." `http://www.ripe.net/`.

[11] N. Spring *et al.*, "Measuring ISP Topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, 2002.

[12] R. Mahajan *et al.*, "Inferring Link Weights using End-to-End Measurements," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, 2002.

[13] "Rocketfuel: An ISP Topology Mapping Engine." `http://www.cs.washington.edu/research/networking/rocketfuel`.

[14] A. Feldmann and J. Rexford, "IP Network Configuration for Intra-domain Traffic Engineering," *IEEE Network Magazine*, September 2001.

[15] P. Aukia *et al.*, "RATES: A server for MPLS traffic engineering," *IEEE Network*, pp. 34–41, March/April 2000.

[16] R. Siamwalla *et al.*, "Discovering Internet Topology." Unpublished manuscript, `http://www.cs.cornell.edu/skeshav/papers/discovery.pdf`, July 1998.

[17] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," in *Proc. IEEE INFOCOM*, March 2000.

[18] R. Teixeira *et al.*, "Dynamics of Hot-Potato Routing in IP Networks," in *Proc. ACM SIGMETRICS*, June 2004.

[19] "GNU Zebra." `http://www.zebra.org`.

[20] "Configure Router Calculation Timers." `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/%np1\_c/1cprt1/1cospf.html\#xtocid2712621`.