QUALITY OF SERVICE MONITORING IN IP NETWORKS BY USING TWO-WAY ACTIVE MEASUREMENTS

Tamás Varga[1] *András Oláh*[2]
[1]Budapest University of Technology and Economics, Telecommunications and Telematics Dpt.
H-1117 Budapest, Pázmány Péter sétány 1/D, Hungary E-mail: varga@ttt-atm.ttt.bme.hu
[2]ERICSSON Hungary, Traffic Analysis and Network Performance Laboratory
H-1300 Budapest, P.O.B. 107, Hungary E-mail: andras.olah@eth.ericsson.se

ABSTRACT

Quality of Service monitoring is an important function in near-future network management systems that operate an Internet Protocol (IP) based real-time communication infrastructure. Such networks carry thousands of quality-sensitive data flows and their performance is difficult to check against the contracted parameters at the network boundaries. Active measurements, which inject test traffic into the network, play an important role in the estimation of user perceived quality. In this paper we focus on delay and loss measurements and we present ideas on how to build a scalable distributed monitoring system. The issue of estimating the end-to-end delay and loss along a given path from link-by-link round-trip measurements is discussed first. Then a heuristic algorithm is presented which attempts to distribute the load of driving the round-trip measurements evenly across the routers in a domain.

1. INTRODUCTION

Offering Quality of Service (QoS) requires three main activities from network operators: (i) deploy an infrastructure which is able to provide premium service, (ii) administer and configure traffic handling parameters in network equipment according to service requirements, (iii) and continuously measure network performance to initiate traffic control mechanisms if necessary. At first, the Integrated Services and Differentiated Services models provide means to realize QoS mechanisms at the network layer [4]. The second area is covered by Bandwidth Brokering [8] and policy management, the latter is the topic of several working groups of the Internet Engineering Task Force [9]. The third area, which covers performance management functions (monitoring and controlling), is lightly studied yet as we see the literature.

Our paper deals with network performance monitoring and the primary focus is on delay and loss monitoring with active measurements. The main purpose of this activity is to check the current network performance against the target values. Before the operator starts providing a service to one of its customers, they record also quality issues in the *Service Level Agreement* (SLA), see [6] for details.

The challenge in passive delay monitoring is that we need keep track the arrival time and the departure time of individual packets at network edges, whose difference yields the packet delay. This model is very difficult to implement, because the entry and exit points should continuously communicate with each other to inform each other on packet arival time. Fortunately, active measurements, which inject some test traffic into the network, can help to approximate the delay between network edges. This approach is already used for Internet performance monitoring [1].

The problem with active measurements is that they generate traffic, which may disturb the properties of the carried traffic. If each network edge generates a measurement flow to the other edges at the same time, these flows will meet each other at buffers and may cause overflow. This problem can be solved by clever scheduling of these flows, but the measurements will not be continuous in time.

To avoid flow overlapping, we try to approach the problem from a different aspect. The idea is that each router measures all of its links and the overall delay is combined from these link-by-link delay measurements. We try to make use of round-trip measurements, because they are much easier to run

(they do not need clock synchronization from the network). In order to see whether this approach is usefull, we conducted a few experiments on a testbed. From these measurements, it seems that round-trip times in opposite direction have the same statistical properties. Therefore, one measurement is enough and the two routers on a link should agree, which of them will carry out the measurements. For this purpose, we developed a heuristic graph algorithm that optimize the measurement task among routers.
The rest of the paper is organized as follows. In Section 2, we discuss quality of service monitoring issues in details and we present the monitoring architecture. In Section 3, we show a simple analysis of our round-trip measurements. Section 4 deals with the graph optimization problem and Section 5 concludes the paper.

