Research Report No. 2007:01

Congestion and Error Control in Overlay Networks

Doru Constantinescu, David Erman, Dragos Ilie, and
Adrian Popescu

Department of Telecommunication Systems,
School of Engineering,
Blekinge Institute of Technology,
S–371 79 Karlskrona, Sweden

Blekinge Institute of Technology
Research Report No. 2007:01
ISSN 1103-1581
Published 2007.
Printed by Kaserntryckeriet AB.
Karlskrona 2007, Sweden.

This publication was typeset using LaTeX.

Abstract

In recent years, Internet has known an unprecedented growth, which, in turn, has lead to an increased demand for real-time and multimedia applications that have high Quality-of-Service (QoS) demands. This evolution lead to difficult challenges for the Internet Service Providers (ISPs) to provide good QoS for their clients as well as for the ability to provide differentiated service subscriptions for those clients who are willing to pay more for value added services.

Furthermore, a tremendous development of several types of overlay networks have recently emerged in the Internet. Overlay networks can be viewed as networks operating at an inter-domain level. The overlay hosts learn of each other and form loosely-coupled peer relationships.

The major advantage of overlay networks is their ability to establish subsidiary topologies on top of the underlying network infrastructure acting as brokers between an application and the required network connectivity. Moreover, new services that cannot be implemented (or are not yet supported) in the existing network infrastructure are much easier to deploy in overlay networks.

In this context, multicast overlay services have become a feasible solution for applications and services that need (or benefit from) multicast-based functionality. Nevertheless, multicast overlay networks need to address several issues related to efficient and scalable congestion control schemes to attain a widespread deployment and acceptance from both end-users and various service providers.

This report aims at presenting an overview and taxonomy of current solutions proposed that provide congestion control in overlay multicast environments. The report describes several protocols and algorithms that are able to offer a reliable communication paradigm in unicast, multicast as well as multicast overlay environments. Further, several error control techniques and mechanisms operating in these environments are also presented.

In addition, this report forms the basis for further research work on reliable and QoS-aware multicast overlay networks. The research work is part of a bigger research project, "Routing in Overlay Networks (ROVER)". The ROVER project was granted in 2006 by EuroNGI Network of Excellence (NoE) to the Dept. of Telecommunication Systems at Blekinge Institute of Technology (BTH).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, the Internet has experienced an unprecedented growth, which, in turn, has lead to an increase in the demand of several real-time and multimedia applications that have high Quality of Service (QoS) demands. Moreover, the Internet has evolved into the main platform of global communications infrastructure and Internet Protocol (IP) networks are practically the primary transport medium for both telephony and other various multimedia applications.

This evolution poses great challenges among Internet Service Providers (ISPs) to provide good QoS for their clients as well as the ability to offer differentiated service subscriptions for those clients who are willing to pay more for higher grade services. Thus, an increased number of ISPs are rapidly extending their network infrastructures and resources to handle emerging applications and a growing number of users. However, in order to enhance the performance of an operational network, traffic engineering (TE) must be employed both at the traffic and the resource level.

Performance optimization of an IP network is accomplished by routing the network traffic in an optimal way. To achieve this, TE mechanisms may use several strategies for optimizing network performance, such as: load-balancing, fast re-routing, constraint-based routing, multipath routing, etc. Several solutions are already implemented by ISPs and backbone operators for attaining QoS-enabled networks. For instance, common implementations include the use of Virtual Circuits (VCs) as well as solutions based on Multi Protocol Label Switching (MPLS). Thus, the provisioning of the QoS guarantees are accommodated mainly through the exploitation of the *connection-oriented* paradigm.

Additionally, a tremendous development of several types of *overlay networks* have emerged in the Internet. The idea of overlay networks is not new. Internet itself began as a data network overlaid on the public switched telephone network and even today, a large number of users connect to Internet via modem. In essence, an overlay network is any network running on top of another network, such IP over Asynchronous Transfer Mode (ATM) or IP over Frame Relay. In this report however, the term will refer to application networks running on top of the IP-based Internet.

IP overlay networks can be viewed as networks operating at inter-domain level. The overlay nodes learn of each other and form loosely-coupled peer relationships. Routing algorithms operating at the overlay layer may take advantage of the underlying physical network and try to accommodate their performance to different asymmetries that are inherent in packet-switched IP networks such as the Internet, e.g., available link bandwidth, link connectivity and available resources at a network node (e.g., processing capability, buffer space and long-term storage capabilities).

