

Fault Detection in Routing Protocols *

Daniel Massey †

Bill Fenner ‡

May 3, 1999

Abstract

Routing protocol faults cause problems ranging from an inability to communicate to excessive routing overhead. This paper proposes a system for detecting a wide range of routing protocol faults. Our system deploys virtual routers called RouteMonitors to monitor a routing protocol. We deployed RouteMonitors in the Mbone's DVMRP infrastructure and uncovered a number of faults. We were also able to identify the cause of these instabilities and deploy fixes. The result was a substantial improvement in DVMRP route stability. This paper reviews the RouteMonitor approach and details the results achieved in the Mbone.

1 Introduction

Detecting routing protocol faults remains an open challenge for protocol designers as well as network administrators. Some routing protocol faults result in obvious problems. For example, end systems will be unable to communicate with each other if a fault created an invalid route between them. But other faults may result in more subtle problems. Packets might be routed along suboptimal or frequently changing routes. Even if packets arrive correctly, faults may cause extra communication and computation in the routing protocol. As networks grow in size and complexity, these subtle problems can

develop into widespread breakdowns. An ideal network would have a mechanism to automatically detect problems occurring in routing protocol.

This paper presents the *RouteMonitor* approach for monitoring a routing protocol and describes the results achieved by deploying this system in the Mbone's DVMRP routing infrastructure. Beginning in summer 1997, our *RouteMonitor* was used to collect data from the Mbone's DVMRP infrastructure. Our observation showed that only 67% of DVMRP routes remained stable during an average hour and some routes changed an over 150 times every hour. Using RouteMonitor, we were able to identify faults which we believed were responsible for this high level of instability. In the intervening time, we worked to encourage the deployment of patches which corrected these faults. By fall 1998, most of the instability had been eliminated and we now observe an average of 95% of DVMRP routes remain stable during each hour. This paper presents the approach used to detect routing protocol faults, a detailed picture of the instability discovered in the Mbone, and the specific faults responsible for this instability.

In the remainder of this section we identify four categories of faults that can impact a routing protocol and provide simple examples for each category. The section concludes with a discussion of the existing methods of coping with faults and related work. Section 2 presents a high level overview of the RouteMonitor system for monitoring a routing protocol. Section 3 details our DVMRP RouteMonitor system and describes the results achieved using RouteMonitor in the Mbone's DVMRP routing infrastructure. Section 4 summarizes our results and addresses areas of future work.

*This material is based upon work partially supported by the National Science Foundation under Grant No. 9729498. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

†Computer Science Department, UCLA, Los Angeles, CA 90095, Email: masseyd@cs.ucla.edu

‡AT&T Labs Research, 75 Willow Rd. Menlo Park, CA 94025, Email: fenner@research.att.com

1.1 Types of Routing Protocol Faults

1.1.1 Hardware Faults

The most obvious type of problem faced by a routing protocol is the failure a link or router. Most routing protocols were designed to cope with simple *fail-stop* behavior by links or routers. Unfortunately, routers and links often fail but do not stop operating. When a link or router fails but does not stop operating, it can create a wide range of routing problems.

One of the oldest and best known examples comes from the original ARPANET routing algorithm [MW77]. An east coast router with faulty memory prevented traffic from reaching UCLA by reporting a corrupt zero cost route to UCLA[MFR78]. The original ARPANET routing algorithm and the more recent protocols RIP[Hed88] and DVMRP[WPD88] require a router to periodically report its distance to each destination. These protocols were designed to detect a router which stopped reporting distances, but there is no mechanism for verifying the distances a router reports. Any distance reported is simply believed.

Similar issues apply to OSPF [Moy89], BGP [RL94], and other routing protocols. The information reported may include the state of link (as in OSPF) or existence of an AS path (as in BGP), but the information reported assumed to be correct. Some routing protocols have been specifically designed to cope with routers which send corrupt information [Per88, SMGLA97], but these protocols have never been widely deployed in the Internet.

1.1.2 Implementation and Configuration Faults

More complex routing protocol faults occur due to implementation errors. Most routing protocols are published as well known standards and numerous vendors implement the standard. For example, major router vendors such as Cisco produce implementations for their state of the art backbone routers and organizations such as the Gated Consortium produce free implementations available for workstations serving as routers. While theoretically all implementations should be compatible, subtle incompatibilities often exist.

In addition to implementation incompatibilities, configuration errors can also introduce faults. A router must be correctly configured by an administrator, otherwise it can advertise invalid information. Router configuration is often a non-trivial task. An incorrect configuration may cause a router to advertise routes belonging to another organization or cause the router to fail to report valid routes. As protocols become more complex and more systems join the Internet, the potential for implementation and configuration errors grows steadily.

Section 3 describes three implementation and configuration errors that were responsible for most of the instability in the DVMRP infrastructure. To illustrate the potential for these types of problems, it should be noted that errors occurred even between implementation versions from the same vendor. The incorrect configuration described in section 3 was in fact the default configuration provided by the vendor.

1.1.3 Protocol Design Faults

Routing faults can also occur because of flaws in the routing protocol itself. With the rapid growth in networking, protocols may encounter situations that protocol designers overlooked or did not anticipate. Routing protocols typically last for many years and during that time great changes can occur in the scale of a network as well as the underlying network technology. The need to design protocols that can tolerate any situation or any fault is counter balanced by the need to make protocols efficient and practical to implement. The resulting protocols often have a reasonable, but not absolute, expectation of handling foreseeable situations.

For example, the multicast routing protocol DVMRP had to make the well known distance vector trade-off between network diameter and convergence time. The value 32 was chosen as infinity (i.e. the maximum number of hops in the network is 31). This is a reasonable choice for an intra-domain protocol, but DVMRP has been used as an inter-domain routing protocol. For inter-domain routing, the choice of 32 hops is inadequate. The situation is being corrected but the introduction of inter-domain protocols such as MBGP [BCKR98]. But this example illustrates how protocols may be

used in ways the designers did not envision and this may create unexpected problems.

A more extreme example is the well known break down of the revised ARPANET routing algorithm [MRR80]. The revised ARPANET algorithm used sequence numbers to determine which route updates contained more recent information. It was assumed that a particular combination of sequence numbers could never occur, but unexpected events proved this assumption incorrect. Once the unexpected combination did occur [Ros81], the routing protocol collapsed and a patched version of the protocol had to be down-loaded onto every IMP¹ in the ARPANET.

1.1.4 Byzantine Faults/Intentional Attacks

The strongest type of fault a routing protocol must cope with is a Byzantine [LSP82] fault. Rather than simply report faulty information, a router could send messages intentionally designed to confuse its neighbors. An attacker could attempt to deny service to some organization by attacking the routes to that organization. The goal is not necessarily to direct the traffic to a particular destination. Instead, the attacker may simply try to prevent the correct route from remaining stable.

For example, consider an attempt to make a site unreachable by advertising false routes to that site. An attacker need not gain access to any device at the target site. From some distant location, the attacker simply advertises a false route to the target site. Depending on the routing policy, the distributed routing protocols can spread this invalid information over potentially large portions of the Internet and packets for the target site can be diverted onto the false routes.

Designing routing protocols that tolerate Byzantine faults has been the subject of much theoretical research [SMGLA97, HPT97, Per88], but the resulting protocols tend to be very resource intensive and have not been deployed. An alternate approach is to design monitoring tools which can detect an attack in the currently deployed protocols. Our RouteMonitor system is designed to address routing protocol faults ranging from faulty hardware to intentional attacks.

¹An Interface Message Processor (IMP) was the ARPANET equivalent of a router.

1.2 Existing Methods and Related Work

The problem of detecting routing protocol faults is at the intersection of protocol design, network management, and performance measurement. Given this intersection, it is important to clarify what our system does and does **not** try to achieve. Our objective is not argue for the deployment of new protocols. Our results do illustrate some limitations of the DVMRP protocol in particular, but it also shows the majority of the problems were overcome once they were detected. We do not intend to advocate or discourage the use of any particular protocol, we only assert that *any* routing protocol requires an effective system for analysis and fault detection.

Another interesting area that has received much attention is performance measurement. Projects such as NIMI [AMMP98] and MINC [CDL⁺] concentrate on gathering performance statistics on the network. These works tend to focus on higher layer issues such as congestion, measuring flows, and so forth. Our objective is simply to determine whether the routing protocols are working as they should be.

