

mPlane: An Architecture for Scalable Fault Localization

†Ramana Rao Kompella, Alex C. Snoeren, and George Varghese

†Purdue University and University of California, San Diego

ABSTRACT

Customers are increasingly demanding service-level guarantees from ISPs. ISPs use active probes for monitoring network health and use tomographic approaches to localize any end-to-end problems observed, which are typically postulated as underconstrained problems, and hence, often limited in accuracy. Active probes are also fundamentally unscalable; operators cannot afford to inject them at high frequencies. We present an architecture, mPlane, that addresses these problems. The key idea in mPlane is to break paths into segments consisting of router forwarding paths and links, and conduct measurements on a per-segment basis. Node measurements are obtained through scalable high-fidelity hardware primitives, while the link measurements are conducted using segment-level active probes at low frequencies. Unfortunately, upgrading all routers with these primitives faces significant deployment hurdles; we propose an incremental deployment strategy that picks the most important routers to upgrade. Our simulations with RocketFuel topologies indicate a partial deployment on 15% of an ISP's routers can yield two orders of magnitude decrease in measurement overhead, and reduce the average localization granularity from 4 hops to about 1.5.

Categories and Subject Descriptors

C.2.3 [Computer Communication Networks]: Network management

General Terms

Algorithms, Measurement

Keywords

fault localization, measurement, routing algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ReArch '09, December 1, 2009, Rome, Italy.

Copyright 2009 ACM 978-1-60558-749-3/09/12 ...\$10.00.

1. INTRODUCTION

The Internet began as a best effort communication medium, with no explicit performance or reliability guarantees. This feature allowed the Internet architecture to be relatively simple with the complexity pushed towards the end points. It is this simplicity that allowed the Internet to scale significantly, both in terms of performance as well as in the number of users. Despite its tremendous success, the Internet still lacks the kind of reliability and robustness we expect from critical infrastructure. Network events, such as equipment failures, often cause disruptions in service, or even complete lack of connectivity between end hosts, often causing frustration to end users and financial damage to enterprises. With the increasing reliance on the Internet for mission-critical applications and with more enterprises relying on 'cloud' applications, disruptions, however small they might be, are quite significant in their impact.

Unfortunately, it is not easy for ISPs to debug their network should performance problems occur. ISPs need to instrument mechanisms to actively measure and monitor their network and *detect* end-to-end problems in the network, such as delay spikes or burst losses in their network. Operators today use active probing techniques for detecting problems. Active probing approach, however, requires $O(n^2)$ probes¹ between n end points and hence, can lead to scalability issues, especially when they need to be issued at high frequencies for high-fidelity measurements. To cope with the scaling issues of active probes, network operators today probe less frequently (say one probe every few seconds or minutes), or reduce the value of n by aggregating end-points, or measure between fewer points and extrapolate (e.g., iPlane [9] assumes that delays to "nearby" nodes will be similar). The downside is that many performance problems may not be easily detectable.

Once operators detect the presence of these problems, operators need additional tools to identify their root causes and fix them permanently; a key intermediate step in diagnosis is *localizing* the problem. Operators today use *inference* algorithms [1, 14, 3] for localization; the postulated inference problems are fundamentally under-constrained. Thus, even

¹It is possible to reduce this to perhaps $O(n \log n)$ but, most operators just issue all-pair probes to simplify measurement process.

for the most sophisticated of these tools, operators still need to perform significant manual debugging and troubleshooting to locate the root cause of any performance problem. We believe that these problems are fundamental in nature and unless we make changes at the architectural level, these problems are not likely to disappear.

