Network Aware Distributed Applications*

Deborah Agarwal, Brian L. Tierney, Dan Gunter, Jason Lee, and William Johnston
Contact Person: Deborah Agarwal, e-mail - DAAgarwal@lbl.gov,
phone - (510)486-7078, fax - (510) 486-6363
Distributed Systems Department
Lawrence Berkeley National Laboratory
1 Cyclotron Rd, MS50B-2239
Berkeley, CA 94720

Abstract

Most distributed applications today manage to utilize only a small percentage of the needed and available network bandwidth. Often application developers are not aware of the potential bandwidth of the network, and therefore do not know what to expect. Even when application developers are aware of the specifications of the machines and network links, they have few resources that can help determine why the expected performance was not achieved. What is needed is a ubiquitous and easy-to-use service that provides reliable, accurate, secure, and timely estimates of dynamic network properties. This service will help advise applications on how to make use of the network's increasing bandwidth and capabilities for traffic shaping and engineering. When fully implemented, this service will make building currently unrealizable levels of network awareness into distributed applications a relatively mundane task. For example, a remote data visualization application could choose between sending a wireframe, a pre-rendered image, or a 3-D representation, based on forecasts of CPU availability and power, compression options, and available bandwidth. The same service will provide on-demand performance information so that applications can compare predicted with actual results, and allow detailed queries about the end-to-end path for application and network tuning and debugging

Introduction

Our vision is that in 5 years a distributed application, for example an application to visualize remotely stored data, would operate as follows. At startup, the application would ask "the network" what bandwidth to expect for data from the remote data source. In this case "the network" would include a collection of monitoring services, but this would be completely transparent to the application, much the same as DNS appears to applications today. Currently, if an application needs to do a hostname to IP number translation, it simply does a system call which returns the answer; the application is unaware of the complex hierarchy of distributed name servers involved in getting the correct answer. The network bandwidth estimation service should be just as transparent, reliable, and easy to use.

Along with the bandwidth estimate, the service would also give advice on the best way to achieve this bandwidth. For example which protocol to use, what buffer size, which QoS level, and so on. A remote visualization application could then modify it's behavior to present the user with the best visualization based on this bandwidth estimate. The application could also request to be informed if this estimate changes for some reason, for example due to unexpected congestion at some point in the path. Additionally, if the application notices that it is not getting the estimated bandwidth, it could automatically request that diagnostic tools be run, which would look for problems such as misconfigured routers, nonconformant TCP, and so on.

The members of the Distributed Systems Department LBNL have a great deal of experience building and tuning data intensive wide-area distributed applications. At the Supercomputing 2000 conference we received the "Fastest and Fattest" Network Challenge award for the application that best utilized the wide-area network. The application was a remote data visualization application called Visapult [17], which used a peak rate of 1.5 Gbits/second, and a sustained data rate of 680Mbps. This performance was the result of a great deal of hand tuning of the SC2000 network. Our envisioned future network would eliminate the need for hand tuning such applications.

*. This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Background

Application developers currently have very few tools to aid in developing applications that effectively utilize the network. They also have few tools to identify and diagnose problems with the network. For a distributed application to fully utilize the network, the application must first know the current network properties. This knowledge is particularly critical in the case of high-speed wide-area networks. Knowledge of current and maximum bandwidth, and of current and minimum latency, make it possible for the application to adapt to the network conditions by, for example, using the optimal TCP buffer size and the optimal number of parallel streams. The recently funded Web100[7] project has the potential to produce TCP protocol implementations that automatically tune themselves to provide optimal performance.

Because the network is dynamic, its properties need to be continuously tracked so that the application can be informed of the changes in network characteristics during execution. Currently, information about the network is very difficult to obtain and must generally be manually collected on a per-usage basis. A reliable, ubiquitous service, similar to DNS, needs to be available to allow applications to query for network properties. Although a continuous monitoring service can be deployed in primary portions of the network, there will inevitably be applications that access sites located outside this area. For these sites, an on-demand measurement capability is needed.

Sensors themselves are evolving. "Active" sensors, such as iperf or pipechar, which measure the network by injecting traffic into the network, provide a wide variety of parameters to more closely model the actual application. "Passive" sensors, such as tcpdump or counters in routers, which observe and sometimes record network traffic, can be programmed to interpret the network stream and trigger actions based on certain traffic patterns. "Application" sensors, such as NetLogger instrumentation, provide the mechanisms to obtain detail about actual distributed application timings. Coordinating these capabilities requires a monitoring framework that can activate and control the individual sensors while maintaining some notion of the current sensor activities system-wide. But, comprehensive network monitoring also introduces an opportunity for malicious network attacks, such as theft by gaining access to private traffic, denial-of-service by running massive amounts of active probes, or spoofing of sensor results to misdirect or redirect traffic. Therefore, the monitoring framework must be designed with security from the ground up.

