A QOS SCHEME TO ADDRESS COMMUNICATION LATENCY ISSUES FOR CRITICAL NETWORK FLOWS IN BEST-EFFORT NETWORKS USING MOBILE AGENTS

Visvasuresh Victor * Govindaswamy
University of Texas at Arlington, Texas, USA.
victor@uta.edu

Gergely Zaruba
University of Texas at Arlington, Texas, USA.
zaruba@cse.uta.edu

G. Balasekaran
Nanyang Technological University, National Institute of Education, Singapore.
gbalas@nie.edu.sg

Abstract

Flows can be mission critical; missed deadlines could result in grave consequences in systems such as hospital, military, automotive safety, and air-traffic control. It could be essential for a user to view essential information in a timely manner to make a critical decision of utmost importance. However, due to the present congestive nature of the internet, this could not be done in a timely manner. In order to ensure that such a flow meets its goal in a timely manner, there must be a means to monitor and improve its quality of service within the network.

This paper introduces a quality of service (QoS) scheme, using agents, that is scalable, congestion avoiding and controlling, network performance monitoring, congestion forecasting, diagnosing, and resource allocating and enforcing scheme aimed at providing end-to-end communication latency and jitter for these flows in a scalable, proactive and reactive manner. Unlike previous schemes, it does not do any QoS negotiation and renegotiation between the agents; thereby, not adding on to the overhead of providing QoS to these critical flows in terms of latency. It does this by combining quality of service monitoring, detection and prediction with explicit window adaptation techniques as resource allocation techniques. It also introduces a rescheduling scheme for non-critical flows within the links that share bandwidth with these critical flows. The scheme aim to create "lanes" large enough for the critical flows to flow through the soon-to-be or already congested link, thereby reducing the communication latency and jitter for these flows.

Keywords: Quality of Service; Congestion Control; Congestion Avoidance; Mobile Agents; TCP.

1. INTRODUCTION

This paper introduces a quality of service (QoS) scheme, using agents, that is scalable, congestion avoiding and controlling, network performance monitoring, congestion forecasting, diagnosing, and resource allocating and enforcing scheme aimed at providing end-to-end communication latency and jitter for these flows in a scalable, proactive and reactive manner. It uses 3 types of agents: Monitor (MA), QoS Manager (QM) and Resource Allocation (RA) agents. MA agents are monitoring agents that are placed on each router to constantly monitor and obtain performance metrics on flows. They send this information periodically to their domain QMs sitting on the Domain Name System (DNS) servers. These QMs, then, dynamically forecasts and detects congestions, diagnoses the violations for these congestions, and decides on possible solutions to remedy these violations. All these actions by the QMs are done within their own domain independently of other QMs without the need for costly QoS negotiations or renegotiations. After deciding on the action to remedy the violations, the QMs create the RAs to carry out congestion control and avoidance schemes on chosen links to ensure timely service for critical flows. These schemes, carried out by the RAs, aim to create "lanes" large enough for the flows belonging to the critical flows to travel through the soon-to-be or already

CCECE 2004- CCGEI 2004, Niagara Falls, May/mai 2004
0-7803-8253-6/04/$17.00 ©2004 IEEE

congested link, thereby reducing the communication latency for these critical flows.

In terms of reserving resources, the scheme is similar to Integrated Services (IntServ) [5] but unlike IntServ and Differentiated Services (DiffServ) [6], it handles congestions by predicting them within the network.

The rest of the paper is organized as follows: section 2 describes the QoS management scheme in detail and in section 3 discusses congested link selection strategies used by the QMs. Finally in section 4, the paper ends with the conclusion.

2. QOS MANAGEMENT

The QoS scheme consists of 4 agents that carry out its QoS management tasks. They are User (UA), Monitor (MA), domain QoS Manager (QM), and Resource Allocation (RA) agents. Their functions and their interactions with each other are described below.

2.1 User Agents

These agents are found at the host machines. If their applications require quality of service support, they will notify their QMs, located at the DNS servers.

The following example explains it more clearly. A web cast's User Agent (UA) needs quality of service support for its application to meet end-to-end communication latency and jitter requirements for its flow. It executes the *traceroute* program to discover the path from the source to the user, i.e. the destination. The UA then sends the information of the path and the application's bandwidth requirement to all the QMs found in the domains where the flow travels through. The QMs, on receiving this information, activates the MAs in their domain to do the monitoring of the routers through which the flow travels. When the transmission of the web cast is over, the QMs deactivate these MAs provided that they are not monitoring other critical flows that are using their routers. The MAs stop monitoring their routers and the QMs, after terminating any existing RAs, terminate their management obligations to the web cast flow.

