LOW OVERHEAD CONTINUOUS MONITORING OF IP NETWORK PERFORMANCE

Mandis Beigi, Raymond Jennings, and Dinesh Verma
IBM Thomas J. Watson Research Center
30 Saw Mill River Road,
Hawthorne, NY 10532
e-mail: {mandis, raymondj, dverma} @ watson.ibm.com

KEYWORDS
monitoring, measurement, performance, low overhead

ABSTRACT

An Internet Service Provider or a corporate Intranet operator often needs to monitor the utilization and performance of its IP network. The performance metrics typically needed are the traffic, delay and loss-rate between two access-points. In many cases, the network provider needs to have the network operate within specific performance expectations, such as those specified within a Service Level Agreement (SLA) contract. A continuous measurement of the performance metrics is needed to ensure compliance with the SLA. However, traditional IP performance approaches typically include snap-shots of network performance at one single point, and do not offer much support for network performance between two customer access points. The main difficulty associated with the measurement is that neither of the access-points are the source or destination of an IP packet. In this paper, we present a scheme for network performance monitoring that can be used to monitor the IP performance problem between two access-points, and to verify that a specific SLA contract is being satisfied.

INTRODUCTION

As the demand for IP-based networks has grown, there has been a significant increase in the number of companies that provide a network to their customer. Such companies may be Internet Service Providers, or operators of private corporate Intranets. In either configuration, the provider network interfaces with multiple customer networks at specific access-points, and may interface with another ISP network at other access-points. Such a configuration is shown in Figure 1.

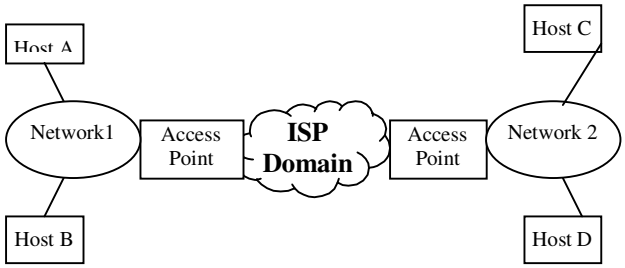

Figure 1: Network topology for performance monitoring

It is also usual for a network provider to have specific Service Level Agreements (SLAs) with its customers. It is more likely when the customer is a large enterprise. While several aspects of a SLA pertain to business practices, it also outlines the performance metrics that would be expected between two customer access-points. These metrics may include a bound on the traffic to be injected or delay between two specific customer access-points. The network operator needs to ensure that the network performance meets the criteria outlined in the SLA contact.

It is common practice in the industry to satisfy SLAs by provisioning the network. This approach works satisfactorily if the load on the network meets the assumptions made during provisioning. More often, the load on the network grows beyond the original assumptions, and it becomes uncertain if the SLA performance criteria are being met.

In order to verify that SLAs are being satisfied, the network operator needs to continuously monitor the performance of the network between a pair of customer access points. One way to do it would be to run a network performance monitor between all pairs of access points. However, most network performance measurement mechanisms tend to be relatively heavyweight. They rely on sending a continuous sequence of ping packets or other network probes between two end-points. The load generated by network measurement mechanisms is usually

quite prohibitive. A typical ISP or a corporate Intranet backbone has a very large number of customer access-points. It is not practical to have a continuous monitoring of network performance between all the end-points.

Another measure of interest to most network operators is the amount of traffic between a given pair of access-points. This measurement can be used to verify if a customer is generating traffic load in accordance with the SLA. Even in the absence of a formal SLA, it is useful to collect the traffic data in order to gain an understanding of network traffic patterns, and can be used in the future provisioning and upgrading of the network.

The traditional methods for measuring traffic have relied on counters and information collected at a single point in the network. Most IP routers collect a running count of packets and bytes transmitted through them, which can be accessed using SNMP (McCloghrie 1991) (Waldbusser 1997). An alternate option is to collect information about the different packet headers, and identify information about the different traffic flows in the network. This is the approach used in RTFM [Brownlee, Mills andRuth 1997).