Our results do illustrate some interesting measures of the DVMRP infrastructure, such as the number of total routes. Certainly routing protocols faults have an impact on performance measurement and performance measurement can lead to the discovery of routing fault. But problems such as congestion and throughput are affected by many factors unrelated to whether the routing protocols work correctly. Our fundamental objective is only to detect when the protocol acts in a way contrary to their specification.

Finally, network management is a broad area concerned with problems ranging from fault detection to billing. We are not suggesting a new network management paradigm. We are, however, suggesting an important new tool in an area that is often overlooked. Identifying and correcting routing protocols faults can substantially improve a network. It is not a comprehensive solution to network management, but is an important component.

In practice, routing protocol faults are often not detected until someone reports a problem and then ad hoc methods are used to solve the problem. For

example, a user notifies the network administrator that some site is unreachable or has a high packet loss. The network administrator then checks the local network and corrects any local problems. If the problem is not local, the administrator can notify the destination's administrator and ask her to investigate. However if the problem is not local to either the source or destination, neither side will be able to correct the problem. At this point, the best they can hope for is to narrow down where the problem might be occurring, contact administrators in that area, and hope those administrators will assist them.

The simplistic example above illustrates two main drawbacks of ad hoc fault detection. First, faults are often not detected until someone encounters a noticeable problem. Aside from the obvious frustration for network users, this means that subtle problems are often not detected and the overall performance of the network suffers. Second, the distributed nature of routing protocols allows any device to cause a problem but an administrator often only has access to local devices. To investigate a non-local fault, the administrator needs to interest other sites in the problem. We would like to have systems that detect faults and tools for tracking the source of a fault.

Most of the systems and tools for detecting routing protocol faults can be classified as either active monitoring tools or passive monitoring tools. Active monitoring tools add something to the system and watch for an expected result. Passive tools observe the system and report on unusual events. We briefly outline some of the most widely used systems and tools below.

1.2.1 Active Monitoring

An active monitoring tool introduces something into the network and measures its effect. For example, the ping tool is the simplest form of active fault detection. Ping sends a packet to a destination and waits for a response. Ping measures quantities such as the round trip time and loss rate. The ping idea is easily generalized into a number of automated fault detectors. Some points in the network introduce traffic and other points react to this traffic. The monitoring tool knows that some specific event was generated and can then detect if

the expected results did or did not occur.

Some of the best known active fault detection tools are route tracing tools. The *traceroute* tool sends UDP[Pos80] packets of varying lifetimes to learn the route to a destination. An advanced version of traceroute, *pathchar*[Jac], sends more packets and generates more statistics such as estimated throughput on each link. The *mtrace*[FC] tool traces the multicast route from a destination to a source using special IGMP protocol messages.

The Real-Time Transport Protocol (RTP)[SCFJ96] specifies a mechanism with which the participants in a multicast session report their observed loss rates from each traffic source in that session. These loss rates can then be passively monitored by any site capable of receiving the multicasted session. Systems such as *rtpmon*[SCFJ96] and *MHealth*[MA] combine these statistics with knowledge of the network topology to help track down faults. Since these statistics can only be gathered with active senders and receivers, *MRM*[Wei] specifies a protocol to cause test stations (routers or hosts) in the middle of the network to source or sink RTP streams.

Active monitoring tools have proven extremely useful and will continue to play a critical role in fault detection. However, wide scale information can not be easily obtained by active monitoring tools. For example, it would clearly be infeasible to obtain a wide scale picture of routing by actively monitoring every IP destination.

One could instead attempt to limit active monitoring to some representative destinations, but defining a representative destination is problematic. One could use the network topology to select representative destinations, but routing protocol faults need not be based on topology. In section 3, we detail routing protocol faults which did not follow a topological pattern. These faults were based on where a route was located in the routing table or whether the route was an aggregate or specific route. This information can be completely unrelated to the network topology and can vary from implementation to implementation.

As networks grow in scale, wide scale active monitoring requires more traffic generation or more reliance on generating representative forms of traffic. One must generate enough traffic to find the problem, but generating traffic may be costly. Adding

traffic is the exact opposite of what one should do if congestion is causing the problem. The test traffic itself may become a source of problems. Active monitoring tools remain essential for some types of routing faults and have proven very useful in characterizing broad network performance. But we contend RouteMonitor provides a form of routing protocol fault detection that could not be achieved by active tools alone.

1.2.2 Passive Monitoring

Passive fault detection is most commonly done using the well known management protocol SNMP [CFSD90, Ros91]. SNMP allows one to query a fixed a set of variables. The collection of variables maintained for a particular protocol is listed in a management information base (MIB). An SNMP agent sends a request asking for a particular MIB variable and the device maintaining the MIB returns the result. SNMP also defines methods for authorizing access to data so that anyone with proper permissions could query the device.

SNMP is an effective system for management of a network. SNMP also allows one to set MIB variables. By setting MIB values, one can essentially perform remote configuration of a router or other device. This allows a central network operations site to easily control a wide range of devices from a single control point.

For example, a RIP router maintains a MIB with RIP related variables. A monitoring tool running on some host could query the router and request the current distance to a particular destination. The device could also be remotely configured by setting various MIB values. If the administrators would be willing to share SNMP authorization, some distant administrator could query the router. However, sharing some types of authorization may also provide access to information that should not be shared, such as sharing network configuration and topology information with competitors.

SNMP allows the operations center to monitor the status of network devices and potentially detect problems by polling the device or using SNMP traps. A network management center could request state from each device periodically, or request that the device notify the management center via a trap when an event occurs. A trap instructs the device

to automatically send a SNMP message whenever a MIB value reaches a particular state. For example, a trap could signal if the number of packets per second sent out a particular interface falls below a threshold. Using polling or traps, an SNMP management station may be able to automatically detect some faults.

SNMP provides access to a wide range of low level information. This low level access has tremendous advantages for some types of network management, but also creates limitations for routing protocol fault detection. The type of routing protocol data available is fixed by the MIB. Reasonable limitations on MIB variables are required since the device must use resources to maintain the MIB. This limits the type and amount of data available for fault detection. In addition, one is relying on the potential faulty device to correctly maintain and report its state. Finally, even if the device does report correctly, frequently sending SNMP queries to a production router adds additional duties and loads to those crucial devices.

Recently a number of experiments have used passive monitoring approaches that rely more heavily on observing packets. In [LMJ97] Labovitz, Jahanian, and Malan observed BGP route updates exchanged at major interchange points and noticed BGP faults such as duplicate withdraws for routes which were never advertised. A more general packet filter/monitoring tool by Malan and Jahanian called *windmill* is described in [MJ98]. In [TMW97], Thompson, Miller, and Wilder designed an OC-3 packet capture system and used it to analyze traffic and routing in MCI's backbone.

The JiNao system [WW98, VWW97] looks at detecting intentional attacks against the routing protocol OSPF. The system works by observing OSPF updates and raising flags when known types of attacks occur. The system always warns of statistically anomalous protocol behavior and attempts to fight back against routing attacks.

2 RouteMonitor

The fault detection approach taken here is to deploy a collection of monitoring devices called RouteMonitors. RouteMonitors observe the protocol messages (i.e. route updates) exchanged between routers. A RouteMonitor essentially imi-

tates a router by listening to route updates and constructing a route table based on the messages it hears. RouteMonitors do not change the route computation in a network, they simply offer points at which to observe how the routing protocol is performing. The RouteMonitor system is designed with the following objectives in mind:

- Observe the protocol by watching the protocol messages exchanged.
- Add a minimal impact to the network being monitored.
- Provide small, easily deployed data collection components.
- Provide easily displayed and easily shared data.
- Do not limit observation to a fixed point.
- Correlate the views from numerous observation points.

Each one of these factors is discussed briefly.

2.1 Observation of protocol messages

Our focus is on watching how a router behaves instead of asking the router to report its state. This is done by observing the protocol messages (i.e. route updates) sent by a router. Our RouteMonitor simply imitates the actions an actual routing would perform to obtain protocol messages. The protocol specification details the required actions and the format of messages. For example, RIP and DVMRP messages are multicast and RouteMonitor need only join the appropriate multicast group to receive updates. BGP and MBGP messages are sent via TCP and RouteMonitor collects updates by exchanging the *keepalive* messages required by BGP/MBGP peer.

