Congestion Avoidance for Unicast and Multicast Traffic

Anca Dracinschi, Serge Fdida

LIP 6, "Pierre et Marie Curie" University
8 rue du Capitaine Scott, 75015 Paris France
{anca.dracinschi, serge.fdida}@lip6.fr
tel. +33 1 44 27 71 26
fax +33 1 44 27 53 53

Abstract

This paper proposes a congestion avoidance mechanism that addresses unicast (TCP or UDP) as well as multicast (UDP) best effort flows. Our mechanism originally combine Explicit Congestion Notification (ECN) based on router active queue management to detect congestion, with ICMP Source Quench messages to inform the involved sources of the network congestion state. The fairness achieved is a trade-off between TCP-like and Max-Min fairness. We demonstrate through simulation that the proposed mechanism enforces fairness and provides an efficient and simple solution to congestion avoidance.

Keywords: multicast, active queue management, ICMP source quench, fairness, best effort

1 Introduction

In the IP framework, the flows without end-to-end bandwidth guarantees (i.e. best-effort flows) have to adapt efficiently to network dynamics to avoid congestion. Network resources are shared by competing unicast (TCP and UCP) and multicast (UDP) communications.

(1) For unicast flows, adaptation is enforced by TCP using a congestion control mechanism. The UDP flows are *unresponsive* because there is no built-in mechanism akin to TCP congestion control. [1, 2, 3] propose end-to-end congestion control mechanisms for UDP flows mainly for unicast applications. The solution is similar to the TCP one, i.e. a feedback mechanism is used to allow loss indication to be conveyed backwards from the receiver to the sender. Then, the source adapts its sending rate.

(2) For multicast communications, UDP is widely used as the transport layer. If no additional mechanism is built on top of UDP in order to ensure responsiveness, these are *unresponsive flows*. In such situations, the TCP-like congestion control approach is difficult to apply since it raises scalability and aggregation problems. However, TCP-like solutions were presented by [4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. These solutions, either try to combine congestion control with multicast reliability, are protocol dependent and, therefore, not adapted for protocols without embedded reliability mechanisms, or use the RTCP (Real Time Control Protocol) side of RTP (Real Time Protocol) [14] which raises new problems with the IGMPv3 (Internet Group Management Protocol version 3) [15] when only the source is allowed in sending multicast.

As long as there is UDP, there is the possibility for unresponsive flows to be generated. In order to cope with network congestion, one approach is for all UDP flows to use a congestion control mechanism and become responsive, another is to build into the network the capacity to handle unresponsive flows effectively (e.g. by dropping packets).

This paper proposes a congestion avoidance mechanism that addresses both unicast and multicast best-effort flows, offering to all (responsive or not responsive) flows involved in a congested router the possibility to be responsive. Section 2 gives a short overview of topics tightly linked to our mechanism. Section 3 describes the algorithm. An analysis of the proposed algorithm is performed in section 4. Section 5 shows simulation results produced by NS-2. Section 6 concludes the paper.

2 Assumptions and goals

An important challenge in schemes addressing congestion is how to keep routers from entering the congestion state rather than having to recover once congestion has occurred. A scheme that allows the network to prevent congestion is called a *congestion avoidance* scheme [16].

In order to provide congestion avoidance, TCP congestion control has been enhanced with Explicit Congestion Notification (ECN) [17], an explicit feedback in the TCP header (see Figure 1) to indicate incipient congestion from the receiver to the source using acknowledgments. ECN builds on the Random Early Detection (RED) [18] mechanism for router active queue management. Two experimental ECN bits are used in the IP header (see Figure 2). RED detects incipient congestion by computing the average queue length of the router buffer using a low-pass filter with exponential weighted moving average. The average queue size is compared to two

0-7803-6419-8/00/$10.00©2000 IEEE

thresholds: min_{th} and max_{th} to detect the congestion state for the router. The mechanism works as follows:

1. When the average queue size is less than min_{th}, no action is done.
2. When the average queue size is between min_{th} and max_{th}, the incoming packets are ECN marked with a certain probability. Each time a packet is marked, the probability that a packet is marked from a particular connection is roughly proportional with the corresponding share bandwidth of the connection at the congested queue.
3. When the average queue size is greater than max_{th}, all the arriving packets are ECN marked.

Thus, the receiver is informed of the congestion state.