2. QUALITY OF SERVICE MONITORING

A customer makes use of network service either directly, when the target resource (e.g. a WEB server) resides within its home operator network, or indirectly, when it is located in another network. In both cases the operator is responsible for quality assurance within its network. When we investigate the overall performance between any two hosts, we refer to it as *end-to-end* performance. If we consider performance in a network between any two entry/exit points, we speak about *edge-to-edge* performance, see Figure 1.
The standardization of performance metrics and methods are published in ITU-T recommendation I.380 [3] and in several IETF documents, see [4] and RFC 2678-2681. The operator's challenge is to measure flow properties at the incoming edge and the outgoing edge for each contracted flow and to compare in real-time to the contracted parameters, see Figure 2. If we want to accomplish this task by passive methods, we should deploy packet capturing hardware on exchange links. We have to record packet arrival times for each flow and to compare the incoming series to the outgoing series. The difference between arrival and departure time of the packet yields the packet transfer delay, while difference of the sent and the received packet gives packet loss. This requires continuous feedback communication, which would generate significant amount of traffic.

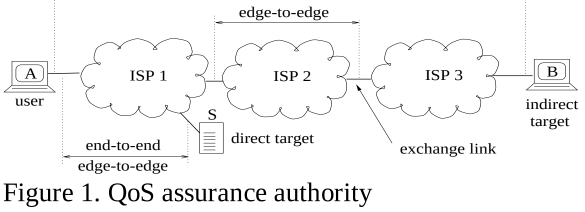

Figure 1. QoS assurance authority

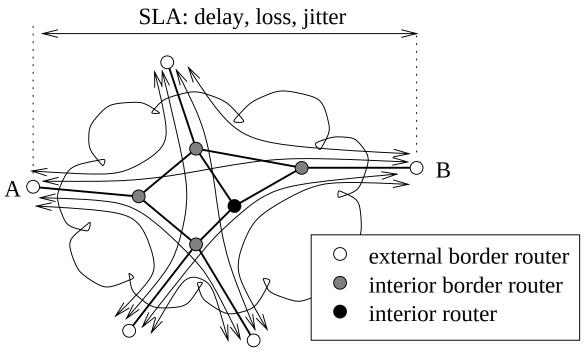

Figure 2. Verifying each edge-to-edge SLA

Instead of trying to measure each flow individually, we seek other ways to know something about edge-to-edge performance. We generate test traffic at one network edge, whose characteristics are known, and estimate the quality of the flows leaving at a remote edge by investigating the distortion of the test traffic. This approach fits well into the view of Differentiated Services model, where individual flows are not distinguished, only different classes of traffic. In this case each traffic class should be measured independently.
We may also exploit the fact, that most of communication is driven by humans and conversations are bidirectional. Even if a voice connection is implemented with two unidirectional flows, both directions must operate properly for a clear communication between two parties. For this reason we may use two-way measurements for delay monitoring rather than expensive one-way measurements. One-way measurements require proper clock synchronization from the network in order to measure packet transfer delay between any two points. For this purpose, the in-band Network Time Protocol (NTP) is unsuitable and expensive off-band hardware is required. The clock synchronization errors in NTP are in the same order as the network delays, because NTP synchronizes the computer clocks by communicating over the network. Therefore, the accuracy provided by NTP is not sufficient to measure one-way transit delays. Nowadays, the most popular external clock synchronization solutions are based on GPS receivers[1].
Before we plan any measurement we should clarify where the boundary of the network is. It is obvious that all the equipment, which have common management, belong to one network along with those

[1] The satellites of the Global Positioning System broadcast the Universal Coordinated Time (UTC) and geographically distant clocks can be synchronized to UTC in microseconds order.

links that connects any two of these equipment. But who is responsible for *exchange links*, i.e. links between peer networks? It belongs to either our or the peer's management authority, but the contribution of the exchange link cannot be neglected. Thus we should monitor exchange links from border routers.