In this paper, we propose a fault localization architecture called mPlane, that addresses the two afore-mentioned problems of scalability and localization. In mPlane, end-to-end paths are broken down into ‘segments’, the properties of each of which are individually measured and monitored at high fidelity. Segments constitute both router forwarding paths and router-to-router links. A network consisting of n routers and m links can be broken down into $O(nd^2)$ per-router forwarding paths with d representing average degree, and $O(m)$ measurements for individual links. Breaking an end-to-end path into relatively short segments allows higher fidelity measurements with fewer resources compared to the end-to-end approach, since it eliminates redundancy stemming from probes between different pairs of end nodes measuring shared common links. Measuring these segments individually also allows easy and direct fault localization compared to the indirect inference approach used today. Note however that our architecture does not obviate the need for active probes; it merely allows active probes to be less frequently injected and allows direct fault isolation.

While the above architecture clearly makes sense, there are two immediate issues one needs to address for it to become reality. The first concerns devising appropriate primitives that can perform high-fidelity measurements scalably at each of the individual segments. In our recent effort [6], we have outlined a data structure called lossy difference aggregator (LDA) that can estimate router delay and loss measurements with high fidelity on a per-segment basis. The design of LDA is based on simple coordinated streaming algorithms we have developed to contain the space and communication complexity of conducting measurements that involve two monitoring points (as is required for measuring one-way delays). While we can use LDA for router-level measurements, mPlane itself is oblivious to LDA; any scalable measurement data structure would suffice.

While ubiquitous deployment of primitive such as LDAs within routers will significantly improve the diagnosis capabilities of today’s routers, such a fork-lift upgrade of the network is going to be hard. Thus, a major issue in realizing the architecture is partial deployment, which is the *main focus* of this paper. Specifically, fault localization requires covering all the segments in the network with direct measurements, and thus, upgrading only a few routers in the network with LDA-like primitives will be insufficient. Another issue concerns which routers to pick for upgrading; specifically, the question is whether there exists a specific order in which one should upgrade the routers.

In mPlane, we propose to solve the first problem by extending the notion of a segment to include ‘virtual’ segments

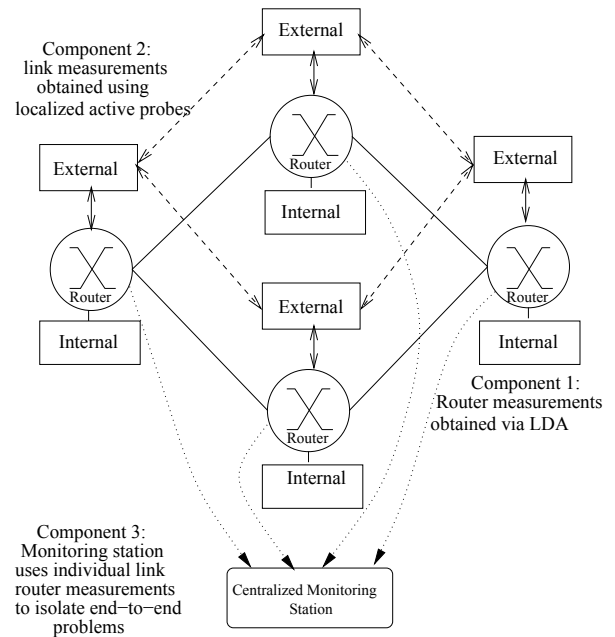


Figure 1: mPlane architecture. Each router is equipped with link-level and router-level measurement modules.

consisting of a path between two upgraded routers. To measure these segments, we propose that the upgraded or measurement friendly routers issue active probes to other upgraded routers to directly measure the properties of that virtual segment. One challenge here is for routers to determine *dynamically* which other routers to probe in order to completely cover the network. We propose a dynamic self-organizing mechanism using simple extensions to OSPF to allow these upgraded routers to form a virtual overlay and dynamically compute their measurement responsibilities (see Section 2). For second problem, namely the strategy for upgrade, we propose an intelligent strategy that identifies the ‘hot’ routers which are positioned among the largest number of shortest paths. We show in Section 3 that this approach results in significantly reducing the probe bandwidth compared to a naive random upgrade strategy.