An alternate approach would be to rely on statistics collected by the router, such as the data stored in an SNMP MIB. Both approaches are valid, but we choose to observe updates. We assume that some unknown number of routers are faulty. Asking the faulty device to correctly report its state is problematic. In addition, we want a large quantity of data (not all of which is contained in a MIB) and want to understand the interaction between

devices. By observing the protocols messages, we can reconstruct how a neighboring router should react and contrast it with the actual reaction observed in neighboring routers.

2.2 Requiring minimal impact

Requiring minimal impact on the network is another reason for observing updates. A monitoring system should not change the system it is observing or add high loads to crucial devices. However, our monitoring system would also like to collect large volumes of data, make this data immediately accessible, and allow for experiments and modifications to the monitoring process. Continuously querying a production device for large volumes of data adds a load to that device. Any action applied to the router must also keep in mind that the action should never crash the router.

By observing updates, we essentially add a *virtual router* to the network. This *virtual router* is called a RouteMonitor and it can be modified, queried, and experimented on without the concern that these actions might negatively impact the actual network routing. Our RouteMonitors are passive devices which never advertise routes and thus are never asked to forward packets. The only load we introduce is additional route updates sent to the RouteMonitor and the (application level) network traffic used to communicate with the RouteMonitors.

2.3 Small, Easily Deployed Data Collection

Another important aspect of a monitoring system is the ability to pursue problems to their source. Routing is a distributed process and the most interesting observation point can change or evolve in response to new information. An important practical concern is that the source of a fault can cross administrative boundaries, but administrators are justifiably hesitant to add anything that might impact their routing infrastructure or reveal internal configuration information.

Complex packet filter software or special hardware devices can play an important role, but we would like to have the ability to quickly contact a system administrator and ask her to temporarily run a data collection tool. A very small, very easily

verifiable data collector that produces clearly understandable text output can be a crucial factor in gaining access to remote sites.

The portion of our system which observes protocol updates is an extremely simple piece of code. The data collection part of our DVMRP RouteMonitor simply joins the appropriate group and listens for updates. The updates are filtered in very rudimentary fashion and some simple statistics (such as average number of changes) are collected. While other packet filter systems provide extremely sophisticated code, our data collector claims extreme simplicity and lack of sophistication as one of its practical strengths.

2.4 Easily Displayed and Shared Data

While our data collection component is a very small, very safe piece of code that can be easily deployed in any location, the remainder of a RouteMonitor can afford to be more complex. The output from the data collection code can be directly fed into the more complex analysis component or the data can be written to a text file for later input into the analysis component. The analysis component is a collection of C, Perl, and Java programs. The analysis component flags anomalous behavior (such as the most frequently changing routes), displays current and long-term data to the web, and allows for interactive searches on the data.

The practical aspect of making the data accessible should not be underestimated. To convince a site to deploy a data collector, one needs to provide reliable evidence that a problem exists. Even once faults are detected, deploying the patched implementations which correct the problem can be a considerable challenge. In the DVMRP results discussed in section 3, the primary faults were discovered in a matter of months. But it took approximately a year before the faulty implementations had been replaced. The ability to share data and promote changes can not be overstated.

2.5 Multiple Observation Points

Using the view from a single RouteMonitor, one can detect problems and focus an investigation on questionable behaviors. Previous work has focused on the view from a single (or small number) of key locations. In our results, the view from a single

RouteMonitor was also enough to identify interesting problems. But this limited view was not sufficient to detect the source router or the direct cause of the problem.

Using views from a collection of RouteMonitors, one can detect problems, locate the source of the problems, and isolate a specific fault. As needed, additional RouteMonitors can be quickly deployed in response to a problem. Our focus is on distributed data collection. RouteMonitors can exchange views in an attempt to isolate a problem. This remains a topic of active work in current RouteMonitor work.

In the next section, we illustrate the RouteMonitor details and advantages with results from our DVMRP RouteMonitor. We detected a number of problems that had gone relatively unnoticed. We were able to identify the sources of these problems and motivate the deployment of corrections. Overall, a substantial improvement in the performance of the MBone's DVMRP infrastructure was achieved.

3 The DVMRP RouteMonitor

Beginning in summer 1997, we used the RouteMonitor approach described in section 2 to analyze routing protocol faults in the MBone's DVMRP infrastructure. Our results showed that during an average hour, only 67% of the DVMRP routes remained stable. Some of the worst routes changed an average of over 150 times per hour. This route instability created both communication problems for a number of sources and also generated extra load for all routers in the DVMRP infrastructure.

We were able to identify the main causes for this instability and implement fixes. By late 1998, the fixes for these problems had been deployed at most problem sites and 95% of DVMRP routes remained stable during an average hour. No routes averaged over 100 changes per hour. The complete results are detailed below.

3.1 MBone and the DVMRP RouteMonitor

IP multicast allows a sender to address a packet to a group rather than to a specific receiver. The sender need not belong to the group and need not

know any of the group members. The sender simply addresses the packet to an IP group address. The routing protocol is responsible determining who belongs to the group and delivering a copy of the packet to each group member. The collection of multicast capable hosts, routes, and links forms the Multicast Backbone or MBone[Eri94].

In the MBone, the hosts and routers use the IGMP[Dee89] protocol to learn group membership information. The MBone uses routing protocols such as DVMRP[WPD88], PIM[EFH⁺98], CBT[Bal97], and MBGP[BCKR98] to compute a routing tree and forward packets from the sender to the receivers. While the details of these protocols are not essential for the remainder of this discussion, it should be clear that some underlying topology information² is necessary to compute the forwarding trees. In particular, the routing protocols rely being able to compute the shortest path to any given source. If this shortest path information is invalid, the resulting tree will also be invalid. We used RouteMonitor to look for faults in one of the routing protocols used to compute this topology information.

In summer 1997, nearly all MBone routers used the DVMRP³ topology learning algorithm. DVMRP is a distance vector algorithm [FF62, Hui95]. A distance vector algorithm finds the neighboring router which is closest to the source and the distance to the source. A router periodically sends to its neighbors updates listing its distance to each source. Upon receiving an update, a router compares the distance learned from the update⁴ with the router's own distance to the source. If the distance learned from the update is shorter,

²Note that the MBone topology is not identical to the Internet topology. Non-multicast capable routers and links are present in the Internet topology, but not in the MBone topology. Some MBone routers create tunnels to bridge the gaps over non-multicast portions of the Internet. These tunnels appear as links in the MBone topology, but are not present in the Internet topology.

³DVMRP specifies both a topology learning protocol and a tree construction protocol. Although less than half of the routers in the MBone used the DVMRP tree construction protocol, nearly all of them used DVMRP for topology learning. Unless specifically stated otherwise, the term DVMRP will refer only to the topology construction protocol.

⁴The distance learned from the update is computed by adding the metric reported in the update to a metric associated with the update sender. The metric associated with the update sender is specified in a configuration file.

then the sender of the update becomes the previous hop to the source and the distance to the source becomes the distance learned from the update.

The DVMRP topology database maintained at a router consists of one entry for each known source network. The entry for a source contains the source network address, the netmask for the address, the previous hop router along the path to the address, and the distance to the address. The maximum distance allowed by DVMRP is 31. A distance of 32 (called "infinity") indicates that a source is unreachable. The DVMRP algorithm used in the MBone requires a router to send at least one update for a source every sixty seconds. If 180 seconds elapse without hearing an update from the previous hop to a source, the route to the source times out. If the distance to a source changes, a router may trigger an update for the source rather than wait for the next sixty second update interval. Routers multicast updates onto each of their attached subnets or links.

A DVMRP RouteMonitor subscribes to the multicast group on which reports are transmitted, 224.0.0.2. This allows RouteMonitor to receive reports from all multicast routers attached to the subnet. The reports can be filtered based on the sender's IP address so that only reports from a particular router are accepted. The reports from the selected router or routers are processed and an entry is made for each source address appearing in a report. Along with the source, we record the distance to the source reported in the update. If a later update reports a new distance, we change the distance to the source and record a route change, called a "flap", for that source.

The data collection portion of RouteMonitor listens continuously for reports, processing each report received from the selected router. Once each hour RouteMonitor outputs the list of sources for which it received reports, the number of flaps for each source, and the distance(s) reported for that source during the hour. The number of flaps is then reset to zero and the update monitoring process continues. RouteMonitor can also output each update individually, but this feature is used sparingly.