2.2 Monitor Agents

The MAs continually and autonomously collect network performance and other relevant data from each router along the path of critical flows. The data that are collected from each router are 1) maximum bandwidth of the link that is being monitored, 2) used bandwidths by the flows using that link, 3) IP addresses of the router, the next hop and previous hop and 4) flow identifications. These data, collected by the MAs, are sent and used by the QMs, whose domain the MAs reside in, in making QoS management decisions for the critical flows flowing through the domain.

2.3 QoS Manager Agents

The QMs, located at the DNS servers, collect network performance and other relevant data, beneficial for QoS management, from the MAs found within their domain, along the path of the critical flows. Using these data, the QMs make decisions relating to QoS management issues for these critical flows that they are providing QoS support for. They detect QoS violations by predicting and identifying congested conditions that may increase communication latency and jitter for these critical flows. These *QoS violations*, i.e. *congested conditions*, occur whenever the summation of all of the used bandwidth for a certain link used by the non-critical flows and the required bandwidth for the critical flows using that particular link is greater than the maximum bandwidth for that link. They then decide on remedies for these violations and create the RAs to carry out their decisions.

The QMs are responsible for QoS management within their own domains only. They carry out their tasks independently of each other. There are no QoS negotiations or renegotiations among them. The resource allocation techniques that are carried out by the RAs on behalf of the QMs make this possible. Moreover, detections of QoS violations, i.e. congested conditions, are done proactively and reactively.

The following example uses a "web cast-user" critical flow to explain the QMs' ability to detect QoS violations proactively within the network.

In figure 1, the MAs along the path of the critical flow, send their collection of network performance and other relevant data to their domain QMs.

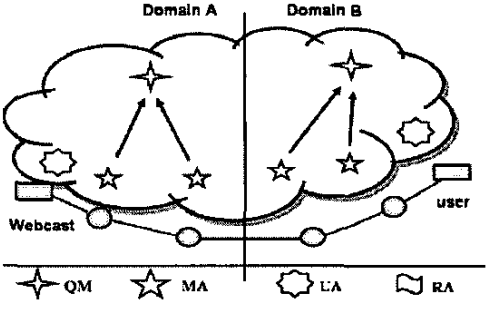

Fig. 1. MAs send their network performance and other relevant data for QoS management to their domain QMs

In figure 2, the QM, in domain A, detects a possible QoS violation in the link between the two MAs within its domain. This violation occurs, as previously mentioned,

when the summation of all of the used bandwidth for a link on the path of the critical flow and the required bandwidth requested by the web cast application is greater than the maximum bandwidth for that link. Hence, there is a possibility of QoS violation in the link in the form of congestion. If left unheeded, the link may become congested, thereby increasing the communication latency between the web cast and the user. Packets may be lost leading to unnecessary retransmissions of these packets. Proactive QoS action needs to be taken on this link to prevent congestion from occurring. Details of this QoS Resource Allocation are covered in section 2.4. The QMs also select the congested routers within its domain to remedy the congestion. The reader is to note that congestion may occur in more than one link. Due to space constraint, this selection process is not covered in this paper. The next example explains the QMs' ability to detect QoS violation *reactively* to support the web cast-user flow.

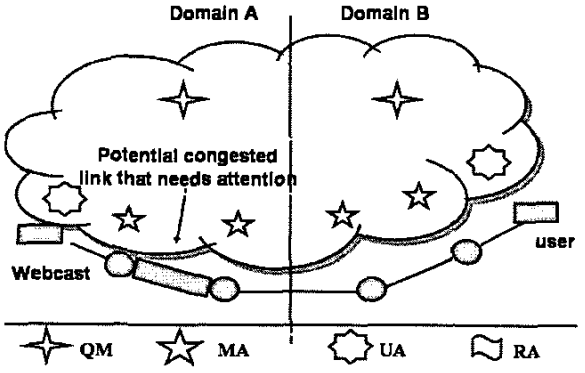

Fig. 2. Proactive detection of possible congested link by the QM in domain A

In figure 3, the web cast application is sending data to the user. There is no congestion for its flow to the user. All of a sudden, in figure 4, a non-critical flow enters the link, being used by the flow of the web cast application, congesting the link. If left unheeded, the communication latency for the flow between the web cast and the user will increase. Through the network performance data it receives from its MAs in its domain, the QM in domain A detects this QoS violation. *Reactive QoS action* needs to be taken on this link to remedy this. Details of this QoS Resource Allocations are covered in Section 2.4.