2. M-PLANE ARCHITECTURE

The key idea in the architecture of mPlane consists of breaking an end-to-end path into individual router- and link-level segments. Once these segments are individually monitored, localizing the root cause of any end-to-end problem will be easy and direct from the individual segment measurements. Accordingly, the mPlane architecture shown in Figure 1 consists of two components—internal and external measurement modules. The internal measurement module measures performance metrics of interest (average delay, variance, and loss) for all the internal forwarding paths within a router (*e.g.*, every ingress-egress pair). The external measurement module, on the other hand, is responsible for measuring link-level properties from the egress of a router to the ingress of

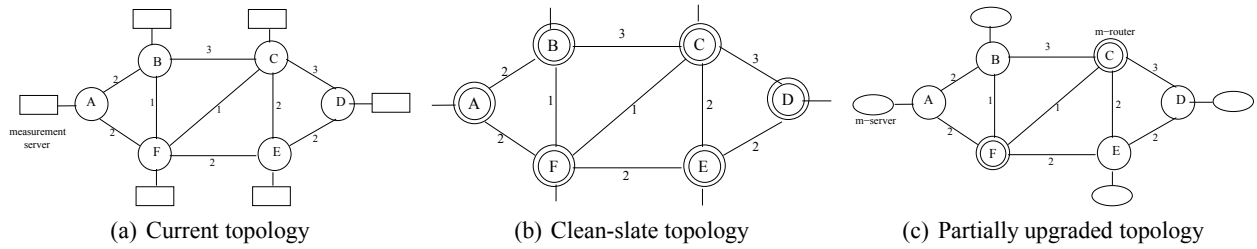


Figure 2: Partial deployment and clean-slate design of the mPlane architecture.

the adjacent one. The router reports the measurements generated by these modules to a centralized monitoring station that is equipped with a topology monitor (*e.g.*, OSPF monitor [12]). The monitoring station first identifies the forwarding paths in the network using the topology monitor, and then isolates the root causes of any perceived problems by identifying the relevant router- and link-level measurements responsible for that path and observing their properties.

Internal measurements: Most variability in end-to-end measurements lies within the routers, where queuing and other functions such as switch scheduling cause non-deterministic delays and losses within the router. Fortunately, we note that in many real routers, forwarding metrics (*e.g.*, loss, delay) depend on the forwarding class more than the particular flow. For example, all flows traveling between the same input and output ports of a router in a given QoS class are often treated identically in terms of queuing and switch scheduling. Thus, we group such flows into what we call a measurement equivalence class (MEC).

We propose the use of a scalable measurement primitive such as LDA [6] per MEC within the router. Briefly, LDA works by maintaining by accumulating timestamps in a counter for a set of packets at both the sender and receiver. The sender, at the end of a measurement interval, transmits this timestamp accumulator to the receiver which then computes the average delay. In order to deal with loss, LDA randomly hashes packets to different buckets, and maintains individual accumulators in each buckets. It also uses sampling to control the number of buckets that can get corrupted due to packet loss. As we mentioned before, our architecture is not tied to LDA; any scalable measurement primitive that can report our measurements would work equally well.

For n routers in the network, the number of measurements in the network equals $\sum d_i^2$ where d_i is the degree of router i , assuming we instrument one LDA per ingress-egress interface pair in each router. Passive measurement solutions such as LDAs do not inject any active probes, and so, a positive side-effect of LDA-like primitives is that there is very minimal probe bandwidth, which in turn, allows our architecture to scale quite well, in addition to allowing direct fault isolation of end-to-end problems.