The analysis component of RouteMonitor uses the hourly data to calculate additional statistics such as the average number of flaps for each source.

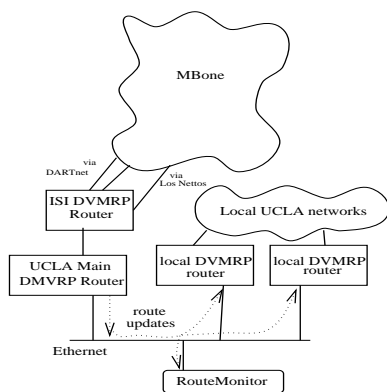


Figure 1: Data Observation Point

A large, long-term database listing statistics for each source that ever appeared in an update is maintained and *suspect* behavior is reported. When a new set of hourly data is received, the RouteMonitor analysis component updates both the database and a series of web pages. The web pages list current DVMRP topology statistics, including the most unstable routes for the current week. To help analyze the data and locate problems, the web pages allow for various searches of the long term database.

Search parameters available from the web include number of route changes, number of updates, up time, first advertisement, last advertisement, and so forth. For example, one can use the web page to report all sources matching a CIDR address, changing some range of values during the current hour (week/month), and first reported on or before some date.

3.2 A View from a DVMRP RouteMonitor

Our main RouteMonitor was deployed at the UCLA Computer Science Department. UCLA had a single connection to the MBone via ISI. The local MBone topology is shown in Figure 1. Since UCLA has only one connection to the general MBone, the view from the subnet attached to the UCLA to ISI router accurately reflects UCLA's view of the MBone topology.

It is important to note that the data gathered by a RouteMonitor is strongly dependent on its location. If the UCLA to ISI link fails, the UCLA RouteMonitor will detect that all non-local

routes in the MBone have changed and all non-local sources have become unreachable. But other portions of the MBone do not depend heavily on the UCLA to ISI link and they would see a far less dramatic change. To another site, only the UCLA sources become unreachable and all other routes in the MBone would remain unchanged. We do not claim the data collected by the UCLA RouteMonitor represents the definitive view of the DVMRP infrastructure; instead we will show that by starting from the UCLA view we were able to detect faults that impacted virtually every portion of the infrastructure.

To understand the following data, it is important to note that the DVMRP topology learning algorithm does not depend on any dynamic factor such as network congestion or which hosts are actively involved in a multicast session. If no faults were occurring, the topology database would remain constant. Allowing for the occasional stop-failure of links or routers, one would expect that topology of the MBone changes very little, if at all. The DVMRP topology database observed by RouteMonitors, however, changed frequently during every hour.

3.2.1 DVMRP Hourly Data

The results displayed here were compiled from the RouteMonitor database and are intended to illustrate the scope and type of problems occurring when our project began.

Figures 2 and 3 show the number of total routes (solid line) and the number of stable routes (dotted line) observed each hour at the UCLA RouteMonitor during July and August of 1997. The number of routes is computed by counting how many distinct sources appear in route updates during an hour. A route to a source is said to be stable during the hour if the distance reported to that source did not change. On average routes for 5600 sources are seen during each hour and 3700 routes remain stable. Only 67 percent of the routes remain stable during an hour.

The large drops in figures 2 and 3 correspond to fail-stop behavior by links or routers critical to UCLA. All but 1200 routes rely on an particular ISI link. (see figure 1). The large drops in the number of stable routes very likely correspond to failures

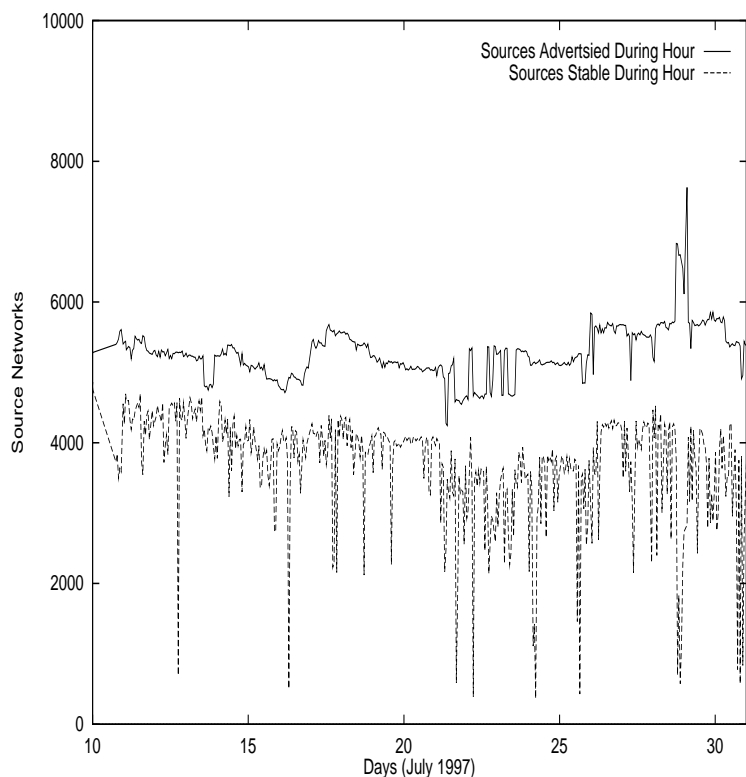


Figure 2: July 1997

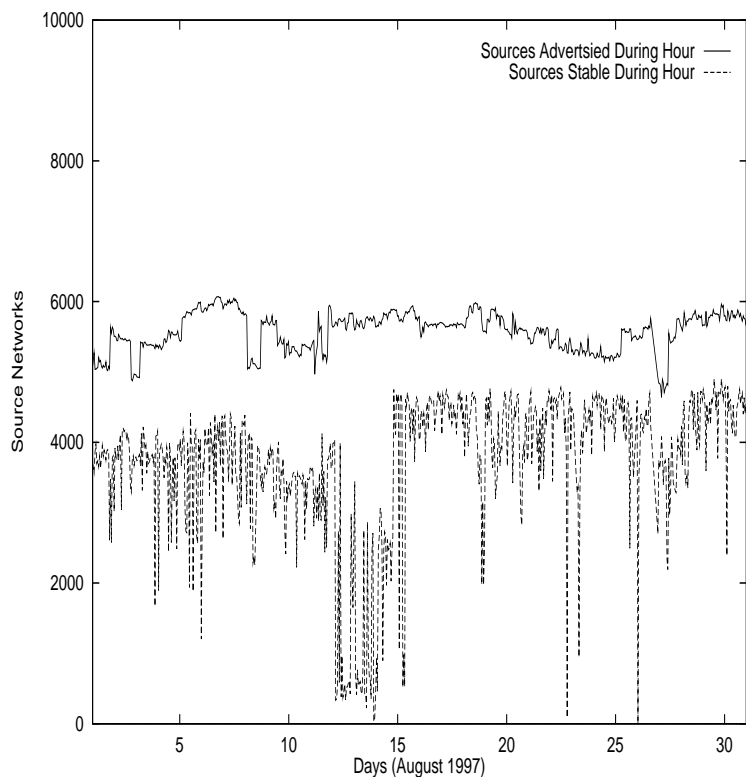


Figure 3: August 1997

of this link. These large drops only indicate that critical links near UCLA failed during that hour and do not reflect faults in the routing protocol itself. Fail-stop behavior by links and routers can only be eliminated by better maintenance of the underlying network hardware and software.

Even after considering the failures of critical links, the overall number of stable routes is considerably lower than anticipated. During any hour, some links and routers may fail. The failure of a few critical links or several less critical links can create a large number of route changes. But the level of instability suggested that other problems may be occurring. In section 3.2.2, we provide data that suggests many route changes are not due to the failure of links or routers.

3.2.2 Average Number of Flaps Per Route

As the previous section indicates, failure of a single critical link may cause many changes. The distance to a source may change a few times as news of the link failure spreads; but it should stabilize and remain constant (at least until another link fails). Figure 4 shows the number of changes that occurred per route during a representative one hour period. Link failures could explain the routes that change a few times during the hour, but link failures are unlikely to account for over 60 changes during only one hour. Figure 5 shows the average number of changes per hour, measured over a several month period. Note a large number of routes consistently change over 60 times an hour.