A monitoring system can provide the framework to allow on-demand or permanent deployment of network measurement sensors with built-in security mechanisms to prevent unauthorized monitoring. In addition, tools for instrumenting the application data stream should be available and integrated into the monitoring system to allow achieved performance to be monitored by the application and recorded by the system. The system needs to incorporate a data forecasting service that can aggregate the real-time data from the sensors and historical data to produce a performance prediction. The monitoring system is an essential tool for application performance tuning and for diagnosing network problems. A critical component of the information service must be an ability to aggregate information from sensors using appropriate statistical methods to determine accuracy and confidence values so that the information known about the characteristics and capabilities of a particular network element increases over time.

There are several current efforts to design and deploy a network monitoring system which allows high-bandwidth and critical areas of the network to be continuously monitored and tested, enabling hop-by-hop analysis of the network paths. Existing monitoring tools such as the Network Weather Service (NWS)[5] are typically deployed and configured in a very static manner, and provide no security mechanisms. The Web100 project[7] is focused on solving one specific aspect of the larger problem: end-to-end TCP performance. The NIMI[8] project is developing a software system for development of network measurement infrastructure for dedicated sensors. There are also some excellent application tuning advise guides that are starting to appear[15].

Motivation

Currently, obtaining optimal TCP performance, even from an uncongested link, requires extensive hand-tuning. For example, if the TCP buffer is too small, it is impossible for the TCP window to fully open up to the size of the network pipe. Setting the buffer too large can lead to receiver overflows, reducing

performance up to 50%. Computing the optimal buffer size requires measuring bandwidth and delay between the client and server. Making these measurements currently requires having accounts on both the client and server hosts, and using tools such as *pchar*[2], *iperf*[1], and *ping* to measure bandwidth and delay. It is important to measure these quantities in both directions, as routes are often asymmetric. There are a few tools that allow the one-way measurement of these values using an account on either the client or server, but the results from these tools are not reliable. For example, *treno*[4] running on the Solaris operating system will never accurately report bandwidth on high-speed links, because *treno* relies on ICMP responses to simulate TCP, but the Solaris operating system limits the frequency of ICMP responses. Another example, the *pchar* tool, is very sensitive to fluctuations in traffic and often reports negative throughput on very high speed links. Coordinating this series of network tests and reconciling the results is a difficult and tedious process, even for experienced network engineers.

In addition, application tuning often runs into a brick wall when the end-to-end tools report results different from the theoretical path characteristics. In general, end-to-end tools are limited by the fact that they are unable to accurately determine the behavior of the network's interior nodes. Because existing tools for monitoring routers, such as SNMP, pose a serious security hazard and are difficult to set up and administer, personal contact with network administrators is currently necessary to obtain router statistics. Thus, for most developers, networks usually appear to be "black boxes" between the end hosts. This already leads to a very significant gap between available network bandwidth and utilized network bandwidth for most applications.

The need for application tuning is increasing. Future networks will be faster, less homogeneous, and will allow more traffic shaping and network engineering. Optical networks, which already have a high percentage of unused potential bandwidth ("dark fiber"), are rapidly approaching terabit speeds, and with fully optical switches petabit speeds will be possible. On the other end of the bandwidth spectrum, wireless networks are becoming more common, and applications need a way to detect they are running on a wireless link and be able to adapt to the higher loss rates and available bandwidth fluctuations that these networks often have.

Knowledge of the bandwidth and latency is an essential first step to allowing an application to efficiently use the network. Both the client and server applications must have access to mechanisms for using this information to, for example, set the socket buffer sizes on a per connection basis. On a congested network, the application may also need to use parallel data streams to achieve needed throughput, or take other application specific actions.

A configurable, on-demand and continuous network monitoring service available to applications to easily query for network characteristics is needed. An application should be able to query for characteristics from host A to host B and for the segments of the path between the hosts. This would allow the application to use that information for tuning and debugging. This includes setting the optimal TCP buffer, determining the need for parallel streams, requesting QoS, and identify network problems. This facility has the potential to significantly improve application throughput.

