Network Performance Management

S. Keshav

The rapid growth of the Internet is greatly straining the existing infrastructure. Managing its performance (in terms of providing connectivity and an acceptable mean response time and mean bandwidth per connection) is a challenging, if not impossible, job. Users commonly experience painfully slow access to networked file systems in the local area, and loss of connectivity to web sites due to route flapping in the wide area. In time, the situation is only going to get worse. Networks are being added to the Internet at a rapid clip. Moreover, ADSL and cable modem technologies are poised to solve the last mile problem, providing high-bandwidth access from home. This will not only further strain backbone networks but may place a potentially misconfigured router in every household! In view of these problems, and particularly given the need to provide users with some form of quality of service guarantees, performance management of the networking infrastructure will soon become critically important.

Providing users with a consistent and reliable quality of service has usually been studied from the perspectives of scheduling, traffic profiling, signaling, and reservations[1]. However, in practice, this may prove to be too narrow a view of the network. Indeed, the user experience, in terms of service quality, is probably determined more by the quality of network management than by specific scheduling and reservation choices. This is true particularly for legacy networks, which have few mechanisms for providing per-user or per-application quality of service guarantees. In such networks, judiciously sizing link capacities and routing metrics may do more to improve user perception of service quality than the introduction of complex mechanisms such as Weighted Fair Queueing, RSVP, and differential pricing. Even if such schemes are introduced and become popular, it is still necessary to administer policies and manage network performance by tuning parameters such as the degree of aggregation and the weights associated with each service class. None of these is possible with existing network management tools.

Network performance management has not been the focus of much research, which is reflected in the poor quality of most network management tools. In the absence of good tools, scientific process and good engineering practice gives way to trial-and-error and a 'gut-feeling' approach to network management. For instance, network operators plan the layout and configuration of their network in an ad hoc manner. However, more often than not, naively adding capacity to a network can actually lead to a decrease in the performance of the network. Similarly, at present, there is no way to test a network component without actually plugging it into the network. As a case in point, the only way to completely test the BGP configuration for a router is to plug it into the Internet. However, a wrongly configured backbone router can accidentally partition the Internet, a fact that is hard to discover, and harder to correct! Clearly, we need better tools to plan network topology, configure components, and to test a component without compromising the integrity of the network.

My group's research goal is to create a suite of tools to manage the performance of enterprise and wide area networks by exploiting innovative technologies for network simulation, web-based file systems, policy servers, router configuration protocols, and browser-based GUIs. We will develop algorithms for discovering the existing network topology, monitoring performance, identifying and removing hotspots, and consistently configuring different components of the network. *We believe these algorithms and technologies form the basis for the next generation of network management systems, that not only observe network functioning, but also actively participate in diagnosing and managing network performance, and exploit Web-based technologies for visualization.*

Our approach to network management is based on the observation there are five stages in the evolution and management of networks: (a) discovering existing topology, (b) collection of statistics, (c) identifying performance problems such as hot spots, (d) algorithms for dealing with performance problems, and (e) configuration of new hardware. We plan to develop algorithms and tools for each of the five stages of the network management process, as outlined next. Our work so far has concentrated mostly on stage (a), which therefore is discussed in most detail. (We note in passing that current network management solutions deal only with stages (a) and (b).)

---

[1] I must add that my past work in the areas of Fair Queueing, Hierarchical Round Robin scheduling, Xunet II wide-area Gigabit ATM testbed design and implementation, Renegotiated Constant Bit Rate service, and end-to-end native-mode ATM protocols has also been limited to this simplified model of the network.

Discovering topology: Network topology refers not only to the physical layout of components and cables, but also the logical partitioning of the network into administrative domains, routing areas, and other virtual views. We have come up with three novel technologies to *discover*, *store*, and *display* multiple views of network topologies. First, we automatically discover a network's topology by using a multi-pass approach, exploiting tools such as 'ping', 'traceroute', and 'nslookup', together with SNMP 'get' requests on routers. Second, we store the topology of the network in a innovative structure we dub the 'Web File System' or WebFS. WebFS treats an HTTP page as a directory, and a hyperlink as a file in this directory. Since we can associate arbitrary semantics with a hyperlink, and also create HTML pages on-the-fly, WebFS allows us to cleanly store multiple dynamic views of a network, and makes these views accessible anywhere in the Internet. Third, the display of network topology is within a standard browser, exploiting Dynamic HTML, Javascript and Java. We have created a scripting language that extends Javascript to allow Javascript functions to register with the Java AWT. This allows us to create an entire windowing system within a browser, where each window represents a network view. Thus, our work, which already exists as a prototype, allows us to automatically discover a topology, represent it in WebFS, and visualize it in a browser using Javascript and Dynamic HTML.