In the case of unicast UDP communications, congestion avoidance using ECN can be similarly performed by schemes (e.g. [1, 2, 3]) that provide congestion feedback notification from the receiver to the source as TCP does. The control overhead introduced in order to inform the source of the congestion state is comparable to the additional traffic introduced by TCP acknowledgments. In the case of multicast communications, the control overhead could be much higher, depending on the number of receivers and, therefore, new issues have to be solved like source implosion and adaptation choice.

Our mechanism is based on the utilization of the *two ECN bits in the IP header* to mark like ECN the packets experiencing congestion, but uses *ICMP source quench messages* (see 2.1) as a feedback mechanism to inform the sender of the network congestion state. The objectives are:

1. to allow all flows to be responsive,
2. to minimize the additional control overhead,
3. to avoid scattering the congestion information towards receivers and aggregating it afterwards for the sender in the case of multicast,
4. to be efficient in terms of delay by allowing the congested router to directly inform the sender about its status, and
5. to provide fairness among the flows experiencing congestion by addressing only the scarce resources between the sender and the worst bottleneck.

When the active queue management detects congestion on a router, an ICMP source quench message is sent from the router to the packet's source host to inform of congestion state.

2.1 ICMP source quench message

[16] proposes source quench messages to be sent back by a gateway to the Internet source host of a discarded datagram when the gateway does not have the buffer space needed to queue the datagram for output to the next network on the route to the destination, or when the gateway approaches its capacity limit rather than waiting until the capacity is exceeded. In the latter case the datagram which triggered the source quench message may be delivered.

3 A Congestion Avoidance Mechanism for Unicast and Multicast

As described above, our mechanism relies on active queue management on the routers to detect incipient congestion and uses ICMP source quench messages to indicate the congestion state backwards to the senders whose packets are involved in congestion, as shown in Figure 3.

3.1 Router mechanisms

We term a group of successive packets in a queue, having the same IP source-destination addresses and the same port numbers, as "burst of packets belonging to the same flow".

The router active queue management algorithm detects *incipient congestion*. In this case, the packet to be sent is checked in order to identify two different conditions: 1, the packet is ECN marked, and 2, the packet is the last packet of a burst of packets belonging to the same flow and stored in the router's queue. When the packet is not marked (condition 1 does not hold), the ECN bit is set in the IP header. If condition 2 applies and if the packets of the burst do not hold any ECN mark, an ICMP source quench message is returned to the source of the burst (Figure 4).

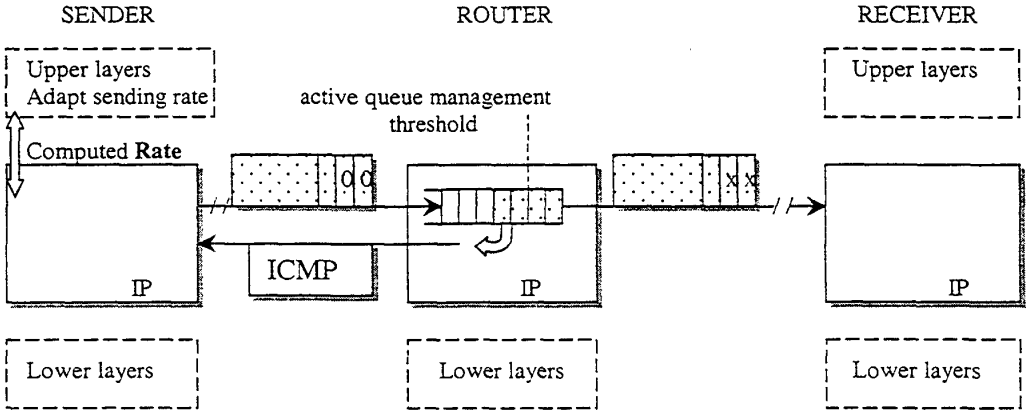
16-bit source port number										16-bit destination port number
32-bit sequence number										
32-bit acknowledgment number										
4-bit header length	Reserved	CWR	ECN	URG	ACK	PSH	RST	SYN	FIN	16-bit window size
16-bit TCP checksum										16-bit urgent pointer

Figure 1: TCP header

8-bit Type of Service/DS Field

4-bit version	4-bit header length	DSCP	ECT	CE	32-bit total length (in bytes)
16-bit identification			3-bit flags	13-bit fragment offset	
8-bit TTL	8-bit protocol		16-bit header checksum		
32-bit source IP address					
32-bit destination IP address					

