Achieving Quality of Service through Network Performance Management

S. Keshav and R. Sharma
Cornell Network Research Group
Department of Computer Science
Cornell University, Ithaca, NY 14853
{skeshav, sharma}@cs.cornell.edu
http://www.cs.cornell.edu/cnrg

Abstract

We believe that current mechanisms for providing quality of service guarantees in integrated services networks have two major flaws. First, these mechanisms, such as signaling, scheduling, and reservations, require significant changes to the existing Internet. Second, they do not provide mechanisms for management of QoS state. In this paper, we present an alternative approach which attempts to provide 'macroscopic' QoS guarantees through effective network management.

1. Introduction

The Internet is exponentially increasing in *breadth* and *depth*. By an increase in *breadth*, we mean that the Internet is reaching ever more users, doubling its user population every year. This increase is likely to continue as developing countries get connected and small-footprint TCP/IP implementations bring embedded computers online. There is a parallel increase in *depth*, that is, in the set of services available to an endpoint. For many years, this set was small, and was provided by applications such as FTP, telnet, Usenet, and email. In recent years, the set has rapidly increased and now includes the World Wide Web, financial services like stock trading, distance education through audio and video multicast, and access to numerous databases.

Because of its rapid increase in breadth and depth, the Internet is widely accepted as an essential part of the global communication infrastructure. How well does it fit this role? A study of Internet performance reveals that the user experience is highly variable and often poor. While there are many millions of satisfied users, it is not hard to find angry customers unable to complete a stock trade, confused students whose video feed has been mysteriously cut off, or users who just cannot connect to popular websites. In view of the social and economic cost of these problems, there has been a great effort, over the last decade, to build networks that can guarantee some form of quality of service. (The solution of this general problem automatically solves the more restricted problem of supporting continuous-media applications in an integrated services network.)

Providing users with a consistent and reliable quality of service has usually been studied from the perspectives of scheduling, traffic profiling, signaling, and reservations. In practice, this may prove to be too narrow a view of the network. *We believe that the service quality experienced by a typical Internet user today is determined more by the quality of network management than by scheduling and reservation mechanisms*. This is particularly true of users on legacy networks that rarely support per-user or per-application quality of service guarantees. In such networks, network performance management tools that help in judiciously selecting link capacities and routing metrics may do more to improve a user's service quality than the introduction of complex mechanisms such as Weighted Fair Queueing, RSVP, and differential pricing. Moreover, even if more sophisticated mechanisms are introduced and become popular, we would still need tools to administer policies and manage network performance by tuning parameters such as the degree of aggregation and the weights associated with each service class.

Unfortunately, existing network management systems do a poor job of managing current networks, let alone the complex mechanisms that may be introduced in future networks. They are usually 'GUI-centric', painting pretty pictures of network status, rather than helping to find and fix network performance problems (for example, see HP's NetMetrix [1]). In the absence of good tools, scientific process and good engineering practice gives way to trial-and-error and a 'gut-feeling' approach to performance management. For instance, network operators plan the layout and configuration of their network in an *ad hoc* manner. However, naively adding capacity to a network can actually lead to a decrease in the performance of the network! Similarly, at present, one cannot test a network component without actually plugging it into the network--the only way to completely test a router's BGP configuration is to plug it into the Internet. However, a wrongly configured backbone router can accidentally partition the Internet, a fact that may be easy to discover but hard 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.

Our research goal, therefore, is to create a suite of network performance management tools that allow network administrators to provide quality of service to users. We do so by developing algorithms and tools for network discovery and monitoring, visualization, fast simulation, capacity planning, router configuration, and fault diagnosis. We believe these form the basis for the next generation of network performance management systems, which not only observe network functioning, but also actively participate in diagnosing and managing network performance. In the rest of this paper, we outline the current state of our tools.

2. The network lifecycle

Our approach to network management is based on the observation there are five stages in the evolution and management of networks (Figure 1).

