Low-Weight Congestion Control for Multi-sender Applications

Jeremiah Scholl, Peter Parnes

Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
{jeremiah, peppar}@cdt.luth.se

Abstract. This paper presents a prototype for single-rate reliable multicast congestion control, which has been built into an existing commercial whiteboard. The prototype was developed using a novel scheme that was engineered around conflicting industry provided requirements for collaborative workspaces. This required the scheme to be both low-weight when used with many senders and compatible with NAT, firewalls and reflectors. The key to overcome this conflict was to combine congestion control and recovery feedback. This differs from many current solutions in that they are often designed for use with a wide variety of protocols and thus operate independent of the recovery mechanism. This paper does not go into the detail required to specify a protocol but instead discusses a few important design requirements for multi-sender applications, which are generally not considered by current research, and describes an approach towards meeting these requirements.

1 Introduction

Over the past several years there has been an increase in the demand for scaleable real-time media applications. This demand combined with the popularity of IP-networks has lead to an increase in the number of applications that take advantage of IP-multicast. Currently, for many reasons, there remain reservations about the wide deployment of the current generation of these applications. One such reason is that they lack effective mechanisms for fair bandwidth sharing with TCP, which could lead to massive congestion problems if they are deployed on a large scale.

For many to many applications the traditional congestion control approach has been to keep the aggregate bandwidth for the session below a static level [1]. This method can provide good performance if network conditions are constant and the administrator of the session selects the correct bandwidth limit. However, the amount of bandwidth available in IP-networks is dynamic by nature causing this static approach to perform very unreliably in practice. As the network becomes congested this becomes a serious issue because the applications will use bandwidth in an over aggressive manner and can severely impact the performance of other multicast applications as well as traditional TCP applications (for instance email and the web) running over the same network.

It has been generally recognized that in order for multicast applications to become popular, effective congestion control schemes must be implemented that allow them to adjust their bandwidth to current network conditions. Because these applications must coexist with TCP, a large number of researchers now share the conservative viewpoint that a flow is acceptable only if it has a long or medium term throughput to any receiver that does not exceed the rate that would be achieved by TCP between the multicast sender and that receiver [2], [3], [4], a state often referred to as TCP-friendly. This viewpoint is especially strong regarding reliable multicast because there is a general view that reliable multicast transport protocols are more likely to cause severe congestion problems than best-effort protocols [5]. So, achieving TCP-friendliness has become the target of the majority of reliable multicast congestion control research.

Up to this point there has been a lot of theoretical research in congestion control but much of this research is not applicable to many current applications. The primary reason for this is that many of these applications have requirements that have not been considered by the designers of existing schemes. For example, much research has focused specifically on single-sender applications based on the assumption that as long as each sender can act independently these schemes could be used effectively by multi-sender applications. However this is not always the case as multi-sender applications must scale in terms of the number of senders as well as the number of receivers. The reality is that multicast protocols are complex and much of this complexity is extended to the design of congestion control schemes. So, the fact that existing schemes are not appropriate for some applications in no way suggests design flaws in these schemes but rather shows that just like there is no "one-size-fits-all" reliable multicast protocol there will also not be a "one-size-fits-all" reliable multicast congestion control scheme.

Therefore, as part of the SIRAM project [6] at Luleå University of Technology we are exploring congestion control for specific use with multimedia collaborative workspaces with an emphasis on creating real world solutions and a highly deployable implementation. In the long term this demands the creation of dynamic schemes for all common media used by collaborative workspaces (whiteboard, audio, video and other) as well as effective bandwidth sharing mechanisms that allow the media to interact in a way that provides the best user experience. Because congestion control for reliable multicast is seen as a priority by standardizing bodies [5], we view the first step to be the implementation of a reliable multicast congestion control scheme for whiteboard traffic and similar reliable media. This paper describes both the requirements for such a scheme as well as an implementation based on those requirements.

Using a real-world approach has encouraged the utilization of a close existing partnership between the University and Marratech AB [7]; a Swedish software company that creates multicast based e-meetings products. The main benefit of this partnership is that it has enabled the design requirements and the general assumptions for the scheme to be based on Marratech's experience deploying IP-multicast-based applications in the general market. In addition, in order to demonstrate compatibility

with existing applications the prototype discussed in this paper has been added to the whiteboard that is part of Marratech's e-meetings product suite.