A flow is usually identified by means of source and destination IP addresses contained in the captured packet. Since the ISP access-points typically tend to be intermediate points in a typical customer traffic flows, single point measurement approaches such as used in RMON (Waldbusser 1997) or RTFM (Brownlee, Mills and Ruth 1997) are not adequate in estimating the amount of traffic flowing between a pair of access-points. While some information can be obtained about traffic patterns if one makes assumptions about the routes used by specific traffic flows, it is difficult to determine the correct mapping between traffic flows and customer access points without keeping track of the routing tables in the network. If a customer network has multiple access points to the provider's network, (a fairly typical scenario), it is hard to predict which of the many access-points is used by packets belonging to a specific flow. Furthermore, no information about the performance (delay etc.) between two access-points can be obtained.

One way to collect the traffic information would be to determine the egress access-point for each packet at the ingress access-point, and to collect counters based on that information. However, determining the egress access-point requires that an additional set of routing tables be maintained within the IP forwarding path. This approach is undesirable both from the point of additional processing delays in packet forwarding, and from the space requirements of storing the additional routing information.

In this paper, we describe a method that can provide traffic accounting between different access-points of an ISP. We also discuss an extension to the method, which will permit an efficient pro-active monitoring of the network performance between two access-points. The method used for bandwidth accounting and performance monitoring provides a way to continuously monitor performance problems in the network with a very low overhead. This monitoring mechanism acts as a trigger, which detects potential violations of SLA contracts. The trigger, in turn, can be used to activate more precise heavyweight monitoring schemes like `netperf` (Netperf), which can accurately verify network performance between two specific access-points.

PERFORMANCE MONITORING

In order to monitor the performance of an IP network, probe packets are sent from ingress access-points to egress access-points. These probes are generated by copying the contents of the normal data packets, which follow the usual IP routing paths. The probe packets are identified and detected by the egress access-points. The probe packets are structured so that any packets that do happen to reach their destinations would be discarded. Within the context of this scheme, we can provide low overhead monitoring of traffic between two access-points as well as the network performance in terms of one-way or round-trip delays.

Traffic Monitoring

Our proposed traffic-monitoring scheme works as follows. Each ingress access-point is configured with a specific packet count number (say N). The ingress access-point keeps track of the packet counts it sends into the network. After every N packets it generates a new probe packet into the network. The probe packet has the same destination IP address as the Nth packet into the network, but the source IP address is that of the ingress access-router. The probe-packet contains special data that identifies it as a probe packet in the network. Additional data in the probe packet can be used for network monitoring.

At the egress router, the probe packets are identified and removed from the network data-stream. They are sent to a local process, which would analyze the information contained in the probe packet. For bandwidth accounting, the receiving process needs to simply increment the count of packets received from the ingress access-router. At periodic intervals, the accumulated number of packets received from each ingress access-routers is converted into

an estimate of the total number of packets being sent from that access-router.

In order to avoid synchronization with specific network traffic patterns, the exact probe generation will need to be varied with a small random distribution about the mean. While we have presented a mechanism which sends a probe packet on the basis of packet counts, simple variants of the scheme, for example sending probe packets after a specific number of bytes have been transmitted, or when a specific time-period has elapsed, can also be used with the same effect.

The overhead of the probe packets can be constrained to be only a limited percentage of the total network load by choosing the configuration packet count number. The choice of N=100, a typical use, would keep network overhead to 1 percent of data traffic. The probe packets can be structured so that a packet, which does not encounter an egress access-point, can be discarded at the receiver. Thus, one can deploy the bandwidth monitoring mechanism at only a subset of all the customer access-points in the network. When the traffic characteristics in the network change, the probe packets adjust according to them in a statistical manner.

While there are multiple ways to create the probe packets, we present a scheme that can be implemented relatively easily on most platforms. The probe packets are created with an IP header and a UDP header. However, the protocol field in the IP header is not marked with the UDP protocol-id, but a special reserved protocol number. The source port field in the UDP header is filled with a reserved pattern, and the destination port field is filled with a number that is unlikely to be used at any receiving end-host. The reserved protocol number would be used by the egress access-router to identify the probe packets to extract them from the stream. It then matches the reserved pattern in the source port field to double-check the probe packet, and replaces the protocol field with the UDP protocol number and the destination port with a local port number where a monitor process is active. This allows the egress access-router to handle the packet as a normal UDP packet, while providing a relative simple implementation of the receiver monitoring.

Performance Monitoring Protocol

In order to monitor the performance of the network, we use the UDP payload of the probe packet to carry information about network delays. The source access router includes the time-stamp of the creation in the probe packet. It can also contain an optional indication for the destination access router to reflect the packet back to the source.