1.2 Motivation

The major advantage of overlay networks is their ability to establish subsidiary topologies on top of the underlying network infrastructure and to act as brokers between an application and the

required network connectivity. Moreover, new services that cannot be implemented (or are not yet supported) in the existing network infrastructure are easier to realize in overlay networks, as the existing physical infrastructure does not need modification.

In this context, IP multicast has not yet experienced a large-scale deployment although it is able to provide (conceptually) efficient group communication and at the same time maintain an efficient utilization of the available bandwidth [22]. Besides difficulties related to security issues [35], special support from network devices and management problems faced by IP multicast, one problem that still need to be addressed is an efficient multicast Congestion Control (CC) scheme.

Consequently, multicast overlay services have become a feasible solution for applications and services that need (or benefit from) multicast-based functionality. Nevertheless, multicast overlay networks also need to address the same issues related to efficient and scalable CC schemes to attain a widespread deployment and acceptance from both end-users and various service providers.

This report aims at providing an overview and taxonomy of different solutions proposed so far that provide CC in overlay multicast environments. Furthermore, this report will form the base for further research work on overlay networks carried out by the ROVER research team at the Dept. of Telecommunication Systems at the School of Engineering at Blekinge Institute of Technology (BTH).

1.3 Report Outline

The report is organized as follows. Chapter 2 provides an overview of congestion and error control protocols and mechanisms used in IP unicast environments. Chapter 3 gives a brief introduction to IP multicast concepts and protocols together with several solutions proposed that concern congestion and error control for such environments. Following the discussion on IP multicast, Chapter 4 presents congestion and error control schemes and algorithms operating at the application layer in multicast overlay environments. Finally, the report is concluded in Chapter 5 where some guidelines for further research are also presented.

Chapter 2

Congestion and Error Control in Unicast Environments

2.1 Introduction

The dominant network service model in today's Internet is the *best-effort* model. The essential characteristic of this model is that all packets are treated the same way, i.e., without any discrimination but also without any delivery guarantees. Consequently, the best-effort model does not allow users to obtain a better service (if such demand arises) in spite of the fact that they may be willing to pay more for a better service.

Much effort has been put into extending the current Internet architecture to provide QoS guarantees to an increasing assortment of network-based applications. Therefore, two main QoS architectural approaches have been defined: *i)* Integrated Services (IntServ)/Differentiated Services (DiffServ) enabled networks, i.e., Resource ReSerVations (RSVs) and per-flow state implemented in the routers, edge policies, provisioning and traffic prioritization (forwarding classes). *ii)* Overprovisioning of network resources, i.e., providing excess bandwidth thus providing conditions for meeting most QoS concerns.

Both approaches have their own advantages and disadvantages but it is often argued that the best effort model is good enough as it will accommodate for many QoS requirements if appropriate provisioning is provided. However, in many cases, service differentiation is still preferable. For instance, when concentrated overload situations occur into sections of the network (e.g., a Web server that provides highly popular content), the routers must often employ some types of differentiation mechanisms. This rises from the fact that, generally, there are not enough network resources available to accommodate all users.

Furthermore, network resources (in terms of, e.g., available bandwidth, processing capability, available buffer space) are limited and when these requirements are close or exceed the capacity of the available resources, *congestion* occurs. Consequently, network congestion may lead to higher packet loss rates, increased packet delays and even to a total network breakdown as a result of *congestion collapse*, i.e., an extended period of time when there is no useful communication within the congested network.

This chapter provides a short introduction to CC and error control schemes employed in unicast environments. The main focus is on the behavior of Transport Control Protocol (TCP) as it incorporates the desired properties of most CC mechanisms and algorithms considered later in this report. CC schemes for unicast transmissions are presented based on the characteristic mechanism employed by the particular scheme, e.g., window-based CC, rate-based CC or layer-based CC. Further, several available solutions for congestion and error control are also described.