Active measurements require bandwidth, the exact amount depends on packet size and on sampling frequency. It is obvious that we cannot measure performance information at any instant, thus a proper sampling rate has to be chosen. For example, if the network carries voice-over-IP flows with *20 ms* sending rate, we should set our sampling frequency to at least *50 Hz*. Thus the trade-off between measurement accuracy and overhead has to be considered Cottrell conducts large scale two-way measurements in the Internet based on the ping utility [2], and this architecture was also adopted by the Cross Industry Working Team [7]. A sequence of packets (actually 10 packets spaced at 1 second) is sent to remote hosts at regular intervals (30 minutes). This produces a very light measurement traffic, although they monitor hundreds of Internet paths from each monitoring site.. In order to monitor quality sensitive applications, such sampling frequency is inadequate.

As it was mentioned before, we assume that edge-to-edge performance of individual flows can be estimated from one active measurement for each pair of edges. Therefore, we have to run *N(N-1)* active measurements in a network with *N* border routers because forward and backward paths are distinguished from each other. It is obvious that often more than one measurement flow traverses a link, and the consumed bandwidth is proportional to the number of the measurement flows on each link.

To reduce this overhead, we propose the following idea: measure individual link performance parameters and try to estimate the total edge-to-edge performance from these measurements as indicated in Figure 3. A monitoring system using this approach works as follows. After network initialization or repairment, routers start to collect link performance information by using round-trip measurements. They derive basic statistics (e.g. minimum round trip delay, variance) and they automatically calculate delay thresholds that indicates high load on that link. When exceeding a threshold, a notification is sent to the performance monitoring center, where the affected edge-to-edge SLA(s) can be identified by incorporating routing information. Routing tables can be obtained from the routers in advance via the Simple Network Management Protocol (SNMP). From these tables,

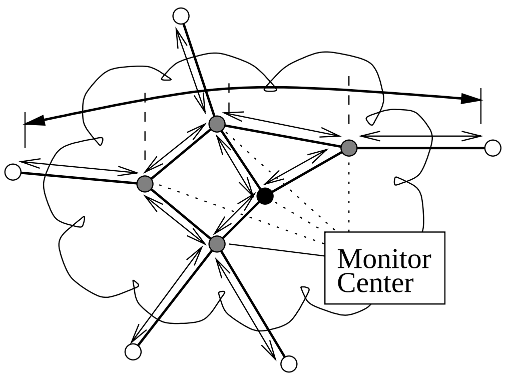

Figure 3. Measure links individually

edge-to-edge routes can be calculated. Routing changes are indicated by SNMP TRAP messages, thus routes can be recalculated then. It is then up to the performance managemenent system whether immediate actions are taken or direct, more accurate measurements of the affected SLAs are initiated.

The realization of such a monitoring system raises some issues that are investigated in more detail here. The first concern is how to estimate edge-to-edge performance from individual link measurements? In particular, we are interested in delay and loss as the main performance measures. Secondly, round-trip measurements on a particular link can be initiated from either ends, but in practice one wants to use only one measurement provided that the outcome of the other one can be deduced. These issues are investigated in Section 3.

Another question pops up if round-trip measurements are done from only one end of each link. The router initiating the measurement has considerable more processing overhead than the one which only reflects the measurement traffic. Therefore it is natural to ask how to distribute the burden of driving the measurements evenly among the routers. A heuristic algorithm is shown in Section 4 to solve this problem.

3. MEASUREMENTS

We wanted to check whether we are able to estimate the overall edge-to-edge performance from round-trip measurements of individual links. For this purpose, a simple test network was built. Four Linux based computers were connected together, the two in the middle (B and C) enable packet forwarding among their interfaces, thus machine A and D can reach each other, see Figure 4. Our measurements require full-duplex communication channels, because core networks are build upon these. At first time we wanted to use point-to-point 10Mbps Ethernet links, but we were unable to set the interface card to full-duplex mode. Finally, we implemented full-duplex

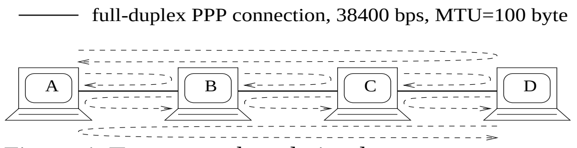

Figure 4. Test network and ping loops

PPP connections over serial lines, which provide 38400 bps. We reduced the Maximum Transfer Unit (MTU) down to 100 bytes, because four 1500 byte packets would increae queuing time over 1 second and it would rather disturb our measurements. Transmission of 100 bytes over 38400 bps link lasts nearly 15 ms, which enables outgoing buffer to grow up to 66 packets within one second.