Although the scheme has been designed specifically to intertwine with Marratech's existing reliable multicast implementation this paper should still be of value to many developers of other interactive, time-critical applications that require delivery guarantees. This is because Marratech has implemented well-known and widely used protocols, (SRM and RTP) thus making the methods described applicable to applications of a similar structure.

The next section of the paper gives a discussion of the requirements, which were the result of placing real-world demands on the scheme. In section 3 we go on to discuss existing schemes in the context of these requirements and in section 4 we describe the prototype. A summary and future work then conclude the paper.

2 Requirements and general assumptions

In order to create a scheme that is useful for today's applications it is critical to keep a real-world perspective. For example, while the definition for TCP-friendliness given in the introduction is gaining general acceptance, the reality is that collaborative workspaces are sometimes run over private networks where the old static approach is attractive because it can give the session priority bandwidth usage over other traffic. In the near future this scenario is likely to be common as the availability of intra-domain multicast (i.e. between the customers of an Internet Service Provider (ISP)) increases while the availability of inter-domain multicast (i.e. from one ISP to another) is expected to remain scarce. Therefore, in order to handle both "friendly" and "non-friendly" deployments the scheme was designed with the intent of providing TCP-friendliness but also to allow for some non-friendly configuration of bandwidth usage. The following requirements and assumptions were used throughout the design process and were developed during meetings with Marratech in order to ensure the practical view necessary for use with real applications.

1. The scheme must not harm the ability of the group to communicate. This means that the scheme cannot destroy interactivity among users nor the reliability of data delivery.
2. All hosts are potential senders and many sessions may run within the same multicast domain. Therefore, special consideration is necessary for keeping the scheme as low-weight as possible both in the amount of bandwidth consumed and in the number of multicast addresses used.
3. The scheme must be end-to-end in nature because router assistance is not currently available.
4. The scheme must be compatible with the use of reflectors, Network Address Translators (NATs) and firewalls. This implies that unicast connections between all hosts in the session will not always be available.

2.1 Sender-Based or Receiver-Based

Because the scheme must preserve interactivity and remain low-weight a single-rate rather than a multi-rate scheme seems to be appropriate. Multi-rate congestion control can be attractive because it allows receivers to be more independent, and does not penalize faster receivers for operating in a session with a few slow receivers. This receiver independence is achieved by having the sender layer the data across several channels making each receiver responsible for subscribing to the channel(s) that have an aggregate send rate within its acceptable reception range [8].

However, when used in conjuncture with reliable media, layered congestion control schemes have high overhead and can make it difficult for receivers with dissimilar reception rates to communicate. The problem is that unlike in a best effort setting, reliable media cannot be layered in a way that allows increasing quality of reception with each layer. Each receiver must receive the entire data set, so redundant data must be passed in the layers [1]. This results in a different reception rate for each receiver rather than a different quality of reception for each receiver. The overhead created by this redundant data can be large and the different reception rates between receivers can create severe interactivity problems over the long term. For these reasons we have focused specifically on a single-rate rather than multi-rate scheme.

2.2 A potential conflict

The fourth requirement (pure multicast) is not intuitive and can be seen as inconsistent with keeping the scheme low-weight because unicast traffic is sometimes used to reduce overhead by keeping packets from reaching uninterested receivers [3]. However, pure multicast often becomes the only solution for multi-sender distributed applications due to the lack of a single access point (server) which can be opened up for all clients. The difficulty that pure multicast imposes on designers is that in order to keep the scheme low-weight control traffic must be kept to an absolute minimum.

One could make attempts to avoid this entirely because a unicast connection can be emulated perfectly by creating a multicast session with two members. However, this has its own drawbacks in that, depending on the type of unicast traffic desired it could potentially require one multicast address to be reserved for each sender-receiver pair, leading to the reservation of an unacceptable number of multicast addresses for each session. For n hosts the number of addresses needed would be:

$$n(n-1)/2 \ . \qquad \textbf{(1)}$$

It should be possible to reduce the number of addresses needed for many situations by creating some sort of address sharing mechanism. But the number of addresses needed should still be least n. This has the potential to significantly reduce the number of concurrent sessions that can run within a multicast domain as it might require each session to reserve addresses for the maximum allowable session size in order to guarantee available address space for each new host. In any case, as shown in section 4, by maximizing cooperation between the congestion control scheme and

the underlying delivery mechanism it is possible to create a pure multicast scheme that is lightweight enough to make such address hogging unnecessary.