2.2 Congestion Control Mechanisms

A simple definition of network congestion can be as follows:

Definition 2.1. *Congestion is a fundamental communication problem that occurs in shared networks when the network users collectively demand more resources (e.g., buffer space, available bandwidth, service time of input/output queues) than the network is able to offer.* □

Typical for packet-switched networks, the packets transit the input/output buffers and queues of the network devices in their way toward the destination. Moreover, these networks are characterized by the fact that packets often arrive in "bursts". The buffers in the network devices are intended to assimilate these traffic bursts until they can be processed. Nevertheless, the available buffers in network nodes may fill up rapidly if network traffic is too high, which in turn may lead to discarded packets. This situation cannot be avoided by increasing the size of the buffers, since unreasonable buffer size will lead to excessive end-to-end (e2e) delay.

A typical scenario for congestion occurs where multiple incoming links feed into a single outgoing link (e.g., several Local Area Networks (LANs) links are connected to a Wide Area Network (WAN) link). The core routers of the backbone networks are also highly susceptible for traffic congestion because they often are under-dimensioned for the amount of traffic they are required to handle [67]. Moreover, IP networks are particularly vulnerable to congestion due to their inherent *connectionless* character. In these networks, variable sized packets can be inserted into the network by any host at any time making thus traffic prediction and provision of guaranteed services very difficult. Therefore, mechanisms for managing and controlling network congestion are necessary. These mechanisms refer to techniques that can either prevent or remove congestion.

CC mechanisms should allow network devices to detect when congestion occurs and to restrain the ongoing transmission rate in order to mitigate the congestion. Several techniques, often conceptually related, that address CC are as follows:

- Host-based: when the sender reduces the transmission rate to avoid overflowing the receiver's buffers.
- Network: the goal is to reduce the congestion in the network and not in the receiver.
- Congestion avoidance: the routers on a transmission path provide feedback information to the senders that the network is (or is about to become) congested so that the senders reduce their transmission rate.
- Resource ReSerVation: scheduling the use of available physical and other network resources such as to avoid congestion.

Furthermore, based on *when* the CC mechanisms operate, they can be divided into two main categories: *open-loop* CC (i.e., prevention of congestion) and *closed-loop* CC (i.e., recovery from congestion). A brief description of these mechanisms is as follows [31]:

a) *Open-Loop* – congestion prevention

- Retransmission policy – a good retransmission policy is able to prevent congestion. However, the policy and the retransmission timers must be designed to optimize efficiency.
- Acknowledgment (ACK) policy – imposed by the receiver in order to slow down the sender.
- Discard policy – implemented in routers. It may prevent congestion while preserving the integrity of the transmission.

b) *Closed-Loop* – congestion recovery

- Back-pressure – a router informs the upstream router to reduce the transmission rate of the outgoing packets.
- Choke point – a specific *choke point* packet sent by a router to the source to inform about congestion.
- Implicit signaling – a source can detect an implicit warning signal and slow down the transmission rate (e.g., delayed ACK).

- Explicit signaling – routers send explicit signals (e.g., setting a bit in a packet) to inform the sender or the receiver of congestion.

Another important concept related to CC is that of *fairness*, i.e., when the offered traffic must be reduced in order to avoid network congestion, it is important to do it fairly. Especially in best-effort networks fairness is of major importance as there are no service guarantees or admission control mechanisms. In IP networking, fairness is conceptually related to CC and is defined as *max-min fairness*. Max-min fairness can be briefly described as follows:

1. Resources are allocated in increasing order of demand.
2. A user is never allocated a higher share than its demand.
3. Users with unsatisfied demands are allocated equal shares from the remaining unallocated resources.

In other words, all users initially get the same resource share as the user with the smallest demand. The users with unsatisfied demands equally share the remaining resources. However, fairness does not imply equal distribution of resources among users with unsatisfied demands. Thus, several policies may be employed such as *weighted max-min fairness* (i.e., users are given different weights in resource sharing) or the *proportional fairness* (introduced by Kelly [40]) through the use of logarithmic utility functions (i.e., short flows are preferred to long flows).

Based upon how a particular CC mechanism is implemented, three main categories can be defined:

a) *Window-based* – congestion is controlled through the use of buffers (windows) both at sender and receiver.

b) *Rate-based* – the sender adapts the transmission rate based on the available resources at the receivers.

c) *Layer-based* – in the case of unicast transmissions, we look at CC from a Data Link Layer (DLL)-layer perspective since the mechanisms acting at DLL are often adapted for congestion and error control at higher layers.