Background test traffic were originated and terminated on machine A and D in both directions. The traffic mix were consisted of one bulk FTP data transfer, 2 telnet sessions to the chargen port[2] and multifile FTP data retrieval with short files. These sources generated a medium load and a varying traffic over the time. The slow transmission lines were the limiting factor and this simple traffic mix was empirically selected.

We were interested in, how round-trip times change between neighbouring computers and between the two edge. We used the `ping` program, which utilizes the ICMP_ECHO facility, since processing ICMP messages requires not too much resource. It takes 30..50 μs, which is 3 order less than the transmission time, thus can be neglected. Successive packets were sent at the default 1 second interval with payload size 64 bytes plus the 20 byte IP header. The measurements were running for 3 hours which produced approximately 11000 samples. Because two-way measurements do not require synchronized clocks from the network, we did not force to start the 8 ping sessions all at once. Instead, we synchonized the sequence numbers only after the measurements. In the output logfiles we were seeking that sequence number where the difference of the current and the previous delay sample exceeds 5% of the current value (we started sampling before traffic generation).

3.1. Round-trip delay analysis

The first thing we have investigated, how behaves that process, which was constructed from the sum of the three round-trip delays. This was motivated by a simulation result, where it was found that per hop delays seems to be independent from each other [10]. Let us denote the round-trip time of the nth packet with r_n, then we have

$$r_n^{ABCD} = r_n^{AB} + r_n^{BC} + r_n^{CD}$$

where *n* takes those values only where all three data exist (i.e. no packet loss occured). Then we derived the error process, which is the sample-by-sample difference of the edge-to-edge delay (r^{AD}) and the sum of delays (r^{ABCD}):

$$\varepsilon_n = r_n^{AD} - r_n^{ABCD}$$

In a similar way, we define the quantities for the opposite direction. Table 1. summarizes the statistics of the round-trip times between the machines. At first look, we can see, that the forward and the backward direction have a little bit different mean values. By focusing on the sum of the three average round-trip times (A-B, B-C and C-D), we find that the sum underestimates the average edge-to-edge round-trip time (A-D) by 20...30%. Upto this time we could not find the source of this difference. It is also interesting, that the standard deviation of the error series and the edge-to-edge series are almost the same. The estimation silently assumed, that we have normally distributed time series, but this is not the case. We can see the histograms of the round-trip times in the forward direction in Figure 5 (we get very similar distributions in the backward direction also), and these do not remember us to a bell-shaped curve, especially A-D, which have two humps. However, the error (ε^{ABCD} and ε^{DCBA}) seems to show normal distribution as the QQ-plot suggest us in Figure 6. The ordered error samples are plotted against the normal distribution and the closer the points fall to the line (determined by sample mean and deviation), the higher is the probability that we have normally distributed samples. It is also interesting that the round-trip time on the center link (B-C) shows discretized values spaced by 10ms apart from each other (bin width is 2.5ms). This can be due to that the queues never became empty and self synchronization occured among the TCP flows.

Link	Mean	St.dev	Link	Mean	St.dev
A-B	193.2	69.8	B-A	203.7	77.5
B-C	70.5	12.6	C-B	78.5	11.7
C-D	144.7	65.6	D-C	167.7	68.2
A-D	562.6	117.4	D-A	568.3	99.6
ABCD	408.5	98.0	DCBA	449.9	104.6
Error	154.0	110.3	Error	118.3	99.6

Table 1. Round-trip time statistics summary, base unit is 1 milisecond.

[2] The TCP chargen port sends MTU sized packets at the rate as acknowledgements are received.