3 Related Work

The available TCP send rate of a host is directly related to the way that TCP varies the number of packets sent per round trip time (rtt) based on loss-events [9]. This behavior can be summarized as incrementing the send-rate for each rtt where a loss-event does not occur, and cutting the send rate in half when a loss-event does occur.

In order to calculate the proper TCP-friendly send-rate for a multicast session, the sender must identify the slowest receiver, which can change over time, and obtain rtt and loss-event information about this receiver. Therefore, the primary design issues for single-rate schemes revolve around obtaining and processing feedback to make this possible. This must be done in a scalable way and must be robust when facing feedback suppression. In general, modern single-rate congestion control schemes follow the same basic architecture in that they identify the worst receiver in the session, and then calculate the send rate based on loss events by this receiver. Three schemes that follow this model are LE-SBCC [10], PGMCC [3] and TFMCC [4].

LE-SBCC is an extreme attempt at low-weight congestion control in that it attempts to provide TCP friendliness without creating any additional feedback and instead relies entirely on feedback provided by the recovery mechanism. The only assumption made by LE-SBCC regarding the nature of recovery feedback used is that time stamps are included in loss-indications (NACKs or ACKs) giving the sender the ability to calculate each receiver's rtt. The sender uses this rtt information along with the arrival times of loss-indications in order to identify loss-events by each receiver. It then adjusts its send rate in a similar way to TCP for each loss-event by the receiver with the highest loss-event rate.

The advantage of LE-SBCC is that it can be deployed with existing single-sender applications rather easily because it is completely source based, requiring only the multicast sender to be updated. Thus, it can be quite useful if updating the receivers presents logistical problems. However, the fact that it does not take measures to handle feedback suppression causes it to perform unreliably. It has clearly shown to act aggressively when NACKs from the worst-receiver are suppressed causing the sender to misidentify the worst-receiver and/or calculate its available data rate as too high. For this reason it is considered unpractical for general use.

A more complete single-rate reliable multicast congestion control scheme is PGMCC, which has demonstrated the robustness that LE-SBCC lacks. It includes two mechanisms for overcoming feedback suppression, one to help the sender correctly identify the worst-receiver and one to make sure it can identify loss-events by this receiver when facing feedback suppression.

The process of selecting the worst-receiver is aided by having each receiver include loss-rate as well as time stamp information in NACKs. The sender can then compare the available send rate of the receivers using the formula,

$$1/RTT\sqrt{r} \quad , \tag{2}$$

where r is a receiver's loss rate and RTT is its round trip time. The formula serves as a simple model of the additive increase and multiplicative decrease aspect of TCP and is adequate for comparative purposes. The advantage of have receivers include loss-rate information in NACKs is that feedback suppression can cause a delay in identifying the worst-receiver, but not long term problems, because only one NACK by the worst-receiver needs to reach the sender in order for it to be identified.

Once the worst-receiver (referred to as "the acker") is selected it is required to unicast acknowledgments of each packet received back to the sender. As only one receiver at a time can be the worst-receiver, this provides very robust TCP-like feedback without impacting the scalability of the application. However, if unicast connections are not available between the worst-receiver and sender then ACKs must be multicasted and the overhead incurred by the congestion control scheme becomes considerable. With n senders sending m packets each the number of unwanted acknowledgments received by each host is:

$$m(n-1) \quad . \tag{3}$$

Due to its effectiveness and simplicity PGMCC has been well received by the research community for use with one to many applications. However, in the context of the requirements discussed in section 2, giving such generous feedback to the sender can be viewed as undesirable for multi-sender applications.