External measurements: Most link-level properties remain invariant regardless of the traffic distribution. For example, the propagation delay and transmission times of a

packet on the link remain similar irrespective of whether it is a delay-sensitive VoIP packet or a best-effort TCP packet. Nevertheless, it is required to measure link properties constantly since, lower-layer optical devices can intelligently re-route traffic (*e.g.*, a SONET ring can mask failures), affecting propagation delay and loss properties. Direct link measurements can be conducted using lower frequency probes since there is usually not much variability in link metrics. One way to essentially remove the need for any additional probes, is to rely on packets that are already exchanged between routers (*e.g.*, time synchronization packets, or OSPF Hello probes as done in [11]). However, this scheme is likely not going to work if we assume virtual segments between two upgraded routers that are not directly adjacent to each other. For such measurements, mPlane requires that routers inject direct active probes to another upgraded router. We explain this scenario in detail in Section 2.2.

2.1 A clean-slate deployment

Our objective is to cover all possible measurement segments in the network, since performance problems can occur anywhere in the ISP network (or the AS). Currently, without support from routers, measurements are performed through measurement servers (m-servers). In Figure 2, we show a toy topology with six routers connected via undirected edges and associated edge costs. Attached to each of the routers is a measurement server (shown in Figure 2(a)) that issues data-plane probes to other such measurement servers to measure path properties of interest.

Figure 2(b) shows the sample topology with all routers upgraded (called m-routers) to perform internal and external measurements outlined before. Following the shortest-path routing, the path between A and C goes through F , *i.e.*, $A.F.C$. This path $A.F.C$ can be broken down into the following segments: 1) A 's ingress to A 's egress (router-level), 2) A 's egress to F 's ingress (link-level), 3) F 's ingress to egress (router-level), 4) F 's egress to C 's ingress (link-level), 5) C 's ingress to C 's egress (router-level). Each router needs to individually reports measurements for each of the path segments either internal to or originate/terminate at the router.

2.2 Incremental deployment

In a partial deployment, we assume that only a subset of routers is upgraded. In Figure 2(c), we show the topol-

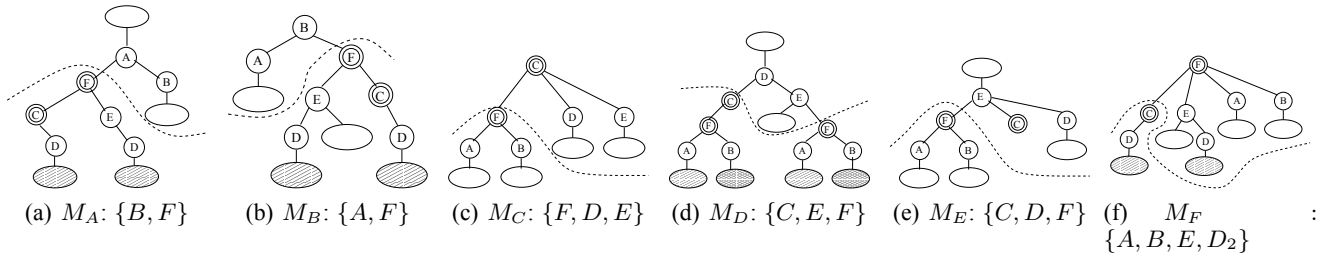


Figure 3: In this figure, we show the various shortest-path trees constructed locally by the m-servers and the m-routers to determine which set of segments to monitor. X_i refers to the i th-shortest path to X , in the case when a router X can be reached via multiple shortest paths.

ogy when two out of the six routers are upgraded to m-routers. This incremental deployment of mPlane proceeds in three steps discussed below. In the first step, the set of measurement servers connected directly to the m-routers are removed since their functionality is subsumed by the m-routers. Further, the set of measurement servers directly connected to non-upgraded routers are transformed into m-servers that also listen to the topology updates (OSPF LSAs) in the network. Thus, the m-servers are capable of reconstructing the forwarding paths in the network similar to the m-routers.