The following sections will present the operation of these mechanisms as well as several available implementations for respective CC scheme.

2.2.1 Window-based Mechanisms

The tremendous growth of the Internet both in size and in the number of users generated one of the most demanding challenges, namely how to provide a fair and efficient allocation of available network resources. The predominant transport layer protocol used in today's Internet is TCP [63]. TCP is primarily used by applications that need reliable, in-sequence delivery of packets from a source to a destination. A central element in TCP is the dynamic window flow control proposed by Van Jacobson [38].

Currently, most Internet connections use TCP, which employs the *window-based flow control*. Flow control in TCP is done by implementing a *sliding-window* mechanism. The size of the sliding window controls the number of bytes (segments) that are in transit, i.e., transmitted but not yet acknowledged. The edges of TCP's sliding-window mechanism can be increased from both sides, i.e., the window slides from the right-hand side when a byte is sent and it slides from the left-hand side when an ACK is received. Thus, the maximum number of bytes awaiting an ACK is solely determined by the window size. The window size is dynamically adjusted according to the available buffer space in the receiving TCP buffer.

For the purpose of flow control, the sending TCP maintains an *advertised window* (`awnd`) to keep track of the current window. The `awnd` prevents buffer overflow at the receiver according to

the available buffer space. However, this does not address buffer overflow in intermediate routers in case of network congestion. Therefore, TCP's CC mechanism employs a *congestion window* (`cwnd`) by following an Additive Increase Multiplicative Decrease (AIMD) policy to implement its CC mechanism. The idea behind this is that if somehow a sender could learn of the available buffer space in the bottleneck router along the e2e TCP path, then it could easily adjust its `cwnd` thus preventing buffer overflows both in the network and at the receiver.

The problem however is that routers do not operate at the TCP layer and consequently cannot use the TCP ACK segments to adjust the window. The circumvention of the problem is achieved only if TCP assumes network congestion whenever a retransmission timer expires and reacts in this way to network congestion by adapting the `cwnd` to the new network conditions. Hence, the `cwnd` adaptation follows the AIMD scheme, which is based on three distinct phases:

i) *Slow-Start* with exponential increase.

ii) *Congestion avoidance* with additive (linear) increase.

iii) *Congestion recovery* with multiplicative decrease.

The AIMD policy regulates the number of packets (or bytes) that are sent at one time. The graph of AIMD resembles a sawtooth pattern where the number of packets increases (additive increase phase) until congestion occurs and then drops off when packets are being discarded (multiplicative decrease phase).

Slow-Start (Exponential Increase)

One of the algorithms used in TCP's CC is *slow-start*. The slow-start mechanism is based on the principle that the size of `cwnd` starts with one Maximum Segment Size (MSS) and it increases "slowly" when new ACKs arrive. This has the effect of probing the available buffer space in the network. In slow-start, the size of the `cwnd` increases with one MSS each time a TCP segment is ACK-ed as illustrated in Figure 2.1(a). First, TCP transmits one segment (`cwnd` is one MSS). After receiving the ACK for this segment, after a Round Trip Time (RTT), it sends two segments, i.e., `cwnd` is incremented to two MSSs. When the two transmitted segments are ACK-ed, `cwnd` is incremented to four and TCP sends four new segments and so on.