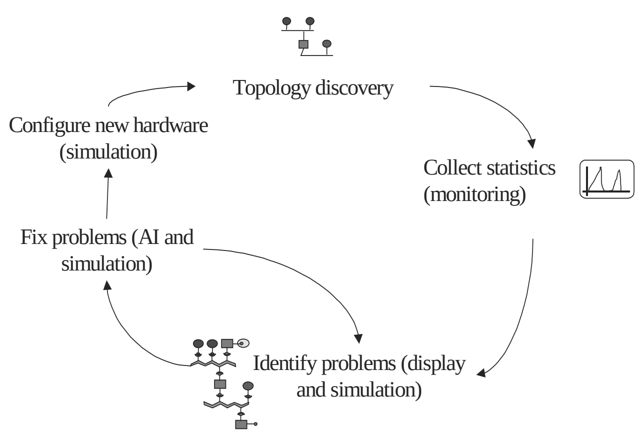

Figure 1. Network performance management lifecycle

These are (a) discovering existing topology, (b) collecting statistics, (c) identifying performance problems, (d) dealing with performance problems, and (e) configuring new hardware. We are developing algorithms and tools for each of these stages, as outlined next. We note in passing that current network management solutions deal only with stages (a) and (b). A significant exception is the Netsys suite of tools from Cisco [2], which offers tools for all five phases of the lifecycle. This product, however, is proprietary, and we are not aware of public descriptions either of its algorithms or its effectiveness. Moreover, it is targeted primarily at dealing with L-2 problems in networks of Cisco routers: we hope to model and manage heterogeneous L-3 networks.

2.1. Discovering topology

Network topology refers not only to the physical layout of components and cables within a given domain, but also the logical partitioning of the domain into subnets. The basic idea in our algorithm is to mark discovered nodes (corresponding to IP addresses) as 'temporary' or 'permanent' (by analogy with Dijkstra's algorithm). Each temporary node, when accurately identified as a router or an end-system, is added to the permanent set. Neighbors of the newly added permanent node are then added to the temporary set for further processing. Thus, network discovery problem reduces to coming up with algorithms for discovering temporary nodes and for identifying permanent nodes. Some of the heuristics we use for generating temporary nodes are to (a) do a DNS domain transfer, which lists every name and IP address in the domain, (b) `ping` a random IP address in the domain, adding it to the temporary set if it responds (c) `traceroute` a random IP address in the domain and add every node along the discovered path, and (d) use SNMP to get a router's routing table, then extract all the nodes in the table. Next we cast out spurious addresses, then discover routers, links, and link speeds. We discover a router either using SNMP (an SNMP GET returns the identity) or DNS (a lookup returns multiple IP addresses). We discover links by SNMP queries on routers or by carefully chosen `traceroutes`. Finally, we discover link speeds using Jacobson's `pathchar` program [3]. The discovered topology information is stored in a hierarchical file system. The attributes of each node are stored as files within a directory devoted to the node. This simplifies access to the topology. For instance, to ping all the hosts in subnet 128.84.154 we simply type `cd 128.84.154; foreach i(*); ping $i; end;`

Although our discovery tool still needs tuning, it can discover all the 10,000 hosts and routers in the `cornell.edu` domain, and over a million nodes in the Internet at large. We are currently experimenting with a new discovery algorithm that correlates `traceroutes` to the same randomly chosen IP address from different sites in the Internet to discover Internet backbone topology. For instance, a `traceroute` to `ncst.ernet.in` from hosts in Stanford and Berkeley uniquely identifies the ISP connecting this host to the Internet.

2.2. 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. However, if there is a large delay between the manager and the managed object, we cannot do fine-grained polling or implement efficient network control algorithms. Moreover, during periods of congestion, precisely when control is important, 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 these problems [4]. 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.