3.2.3 Number of DVMRP Routes

From the graph in Figure 2 or 3, one might infer there were approximately 6000 distinct sources in the DVMRP routing table. But RouteMonitor discovered this was not the case. While the number of distinct sources updated each hour consistently ranges between 5000 and 6000, a different set of sources was updated each hour. Updates for some sources ceased to appear while updates for previously unknown sources appeared. In fact, nearly 10,000 distinct sources appeared in at least one update during August 1997. Every hour, our picture of the MBone consisted of a *different* set of roughly 5600 sources and the true MBone could consist of nearly 10,000 sources. In January 1998,

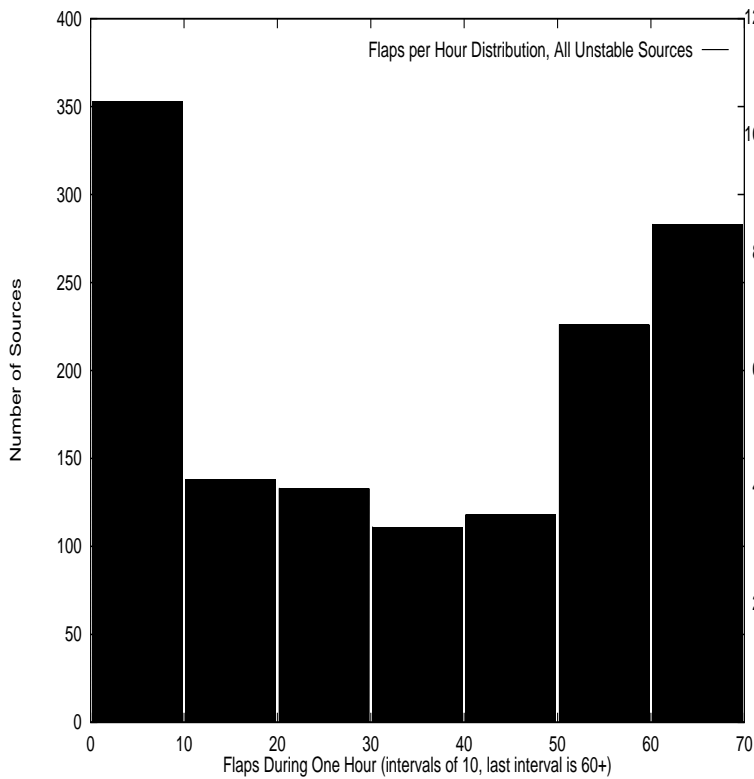


Figure 4: Flaps during one hour (in August 1997, stable routes not shown)

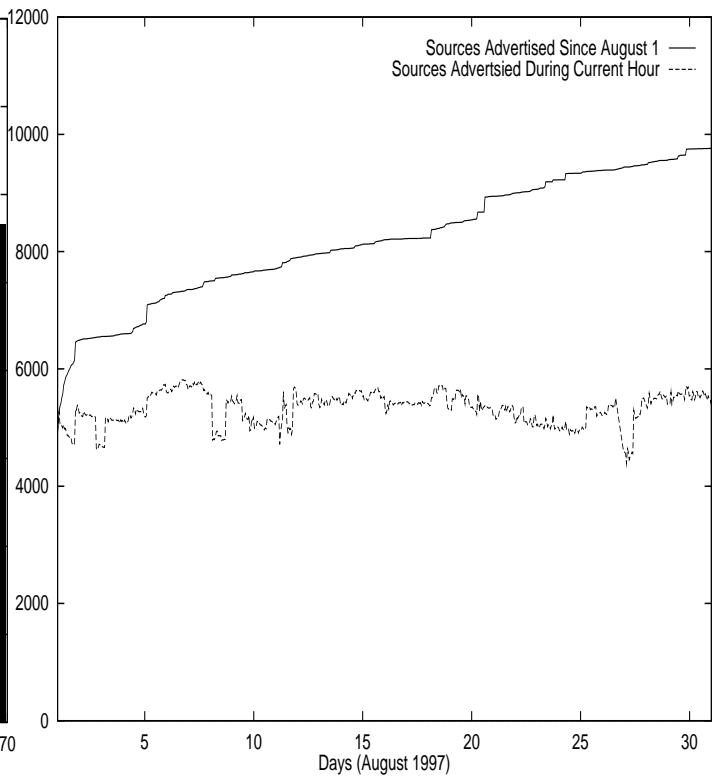


Figure 6: Total sources seen in August

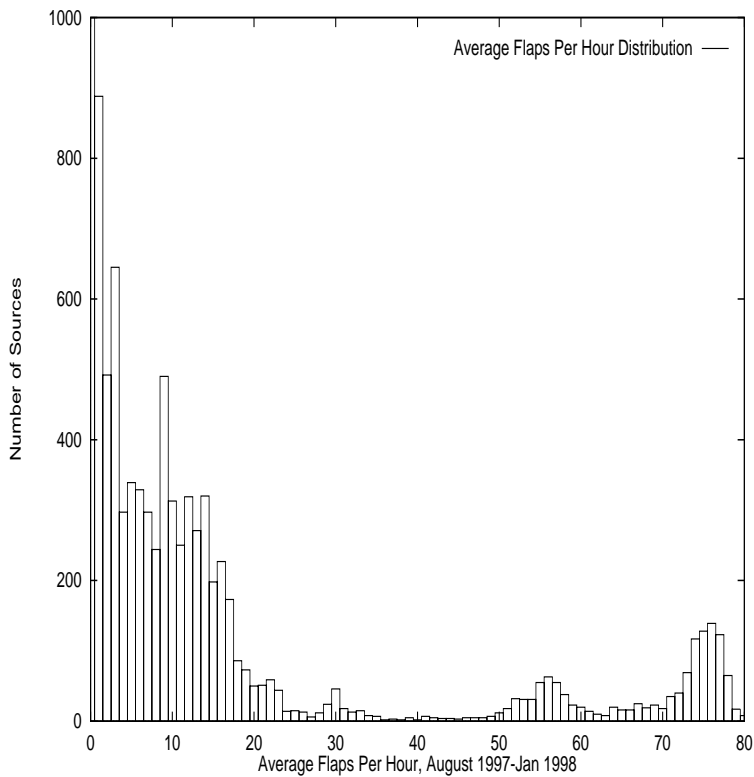


Figure 5: Average Flaps per Hour (y-axis chopped at 1000)

our route database had entries for over 18,000 different sources.

The graph in Figure 6 shows the number of sources advertised each hour and the number of sources advertised in August, 1997. Figure 7 shows the number of sources that **appeared** during an hour and the number of sources that **vanished** during an hour. A source **appeared** if at least one update for the source was received during the current hour and no updates for source were received during the previous hour. Similarly a source **vanished** if no updates for source were received during the current hour, but at least one update for the source was received during the previous hour. The figures show that each hour some sources **appear** while other sources **vanish** and thus UCLA sees a different routing table each hour.

Figure 6 shows the new, previously unknown routes continuously appear. Figure 7 illustrates that this is not due to isolated events. A different number of sources appeared each hour and similarly a different number of sources vanished each hour. The collection of reachable sources in the Mbone was a highly variable set.

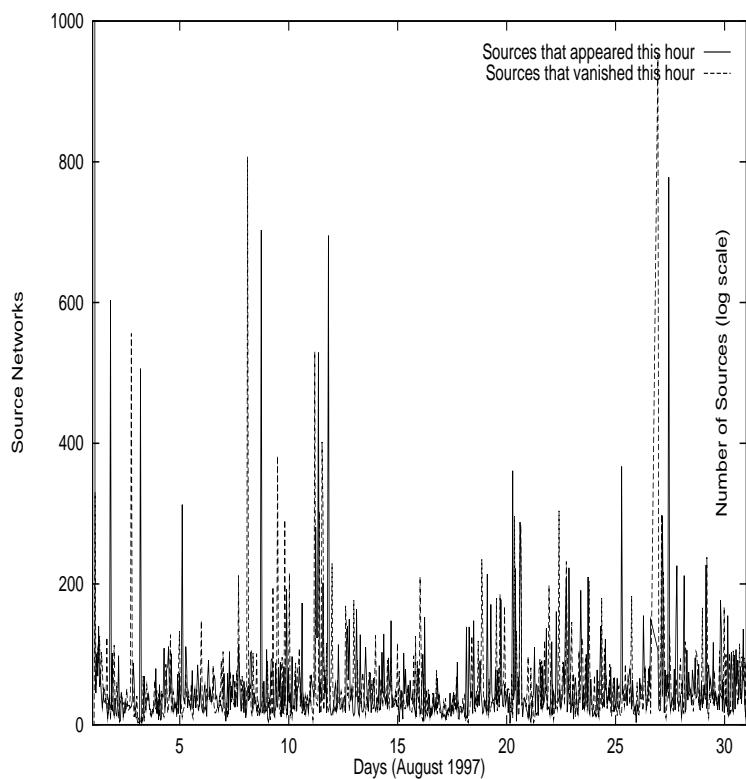


Figure 7: Appearing and Vanishing Routes

A route which **vanishes** or **appears** does not necessarily indicate a fault in the routing protocol. During the span of a month, some valid routes may temporarily **vanish** due to administrative work on their local router. If a critical link is down for an entire hour, this could also cause a valid route to **vanish**. This does not indicate a fault in the routing protocol. It simply represents the protocol correctly reacting to the fail-stop actions of a link or router. But again the large number of changes seemed to indicate some other factor was involved.