In the second step, each m-server or m-router identifies a set of nodes for which it monitors properties, called an *m-set*. It does so by first computing a self-sourced shortest-path spanning tree using Dijkstra’s algorithm. The shortest path trees computed at each of the six nodes is shown in Figure 3. The m-router does not need to explicitly perform this computation and can leverage the existing shortest-path tree already computed by the OSPF process on the router. It then determines the m-set by making a cut in the tree whenever an m-router or an m-server is encountered. If an m-router is encountered, the rest of the paths to various destinations in the subtree of this m-router are monitored by that m-router. An m-server is encountered if no such m-router exists along the path (and hence it has to monitor this path itself). In Figure 3, we show such m-sets for all the routers for the toy topology in Figure 2. Note that in Figure 3(f), the m-set consists of D_2 that corresponds to the second shortest path to D through E . The router F does not need to monitor the first shortest path to D through C .

Finally, m-router to m-router or m-server links (or virtual link consisting of paths through non-upgraded routers) are monitored using regular active probing. The m-routers report the internal measurements and the measurements to the nodes in the m-set periodically to a monitoring station using specialized measurement state packets (MSPs). The m-servers only report the measurements to the nodes in the m-set within the MSP. The monitoring station uses the topology information to build n shortest-path trees (with each node as source), and uses the MSPs to directly diagnose any end-to-end problem in the network.

Our architecture also accommodates topology changes due to link failures or other reasons. Similar to what happens to-

day, the m-routers (and the m-servers) recompute the new shortest paths when they receive OSPF LSAs, and update their m-set by identifying the cut in the shortest-path tree again. During such periods, every m-router must continue to measure properties of the links or virtual links to the routers in both old and new m-sets for a configurable amount of time to ensure all paths are covered before and after reconvergence. Afterwards, they phase out the routers in the old m-set and restrict measurements to only those in the new one.

2.3 Advertising presence of m-routers

Each m-router needs to identify the presence of other m-routers in the network in order to construct its m-set. One way to do this is to configure each of the m-routers with information about the others in the network. Every time a new upgraded router is added, however, all the existing m-routers would need to be reconfigured. Instead, we leverage on the existing OSPF protocol (or IS-IS) to allow m-routers to advertise their presence to other m-routers in the network. We propose to use one of the reserved bits in the *Options* field of the OSPF control-plane messages for this task; the field exists precisely to advertise special capabilities of routers in the network. Currently, two out of eight bits reserved for the options field are used: the T-bit (to indicate type-of-service capability) and the E-bit (to indicate external routing capability). We use one of the six unused bits (which we call the M-Bit) to advertise the presence of an m-router. A router that transmits OSPF control-plane messages with the M-bit set indicates that it is capable of performing and reporting router- and link-level measurements. Legacy routers in the network do not pay attention to this bit.

3. EVALUATING THE BENEFITS

We now attempt to quantify the benefits achieved by incrementally deploying our architecture in real networks. Lacking access to actual tier-1 ISP topologies, we conduct our evaluation using the Rocketfuel topologies annotated with inferred link weights [13]. Despite the known deficiencies of this data, they suffice to demonstrate general trends. We compare the benefits of upgrading in a naive (random) fashion to an intelligent upgrade strategy.

We use a simple metric called *probe hop count* to quanti-

tatively describe the benefit achieved by upgrading existing routers to m-routers. Probe hop count is defined as the sum of all the hops taken by every active probe that traverses the network. When active probes are issued from every measurement server to another, this translates to the sum of hop-lengths of all the $O(n^2)$ shortest-paths (including the multiple paths between a given pair of routers) in the network. On the other hand, in the mPlane architecture, the probe hop count reduces to the total number of links in the network, since each m-router transmits messages only to its adjacent routers. While the complexity of the probes is different in both the cases (active probes versus synchronization messages), we ignore this difference in this metric. The algorithm shown in Algorithm 1.