2.4 Resource Allocation Agents

Once a domain QM detects a possible QoS violation, it makes a decision on how to remedy this violation. Having made the decision, the QM creates a QoS Resource Allocation (RA) agent to enact it.

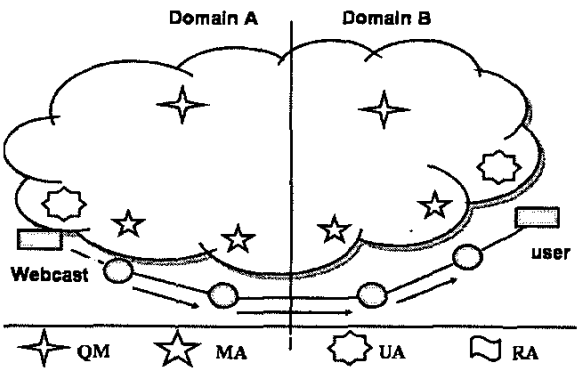

Fig. 3. Web cast application is sending data to the user without any congestion along its path

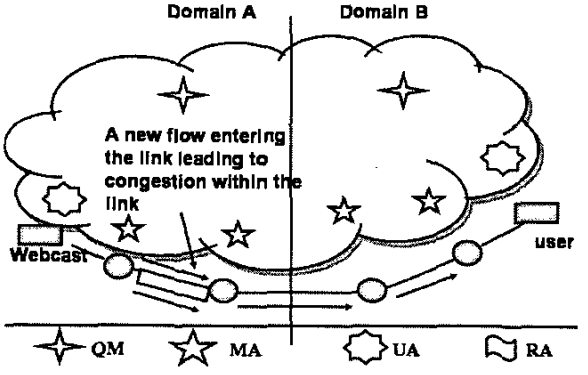

Fig. 4. Reactive detection of congested link by the QM in domain A

There are two scenarios that require two different actions to be undertaken by the RA agent to remedy a QoS violation. The implementations of these two actions are the same whether or not they are done proactively or reactively to remedy the violation.

The two scenarios are:

1) When the summation of bandwidth required for critical flows (ΣCj) through a congested link is less than the bandwidth capacity (B) for that link (i.e. $\Sigma Cj < B$) and

2) When the summation of bandwidth required for critical flows (ΣCj) through a congested link is more than the bandwidth capacity (B) for that link (i.e. $\Sigma Cj > B$).

2.4.1 Scenario One. The action for this scenario is implemented at the routers by the RAs when the summation of bandwidth required for critical flows' sessions (ΣCj) through a soon-to-be or already congested link is less than the bandwidth capacity (B) for that link (i.e. $\Sigma Cj < B$). Here, there is enough bandwidth for all the

critical flows that are about to use or presently using the link to maximize their share of bandwidth. Whatever bandwidth is not used or not going to be used by these critical flows is then allocated among the non-critical flows (see section 2.4.1.1). After the allocation, they need to be enforced. To enforce these allocations, one can modify at each router along the path of the critical flows, at the location of the soon-to-be or already congested link, the receiver's advertised window in TCP acknowledgments returning to the sources, and so as to decrease the number of bytes they may transmit (Section 2.4.1.2). After the enforcement of these flows, "lanes" large enough for the critical flows to travel through the soon-to-be or already congested link are created, thus remedying the QoS violation.

2.4.1.1 Rate Allocation for non-critical flows. Whatever bandwidth is not used by the critical flows, that is, critical flows that are about to use or presently using a link which is about to become or already became congested, is allocated among the non-critical flows. By doing so, enough bandwidth is reserved for all the critical flows are about to use or presently using the link.

The rate allocation scheme is used in such a way that the total throughput of all flows crossing the link does not exceed the link's capacity when the web cast-user flow and/or any other critical flows start transmitting through the link. This will prevent or control congestion in the links.

The approach is to divide the bandwidth available for the non-critical flows by using a variation of the Max-Min Fair Share scheme [4]. In the Max-Min Fair scheme, it is used for all the flows using a link.

The scheme works as follows: The smallest of all the bandwidths among the competing non-critical flows are allocated first. Let the flow demand of the smallest flow be x1 and it is allocated A/k where A is the remaining bandwidth, after the bandwidth allocation for the critical flows, and k is the number of competing non-critical flows in the link. If this allocation is greater than what the flow needs, then A/k – x1 will go back to the pool of leftover bandwidth. This means that the remaining k – 1 flows get an additional A/k + (A/k – x1)/ (k-1) and this process iterates until all of the remaining bandwidth A are exhausted or all demands have been met. For example, suppose the available bandwidth for non-critical flows of the link is 150 Mbps. Assume 5 competing flows have a bandwidth of 23, 27, 35, 45, and 55 Mbps each. Initially, the available bandwidth is divided equally and are allocated 31 Mbps each (155/5). Since the first flow needs only 23 Mbps, the remaining 8 Mbps (31-23) are divided equally among the 4 remaining non-critical flows. This gives the remaining flows 33 Mbps each. However, the second flow needs only 27 Mbps; the residual 6 (33-27) are divided among the 3 remaining non-critical flows. This increases the allocation for flows 3 to 5 to 35 Mbps each. Since all of these sources need 35 Mbps or more, the algorithm stops allocation at this point.