TFMCC is a complex and well-designed scheme intended for best-effort media. One primary difference in creating a single-rate scheme for best-effort traffic is that the underlying recovery mechanisms used by reliable multicast protocols create feedback that congestion control schemes can take advantage of. Since this feedback is not available with best-effort traffic much of the design work in TFMCC focused on creating scaleable "extreme feedback" mechanisms for identifying the slowest receiver where only important receivers are required to pass feedback to the sender. This has shown to be effective while scaling to potentially thousands of receivers. TFMCC also introduced a scaleable way for "important" receivers to calculate their round-trip-times from the sender allowing receivers to report loss-event rates back to the sender. This allows the sender to select the worst-receiver based on more precise loss-event information than the estimations given with formula (2).

The feedback methods employed by TFMCC are clearly effective but could be considered overkill when used with reliable multicast if sufficient information for congestion control can be obtained from the underlying protocols. PGMCC demonstrates in part how to accomplish this by relying on information added to NACKs, which are used by many reliable multicast protocols, for selecting the worst-

receiver. If this idea is extended so that adequate loss-event feedback is also available from the recovery mechanism then very low-weight schemes should be possible.

4 Description of the Prototype

In this section we describe a scheme that was designed using a different approach than the schemes above in that it was specifically designed around the requirements given in section two, and for use with SRM or similar protocols that use random-timer based NACK suppression. Previous schemes do not make assumptions about how the underlying recovery mechanism operates and can be used with protocols of an entirely different nature, for example those based on router aggregation. The advantage in taking a streamlined approach is that by merging congestion control and other control data it is possible to reduce the overheard required for calculating the proper send-rate. This idea by itself is not new, but the closeness by which the scheme works with SRM-like suppression timers is a novel idea and allows the scheme to avoid expensive loss-event feedback.

4.1 Quick NACK Feedback from the Worst Receiver

SRM [1] provides a framework based on random timers, which has become the most widely used method for providing scalability in NACK-oriented protocols. Random timer feedback suppression provides scalability by effectively choosing at random the host to send a NACK when many hosts loose the same packet. This NACK suppression can make it difficult for the sender to identify loss-events by the worst receiver, as it must compete against all other hosts for the right to NACK.

This competition can be removed if the timer mechanism for the worst receiver is altered so that it acts like a “quick nacker”, sending a NACK as soon as it realizes it has lost a packet. This will result in a behavior identical to that as if the worst case receiver is always given a value of 0 for its random timer and will preserve scalability of the recovery mechanism because just like there is only one acker in PGMCC, there should also only be one quick nacker. This feedback method is advantageous because it will incur no overhead beyond that of SRM unless the NACK created by the worst-case receiver results in a redundancy that would not have occurred otherwise. This will occur on a per-packet basis if the following three conditions are met.

1. There is at least one receiver that looses the same packet as the worst receiver.
2. The random timer for one of these receivers expires before it receives the quick NACK from the worst receiver.
3. This NACK reaches the worst receiver before its “normal” random timer would have expired.

In practice this should be uncommon for two reasons. The first is that this situation becomes more likely when there are receivers behind the same bottleneck as the quick nacker that are much closer to the sender than the worst-receiver, which will not

happen in every topology. The second is that even if there are such receivers, the use of exponentially distributed timers [11] causes each of these receivers to likely have long rather than short random timer values, so the quick NACK will most often reach them in time to suppress their NACKs. Further more, the exponential distribution of random timers is now the norm because it has been shown that NACK suppression is improved when there are a few "early NACKers" to fulfill the repair needs of the remainder of the group. The quick nacker helps to fulfill this principle and in some situations will reduce NACK redundancies from receivers farther away from the sender than the worst receiver.

We leave the investigation of the “true” overhead of this feedback mechanism up to future work. For now we only mention that inevitably this will depend on the topology used, so we plan on resolving this by performing tests of our prototype deployed in practice over real networks.

4.2 Selection of the Worst-Receiver

The Marratech whiteboard runs in an RTP-based multimedia environment, which supplies the sender with rtt and loss-rate information about all receivers from RTP receiver reports. This information can be used with formula (2) in order for a sender to identify its worst receiver. The information provided by RTP receiver reports is used by the prototype because it does not require additional overhead in terms of bandwidth in order to identify the slowest receiver. The periodic nature of receiver reports will inevitably cause some delay in identifying a necessary switch in the worst receiver. However, we take the pragmatic stance that over the long term this will not affect the ability of the scheme to clear up congestion. In fact, in some situations this is beneficial because constant switching between receivers can cause single-rate schemes to perform unreliably and is something that needs to be avoided [3].