Key Issues

There are many issues involved in building and deploying a network monitoring service. These include:

- should the measurements be based on active or passive monitoring (or some combination of both)
- effect of active monitoring on the network and the applications on the network (appropriate uses of active monitoring)
- instrumenting the application data stream to achieve active application specific measurements
- accuracy of measurements and how to use them for prediction of future behavior
- techniques to measure and appropriately report and respond to asymmetric routes
- security to prevent unauthorized or inappropriate use of the monitoring system and its active monitors (e.g. for denial of service attacks)
- mechanisms to prevent unauthorized access to sensitive network performance and characteristics information
- deployment, maintenance, and administration of the monitoring service

- measurement of capabilities of multiple protocols, such as IP multicast and UDP
- correlation of network monitoring with host monitoring to determine if the bottleneck is the network or one of the end hosts
- efficient distributed storage and retrieval of monitoring data

In addition to the above, there are implementation issues such as how to make such a service reliable and high-performance. DNS seems like a good model for this, with the ability to cache results of queries locally, and to automatically fail over to alternate servers if the primary server is down. The system needs to be able to satisfy both general and specific queries and allow access to the raw measurement data as well as the synthesized results generated by predictive and aggregation mechanisms. Access to the queries and results need to be controlled using security mechanisms since it may not be desirable to have this service and information publicly accessible.

The issue of how to present the information obtained by the network monitoring service to the application so that it can understand and use the information to make decisions regarding communication behavior is largely unaddressed to date. There are several types of raw information that could be available from the monitoring service includes: expected bandwidth available, current achieved bandwidth, notification of changes in bandwidth availability, availability of alternate paths and notification of alternate classes of service. This raw information needs to be translated into information understandable to the application so that it can make intelligent decisions.

As applications themselves become more complex and widely distributed, the network information becomes more important. A distributed computation that has a choice of several possible compute nodes and several possible replicas of the data, and a limited amount of money to spend on the computation needs several questions answered. For instance, a question it needs answered is not "what TCP buffer do I use from point A to point B," but rather, given my budget what combination of compute node(s) and replica cache(s) can I use. Only once that question has been answered can it ask "what should the TCP buffers be.”

Another issue is providing dynamic accuracy estimates of the measurements. Accuracy is important to allow the intervening layers of analysis and aggregation tools to track the real variations in the data instead of the noise. Static estimates for a single type of network and tool can be hardcoded into the tool itself, but in a real network dynamic accuracy estimates will be more useful. This will require that the monitoring system, in a sense, monitor itself, by comparing for example application throughput and latency on a given path with periodic iperf and ping measurements. Performing this task in an automated way will require archiving, correlation, and analysis to be built into, or on top of, the monitoring framework.

Approach

A critical step in the development of the network monitoring service is a framework for the monitoring architecture. The framework needs to allow for the deployment of autonomous network monitoring boxes that allow tests to be triggered from remote locations without compromising security of the network. The basic idea of the framework is to provide the secure monitoring request mechanisms, loading of new sensors, and a well defined architecture for archiving and for returning to the requestor monitoring data.

Several tools currently under development, such as Network Weather Service (NWS)[5], NIMI[8][9], *pipechar*[16], and *iperf*[1], are a good starting point for a network monitoring infrastructure. These tools provide some of the functionality required by the network monitoring service. NWS is particularly appropriate in that it preforms predictions of future bandwidth, and includes the notions of accuracy of the measurements, uses active network sensors that balance their introduced load, and it includes a publication mechanism. However existing tools such as NWS and *iperf* have a number of shortcomings, including no facility for doing hop-by-hop analysis of the network path and provide no security mechanisms. In addition to the right tools, the right architecture for collecting and distributing the monitoring data is essential. A good starting point for the development of this architecture is the Grid Forum[11] Network Monitoring Architecture[10] which is currently under development.

A scalable framework for launching and accessing information from the monitoring tools is required. This framework must allow incorporation of sensors that are continuous running and sensors that are activated

on-demand in the same system. One significant concern that must be addressed in designing and building the framework is security. Active sensors that inject traffic into the net could potentially be misused by unauthorized users to congest the network and create denial-of-service attacks. Passive monitoring components that record traffic could be activated by unauthorized access to gain access to sensitive information or to create a denial of service attack on the monitor itself. The framework security mechanisms need to allow only authorized users to access the active and passive monitoring system components. Additionally, the system needs to implement intelligent policies that restrict the total amount of monitoring in critical subsystems, so that authorized users do not accidentally introduce large perturbations with a combination of separately tolerable monitoring requests.

Incorporation of security mechanisms in the framework allows the sensors to be incorporated into the system and activated dynamically. In order to scale the system and tailor the monitoring and testing activities to data that is currently needed, dynamic sensor activation mechanisms must be designed into the framework. Ideally the application needs to be able to activate an end-to-end test to monitor a specific traffic stream. One possible approach is to design a probe that can be sent out by the application requesting activation of monitors along the path of the application's own traffic. The framework can be designed to listen for these probes and launch the appropriate sensors to monitor the traffic. Some early work in creating a secure network test infrastructure called nettest[13] may be a good starting point for this work.

The activation of the sensors will be protected by the framework security. The resulting system will consist of both permanently installed network sensors and dynamically invoked sensors that are installed on demand. This architecture will allow continuous monitoring of critical links in the network as well as on-demand measurement of links outside of the critical area. This on-demand capability will allow applications to test all the way to end user sites regardless of whether the end user site is in a domain that is currently running network sensors.