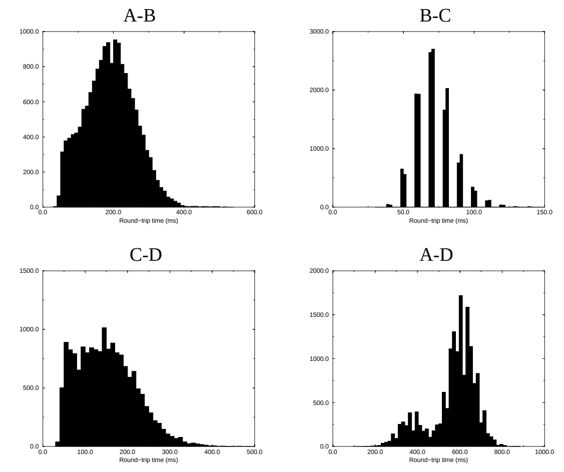

Figure 5. Histograms of round-trip times

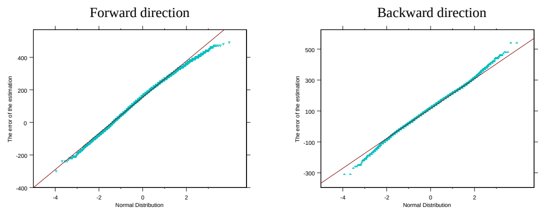

Figure 6. Quantile-quantile plot of the error samples against the normal distribution.

3.2. Round-trip time asymmetry

Next we focus on the round-trip time asymmetry, i.e. whether we can find any relation between the measurements done in opposite directions. If we consider the two machine in the middle (B and C), we can build a simple model drawn in Figure 7. The pinger module generates the packets at fixed intervals, which join the outgoing queue if it is not full. This queue is also fed by other sources that wish to use the same link and its cumulative arrival curve is denoted by *A(t).* A packet delays until all the packets are serviced in the queue including itself. In the peer node, the responder module mirrors the message and puts it into its outgoing queue. The response has to compete with the traffic flowing in the backward direction, which can be described with its arrival curve *D(t).* The other pinger-responder loop can be described in similar way. The only difference is that the packets running in opposite loops visit the same queue in different time.

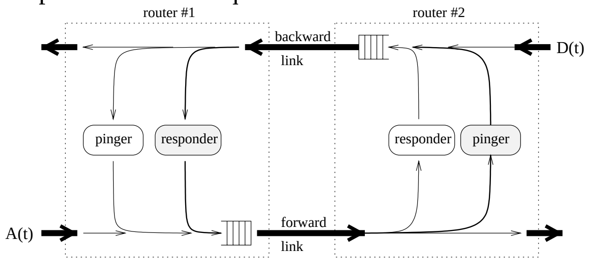

Figure 7. Round-trip loops

Instead of building an analytical model, we started to examine the statistical properties of the datasets. If we consider the round-trip time datasets of B-C and C-B and keep those entries which have a matching pair by the adjusted sequence number in the other dataset (since packet loss can occur), we derive two datasets X_n and Y_n respectively. We assumed that the normalized error

$$Z_n = \frac{X_n - Y_n}{X_n + Y_n}$$

can be approximated by normal distribution. Figure 8 shows the QQ-plot of the error of the three links and the edge-to-edge round-trip time. For the three links, the samples follow normal distribution approximately within the range *(-2σ,2σ)*, which covers the 95% of the distribution.

3.3. Loss analysis

Other interesting quantity, which we can easily extract from the round-trip measurements, is the packet loss ratio. It is simply calculated as

$$l = 1 - \frac{\textit{number of entries}}{\textit{endSEQ} - \textit{startSEQ} + 1}$$

where `startSEQ` and `endSEQ` is the first and last packet sequence number respectively.