RouteMonitor tracked the frequency with routes appeared and vanished and discovered a large number of routes vanish and reappear periodically. We say a route **changes state** every time it **appears** or **vanishes**. By definition, each route changed state at least once, the first time it appeared. Figure 8 shows the distribution of state changes observed during August 1997. The distribution shows that only 1346 sources appeared and then continued to appear in at least one update for each subsequent hour during the month.

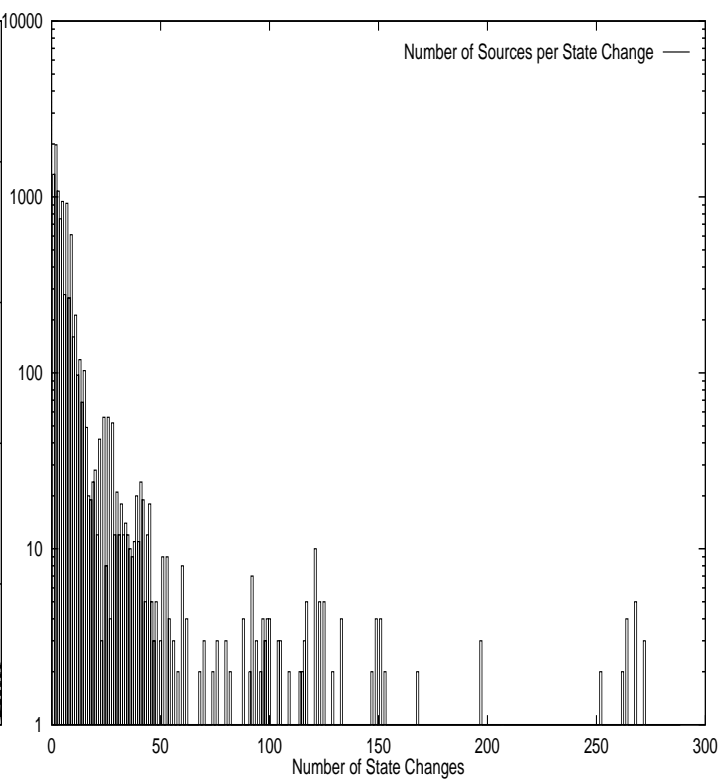


Figure 8: State Changes per Route

3.2.4 Summary of the UCLA View

The view from the UCLA RouteMonitor provided one viewpoint of the DVMRP routing table. We were able to determine that the UCLA routers were behaving correctly, but frequent route changes were being learned from neighboring DVMRP sites. Based on the frequency and overall amount of route changes, it seemed unlikely that these changes were due to fail-stop behavior by links or routers. However, the UCLA view alone was not sufficient to identify the cause of the problems.

Despite the limitations of a single viewpoint, we were able to detect and correct some MBone configuration faults. For example, in late July the number of DVMRP routes suddenly rose by nearly 2000 routes. The RouteMonitor data alerted us to this change and allowed us to easily determine which routes suddenly appeared. The problem was then easily traced to a configuration error at a remote site. The site administrator was contacted and corrected the error. As shown in Figure 2, the number of DVMRP routes quickly soared to over 7600 and then returned to 5600 a few hours later after the administrator corrected the error.

Note that this data is trivial to collect using RouteMonitor, but not easily obtained by other methods. Active tools can not easily capture a picture of the global routing table. By passively observing updates, RouteMonitor collected this data without adding any additional load to the network and without sending any SNMP style query messages to any router. The data did not rely on a polling system which could miss some changes. Every route change reported by the protocol was captured by RouteMonitor and the data is relatively complete from the perspective of the DVMRP protocol. Both the global picture and the specifics of any individual source are easily obtained.

RouteMonitor had allowed us to collect large amount of data without adding any load to local routers. We were able to observe the actions of local routers and provide a real time, easy to view picture of the MBone. Based on this picture, we identified problem behaviors that had not been widely recognized. But based on our single point of observation, we could only determine that the problems were not local.

3.3 Identifying DVMRP Network Layer Faults

To further analyze the problem, we needed to obtain data from other portions of the MBone. While a single RouteMonitor is useful, our approach is based on a collection of RouteMonitors running at different points in the network. The UCLA RouteMonitor identified several sources which change frequently. We needed to see what was happening at these sites or along the path these sites. Since there is no central authority “in charge” of the MBone, there was no single contact to provide the necessary access. Instead, we had to try to determine who was responsible for the sites in question, contact them individually, and try to convince them to give us access to information from the site.

The UCLA RouteMonitor provided two important assets in our attempt to gather data from other sites. The first asset was that we could easily demonstrate the problem by simply pointing to the UCLA RouteMonitor web page. This allowed anyone to see the scope of the problem and helped motivate the interest of other sites. The second important asset was that we did not have to request

access to routers at other sites. Instead, we were able to provide the RouteMonitor software for collecting data. A site could deploy the software on a host, filter the data based on a particular address, and simply send us a text file showing the updates exchanged. This allowed us access to the required information without requiring other sites to give us access to their systems.

We focused on the most frequently changing source addresses and used *mtrace*, the multicast traceroute program, to try and identify interesting points to deploy RouteMonitors. Most sites were receptive to running some RouteMonitor data collection. The data viewed from other sites confirmed the results seen at UCLA. These sites had pathologies similar to those at UCLA and the sources seen as unstable at UCLA tended to be unstable as seen from other sites as well. The system of multiple RouteMonitors allowed us to view the updates sent by various routers and this information led to the discovery of three main routing protocol faults.

3.3.1 Sending Bursts of Updates

When using DVMRP, a router must send an update for each source once a minute. At least one implementation sends the updates for all sources at once. On average, 5600 sources need to be updated. For each source the router sends the source address, netmask for the address, and the distance metric. The updates are sent in approximately 70 consecutive packets, each packet containing updates for about 80 sources.

Neighboring routers receive the burst of updates faster than they can be processed. This results in packet loss as queues overflow. RouteMonitor would also suffer this buffer overflow, but RouteMonitor counts the number of updates received each hour. Looking at the data, we were able to determine that the expected number of updates was not being received and the problem was traced to the buffer overflow. These bursts have been observed to overwhelm several different implementations, including the one that sends the bursts.

Route instability occurs when a router R 's best path to a source is via a bursty router, B . Router R hears an update for the source from B . The route via B is the shortest route so R declares B to be its previous hop toward the source. The bursty

router continues to send updates for the source each minute, but many of these updates are dropped because of buffer overflow at R . If three consecutive updates from B are dropped, R believes the route has timed out and removes it from its routing table. B continues to send updates for the source and eventually some update is not dropped. R then relearns the route via B and the process repeats.

In one particularly bad case, a bursty router advertised routes to 256 local sources. The bursty router sent local routes last and thus the updates for these 256 local routes were frequently dropped by neighboring routers. At our UCLA RouteMonitor, each of these 256 routes changed an average of 68 times per hour. During one particularly bad hour, these routes changed 173 times. At no point during our study were any of these routes stable for an entire hour. This extremely large number of changes is due to initial instability caused by the burst problem combined with other factors that enhance initial instability.

This problem is essentially a protocol implementation error. An implementation needs to send updates at a reasonable rate. The solution to this problem is simply to space out updates rather than send a large burst of updates. Most DVMRP routers distribute the updates evenly over the 1 minute period and do not suffer from the burst problem. We informed the router vendor of the problem and the problem was corrected in more recent software releases.