Figure 2: IP header



00, xx – ECN bit mark in the IP header

Figure 3: An end-to-end scenario for congestion avoidance

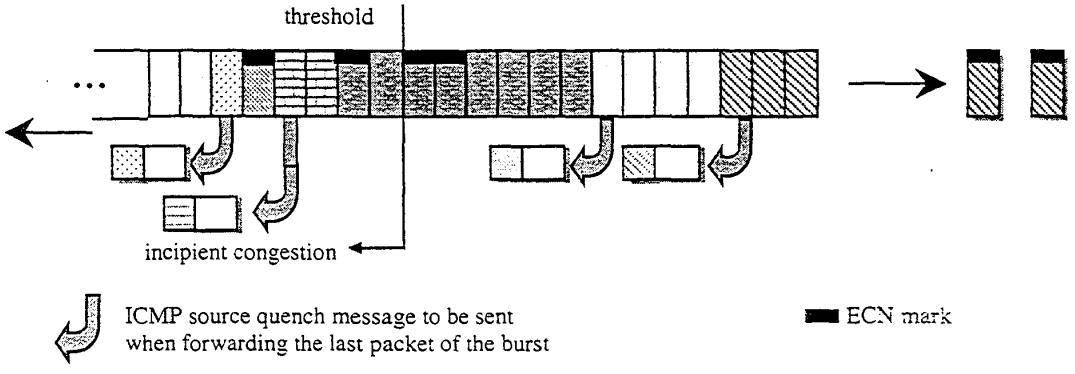


Figure 4: Router mechanisms

The ECN bit set in the IP header of a packet is used by the successive congested routers as an indication that the burst has already passed through at least one congested router. Hence, an ICMP source quench was already transmitted to the sender for this burst and an additional ICMP source quench transmission is avoided.

When the router's queue becomes full, the queue enters the *congestion* state and the arriving packets are dropped. The queued packets to be sent are checked as in the incipient congestion case.

3.2 Sender mechanisms

The sender reaction to network congestion is the most sensitive part of a congestion avoidance scheme, influencing the loss rate at the bottleneck link (high if the sending rate is too high), the efficient utilization of resources (low if the sending rate is too slow), and the fairness among the flows involved.

Negotiations between sender(s) and receiver(s) by means of end-to-end flow control mechanisms allow establishing data transfer rate boundaries agreed by receiver(s). These boundaries are hereinafter called **limits**. In the case of TCP, the flow control is ensured by a window mechanism that uses a field in the acknowledgments to advertise the buffer space of the receiver to the sender. In the case of UDP, flow control mechanisms, if any, are handled at a higher layer. We focus below on the algorithm used to adjust the data transfer rate offered by the sender to ICMP source quench.

As long as no ICMP source quench has been received since the data transfer started, the sources keep on sending data at the rate monitored by the upper layer mechanisms (e.g. TCP for TCP flows, application for UDP flows). When the first ICMP source quench arrives indicating congestion or incipient congestion, additional mechanisms are activated at the sender side.

3.2.1 Additional mechanisms

We propose a congestion window based algorithm at the sender side. The following parameters are used:

1. **cwnd**: the congestion window measured in terms of packets.
2. **rRTT** (router Round Trip Time): a timer delay taking the value of the RTT between the sender and the congested router from which an ICMP source quench has been received. The **rRTT** is the sum of two quantities: the time a packet needs to reach that router and the time the ICMP source quench (having the same sequence number as the packet) generated by that router needs to reach the sender of the packet. This would require an additional layer above UDP that provides: packet sequence_number and a mechanism to compute the **rRTT** based on packets' sending time as TCP computes the RTT. The measured $\mathbf{rRTT_m}$ is computed as:

$$\mathbf{rRTT_m} = \text{time\_packet\_source\_router} + \text{time\_ICMProuter-source}, \quad \text{or}$$
$$\mathbf{rRTT_m} = \text{crt\_src\_time} - \text{packet\_sent\_time} \qquad \text{(eq1)}$$

where crt_src_time is the time when the ICMP source quench has been received and packet_sent_time is the time when the packet with the same sequence number has been sent.

rRTT is smoothed using a low-pass filter:

$$\mathbf{rRTT} \leftarrow \alpha * \mathbf{rRTT} + (1 - \alpha)\,\mathbf{rRTT_m} \qquad \text{(eq2)}$$

where α is a smoothing factor with a recommended value of 0.875.