Collecting statistics: Traditional network management tools use SNMP to collect statistics from managed objects. SNMP requires the manager to poll the managed object to obtain information about dynamically changing MIB variables. The possibility of a large delay between the manager and the managed object precludes the implementation of fine-grained polling and network control algorithms. Moreover, during periods of congestion SNMP requests may be lost. Therefore, in addition to using SNMP to collect network statistics, we have developed an extension to SNMP called 'Active SNMP' that solves the above problems. Active SNMP allows a Java applet to execute on a Java runtime 'near' the managed object, carrying out computation on MIB variables on behalf of the manager. For instance, the applet could monitor the number of IP packet losses and use this to adjust the advertised link weight. By placing processing near the MIB, Active SNMP allows us to perform fine-grained control, and avoid the 'horizon effect' where event information in a remote domain is invisible because of aggregation.

Identifying performance problems: Performance problems in networks can be categorized very roughly into Level-2 and Level-3 problems. Level-2 problems can be detected by closely monitoring collisions at routers and hubs, which we plan to do using the Active SNMP framework outlined above. We will come up with algorithms to map from collision statistics to probable causes of frequent collisions. At Level-3, we can not only passively detect problems such as loss of connectivity, but can *actively* probe the network in two ways. The first approach involves proactively testing paths in the network for routing loops and black holes. This can be done with IP loose source routing. The second approach, which is more sophisticated, draws on our past work in network simulation (the REAL network simulator). Our goal is to extend REAL so that a simulated network appears to be equivalent in every way to actual network. Thus, simulated nodes can be `ping`ed, and will also support `ifconfig` scripts. With this level of simulation, we can create simulated test networks that generate routing updates to test the behavior of the actual network. This is an extremely powerful technique to test networks, and we hope to exploit it for proactive network management. For instance, we can create a virtual node running RSVP that talks to other RSVP-enabled routers in the network. By creating spurious RSVP requests, we can test whether the actual routers correctly obey RSVP, and whether the resource partitioning in these nodes is adequate. We will, in addition, use 'expert ensemble' techniques from AI to identify performance problems in a network. In this approach, an ensemble of experts diagnose performance problems using statistics collected as described above. In a training phase, experts that correctly diagnose problems are given more weight. Iterating over several training examples, we hope to create a finely tuned algorithm to automatically diagnose network problems.

Correcting network performance problems: We again plan to exploit the REAL network simulator for this part of the project. Our approach is to represent network topology and protocols within the simulator, including the details of routing protocols, and use this to recreate observed performance problems. Then, network administrators can test out potential solutions (such as adding new hardware, or changing routing parameters) before implementing them in the network. To successfully carry out this phase of the work, we need to simulate packet transmission and routing for networks with thousands of nodes and tens of thousands of simultaneous sessions. This is impossible with the existing packet-level simulation approaches. Thus, we have come up with a novel technique for *session-level simulation* that will enable us

to meet this goal. In this approach, we create a map from the number of simultaneous TCP or UDP sessions to the expected performance of each such session from empirical observations. The map allows us to simultaneously compute the traffic flows traversing each router (from the routing simulation) and the performance of each flow (from the empirical map). We hope to test this technique and its validity in identifying and solving network performance problems over the course of the project.

Configuring new hardware: One outcome of the previous step may be the need to procure new hardware such as routers and hubs. These network components are not easy to configure. For instance, current routers require users to specify numerous interdependent configuration variables. Mistakes in router configuration are the most common reason for network outages. We plan to solve this problem in two ways. First, we plan to simulate the behavior of a network component before it is plugged in, thus identifying misconfigurations and potential performance problems before they occur. Second, we plan to build and test policy servers (such as those using PEPCI) to centralize the administration of network components. By integrating policy management and simulation, we believe we will have a powerful tool for network performance management. We plan to test this approach in future work.

Related Work: It may be worthwhile to mention that our work is not being carried out in isolation. We are aware of the efforts of others in this area (including that of Abel Weinrib and Raj Yavatkar at Intel). Here are the efforts that we expect will be synergistic with our work:

- IETF Topology Discovery Working Group (aids our work in discovering network topology)
- Microsoft + Cisco Active Directory Initiative (can be made part of WebFS with an LDAP gateway)
- Torrent/Intel Dynamic Router Configuration Protocol (for router configuration)
- Intel's PEPCI (this work allows creation of generic policy servers)
- ISI's Routing Arbiter Project (allows simulation and consistency checking of BGP updates)

Acknowledgment This document owes much to discussions with Rosen Sharma over a period of many months.