3.3.2 Interactions with Unicast Routing

Some routers are used for both unicast and multicast routing. When using the DVMRP topology discovery protocol, these routes go into a separate table, and the router uses both tables to determine routes for multicast tree construction. However, the router can be configured to advertise sources from the unicast table to neighboring multicast routers. In certain topologies, the router becomes confused about how to reach these sources.

Consider the topology shown in Figure 9. The router C learns of local network S from its unicast routing table and is configured to advertise network S into DVMRP. C sends a multicast route update to A and B , listing C 's distance to S as 1. A will select C as its previous hop to S and set its distance

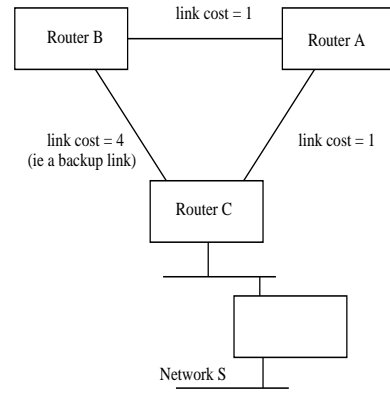


Figure 9: Sample Topology for Section 4.2

to S as 2. When B hears an update from A , it will prefer the route it heard from A (cost 3) over the route it heard from C over the backup link (cost 5). B will then send an update to C listing B 's new distance to S as 3. C should ignore this new route to S since C has a route of cost 1.

However, C 's route to S came from its unicast table. C doesn't have a route for S in its DVMRP routing table, so it accepts the cost 7 route to S from B . This forms a loop where B believes (correctly) its previous hop to S is A , A believes (correctly) its previous hop toward S is C , but C believes its previous hop toward S is B . C sends an update to A informing its distance to S is now 7 (via B). A then informs B that A 's distance to S is now 8. B informs C the distance to S is now 9. Gradually the distance to S increases until the maximum allowable distance is reached, 31 in DVMRP. This behavior where routing updates flow around a loop and count up to the maximum metric is called "counting to infinity." Once the distance to S exceeds 31, C prefers its original unicast cost 1 route to S and the process begins again.

This problem will only occur if C is part of a topology where it may hear an update for a route it introduced and if C is configured to prefer the multicast routes to its own unicast routes. This is the default configuration, because in simpler topologies this is the correct behavior.

In complex systems, where multiple routing protocols can interact, it is extremely difficult to make sure that all cases work correctly. The current state of the art is to make the configuration as flexible as possible, and expect that administrators will configure their systems properly. Since these systems

are so complex, it's hard to even know what the interdependencies are, let alone document them. Unfortunately, this leads to situations like this one where an implicit configuration requirement causes problems for the network.

3.3.3 False Aggregation of Routes

Overlapping routes can sometimes cause confusion in another implementation. Suppose both a specific route 131.179.96/24 and more general route 131.179/16 are both being advertised. This implementation will accept the updates for both routes but advertise only the more general route, 131.179/16, to its downstream neighbors. The correct behavior would be to advertise both the 131.179/16 and the 131.179.96/24 routes to the downstream neighbors.

Instability can occur because a buggy router applies the metric for each route to the more general route in the routing table. In most configurations, the more specific route will have a higher metric than the general route (e.g. the more specific route is injected by a site's internal router and the more general is injected by the border router). However, if the more specific route has a lower metric (e.g. because of "hole-punching" in a CIDR[FLYV93] aggregate), this behavior causes the apparent metric of the general route to vary. The router will accept the specific route with the low metric, update the general route's metric with it, and trigger an update of this new information to its peers. Then, in the next routing update, the router will accept the general route with the higher metric and update the general route's metric back to where it was. This information is triggered also. Then comes the more specific route in the update, and the process starts over again. This causes instability for the general route downstream of this router, and can cause abbreviated counting to infinity in cases where the metric difference is high.

3.3.4 Explanations for the Faults

The RouteMonitor system detected problems in the DVMRP infrastructure that had not been detected by other means. Figures 2 and 3 indicate there was high level of overall instability and this instability had been occurring for at least several months. Figures 4 and 5 indicate some sources

were suffering from severe routing problems. Prior to RouteMonitor, the general problems were not widely recognized and the specific faults had not been detected.

One explanation for this is that routing protocol fault detection is often driven by end user complaints. If the path to a source changes frequently, that source will have difficulty **sending** multicast packets but the ability to receive packets is not directly affected. Only an end user **sending** packets would notice the problem and initiate an ad hoc investigation. Most sites are frequently receivers, but only infrequent senders. Sites with frequently changing routes would only notice the problem if they acted as senders. Other sites might notice an added load, higher packet loss rates, or other general problems. The fact these problems did not lead to more rigorous investigations may simply indicate that end-users will suffer a certain level of poor performance before actively seeking help.

In a more general sense, higher layer protocols are designed to be robust. Legitimate route changes do occur and higher layer protocols should work despite these changes. Higher layer protocols ranging from TCP[Pos81] (unicast only) to audio tools have proven extremely robust in hiding underlying problems. But hiding instability has performance costs for the end user as well as the network. Routing protocol fault detection tools need to be as robust in finding errors as higher layers are at hiding them.

Without a problem report for a specific source, it is not surprising that active monitoring tools such as *mtrace* did not discover the problem. But it may seem surprising that passive monitoring tools such as SNMP did not find the problem. The difficulty lies in noticing the difference between legitimate change and problem behavior. Using traps or polling, SNMP can detect route changes or route time outs. But routes can change or time out for a number of valid reasons such as the failure of a critical link. When a link does fail, the route may change several times before converging on the new correct route. Thus route changes, even several route changes, would not necessarily indicate a problem. To detect the problem, one would have to observe that a route continuously changes over a sufficiently long time such as days or weeks.

This kind of long term observation costs the

router resources. Considering that there are over 4000 active routes, long term observation of all routes may be tedious and unrealistic. Long term monitoring of some routes has a probability of catching the problem, but how should one decide which routes to monitor? What threshold of changes should raise a warning? A high threshold can miss problems. A low threshold can report hundreds of potentially faulty routes. Of these potentially faulty routes, which does one pursue further?

We do not claim it would have been impossible to discover these problems using only a system like SNMP. However, periodic polling via SNMP puts extra load on production devices and only leads us to *suspect* some routes. It does not yet identify the location or the cause of the problem. Finally, note that monitoring the actual faulty router would not have revealed any problems. For example, the bursty router from section 3.3.1 would simply report stable routes. Even if one suspected a problem with lost updates, the bursty router believes it is sending the correct number of updates and one would eventually be forced to observe the packets directly.

RouteMonitor provided a simple method to detect problems and added no cost to the routing protocol or the routers. RouteMonitor is not involved in forwarding packets. It runs on a host and its primary job is to collect and analyze data. For RouteMonitor, a natural question to ask is which routes change most frequently. In fact, RouteMonitor automatically posts a list of the current week's worst routes, the *frequent flappers*. The frequent flapper list is sorted by the average number changes per hour during that week. Based on this data, some suspect routes are immediately obvious. Closer analysis of the data, such as figures 4 and 5, clearly indicates problems are occurring.

3.4 Results Achieved with the DVMRP RouteMonitor

Most of the frequently changing routes could be traced to one of three problems described above. We contacted the vendors of each implementation mentioned and they produced fixes for the bugs. The fixes for sending bursts of updates (section 3.3.1) and false aggregation (section 3.3.3)

have been introduced in newer releases of these implementations. The unicast/multicast route confusion (section 3.3.2) can be solved via the router's configuration file and details on how to do this were posted on the RouteMonitor web page.

RouteMonitor also sent weekly email message to the mailing lists for MBone router operators and its European equivalent. At the end of each week RouteMonitor determined which routes changed the most during the week. Only one route per autonomous system (AS) was included in the list. The email was sent to the mailing lists in order to inform sites of potential problems and help speed deployment of patches which could correct much of the instability in the network.

This approach was successful and the sites responsible for the worst problems deployed the patches. In the current MBone, the average number of stable routes has improved from 67% to 95% and no routes currently average over 60 changes per hour. RouteMonitor data is no longer sent to any mailing lists. The RouteMonitor web site continues to provide hourly reports at <http://ganef.cs.ucla.edu/~masseyd/Route>. RouteMonitor is freely available and has been installed by over a dozen sites.