Once a monitoring framework and architecture has been designed and implemented, a critical deployment step will be to work with network administrators from networks such as DOE's ESNet and Internet2's Abilene network to deploy a collection of monitoring hosts co-located with critical routers, allowing detailed monitoring and analysis of network segment.

The network measurement information that this monitoring system will provide can then be used with systems such as the one being built as part of the LBNL "ENABLE" project[18][12], to improve application throughput. For example, the ENABLE service will make suggestions for TCP settings, optimal number of parallel TCP sessions to use, and whether IP-level pacing should be used. The project is also currently pursuing some promising work looking at collecting data from layer-4 switches.

The applications themselves must be instrumented, so that one may compare the results of the monitoring system predicted throughput and latency with the throughput and latency actually obtained by the applications. The monitoring system should also provide sensors that can be used by the application to instrument its own data stream and contribute performance measurements while the application is running. We have been using NetLogger[6] for the application instrumentation, which has proven to be extremely useful tool for application tuning and debugging.

Conclusion

We have described here a network monitoring service which is capable of providing network-aware distributed applications with a means of maximizing their network utilization to achieve the best possible network throughput, as well as a means of determining when and how to use various future network services, such as QoS. Current network monitoring and analysis tools will not meet the requirements of tomorrows distributed applications. While there are many tools which are a good start, most are in their infancy and don't interoperate with each other. Our idea is to solve this problem with a network monitoring service that builds on existing tools and consolidates them into one, secure, reliable service.

Without such a monitoring infrastructure, it will be very difficult for data intensive distributed applications to fully utilize future high-speed networks. Application developers will never know what their application is capable of, while network engineers will continue to wonder why their networks are never fully utilized. This

problem will just become more and more exasperated as networks grow to be faster and have the capacity for larger amounts of bandwidth. A network monitoring service like the one we have suggested here will overcome many of these obstacles, while facilitating the search for future bottlenecks to network applications.

References

[1]iperf: http://dast.nlanr.net/Projects/Iperf/

[2]pchar: http://www.employees.org/~bmah/Software/pchar/

[3]M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, A. Essiari, "Certificate-based Access Control for Widely Distributed Resources" Proceedings of the Eighth Usenix Security Symposium, Aug. `99, http://www-itg.lbl.gov/Akenti/

[4]treno: http://www.psc.edu/networking/treno_info.html

[5]R. Wolski, N. Spring, J. Hayes, "The Network Weather Services: A Distributed Resource Performance Forecasting Service for Metacomputing," Future Generation Computing Systems, 1999. http://nsw.npaci.edu/

[6]B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, D. Gunter, "The NetLogger Methodology for High Performance Distributed Systems Performance Analysis", Proceeding of IEEE High Performance Distributed Computing conference, July 1998, LBNL-42611. http://www-didc.lbl.gov/NetLogger/

[7]web100 - http://www.web100.org/

[8]V. Paxson, J. Mahdavi, A. Adams, M. Mathis, "An Architecture for Large-Scale Internet Measurement," IEEE Communications, Vol. 36, No.8, pp 48-54, August 1998. (ftp://ftp.ee.lbl.gov/papers/nimi-ieee-comm98.ps.gz)

[9]V. Paxson, A. Adams, M. Mathis, "Experiences with NIMI," to appear in Proceedings of Passive & Active Measurement: PAM-2000.

[10]"A Grid monitoring Architecture," white paper, Grid Performance Working Group, Grid Forum (http://www-didc.lbl.gov/GridPerf/papers/GMA.pdf).

[11]The Grid Forum (http://www.gridforum.org/) is a consortium of institutions and individuals working on wide area computing and computational Grids.

[12]B. Tierney, B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson, "A Monitoring Sensor Management System for Grid Environments," HPDC-9, July 2000. Available at http://www-didc.lbl.gov/JAMM/

[13]Advanced Visualization Communication Toolkit, http://www-itg.lbl.gov/~deba/NGI/AdvVizComm.html, LBNL.

[14] Methods for Network Analysis and Troubleshooting, http://www-didc.lbl.gov/~jin/network/net-tools.html, LBNL.

[15]"TCP Tuning Guide for Distributed Application on Wide Area Networks," white paper, Data Intensive Distributed Computing Group, Lawrence Berkeley National Laboratory (http://www-didc.lbl.gov/tcp-wan.html).

[16]pipechar - http://www-didc.lbl.gov/~jin/network/net-tools.html

[17]W. Bethel, B. Tierney, J. Lee, D. Gunter, S. Lau, "Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization," Proceedings of the SC2000 Conference, November 2000, Dallas, TX.

[18]Enable: http://www-didc.lbl.gov/ENABLE/