We supposed that packet loss occurs indepently on each link, but loss over a path (sequence of links) depends on all previous links. Thus the edge-to-edge loss can be simply derived in a product-form out of the individual loss ratios.

$$l_{ABCD} = l_{AB} + (1 - l_{AB})l_{BC} + (1 - l_{AB})(1 - l_{BC})l_{CD} = \\ = 1 - (1 - l_{AB})(1 - l_{BC})(1 - l_{CD})$$

Table 2 shows the calculated loss statistics for each link. It is interesting that no packet loss occured on the center link (B-C) and side links (A-B and C-D)

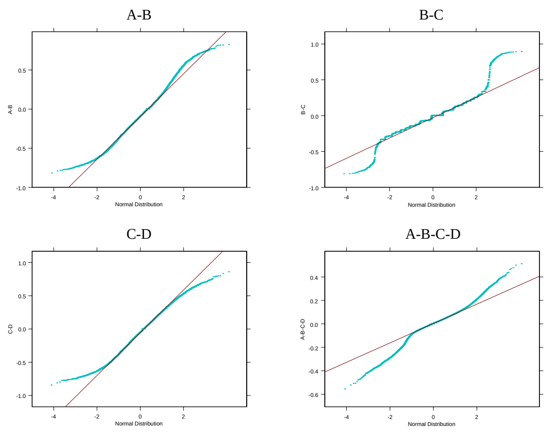

Figure 8. Quantile-quantile plot of the error against normal distribution

experiences rather asymmetic loss ratios. This can be due to the TCP congestion control meachnism that determine the background traffic. The conclusion is that the overall loss ratio (l_{AD}) is underestimated from the individual measurements (l_{ABCD}). This can be due to that we sample queue lengths in different time at regular intervals and TCP flows are synchronized.

Link	Loss	Link	Loss
A-B	0.28%	B-A	5.24%
B-C	0%	C-B	0%
C-D	1.2%	D-C	2.05%
A-D	6.12%	D-A	8.76%
ABCD	1.47%	DBCA	7.18%

Table 2. Loss statistics

4. CONFIGURATION OPTIMIZATION

As we have seen, it is statistically irrelevant that from which endpoint of a link do we initiate the round-trip measurement. If we model our network with an undirected graph, we can derive a directed graph from it, in which we mark the direction from pinger to responder. The question to answer here is how to set up the directions efficiently, because we have several constraints:

- The larger number of pingers running on a system the less accuracy we will achieve, since the accuracy of the timestamps are heavily dependent on system load.
- The load on border routers typically higher than on any internal nodes, because they perform e.g. inter-domain routing.
- We cannot initiate measurements from external border routers, because they do not belong into our authority.

This graph optimization problem can be easily drawn up: give a direction in the graph that tries to keep the difference of the number incoming and outgoing edges under a certain limit for each vertex. We are looking for a fast, convergent algorithm which can be implemented either in the management system or in routers itself as a protocol.

We developed and implemented a very simple heuristic algorithm to test our simulation environment. The idea behind the algorithm is that we eliminate the full constrained edges first, then we try to give a direction for the remaining edges in one step. Finally a greedy optimization is run until we cannot improve by changing the direction of unconstrained edges. Figure 9 shows a sample output.

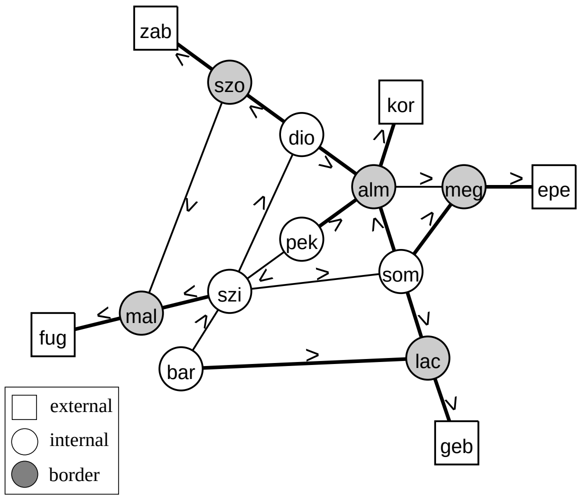

Figure 9. Example configuration, arrows show the direction from pinger to responder

The algorithm operates on an ordinary undirected graph extending vertices and edges by several attributes:

- Edge: **direction** can be either unset, forward and backward; while **lock** status means that the direction cannot be changed anymore.
- Vertex: beside the number of **incoming** and **outgoing** edges we maintain the number of connecting **unlocked** edges and **type**, which can be external, border or internal.