4 Summary

Routing protocol faults can occur due to faulty hardware, incompatible implementations, incorrect configurations, protocol design flaws, and malicious attacks. A fault at a single site can have far reaching consequences for large portions of the network. Routing protocol faults have been a source of trouble since the the creation of the ARPANET and continue to present problems for the modern Internet and the MBone.

Reliable systems for monitoring the routing protocols are required to ensure reasonable performance and to guard against malicious attacks. Current systems are able to cope with only some of the potential threats. We have presented an effective approach to network monitoring and fault detection. Our system added a collection of virtual routers called RouteMonitors into the network. These virtual routers allowed us to monitor the network without adding load to the routing protocols or routers. By focusing on the specific routing pro-

ocols, our system can detect faulty behavior that may elude other methods.

In the MBone, we discovered that a relatively small number of faults had been creating widespread problems. Some sites were unable to send data due to frequent route changes. All sites suffered from some type poor performance, ranging higher packet losses due to excessive DVMRP overhead at the routers. Existing tools and trouble reports by end-users had not detected these problems.

Using the DVMRP RouteMonitor system, we were able to identify the source of these problems. RouteMonitor accomplished this without adding additional load to the DVMRP routers and without generating additional traffic. It also facilitated sharing between different administrations. RouteMonitor allowed sites to share routing data without sharing access to internal routers and hosts. Using this system, we were able to obtain data from several sites, detect and correct the faults, and substantially improve the performance of the MBone DVMRP infrastructure.

DVMRP RouteMonitors continue to provide a system for monitoring and detecting faults in the DVMRP infrastructure. The RouteMonitor software is freely available from <http://ganef.cs.ucla.edu/~masseyd/Route> and has been deployed by a number of sites. RouteMonitor can be installed on any MBone host by simply downloading the package and running the install script.

4.1 Future Work

This work primarily discussed a DVMRP RouteMonitor, but the same principle applies to protocols other than DVMRP. Work is underway on an MBGP RouteMonitor to study the behavior of this new protocol. Protocols such as MBGP and BGP offer a greater potential for automatic fault detection. Route updates for these protocols include a path to the source instead of simply the distance reported by DVMRP. This provides RouteMonitor with more information to use in constructing views of the network and detecting routing protocol faults.

Automatic detection of faults is the objective of any monitoring system. The results presented here

used a combined automated RouteMonitor work with manual analysis of the data. A number of faulty behaviors could be automatically detected by having RouteMonitor automatically check certain axioms. For example, the number of updates for a source must be greater than the number of minutes over which the source has been observed. The number of sources reported by a router must equal the number of sources advertised by neighboring routers. We are investigating enhancements that allow for a higher degree of automated detection.

As we illustrate, fault detection often requires more than one observation point. How to identify the optimal observation points for a given topology is an open question. The optimally deployed RouteMonitors could exchange data to automatically locate the source of network problems.

An ideal scenario would be for cooperating RouteMonitors to be deployed throughout the Internet. While technically possible, sharing information between different administrative domains is often a challenge in diplomacy and Internet-wide RouteMonitors are perhaps unlikely. However, within portions of the Internet RouteMonitors can provide substantial detection ability. Organizations can choose to easily cooperate and share RouteMonitor data in response to problems. RouteMonitors can be deployed within a large ISP or corporate network to provide fault detection within that portion of the network as well as a view of what outside networks are advertising.

A coordinated, distributed system of RouteMonitors can give insight into the behavior of a routing system that is not available using any other method. This insight can help network administrators to better understand their networks, not only to understand what is normal but also to help gain understanding of abnormal behavior.

References

- [AMMP98] A. Adams, J. Mahdavi, M. Mathis, and V. Paxson. Creating a Scalable Architecture for Internet Measurement. In *Proceedings of INET 98*, July 1998.

- [Bal97] A. Ballardie. Core Based Trees (CBT version 2) Multicast Routing. RFC 2189, SRI Network Information Center, September 1997.
- [BCKR98] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol Extensions for BGP-4. RFC 2283, SRI Network Information Center, February 1998.
- [CDL⁺] Ramon Caceres, Nick Duffield, Francesco LoPresti, Joe Horowitz, Jim Kurose, Don Towsley, and Vern Paxson. MINC: Multicast-based Inference of Network-internal Characteristics. <http://gaia.cs.umass.edu/minc/>.
- [CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). RFC 1157, SRI Network Information Center, May 1990.
- [Dee89] S. Deering. Host Extensions for IP Multicasting. RFC 1112, SRI Network Information Center, August 1989.
- [EFH⁺98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol specification. RFC 2362, SRI Network Information Center, June 1998.
- [Eri94] H. Eriksson. MBONE: the Multicast Backbone. *Communications of the ACM*, 37(8):54–60, August 1994.
- [FC] W. Fenner and S. Casner. draft-ietf-idmr-traceroute-ipm-03.txt. Internet Engineering Task Force Draft.
- [FF62] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- [FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519, SRI Network Information Center, September 1993.
- [Hed88] Charles Hedrick. Routing Information Protocol. RFC 1058, SRI Network Information Center, June 1988.
- [HPT97] Ralf Hauser, Tony Przygienda, and Gene Tsudik. Reducing the Cost of Security in Link-State Routing. In *Proceedings of ISOC Symposium on Network and Distributed System Security*, pages 93–99, February 1997.
- [Hui95] Christian Huitema. *Routing in the Internet*. Prentice-Hall, 1995.
- [Jac] Van Jacobson. pathchar. <http://www.caida.org/Tools/Pathchar/>.
- [LMJ97] C. Labovitz, G. Malan, and F. Jahanian. Internet Routing Instability. In *Proceedings of ACM Sigcomm*, September 1997.
- [LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
- [MA] D. Makofske and K. Almeroth. The MHealth project. <http://imj.ucsb.edu/mhealth/>.
- [MFR78] John M. McQuillan, Gilbert Falk, and Ira Richer. A Review of the Development and Performance of the ARPANET Routing Algorithm. *IEEE Transactions on Communications*, 26(12):1802–1811, December 1978.
- [MJ98] G. Malan and F. Jahanian. An Extensible Probe Architecture for Network Protocol Performance Measurement. In *Proceedings of ACM Sigcomm*, September 1998.
- [Moy89] J. Moy. The OSPF Specification. RFC 1131, SRI Network Information Center, October 1989.

- [MRR80] John M. McQuillan, Ira Richer, and Eric Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, 28(5):711–719, May 1980.
- [MW77] J. M. McQuillan and D.C. Walden. The ARPANET Design Decisions. *Computer Networks*, 1, 1977.
- [Per88] Radia Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT Lab. for Computer Science, 1988.
- [Pos80] Jon Postel. User Datagram Protocol. RFC 768, SRI Network Information Center, August 1980.
- [Pos81] Jon Postel. Transmission Control Protocol. RFC 793, SRI Network Information Center, September 1981.
- [RL94] Y. Rekhter and T. Li. Border Gateway Protocol 4. RFC 1654, SRI Network Information Center, July 1994.
- [Ros81] E.C. Rosen. Vulnerabilities of Network Control Protocols: An Example. *Computer Communications Review*, 11(3):10–16, 1981.
- [Ros91] M. T. Rose. *The simple book : an introduction to management of TCP/IP-based networks*. Prentice-Hall, 1991.
- [SCFJ96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, SRI Network Information Center, January 1996.
- [SMGLA97] Bradely R. Smith, Shree Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance Vector Routing Protocols. In *Proceedings of ISOC Symposium on Network and Distributed System Security*, pages 85–92, February 1997.
- [TMW97] K. Thompson, G. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*, November 1997.
- [VWW97] B. Vetter, F. Wang, and S.F. Wu. An Experimental Study of Insider Attacks for the OSPF Routing Protocols. In *Proceeding of IEEE ICNP*”, pages 293–300, October 1997.
- [Wei] Liming Wei. MRM. IETF MBoneD mailing list; <ftp://ns.uoregon.edu/mailling-lists/mboned.archive>.
- [WPD88] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, SRI Network Information Center, November 1988.
- [WW98] F. Wang and S.F. Wu. On the Vulnerability and Protection of the OSPF Routing Protocols. In *Proceeding of IEEE ICCCN*, October 1998.