When the packet is received at the egress access router, the local timestamp of packet reception is computed. The difference in time between the packet reception and transmission is the delay. This difference is rounded to zero if it happens to be negative. The time difference is smoothed out using a running average, and compared to a target delay stored in a configuration file, or in a directory database. If the smoothed delay exceeds the target delay, or approaches it, a trigger is generated for the egress and ingress access-points to activate the heavyweight performance monitoring protocol available on them. This scheme works well when the target delay amounts exceed the amount of clock skew that is expected among the two routers.

When clock skew is an issue, the ingress access router contains the option that probe packets be reflected back to it. The ingress access router then contains the smoothed round-trip delay and activates the heavyweight performance monitoring when it exceeds the target round-trip delays.

Note that the continuous monitoring provides a low overhead trigger that is activated when network performance problems are suspected. A more precise value of network performance is obtained by the heavyweight monitoring protocol, and an appropriate alarm generated for the network operator.

IMPLEMENTATION

We have implemented the above-mentioned performance monitoring protocol on a Windows NT platform. Our monitoring protocol is implemented as an NDIS intermediate device driver, which is placed between the Microsoft IP stack and the token ring device driver. The software was developed within the context of IBM Intermediate Device Driver (Dietrich). The NDIS intermediate device driver keeps a running count of the number of IP packets sent out on an interface. When the trigger is activated, a new IP packet is created by copying the token ring header and IP header of the current packet. The protocol field in the IP packet is changed, and a new payload is created. The source field in the IP packet header is changed to that of the local access-point. This ensures that ICMP messages from untrapped probe packets are received at the intermediate driver and can be prevented from reaching the IP stack. The payload consists of 4 bytes of magic number, the IP address of the access-point generating the probe, the current time at the access-point, a received time field which is created to be zero and an

option field. The option field indicates whether the packet delays and counters are to be computed at the egress access-point or if the packet needs to be reflected back to the ingress access-point.

The intermediate driver also examines all packets received up from the token ring. It checks to see if the packet has the reserved protocol field. If so, it verifies that the first four bytes in the payload matches the specific magic number used for network monitoring. If there is a match, the intermediate driver checks if the reflection option is specified. If so, a new IP packet is created and echoed back to the source access-point. The payload in the packet consists of the IP address of the current interface, and the option field indicates that the packet is a reflected one. If the packet is to be analyzed at the local access-point, the received time-field is filled by the intermediate driver.

While the measurements of bandwidth and delays can be done in the intermediate driver, we have chosen to use a simple hack to move most of the processing to the application layer in NT. The protocol field in the IP header is changed to that of UDP and the destination field is changed to that of the local interface. The magic number field corresponds to the source and destination ports in UDP. The destination port is changed to be one upon which a local application is listening. The local application maintains a table of all probe counters that have been received from different source access-points.

A control server in the network collects information from all the access points in the network. The control server is an NT application, which polls all the access-points at periodic intervals. It obtains the rate of probe generation and the total outgoing bandwidth (pkts/sec) from all access-points in the network. It also collects the probe rate, which is received at each of the access-points in the network from other access-points. Using this information, an estimate of the traffic between each pair of access-points is obtained. Similarly, a smoothed estimate of the average delay in each polling cycle between the pair of access-points is obtained. The control server compares the traffic rates thus obtained, and the value of delays against the targets stored in a directory, and flags a warning message whenever the observed metrics exceed the targets stored in the directory.

EXPERIMENTS AND RESULTS

In order to validate the implementation of the performance monitor, and to explore the characteristics of the protocol, we used the performance monitoring protocol to evaluate the characteristics of network on our campus Intranet. One site for measurement was placed in one of the labs in Hawthorne, New York, while the other site was placed in another lab located in Raleigh, North Carolina. The topology of the network between these two sites is shown in Figure 2. Two access routers (shown as AR1 and AR2 in the figure) were used for the purpose of the measurements. The control server was co-located with the access router located in Hawthorne. The directory server was located on a UNIX machine in the Hawthorne. The path between the access routers traversed several token rings, three T3 lines and one FDDI ring as shown in the figure.