When a report arrives the sender identifies the current worst-receiver in the following way.

a. The first receiver sending a report becomes the initial worst-receiver and this receiver’s throughput is stored. Each time a report comes in from the worst receiver its throughput is updated.
b. If a receiver other than the current worst-receiver reports a throughput that is worse than this stored value it becomes the new worst receiver.

Because these receiver reports are multicasted to the group all hosts should be aware of the current worst-receiver. However, there could be a problem if some hosts loose these reports. This can be avoided by having the sender include the location of the current worst receiver in each data packet. A receiver then sends out quick loss-event feedback only if the last data packet it received identified it as the worst receiver.

An Alternative to Receiver Reports

Receiver reports are not used by every application and the above method of selecting the worst receiver will not be available in this case. In this situation it is possible to have receivers add rtt and loss-rate information to NACKS for use with formula (2), in similar fashion to PGMCC. This slightly increases the size of a NACK but has the advantage that it facilitates switches in the worst receiver as soon as NACKs arrive, rather than requiring the sender to wait for the next report that comes in.

Quick NACK feedback will cause any NACK-based selection process to naturally favor the current worst receiver over receivers behind the same bottleneck because it will suppress NACKs from other receivers that have a similar loss pattern. However, it will not interfere with the ability of the congestion control scheme to identify a new worst-receiver that is not behind the same bottleneck because in order to have a higher loss rate than the worst case receiver, this receiver will have to loose some packets that were not lost by the worst receiver, and these packets are unaffected by the quick NACK mechanism.

4.3 Configurablity

The main drawback of using single-rate congestion control is that single-rate schemes inevitably suffer from the "crying baby" problem where one receiver drastically reduces performance for the entire group. A commonly suggested solution is to set a bandwidth floor and force a host to leave the session if its reception rate falls below this limit [4]. This allows the session to remain TCP-friendly for the entire group without dropping bandwidth usage to unusable levels.

However, for group communication applications it can be necessary to take a softer stance where these receivers can stay in the session even though they will sometimes operate in a congested state. Congested receivers will have difficulty communicating with the rest of the group but this may be considered "the lesser of two evils" as they will still be able to participate in some aspects of the session. In order to keep one receiver from pushing bandwidth usage to unacceptable levels for the rest of the group we have included a mechanism to allow the owner of the session to configure bandwidth usage so that there is a floor and ceiling on the send rate. When this is enabled the mechanism will calculate the rate in normal fashion but the actual send thread will not respond to requests for a rate beyond these limits.

This is realistically just a compromise between the old static approach, which some users of collaborative workspaces prefer, and the new dynamic approach, and will obviously cause problems if configured incorrectly. However, this will allow at least some reaction by the application to congestion, which will be an improvement. We admit at this point that we have taken the simplest solution possible to deal with this problem and in the future more sophisticated and statistical methods of dealing with the crying baby may become a new research topic.

4.4 Initial Testing

Initial testing of the prototype has focused on demonstrating effective bandwidth sharing with TCP on a congested link as well as some simple tests to determine if the application could correctly identify the worst receiver. We have also conducted some preliminary tests with multiple senders in a session in order to demonstrate that effective congestion control for multi-sender applications can be obtained by having each sender react independently.

TCP flows were created using a common windows application (the simple cut and paste of a file onto a remote host) and Dummynet [12] was used as a bandwidth limiter along with EtherPeek [13] as a packet sniffer in order to monitor actual use of network bandwidth by the applications under precisely controlled conditions. Figures 1 – 3 each contain two graphs representing bandwidth usage by two competing flows with a 400 Kb/s bottleneck with a 50-slot buffer as well as 50ms delay time placed on the outgoing link.