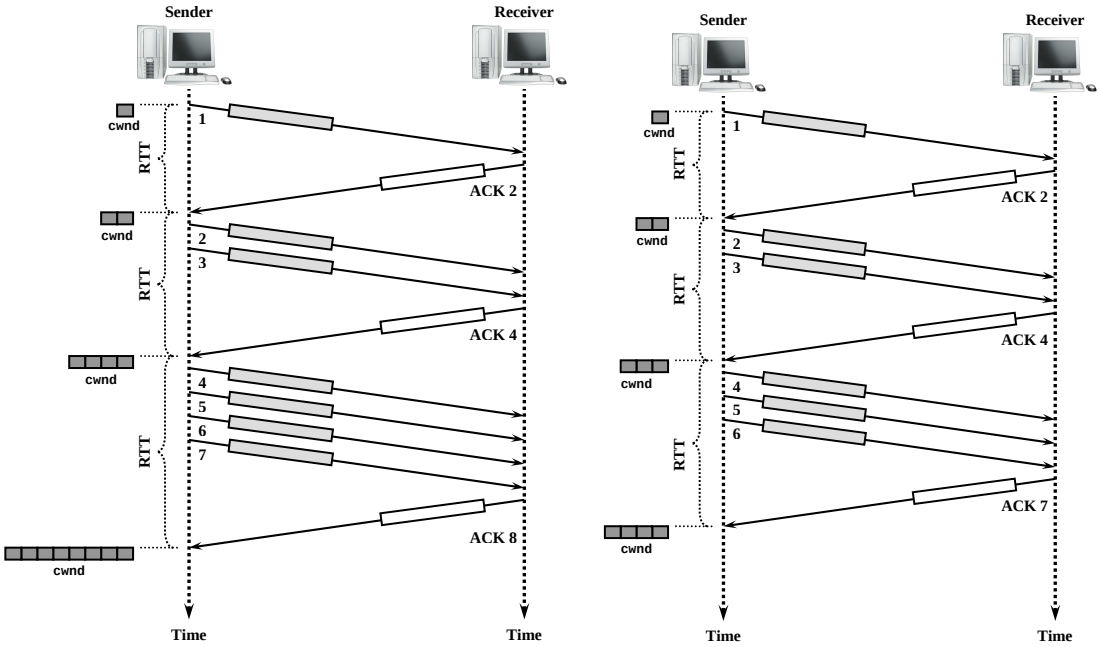
As the name implies, this algorithm starts slowly, but increases exponentially. However, slow-start does not continue indefinitely. Hence, the sender makes use of a variable called the *slow-start threshold* (`ssthresh`) and when the size of `cwnd` reaches this threshold, slow-start stops and the TCP's CC mechanism enters the next phase. The size of `ssthresh` is initialized to 65535 bytes [77]. It must be also mentioned that the slow-start algorithm is essential in avoiding the congestion collapse problem [38].

Congestion Avoidance (Additive Increase)

In order to slow down the exponential growth of the size of `cwnd` and thus avoid congestion before it occurs, TCP implements the *congestion avoidance* algorithm, which limits the growth to follow a linear pattern. When the size of `cwnd` reaches `ssthresh`, the slow-start phase stops and the *additive* phase begins. The linear increase is achieved by incrementing `cwnd` by one MSS when the *whole* window of segments is ACK-ed. This is done by increasing `cwnd` by 1/`cwnd` each time an ACK is received. Hence, the `cwnd` is increased by one MSS for each RTT. This algorithm is illustrated in Figure 2.1(b). It is easily observed from the figure that `cwnd` is increased linearly when the whole window of transmitted segments is ACK-ed for each RTT.

Congestion Recovery (Multiplicative Decrease)

In the occurrence of congestion, `cwnd` must be decreased in order to avoid further network congestion and ultimately congestion collapse. A sending TCP can only guess that congestion has occurred if it needs to retransmit a segment. This situation may arise in two cases: *i)* either the



(a) Slow-Start with Exponential Increase.

(b) Congestion Avoidance with Additive Increase.

Figure 2.1: *TCP Congestion Control Algorithms.*

Retransmission TimeOut (RTO) timer has expired or *ii)* three duplicate ACKs are received and in both these cases the size of threshold variable `ssthresh` is set to half of the current `cwnd`. The algorithm that controls the `ssthresh` variable is called *multiplicative decrease*. Hence, if there are consecutive RTOs this algorithm reduces the TCP's sending rate exponentially.

Further, most TCP implementations react in two ways, depending on what caused the retransmission of a segment, i.e., if it was caused by an RTO or by the reception of three duplicate ACKs. Consequently:

1. If RTO occurs: TCP assumes that the probability of congestion is high – the segment has been discarded in the network and there is no information about the other transiting segments. TCP reacts aggressively:

   - `ssthresh` = `cwnd`/2.
   - `cwnd` = 1 MSS.
   - initiates slow-start phase.

2. If three duplicate ACKs are received: TCP assumes that the probability of congestion is lower – a segment may have been discarded but other segments arrived at the destination (the duplicate ACKs). In this case TCP reacts less aggressively:

   - `ssthresh` = `cwnd`/2.
   - `cwnd` = `ssthresh`.
   - initiates congestion avoidance phase.