As long as no ICMP source quench has been received since the data transfer started, the sources keep on sending data at the rate monitored by the upper layer mechanisms (e.g. TCP for TCP flows, application for UDP flows). When the first ICMP source quench is received, **cwnd** is set to one packet and **rRTT** is computed as $\mathbf{rRTT_m}$ (eq1). The sending rate is computed as:

$$\text{Rate} = \min(\mathbf{cwnd} * \text{MTU} / \mathbf{rRTT}, \mathbf{limits}) \text{ (bytes/s)} \qquad \text{(eq3)}$$

where MTU (Maximum Transmission Unit) is the packet size used on the connection. MTU may be determined using the MTU Discovery Algorithm [19], or the minimum acceptable value for TCP (576 bytes).

Table 1: The congestion window behavior

	Decrease upon receiving a valid ICMP source quench message (except the first one)	Increase when no valid ICMP source quench message has been received for an **rRTT**
Unicast transmission	**cwnd** = 1/2 * old_cwnd[1] (at least 2 MTU);	Linear increase until the sending rate reaches **limits**
Multicast transmission	**cwnd** = $f(n)$[2] * old_cwnd (at least 2 MTU);	Linear increase until the sending rate reaches **limits**

[1] In the case of TCP, this corresponds to slow start mechanism and old_cwnd is the value of cwnd before decreasing. In the case of UDP, old_cwnd has a known initial value.

[2] We specify this function in section 4.

3.2.2 Sender behavior

Upon receiving an ICMP source quench, the sender (i) checks its sequence_number in order to detect whether it is a valid congestion signal and (ii) computes the new $\mathbf{rRTT_m}$ (eq 2).

(i) An ICMP source quench is valid if the sequence_number corresponds to a packet sent after the last **cwnd** reduction. This means that the last sending rate was not decreased enough to avoid congestion. A valid ICMP source quench triggers **cwnd** to decrease according to AIMD algorithms that are different depending whether the flow is unicast or multicast, as shown in Table 1.

(ii) The current **rRTT** is modified by the new $\mathbf{rRTT_m}$ value when the **rRTT** timer expires, smoothly adapting to the **rRTT** (eq2) and, therefore, converging to the worst bottleneck.

If no valid ICMP source quench was received during an **rRTT** period, **cwnd** is increased when the **rRTT** timer expires. If this happens after the first ICMP source quench of the transmission, the increase is exponential until old_cwnd/2 and linear afterwards until the sending rate reaches **limits**. Otherwise, the increase is linear until the sending rate reaches **limits**.

The sending rate is computed (eq 3) when **cwnd** or **rRTT** change in order to adapt to the updated congestion state. If the sending rate reached **limits** for p (known) **rRTT** periods, the **rRTT** timer is set to one and **cwnd** changes in order to preserve the same sending rate (i.e. **limits**) as follows:

$$\mathbf{limits} = \text{old\_cwnd} * \text{MTU}/\mathbf{rRTT} = \mathbf{cwnd} * \text{MTU}/1 \quad \text{(eq4)}$$

In this case, a new congestion or incipient congestion signal will trigger a reaction as when receiving the first ICMP source quench.

4 Discussion

This section shows the reasons for some design choices, as well as their impact on fairness, overhead traffic, scalability and interoperability.

In the case of multicast communication, **cwnd** uses $f(n)$ to adapt upon receiving a valid ICMP source quench. $f(n)$ is related to the theoretic gain (in terms of network resources) allowed by a n receiver multicast transmission compared to n unicast transmissions for n receivers. We measure the network resources in terms of bits occupied by the packets still flowing in the network. In order to be TCP friendly under congestion, we reduce the resources to be used by 50%, since under congestion TCP reduces the used resources by 50% in case of fast recovery. The idea is to adapt the multicast applying the same reduction factor to the amount of resources used by n unicast transmissions as equivalent of the multicast. We consider a binary tree in our simulations and computing $f(n)$ leads to the constant 3/4. For general Internet topologies [20] provides help to analyze the average-case of $f(n)$. This is to be investigated in our future work.

Fairness

We choose different AIMD algorithms for multicast and unicast, i.e. a smaller reduction for multicast than for unicast sending rate in case of congestion. Hence, we allow an *explicit* approach to fairness between multicast and unicast flows, since the former brings implicitly a better utilization in terms of network resources. Our mechanism provides also an *implicit* approach to fairness, by means of its rRTT mechanism. This fairness, based on TCP fairness, could be seen as lying in-between TCP fairness and Max-Min fairness. By means of **rRTT**, we allow "equal" (as depending on the *explicit* fairness) congestion windows per **rRTT** to all flows involved in the bottleneck path, as TCP does with its RTT mechanism. Therefore, our fairness results in sharing only the resources between the source and the most congested router.