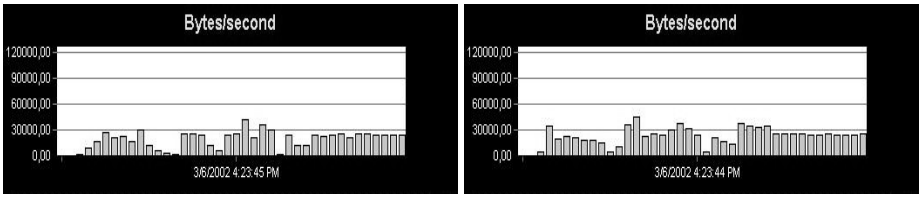

Fig. 1. Bandwidth usage by two competing TCP flows. Each bar represents the mean bandwidth usage of a flow over a 5 second time period

The first figure is included as a reference point and shows two TCP flows in competition. Each "bar" shows the average bandwidth consumption of a flow and as expected both of the flows were able to transfer data without being denied a portion of the bandwidth.

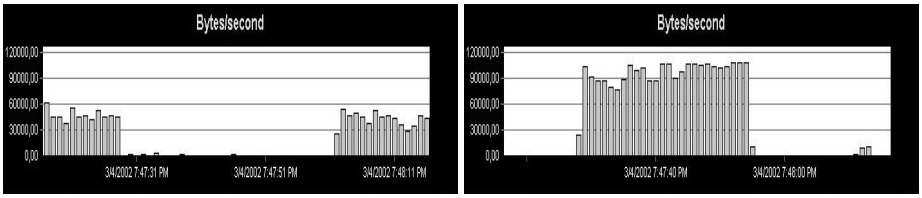

Fig. 2. A TCP transfer (*left graph*) being interrupted by an overaggressive whiteboard transfer (*right graph*). Each bar represents the mean bandwidth usage of a flow over a 1 second time period

However, this is not the case with Figure 2 which shows a severe example of a TCP flow being "dropped to zero" by an overaggressive whiteboard transfer using UDP with a static bandwidth limit that is higher than the available bandwidth. The graphs

demonstrate that once the whiteboard transfer starts, the TCP flow is forced to delay all transfer of data until the whiteboard transfer is finished.

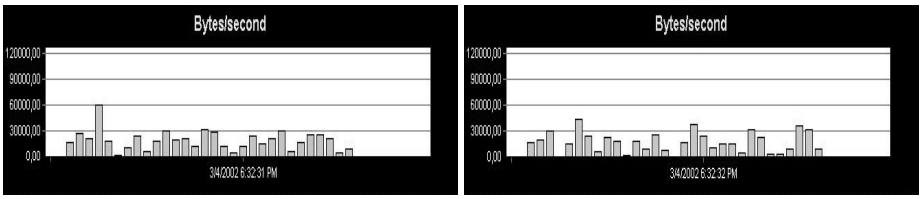

Fig. 3. The graphs show the concurrent transfer of an identical file by TCP (*right graph*) and the whiteboard with congestion control (*left graph*). Each bar represents the mean bandwidth usage of a flow over a 5 second interval.

Figure 3 shows how with congestion control enabled, the whiteboard and a TCP application can coexist while neither of the flows is dropped to zero. As long as each sender uses effective end-to-end congestion control then this idea can be extended for use with multi-sender applications.

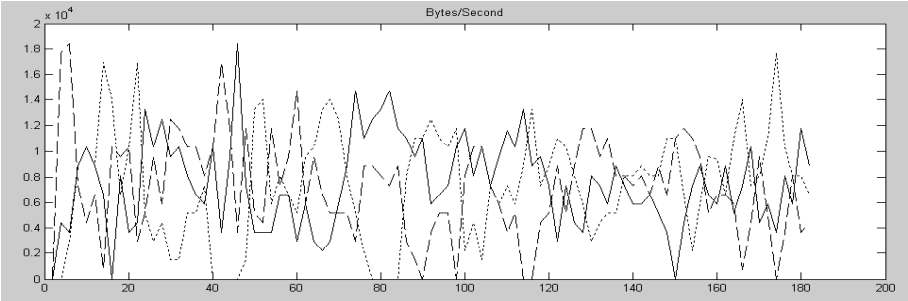

Fig. 4. A plot of bandwidth usage between 3 senders in a whiteboard session. Each host transferred an identical 1.05 MB file and a 200 Kb/s, 50-slot bottleneck with 100 ms of delay was placed between the senders. The first sender (*dashed line*) averaged a throughput of 7301 bytes/second. The second sender (*solid line*) started its transfer 96 ms after the first sender and had an average bandwidth consumption of 7619 bytes/second. The third sender (*dotted line*) started its transfer 1.07 seconds after the second sender and had an average bandwidth consumption of 7485 bytes/second.