While Active SNMP solves some problems with monitoring, the question of deciding *which* nodes to monitor is still open. To solve this, we fit the traces collected from a particular site to an empirically determined mean. If a coarse-grained trace reveals that parameters of a fitted curve deviate substantially from normal, we automatically install an Active SNMP applet to more carefully monitor the situation.

2.3. Topology display

Discovery and monitoring create data sets that need to be efficiently and intuitively displayed. Instead of using a Visual Basic, Tcl, or Xlib GUI, we decided to use a standard Web browser as the user interface. We have created a scripting language that extends Javascript by wrapping methods in the Java AWT. Thus, window events, normally visible only to Java, are now exposed to Javascript. This allows us to rapidly create UI components in Javascript. Moreover, by writing a small windowing system in Dynamic HTML, we can represent multiple views of the topology, as well as current monitoring data within a browser. Our approach allows network management to be carried out from practically anywhere, and to display network performance information to interested users without a costly investment in a proprietary GUI.

2.4 Identifying performance problems

Performance problems in networks can be categorized roughly into Level-2 and Level-3 problems. Level-2 problems can be detected by monitoring collisions at routers and hubs, which we can do using the Active SNMP framework outlined above. (We also intend to 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 using SNMP, 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 with IP loose source routing. The second approach, which is more sophisticated, draws on network simulation techniques. We have designed and implemented an object-oriented network simulator called SurREAL that allows us to run multiple instances of the 4.4 BSD kernel networking code within a single Solaris or Windows NT address space. Moreover, existing applications can be ported to this environment with nearly no change. Our careful design allows the simulated network to appear to be equivalent to an actual network in every way. Thus, simulated nodes can be `ping`ed, and also support `ifconfig` scripts. This is an extremely powerful technique to test networks. For instance, we can create simulated test networks that generate and respond to routing updates to test the behavior of the actual network. As another example, we can simulate link and router failures to test its effect on the rest of the network. We can also deploy and test algorithms for differential service, resource reservation, and routing in a tightly controlled environment.

2.5 Correcting network performance problems

We plan to exploit SurREAL to correct network performance problems. Our approach is to use our discovery and monitoring tools to accurately model network topology, protocols, and workloads within a simulator, then 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 may 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 (including SurREAL). Thus, we plan to come up with techniques for *session-level simulation* that will enable us to meet this goal. In this approach, we plan to 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 will therefore allow us to simultaneously compute the traffic flows traversing each router (from the routing simulation) and the performance of each flow (from the empirical map). While the details of this approach and its effectiveness still are under study, we believe that this will allow us to capture network performance at least to first order.

2.6 Configuring new hardware

One outcome of the previous steps 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 hook up a network component to SurREAL and put it through its paces before it is plugged in, thus identifying misconfiguration and potential performance problems before they occur. Second, we plan to build and test policy servers to centralize the administration of network components. This integration of policy management and simulation is a powerful tool for network performance management.

3. Conclusions

We believe that as the Internet becomes more and more important in daily life, there is a serious need not only to provide QoS guarantees in terms of delay and bandwidth for audio and video traffic, but also in terms of connectivity, robustness, and good performance for traditional best-effort traffic. We believe that current approaches to providing service quality are too narrow to achieve this goal. In contrast, our pragmatic approach combines tools for discovery, monitoring, display, and simulation that allow network administrators to build and maintain high-quality networks. While this paper describes work that is still in progress, our initial results are encouraging. Our experience leads us to believe that, in the long run, our approach may be only one actually able to deliver QoS to users.

4. References

[1] HP NetMetrix, http://www.tmo.hp.com/tmo/ntd/products/products.html

[2] Cisco Netsys, http://www.cisco.com/warp/public/734/toolkit/index.html

[2] V. Jacobson, “Pathchar” binary, ftp://ftp.ee.lbl.gov/pathchar/

[3] R. Sharma, S. Keshav, M. Wu, and L. Wu, “Environments for Active Networks,” *Proc. NOSSDAV ’97*, May 1997.