Overhead traffic

A general drawback mentioned with respect to source quench messages is the additional traffic in the reverse direction on what might be a congested path. Nevertheless, taking into account the need of congestion state information for UDP senders in order to be responsive, besides avoiding the aggregation of congestion signals, we reduce the ICMP source quench generation by using the ECN bits. The simulations show that the number of source quench messages for a given TCP flow is less then 5% of the number of acknowledgements.

Scalability

The scalability issue raises for multicast communications mainly when receivers or router states are involved. No receiver is involved in our mechanism. Concerning the router, our mechanism uses one state per queue, the copy of the current analyzed packet. The complexity is independent of the group size and, therefore, rises no scalability problem.

Interoperability

We propose a sender oriented congestion avoidance mechanism. Receiver oriented multicast congestion control mechanisms also exist in literature [21, 22]. Then, congestion signaling is not suitable at the sender side since the receivers deal with congestion. In order to ensure interoperability with such protocols and, thus, to avoid ICMP source quench to be returned to senders, a specific ECN bit mark can be used in the IP header of the packets sent by sources using these protocols. Upon receiving a packet with this ECN bit mark, a congested or incipient congested router does not send any ICMP source quench back to the sender. The congestion state has to be indicated to receivers in order to react according on their congestion control mechanisms.

5 Performance evaluation

We implemented our mechanism in Network Simulator 2 [23] to analyze its behavior when TCP and UDP unicast and multicast flows interact together and experience congestion. We are particularly interested in fairness and efficiency in avoiding congestion.

We use the NS2 RED, TCP and UDP implementations modified in order to use the parameters (**rRTT** and **cwnd**) as described above. The length of each simulation is about 500 seconds allowing us to analyze the steady state behavior of the system. All TCP flows are FTP sessions with infinite amount of data. All UDP flows are CBR set to consume together less than the bottleneck link capacity, or On-Off flows. The packet size for each flow is fixed to 1000 bytes and both the acknowledgement and ICMP source quench message size to 40 bytes. We start the transmissions at slightly different times in order to avoid synchronization. The routers intended to be congested run RED and their resources are scaled up linearly with the total number of flows (N). Their buffer size is N*16 packets and max_{th} N*15 packets. We have analyzed the behavior of our mechanism varying min_{th} between 10 and 50 packets for different values of N between 20 and 100 flows. It provides fair bandwidth sharing in all cases, the results being slightly similar to those presented below where min_{th} is set to 20 packets. Other RED parameters are the default values in NS2. Table 2 summarizes the used parameters.

This paper focuses on topologies with equal **rRTTs** for the most congested router, where fairness is achieved when the flows get equal average bandwidth which is easily identifiable. In reality most flows experiencing congestion have the senders located at different distances from the congestion. In these cases, the sender with a smaller rRTT gets larger bandwidth

Table 2: Simulation parameters

Number of flows	N = tcp + udp + onoff
RED queue size	N*16 packets
RED max_{th}	N*15 packets
RED min_{th}	20 packets
Packet size	1000 bytes
Ack size	40 bytes
Source quench size	40 bytes
Simulations length	500 seconds

5.1 Analysis for Unicast Flows

The first topology used for our simulations is shown in Figure 5. The bottleneck bandwidth is C = N*50 Kbps. Therefore, each flow is supposed to get an average bandwidth of 50 Kbps for equal **rRTTs**.
All links except the bottleneck link are sufficiently provisioned to ensure that only the bottleneck link experience congestion.

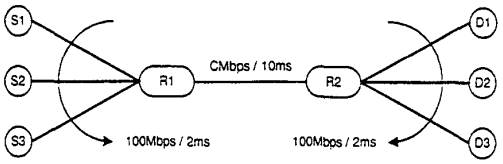


Figure 5: Topology 1

Our goal is to examine the behavior of TCP flows interacting with UDP unicast flows in two situations as follows. In Figure 6, ten TCP flows[3] using only ECN share the bandwidth left by three UDP flows of 200, 266, and 400 kbps. Figure 7 presents the case where the ten TCP flows using our mechanism share the bandwidth left by the same three UDP flows. We can notice when comparing both graphs, that our mechanism preserves the fairness between TCP flows as ensured by the present TCP implementations.