A brief demonstration of this behavior is given in figure 4, which shows three members of a congestion control enabled whiteboard session transferring an identical file. While controlling their send-rates' independently, the difference between their bandwidth usages was less than 5% for the life of the transfer (app. 180 seconds).

4.5 Quality of Feedback

While initial tests have shown a flow using the scheme to compete fairly with a TCP flow they also demonstrate one drawback of using NACK rather than ACK based

feedback in congestion control. Schemes using NACK based feedback react slower to congestion and cannot take advantage of many of the sophisticated congestion avoidance mechanisms used by TCP. A glimpse into the effects of this is given by figures 1 and 3. Due to the use of congestion avoidance the two TCP flows in figure 1 seem to "level out" after a certain period of time and have very similar send rates over the 5 second time scale thereafter and this behavior in not observed by the flows in figure 3. However, because effective TCP-friendly congestion control can be achieved without using ACK feedback and while reacting much slower to congestion than TCP [9] this does not keep the scheme from achieving its goal of TCP-friendliness.

5 Summary and Future Work

We have discussed why current single-rate congestion control schemes have trouble fulfilling industry provided requirements for multiple sender applications. In particular, these schemes either do not handle feedback suppression well, or take such a liberal approach towards providing feedback that they can reduce the number of effective senders that can participate in a session. We have explored a possible solution for reducing this feedback that alters timer-based NACK suppression used by SRM-like protocols so that the slowest receiver sends immediate NACKs. We have implemented a prototype based on this idea into an existing commercial whiteboard that runs in an RTP and SRM environment and have conducted initial testing of this prototype.

In the future we will continue our work towards creating a complete rate control scheme for collaborative workspaces but will shift our focus towards best effort media. In particular, layered schemes for audio/video need to be implemented and effective methods for bandwidth management need to be developed that can delegate available resources to the separate media in the optimal way. We will also conduct further analysis of the "true" overhead of quick NACK feedback over real networks, as this was the essential feature that enabled for low-weight feedback. This could lead to an attempt at standardizing a quick NACK based feedback mechanism with the IETF, as this is the only part of existing schemes that conflicts with our design requirements.

References

1. S. Floyd, V. Jacobson, C. Liu, S. McCanne: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. IEEE/ACM Transactions on Networking. December 1997
2. S. Floyd, K. Fall, Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, August 1999.
3. L. Rizzo: PGMCC: a TCP-Friendly single rate multicast congestion control scheme. ACM SIGCOMM 2000.

4. J. Widmer, M. Handley. Extending Equation-based Congestion Control to Multicast Applications. ACM SIGCOMM 2001.
5. A. Manking, A. Romanow, S. Bradner, V. Paxon. RFC2357: IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols. The Internet Society (1998)
6. The SIRAM project, http://www.cdt.luth.se/projects/siram
7. Marratech AB, http://www.marratech.com
8. S. McCanne, V. Jacobson, M. Vetterli. Receiver-driven Layered Multicast. ACM SIGCOMM 1996.
9. S. Floyd, M. Handley, J. Padhye, J. Widmer: Equation Based Congestion Control for Unicast Applications. ACM SIGCOMM 2000.
10. P. Thapliyal, Sidhartha, J. Li, S. Kalyanaraman. LE-SBCC: Loss Event Oriented Source-based Multicast Congestion Control. Multimedia Tools and Applications, 2002, to appear. Available from: http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html
11. J. Nonnenmacher, E. Biersack: Optimal Multicast Feedback. Technical report, Institute EURECOM, BP 193, 06904 Sophia Antipolis cedex, FRANCE, July 1998
12. L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. ACM Computer Communication Review, Vol. 27, N.1, Jan 1997.
13. EtherPeek http://www.wildpackets.com/products/etherpeek