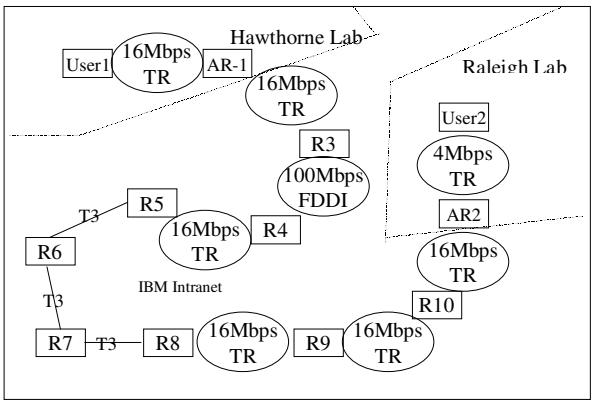

Figure 2: Network used for experimentation

We measured the characteristics of network monitoring protocol by generating known load using a packet generator, and monitoring how closely the traffic delays and bandwidth reported by the monitoring protocol corresponded to the known generated traffic. The packet generator sent fixed-size UDP packets at a configurable rate between the two users shown in Figure 2. A probe packet was created and sent out for every 50 outgoing packets, making the overhead to be 2 percent. Since the two access routers clocks were not synchronized, we used the reflection option in the probe packets to measure round-trip delays.

Figure 3 illustrates the round trip delays measured between the access routers in Hawthorne and Raleigh from 2:30 PM to 4:30 PM on a weekday. The measured delay is typically around a value of 250 ms with some occasional peaks. Figure 3 shows the results of measuring delays at one-minute intervals. The control server displays a rate of approximately 20 probes being sent in each direction during a one-minute interval as reported by the access routers. Since the UDP packet generators are transmitting approximately 16 packets/sec, we calculate the probe packet rate as reported by the control server to be correct.

Each delay is therefore an average of information obtained from approximately 20 probe packets.

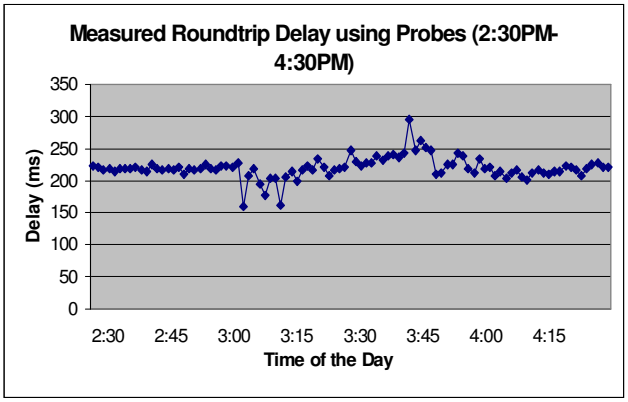

Figure 3

Figure 4 shows the measured roundtrip delay between the same access routers using `*ping*' averaged over one-minute intervals during the same time-period. Due to the change in the traffic load and therefore the traffic delays between the two sites during different times of the day, we started the ping and the bandwidth broker at the same time. The measured roundtrip delay as reported by ping is very similar to the ones reported by the probe packets. One would expect slightly different results to be reported by ping due to the different processing of the ICMP packets in the end routers. However, the measurements were done during a busy period of network operation, and the impact of ICMP processing was small in these studies. On a smaller network, or a less loaded network, the effect would have been more prominent. It can be concluded from the graphs that the probe packets show reasonably accurate results with good precision, which can be used by a network operator for monitoring a network.

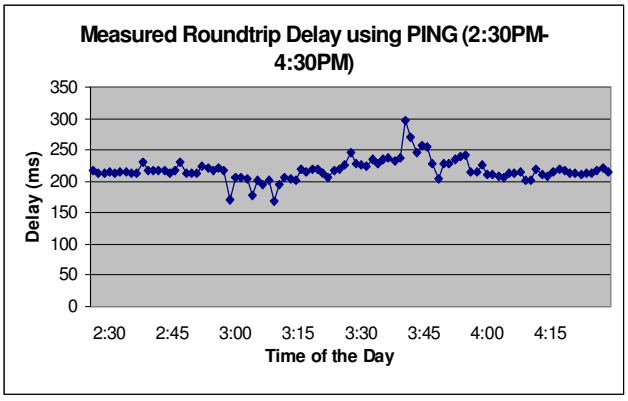

Figure 4

While the delay measurements done with our protocol are about the same as the ping measurements, there are some distinct limitations to using ping within bandwidth brokers. Ping provides no indication of the traffic distribution on the network, and is not able to accommodate for different class of traffic in a differentiated network. The network monitoring protocol can readily provide answers to both of these cases. Furthermore, continuous operation of ping across all access routers generates excess traffic in the network.