In the second scenario, both TCP and UDP flows use our mechanism. Figure 8 illustrates that it ensures the fair sharing of the bottleneck link between all flows, TCP and UDP. When the TCP flows stop their transmission (at 300 seconds on the time scale), the UDP flows try to fairly share the new available bandwidth and get the same bandwidth share, as the sum of the UDP rates is larger than the link capacity. The initial UDP sending rates, as demanded by applications, respect a given ratio (Figure 6). The bandwidth share when UDP1 stops (at 450 seconds) follows the same ratio, as now the sum of the UDP2 and UDP3 rates is less than the link capacity.

The availability for a new arriving flow on the bottleneck link in congestion is also shown in Figure 8. UDP1 arrives at 250 seconds and fairly shares the bandwidth with the existing flows.

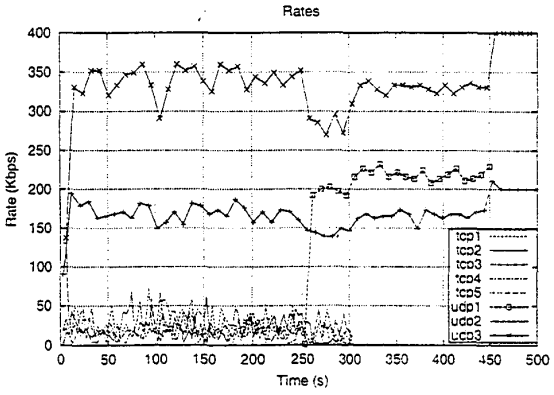


Figure 6: Bandwidth Share TCP and UDP

[3] We plot only the first 5 TCP flows to facilitate reading the figures.

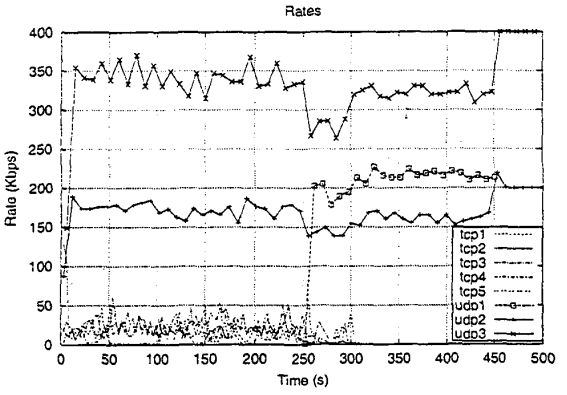


Figure 7: Bandwidth Share TCP-SQ and UDP

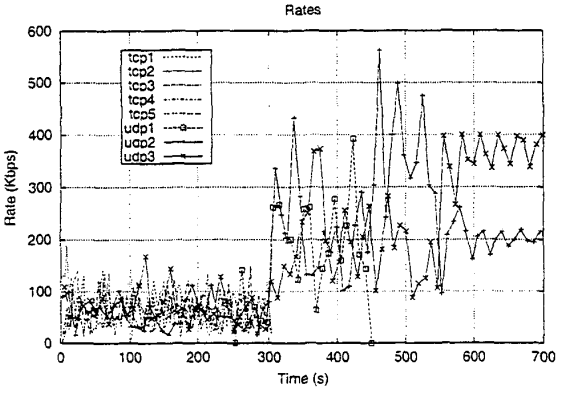


Figure 8: Bandwidth Share TCP-SQ and UDP-SQ

5.2 Abnalysis for Unicast and Multicast flows

We now analyze our mechanism with topologies having two bottleneck links. When multicast flows are involved, these bottleneck links can be on the same path or on different paths of the multicast tree. For the set of simulations where the two bottleneck links are on the same path of the multicast tree, the topology is shown in Figure 9.

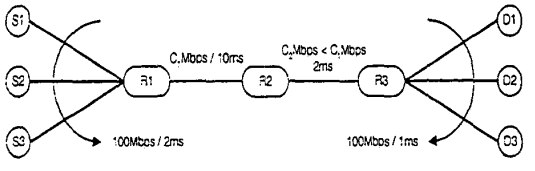


Figure 9: Topology 2

The bandwidth of the bottleneck links are N*80 Kbps for C1 and N*50 Kbps for C2. At the beginning of the simulation, the sending rate reduction is due to the ICMP source quench sent mainly by router R1. As the capacity of link R2-R3 is smaller than the capacity of link R1-R2, router R2 still experiences congestion when the sources adapt their sending rates to link1. Therefore, R2 keeps sending ICMP source quench in order to cope with its congestion state. This triggers the senders to adapt their sending rates to link R2-R3.