In order to identify candidate routers to upgrade, we guide the search in the direction of reducing the probe hop count metric as much as possible. In particular, we select the routers that reduce the probe hop count the most. Our algorithm first calculates the shortest paths between all pairs of end points, including the duplicates (which may be used by equal-cost multi-path routing algorithms). It then computes the number of shortest paths that traverse each router by incrementing the counts for all intermediate routers on each path (excluding the source and destination of a path). Then, it selects the router with the maximum count as the router to upgrade. To select additional routers, the algorithm breaks all paths traversing the selected router into two segments—source to the router and router to destination—and adds them to the set of shortest paths. If any of these segments already exist then they are not added to avoid double counting. Note that only the paths that pass through a router contribute to the count of that router; paths that originate or terminate at a given router do not contribute to its count. This step is important to ensure that the search process always identifies intermediate routers, as opposed to access routers, which do little to break up source-destination paths.

Figure 4 shows the results of both upgrade strategies on three representative Rocketfuel AS topologies (results were similar on all of the topologies we considered). The curve for all the topologies is convex in shape; upgrading the first few routers results in maximum benefit, while the marginal benefit reduces drastically after a while. On average, upgrading about 15% of the routers in an intelligent fashion results in a two order-of-magnitude reduction in the probe hop count. For example, the Sprint topology in Figure 4(a) requires approximately one million end-to-end active probes to measure each path without any upgraded routers. Upgrading 45 routers out of 315 results in a probe hop count of only 10,000—a two order-of-magnitude reduction in measurement overhead.

Figure 5 shows the average localization granularity as well as maximum and minimum localization granularity, in terms of the size of an average segment. We can observe that the average localization granularity also drops down very rapidly (convex) for all the ISP topologies, indicating that

Algorithm 1 IdentifyRoutersToUpgrade($V, E, \text{numUpdate}$)

```

1:  $S = \text{ComputeShortestPaths}(V, E)$ ;
2:  $U = \{\}$ ;
3:  $\text{numiter} = 0$ ;
4: while ( $\text{numiter} < \text{numUpdate}$ ) do
5:   for path  $p \in S$  do
6:     for router  $r \in p - \{\text{src}, \text{dst}\}$  do
7:        $\text{count}[r]++$ ;
8:     end for
9:   end for
10:   $\text{maxRouter} = \text{findMax}(\text{count})$ 
11:   $U = U + \{\text{maxRouter}\}$ 
12:  for path  $p \in S$  do
13:    if  $\text{maxRouter} \in p$  then
14:       $S = S - \{p\}$ 
15:       $p_1 = \text{split } p \text{ from src till maxRouter}$ 
16:       $p_2 = \text{split } p \text{ from maxRouter till dst}$ 
17:       $S = S + \{p_1, p_2\}$ 
18:    end if
19:  end for
20:   $\text{numiter}++$ ;
21: end while

```

upgrading a small number of routers can quickly achieve almost all the benefit. The benefit is achieved much slower, however, in the case of random upgrade. In particular, for the Sprint topology, upgrading about 1/6th of the routers (50 out of about 300) reduces the localization granularity to around 1.5 from 4. Of course, this is assuming only direct localization. If we were to couple this with other inference techniques, we can reduce this even further.

4. RELATED WORK

Our mPlane architecture is based on router-based active and passive measurements. For active measurements, there exists a large body of prior work (e.g., [10]) in the measurement of specific properties such as delay, loss, available bandwidth, per-hop capacity and so on. Our architecture leverages these tools in the localized segment measurements conducted by routers. Our architecture allows end-to-end active probes to be issued at lower frequencies, but does not completely eliminate the need for them. In that sense, our architecture is complementary to the end-to-end active probes.

Tomographic approaches (e.g., [14, 3, 7]) measure end-to-end path properties via active probes and use topology to infer individual hop-properties. As we discussed before, tomography approaches are often based on under-constrained formulations and thus the results may not be as accurate. There also exists some related work in also designing router primitives for measurements. Machiraju *et al.*, argue for a measurement friendly network architecture where individual routers provide separate priority levels for active probes [8]. The use of router support for sampling packet trajectories has been suggested by Duffield *et al.* in [4]. Many other