However, if the allocation for the smallest of all the bandwidths is lesser than what it needs using the above scheme, the above scheme cannot be applied in terms of fairness to all non-community flows. If S is the sum of all the competing non-critical flows and B is the available bandwidth for non-critical flows of the link, then let the ratio of S to B be M. Let the flow demand for the smallest of all these flows be x1 and it is assigned x1/M. Similarly, the bandwidths are allocated for the rest of the remaining competing non-critical flows. For example, suppose B is 50 Mbps. Assume 4 competing non-critical flows have a bandwidth of 20, 20, 20, and 40 Mbps each. The ratio M is 2 (100/50). Thus the allocation for the four flows are 10 (20/2), 10, 10 and 20 (40/2) Mbps.

Once the bandwidths for the competing non-critical flows has been allocated, they are enforced by the RAs on the selected routers by explicitly controlling the window size of these connections as a function of the available bandwidth that has been allocated.

2.4.1.2 Rate Enforcement for non-critical flows. In the current internet, the Transmission Control Protocol (TCP-Reno) [2] is responsible for controlling congestion by using the end-to-end window as a function of the congestion state of the network. Current TCP implementations contain a number of algorithms aimed at controlling network congestion. But these do not guarantee end-to-end communication latency for critical flows. The algorithms used by TCP include *slow-start, congestion avoidance, fast-retransmit,* and *fast recovery.* For the purpose of this paper, only slow-start and congestion avoidance algorithms are covered. These algorithms depend on the senders' and receivers' buffers. The sender has a congestion window, called *cwnd*, which is initialized to a single segment (segment size announced by the receiving side) at the start of a new connection. During the slow-start, at each time an acknowledgement packet, ACK, is received, the *cwnd* is increased by a segment. The sender transmits up to the *minimum of the congestion and advertised windows*. The advertised window is the actual flow control imposed by the receiver based on the amount of available buffer space at the receiver for the connection whereas the congestion window is a flow control imposed by the sender based on its assessment of perceived network congestion.

Hence, slow-start is the way to initiate data flow across a connection. At some point, the limit of an intervening router is reached, and packets can be dropped. *Congestion avoidance* is a way to deal with lost packets.

There are two indications for packet losses: timeout occurring and the receipt of duplicate ACKs. When congestion occurs, the transmission rate of packets into the network needs to be slowed down and slow start is activated to get things going again.

Both these algorithms are meant to work together. The Congestion Avoidance algorithm has a variable called ssthresh which is actually the slow start threshold size. At the start of a transmission, the *cwnd* is initialized to one and *ssthresh* to 65535 bytes. The Sender TCP never sends more than the minimum of cwnd and the receiver's advertised window. When congestion occurs (indicated by timeout or reception of duplicate ACKs), one-half of current window size (the minimum of cwnd and the receiver's advertised window, but at least two segments) is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment. When new data is acknowledged by the other end, the cwnd is increased depending on whether slow-start or congestion avoidance is being performed. If the cwnd is less or equal to ssthresh, then slow-start is activated otherwise congestion avoidance takes place. Slow-start continues until it is half to where it was when congestion occurred, and then congestion avoidance takes over. Slow-start has cwnd start at one segment, and be incremented by one segment, every time an ACK is received. It opens the window exponentially. Congestion avoidance dictates that cwnd be incremented by 1/cwnd each time an ACK is received.