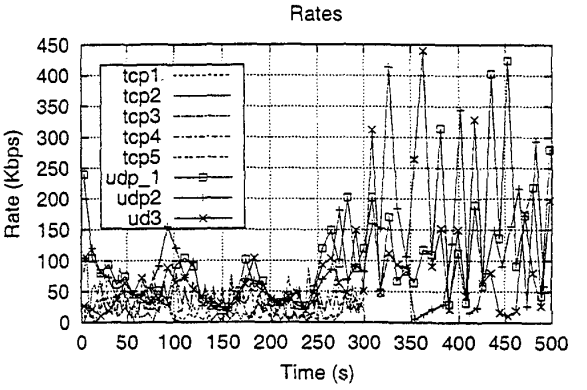


Figure 10: Bandwidth Share TCP-SQ and UDP-SQ Topology 2

Hence, the behavior confirms our expectation concerning the convergence of our mechanism to adapt to the worst bottleneck link. When the most congested router is the first of the congested routers on a path, the sending rates adapt directly to this router.

In the case of multicast flows, it is more interesting to analyze the behavior when the congestion or incipient congestion occurs on different paths of the multicast tree at the same time.

The topology used for this set of simulations is illustrated in Figure 11. The bandwidth of the bottleneck links is C1=500 Kbps for link R2-R3 and C2=1 Mbps for link R2-R4. We consider 21 TCP flows running between S1-S3 and D1-D6 and one multicast UDP flow sent from S1 with the initial rate of 400kbps. On each bottleneck link there are unicast flows and the UDP multicast flow has linkR2-R3 and link R2-R4 on different paths of its multicast tree. All the flows are using the presented mechanism.

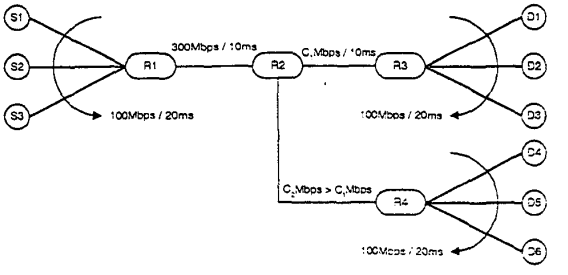


Figure 11: Topology 3

At time 10 seconds receiver D5 joins the multicast group. The sending rate of the multicast UDP flow adjusts to the bandwidth share of the bottleneck link R2-R4. When

receiver D2 joins at time 150 seconds, the multicast flow sending rate adjusts at the slowest link, this is the bottleneck link R2-R3 (C2 < C1). In Figure 12, link R2-R3 bandwidth sharing is illustrated by the curves at the bottom of the graph, while link R2-R4 bandwidth sharing is illustrated by the upper curves. The multicast corresponding curve is among the curves at the bottom of the graph. Therefore, this set of simulations shows also, that the multicast flow adapts to the worst bottleneck link.

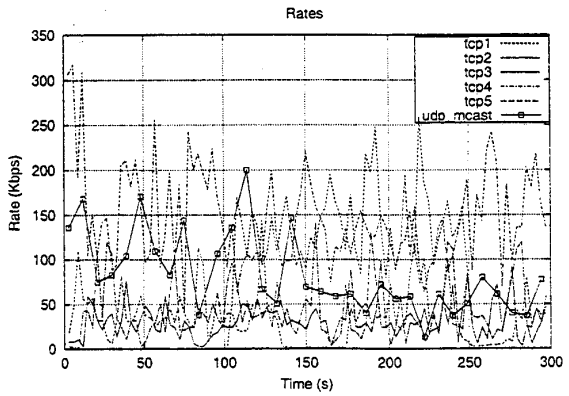


Figure 12: Bandwidth Share TCP-SQ and UDP -SQ Topology 3

The measures per flow of the bandwidth used by the ICMP source quench compared to the bandwidth used by the TCP acknowledgments show that little bandwidth overhead is use (Table 3). When more bottleneck links are on a path, the ICMP source quench are mainly sent by the most congested router. The mechanism stability is shown by router queues that do not grow indefinitely at the bottleneck link as the average queue length stays below max_{th} as target of congestion avoidance.