At initialization we unset **direction** and unlock all edges, while we zero all counters of vertices and record the number of edges in **unlocked**.

Step 1. Take those edges which have one external endpoint (if both are external, then this edge is out of our scope), set direction towards them and lock.

Step 2. Take those edges which have a border endpoint and if the peer endpoint is

- external then skip it
- internal then set direction against it and lock
- border then check whether the peer is in worse situation (i.e. the difference of the incoming and outgoing edges are bigger than current plus two) and set direction according to it

Step 3. For each unlocked edge, calculate the number of connecting edges to its endpoints (i.e. the sum of degree of the two vertices minus two) and sort edges in ascending order by this factor. Take each edge and set direction according to the test which endpoint is in a worse situation.

Step 4. While there is such an unlocked edge, which would improve the situation of its vertices, reverse its direction.

The algorithm works well with arbitrary network topologies. The first three steps give an almost optimal configuration for such network that have no cycle in it. The difference of the incoming and outgoing edges never falls below minus 2 if there are no constraints. The last refinement step is required to balance loops. An example can be seen in Figure 10: node A has two outgoing edges, while we can improve if we change the direction of edge A-D. The algorithm is stable, since we initiate direction change only if the difference of the incoming and outgoing edges between the two vertices is at least two and any direction modification cause change by only one in this value.

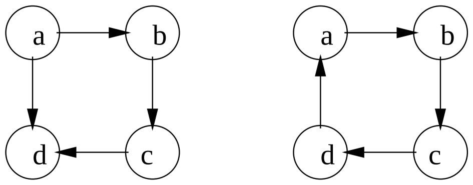

Figure 10. Loop balancing problem

5. CONCLUSION

Quality of Service monitoring requires efficient active measurement and estimation methods to keep track the user perceived performance. We have presented a monitoring architecture, which uses round-trip time measurements on each link to estimate end-to-end delays in the network. Using these estimates, the frequency of more accurate end-to-end delay measurements can be reduced. We have also presented a heuristic graph optimization algorithm which distributes overhead of the measurement task evenly among routers. The ideas discussed in the paper represent work in progress. Since the advantages of a scaleable, distributed monitoring system are promising, further work is aimed at analysing the presented results more rigorously and at refining the methods.

ACKNOWLEDGEMENTS

The authors would like to thank András Veres, János Harmatos, István Maricza and Tamás Éltetö for their valuable comments.

REFERENCES

[1] Les Cottrell, Comparison of some Internet Active End-to-end Performance Measurement project, http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, 1999

[2] L. Cottrell, "Tutorial on Internet Monitoring", Stanford Linear Accelrator Center, http://www.slac.stanford.edu/comp/net/

[3] ITU-T-Recommendation I.380, "Internet protocol data communication service - IP packet transfer and availability performance parameters"

[4] Quality of Service Forum, "QoS protocols & architectures", http://www.qosforum.com/

[5] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics" - RFC 2330, May 1998

[6] Nathan J. Muller, "Managing Service Level Agreements", International Journal of Network Management, Vol. 9, pp. 155-166, 1999

[7] Cross-Industry Working Team, "Customer View of Internet Service Performance: Measurement Methodology and Metrics", September 1998

[8] B. Teitelbaum et al, "Internet2 QBone: Building a Testbed for Differentiated Services", IEEE Network, pp. 8-16, September/October 1999

[9] IETF Working Groups, Operations and Management Area, rap, policy and snmpconf working groups, http://www.ietf.org/html.charters/wg-dir.html

[10] Yates, Kurose, Towsley, Hluchyj, "On per-session end-to-end delay distributions and the call admission problem for real-time applications with QoS requirements", ACM SIGCOMM'93, 1993