In the section 2.4.1.1, the bandwidths for the competing non-critical flows have been allocated. Now, they need to be enforced. These are done by explicitly controlling the window size of these connections at the selected routers, at the location of the soon-to-be or already congested link, as a function of the available bandwidth that has been allocated. This technique [1] is used to enforce the rate of the non-critical flows. Since the Sender TCP never sends more than the minimum of cwnd and the receiver's advertised window, its rate is reduced if the minimum is the receiver's advertised window. If the minimum is cwnd, then the rate has already been reduced by the congestion window. The technique is used explicitly only for the reduction of the sender's transmission rate. The receiver's advertised window, w, can be calculated by using

w = (Allocated Bandwidth x Round Trip Time) -----(1)

The Round Trip Time is approximated by

1) Observing packets and the corresponding acknowledgement packets and vice-versa or

2) Pinging the source and destination

The new window value is then inserted in the receiver window field of the TCP header found in acknowledgement packets by taking the minimum of the old value in the receiver window field of the TCP header and the calculated value obtained in (1). If there is any other congested links further along the path, the technique is repeated on these links by the RAs that has been assigned to these links. Eventually, the Sender TCP receives the minimum calculated value for w along the path. This unique feature enables the domain QMs to function independently of each other without the need for QoS negotiations or renegotiations, thus, preventing communication delay due to costly QoS negotiations or renegotiations. The Sender TCP reads the receiver window field of the TCP header of acknowledgement packets and takes the minimum of w and the cwnd. Whenever the TCP header of acknowledgement packets are modified, the checksum in the TCP header needs to be adjusted for error control:

Delta = w – wold (using one's complement subtraction)

Checksum = Checksum + Delta (using one's complement addition)

This technique is applied for all the competing non-critical flows within the soon-to-be or already congested links. After the enforcement of these flows, "lanes" large enough for the critical flows to travel through the soon-to-be or already congested link would have been created, thus remedying the QoS violation.

2.4.2 Scenario One. The action for this scenario is implemented by the RAs on the selected routers when the summation of bandwidth required for the critical flows' sessions (ΣCj) through a soon-to-be or already congested link is more than the bandwidth capacity (B) for that link (i.e. $\Sigma Cj > B$). Here, there is not enough bandwidth for all the critical flows, which are about to use or presently using the link, to maximize their usage of the bandwidth. Hence, all competing non-critical flows within the link are rescheduled until one or more critical flows cease (section 2.4.2.1). The rescheduling is done by setting at an intermediate network element, at the location of the soon-to-be or already congested link, the receiver's advertised window in TCP acknowledgments returning to the sources to zero. These actions enable only critical flows to travel through the links. The link bandwidth is partitioned among the critical flows using the rate allocation scheme that was used for non-critical flows in section 2.4.1.1. After that, the rate enforcement scheme for non-critical flows in section 2.4.1.2 is used on these flows. These actions help to reduce the communication latency for these critical flows.

2.4.2.1 Rescheduling all competing non-community flows within the link. A flow is rescheduled by setting the receiver window field of the TCP header of an acknowledgment packet to zero at a router, at the location of the soon-to-be or already congested link. Permission is granted later by sending a segment with the same acknowledgement number and the old receiver

window size field. This technique is used to reschedule all competing non-critical flows within a link.

TCP has a persistence timer [2] which is designed to prevent the following deadlock: When the sender receives an acknowledgement with a receiver window size of 0, it waits. Eventually, the persistence timer goes off. The sender sends a probe to the receiver. The response to the probe gives the actual window size from the receiver. If the critical flows has ceased up, the sender receives the actual window size from the receiver. Otherwise, it will receive the modified window size of 0. If it is still zero, the timer is set again and the cycle repeats. Otherwise, the data is sent. The first timeout is bounded between 5 and 60 seconds, exponentially backing off. TCP never gives up sending window probes. It does so by sending probes at 60 seconds interval until window opens up or either of the applications using the connection is terminated.

3. Conclusion

This report addressed the issues of avoiding and controlling congestions, monitoring communications, forecasting congestions, diagnosing QoS violations, and allocating and enforcing resources within and across non-QoS critical flows with the goal of reducing communication latency in a scalable, proactive and reactive manner. A performance evaluation of the mechanism will be provided in the near future.

Acknowledgements

I would like to thank Dr Zaruba for his guidance in this research.

References

[1] R. Satyavolu, K. Duvedi, and S. Kalyanaraman, "Explicit rate control of TCP applications", *Unpublished Manuscript, 1999.* http://www.ecse.rpi.edu/ Homepages/ shivkuma/ research/papers/iwqos99-rate.ps

[2] W. Richard Stevens, *THE PROTOCOLS TCP/IP ILLUSTRATED.* Volume 1, Location: Addison-Wesley Pub Co., 1995.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol", *IEEE-Network*, Vol 7, No 5, 1993.

[4] Sanjay Jha and Mahbub Hassan, ENGINEERING INTERNET QOS. 1st edition, Location: Artech House, August 15, 2002.

[5] Integrated Services (IntServ), http://www.ietf.org/html.charters/intserv-charter.html

[6] Differentiated Services (diffserv), http://www.ietf.org/html.charters/diffserv-charter.html