Table 3: Source quench statistics

	#ICMP source quench message packets / #TCP acknowledgement packets
Topology 1	2.8-4.0%
Topology 2	Link R1-R2 0.0-0.1% Link R2-R3 2.6-4.8%
Topology 3	Link R2-R3 0.7-1.0% Link R2-R4 2.0-2.4%

The presented mechanism satisfies our objectives of providing a fair sharing of scarce network resources and a congestion avoidance mechanism for both unicast and multicast, as well TCP as UDP flows by means of minimum additional control overhead in all simulated topologies.

6 Conclusions and future work

We presented in this paper a congestion avoidance mechanism which addresses unicast and multicast best-effort (reliable or not reliable) flows. We introduced a new approach to fairness by means of the router round trip time concept used by our mechanism. We evaluated the mechanism behavior by simulations and our results show that it achieves fair sharing of network resources using little bandwidth control overhead. Due to space limitations, many details of the algorithm and simulation results are not shown in this paper. Our future work is to carry out more and larger scale simulations and to refine the mechanism based on the gained experience. Another issue we plan to explore is a theoretical analysis of the fairness as defined here.

References

[1] J. Padhyle, J. Kurose, D. Towsley, R. Koodli. A TCP-friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Network. UMASS CMPSCI Technical Report 98-47, October 1998. An Extended Abstract will appear in the *Proceedings of SIGMETRCICS'99.*

[2] R. Rejaie, M. Handley, D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. *Proceedings of IEEE INFOCOM'99*, March 1999.

[3] D. Sisalem, H. Schulzrinne. The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme. *Proceedings of NOSSDAV 1998.*

[4] T. Sano, N. Yamanouchi, T. Shiroshita, O., Takahashi . Flow and CC for Bulk Reliable Multicast Protocols. *Proceedings of INFOCOM 1998.*

[5] D De Lucia, K. Obraczka. A Congestion Control Mechanism for Reliable Multicast. IRTF RMRG meeting, September 1997.

[6] L.Vicisano, M.Handley, J. Crowcroft. B-MART, Bulk-data (nonRT) Multiparty Adaptive Reliable Transfer Protocol. Draft http://research.ivv.nasa.gov/RMP/links.html

[7] T. Montgomery. A Loss Tolerant Rate Controller for Reliable Multicast. Technical Report NASA - IVV-97-011, http://research.ivv.nasa.gov/RMP/links.html, August 1997.

[8] S. Bhattacharyya, D. Towsley, J. Kurose. The Loss Path Multiplicity Problem in Multicast CongestionControl. *Proceedings of INFOCOM 1999*. Also available as CMPSCI Technical Report TR 98-76, August 1998.

[9] H. A. Wang, M Schwartz. Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet. *Proceedings of SIGCOMM '98.*

[10] Reliable Multicast Protocol. http://research.ivv.nasa.gov/RMP/Docs/index.html

[11] RMTP: A Reliable Multicast Transport Protocol. Internet draft - RMTP - II specifications. ftp://ftp.ietf.org/internet-drafts/draft-whetten-rmtp-ii-00.txt

[12] T. Speakman, D. Farinacci, S. Lin, A. Tweedly. PGM Reliable Transport Protocol Specification. Internet Draft. http://ale.east.isi.edu/RMRG/orlando/draft-speakman-pgm-spec-00.txt, 1998.

[13] T. Speakman. PGM Reliable Transport Protocol Specification. Invited talk NGC99, November 1999.

[14] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications RFC1889, 1996

[15] B. Cain, S. Deering, A. Thyagarayan. Internet Group Management Protocol Version 3. Internet Draft November 1999

[16] K. Ramakrishnan, S. Floyds. A Proposal to add Explicit Congestion Notification (ECN) to IP. ftp://ftp.isi.edu/in-notes/rfc2481.txt

[17] S. Floyd, V. Jacobson. Random Early Detection (RED) gateways for Congestion Avoidance. http://www-nrg.ee.lbl.gov/papers/red/red.html

[18] J. Postel. Internet Control Message Protocol. RFC 792, 1981.

[19] J. Mogul, S. Deering. Path MTU Discovery. RFC1191, 1990.

[20] M. Faloutsos, P. Faloutsos, C. Faloutsos. On Power-Law Relationship of the Internet Topology. *Proceedings of SIGCOMM'99.*

[21] T. Turletti, S.F. Parisis, J. Bolot. Experiments with a Layered Transmission Scheme over the Internet. *Proceedings of INFOCOM'98.*

[22] L. Vicisano, L. Rizzo, J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. *Proceedings of INFOCOM'98.*

[23] S. McCanne, S. Floyd. Network Simulator ns2. http://www-mash.cs.berkeley.edu/ns