One of the requirements for efficient bandwidth brokering would be to get good timely estimates of the bandwidth utilization in the network. We wanted to see how quickly the bandwidth broker protocol would react to changes in the load between two access routers. We changed the load on the access routers function manner to see how quickly the bandwidth broker would detect the changed load. Figure 5 shows the results of this experiment. In the figure, the solid line shows the reference traffic load, and the dotted line represents the reported traffic estimate. The bandwidth broker estimations were close to actual measurements, and it reacted within one polling cycle to the changed load on the network. Thus, the calculation of the estimated traffic rate by the control server is relatively accurate and close to the actual value.

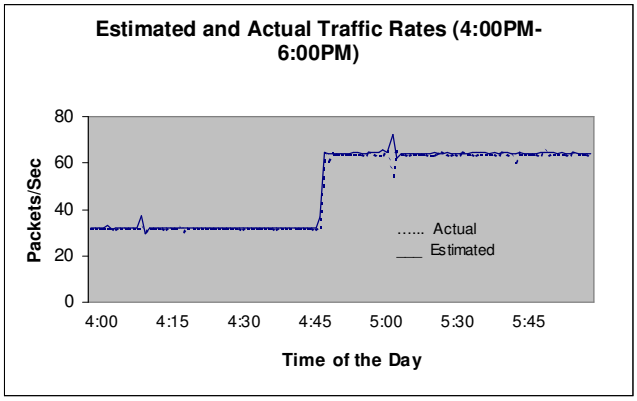

Figure 5

CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a scheme for low overhead continuous monitoring which can be used as a trigger to activate more heavyweight performance monitoring when performance problems are suspected. The scheme reduces the monitoring load on the network substantially, and assists in verifying service level agreements in an ISP network or corporate backbone.

The low-overhead bandwidth estimation protocol outlined above can also be used for detecting and isolating network failures in a TCP/IP network. The egress access-router contains a smoothed estimate of the number of packets to be received from each of the source access-routers. When the smoothed expected number is significantly more than the expected number of packets in the network, a heavyweight fault detection mechanism is activated (e.g. running ping between the two access routers). The heavyweight fault detection mechanism generates the appropriate alarm for the network operator if a fault indeed exists in the network.

We have presented some sample measurements done using the protocols over the IBM Intranet. To refine the measurement of the protocols, we are planning to conduct further studies involving more access routers. We are also exploring methods by which the scheme can yield precise loss information on specific tunnels. We have mainly looked at traditional unicast IP traffic. Estimation of multicast IP traffic requires some modifications to the existing protocols, and we are investigating appropriate means to incorporate multicast traffic into the monitoring scheme.

REFERENCES

Brownlee, N.; C. Mills and G. Ruth , *Traffic Flow Measurement Architecture*, Internet RFC 2063, January 1997

Dietrich, K., *NT NDIS Intermediate Device Driver Software*, Private Communicationn

McCloghrie, K. and M. Rose, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, Internet STD 17, RFC 1213, March 1991.

Netperf, home page:
http://www.netperf.org/netperf/NetperfPage.html

Waldbusser, S., *Remote Network Monitoring Management Information Base*, Internet RFC 2021, January 1997

Mandis Beigi received her Bachelors of Engineering in Electrical Engineering from the State University of New York at Stony Brook in 1993. She received her Masters of Science in the field of Electrical Engineering from Columbia University in 1995. She works at the IBM T.J. Watson Research Center and is also a Ph.D. student at Columbia University. Her research interests are quality of service, service differentiation and network monitoring.

Raymond B. Jennings III received a Bachelors of Science in Electrical Engineering from Western New England College in 1993 and Masters of Science in Computer Engineering from Manhattan College in 1996. He works for the IBM T. J. Watson Research Center and is currently a Ph.D. student at Polytechnic University.

Dinesh Verma received his B. Tech. in Computer Science from Indian Institute of Technology, Kanpur, India in 1987, and Ph.D. in computer networks in 1992 from the University of California, Berkeley. He is currently managing the Enterprise Networking Research Group at IBM T. J. Watson Research Center. His interests include Quality of Service (QoS) in networks, performance evaluation of networks, policy enabled networking, and applicability of TCP/IP in enterprise environments.