The *additive increase* of the `cwnd` described in the previous section and the *multiplicative decrease* of `ssthresh` described here is generally referred to as the AIMD algorithm of TCP.

2.2.2 Adaptive Window Flow Control: Analytic Approach

As mentioned above, TCP uses a dynamic strategy that changes the window size depending upon the estimated congestion on the network. The main idea behind this algorithm is to increase the window size until buffer overflow occurs. Buffer overflow is detected when the destination does not receive packets. In this case, it informs the source which, in turn, sets the window to a smaller value. When no packet loss occurs, the window is increased exponentially (slow-start) and after reaching the slow-start threshold, the window is increased linearly (congestion avoidance). Packet losses are detected either by RTOs or by receiving duplicate ACKs.

This simplified case study aims at illustrating Jacobson's algorithm in a very simple case: a single TCP source accessing a single link [45, 76]. It must be emphasized that this case study is not our work. However, we decided to include it due to its highly illustrative analytical explanation of the behavior of TCP. The interested reader is referred to [45, 76].

Several simplified assumptions are used for this example. Assume c as the link capacity measured in packets/second with $1/c$ being the service time of each packet. The source is sending all data units equal to the MSS available for this link. The link uses a First In First Out (FIFO) queueing strategy and the link's total available buffer size is B. Let τ denote the round-trip propagation delay of each packet and $T = \tau + 1/c$ denotes the RTT as the sum of the propagation delay and the service time. Furthermore, the product cT is the *bandwidth-delay* product. The normalized buffer size β available at the link, with B measured in MSSs, is given by [45, 76]:

$$\beta = \frac{B}{c\tau + 1} = \frac{B}{cT} \tag{2.1}$$

For the purpose of this example, it is assumed that $\beta \leq 1$ which implies $B \leq cT$. The maximum window size that can be accommodated by this link and using (2.1) is given by:

$$W_{max} = cT + B = c\tau + 1 + B \tag{2.2}$$

The buffer is always fully occupied and the packets still in transit are given by cT. The packets are processed at rate c. Consequently, ACKs are generated at the destination also at rate c and new packets can be injected by the source every $1/c$ seconds. The number of packets in the buffer is B. By using (2.2) it is concluded that the total number of unacknowledged packets without leading to a buffer overflow is equal to W_{max}.

When a packet loss does occur the current window size is slightly larger than W_{max} and this depends both on c and RTT. When loss occurs, `ssthresh` is set to half of the current window size. The size of `ssthresh` is assumed to be:

$$W_{thresh} = \frac{W_{max}}{2} = \frac{cT + B}{2} \tag{2.3}$$

Considering the slow-start phase, the evolution of the `cwnd` size and queue length is described in Table 2.1. Here, a *mini-cycle* refers to the duration of a RTT equal to T, i.e., the time it takes for `cwnd` to double its size.

In Table 2.1 the i^{th} mini-cycle applies to the time interval $[i, (i+1)T]$. The ACK for a packet transmitted in mini-cycle i is received in mini-cycle $(i+1)$ and increases `cwnd` by one MSS. Furthermore, ACKs for consecutive packets released in mini-cycle i arrive in intervals corresponding the service time, (i.e., $1/c$). Consequently, two more packets are transmitted for each received ACK thus leading to a queue buildup. This is valid only if $\beta < 1$ so that the `cwnd` during slow-start is less than cT and the queue empties by the end of the mini-cycle.

In conformity with Table 2.1 it is observed that, if we denote `cwnd` at time t by $W(t)$, the following equation describes the behavior of $W(t)$ during $(n+1)^{\text{th}}$ mini-cycle:

$$W\left(nT + \frac{m}{c}\right) = 2^n + m + 1, \qquad 0 \leq m \leq 2^n - 1 \tag{2.4}$$

Similarly, by denoting the queue length at time t with $Q(t)$, then the behavior of $Q(t)$ during $(n+1)^{\text{th}}$ mini-cycle is described by:

Table 2.1: *Evolution during Slow-Start phase.*

Time	**Packet ACK-ed**	`cwnd` **size**	**Packet(s) released**	**Queue length**
		mini-cycle 0		
0		1	1	1
		mini-cycle 1		
T	1	2	2, 3	2
		mini-cycle 2		
$2T$	2	3	4, 5	2
$2T + 1/c$	3	4	6, 7	2 - 1 + 2 = 3
		mini-cycle 3		
$3T$	4	5	8, 9	2
$3T + 1/c$	5	6	10, 11	2 - 1 + 2 = 3
$3T + 2/c$	6	7	12, 13	2 - 1 + 2 - 1 + 2 = 4
$3T + 3/c$	7	8	14, 15	2 - 1 + 2 - 1 + 2 - 1 + 2 = 5
		mini-cycle 4		
$4T$	8	9	16, 17	2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

$$Q\left(nT + \frac{m}{c}\right) = m + 2, \qquad 0 \le m \le 2^n - 1 \tag{2.5}$$

Moreover, maximum window size and maximum queue size during mini-cycle $(n+1)^{\text{th}}$ satisfy:

$$\begin{aligned} W_{max} &= 2^{n+1} \\ Q_{max} &= 2^n + 1 \end{aligned} \tag{2.6}$$

It is observed from (2.6) that

$$Q_{max} \approx \frac{W_{max}}{2}. \tag{2.7}$$

Considering the situation when buffer overflow occurs in the slow-start phase, and given that the available buffer size is B, then the condition for no overflow is given by:

$$Q_{max} \le B \tag{2.8}$$

However, buffer overflow occurs in the slow-start phase when the value of `cwnd` exceeds the `ssthresh`. Hence, by using (2.3) and (2.7) we obtain:

$$Q_{max} \le W_{thresh} = \frac{W_{max}/2}{2} = \frac{W_{max}}{4} = \frac{cT + B}{4} \tag{2.9}$$

Consequently, the sufficient condition for no buffer overflow during the slow-start phase is:

$$\frac{cT + B}{4} \le B \quad \equiv \quad B \ge \frac{cT}{3} \tag{2.10}$$

where $\equiv$ denotes *equivalent to*. Accordingly, two cases are possible during the slow-start phase. If $B > cT/3$ no buffer overflow will occur while if $B < cT/3$ overflow does occur since in this case Q_{max} exceeds the value of B. The two cases are considered separately. Consequently:

1. *No buffer overflow*: $B > cT/3$.

   In this case only one slow-start phase takes place and it ends when $W_{thresh} = W_{max}/2$. The duration of this phase is approximated by a simplified version of (2.4), namely $W(t) \approx 2^{t/T}$. Thus, the duration t_{ss} of the slow-start phase is given by:

$$2^{t_{ss}/T} = \frac{W_{thresh}}{2} = \frac{cT+B}{2} \tag{2.11}$$

$$t_{ss} = T\log_2\left(\frac{cT+B}{2}\right) \tag{2.12}$$

The number of packets transmitted during the slow-start phase is approximated by the `cwnd` size at the end of this period, i.e., W_{thresh}. This approximation is valid since `cwnd` increases during this phase by one MSS with each received ACK starting with an initial value of one. Hence, the number of packets n_{ss} is:

$$n_{ss} = W_{thresh} = \frac{cT+B}{2} \tag{2.13}$$

2. *Buffer overflow*: $B < cT/3$.

This case generates two slow-start phases. We denote, in a similar fashion with the previous case, t_{ss1}, n_{ss1}, t_{ss2} and n_{ss2} the duration and the number of transmitted packets during the two slow-start phases. Hence, in the first slow-start phase with $W_{thresh} = W_{max}/2$, buffer overflow occurs when $Q(t) > B$, and with reference to (2.7), it is concluded that the first overflow situation occurs at approximately $2B$. Thus, t_{ss1} is given by the duration needed to reach this window size (see (2.12)) plus an extra RTT time duration that is necessary for the detection of the loss:

$$t_{ss1} = T\log_2\left(\frac{cT+B}{2}\right) + T \tag{2.14}$$

With the same argument as above, n_{ss1} is given by:

$$n_{ss1} = W_{thresh} = \frac{cT+B}{2} \approx 2B \tag{2.15}$$

The buffer overflow in the first slow-start phase is detected only in the next mini-cycle and during each mini-cycle the window size is doubled. Accordingly, the buffer overflow can be shown to be detected, using a more careful analysis than the scope of this exemplification, when window size is approximately $W^* \approx \min[2W_{max}-2, W_{thresh}] = \min[4B-2, (cT+B)/2]$. Hence, the second slow-start phase $\tilde{W}_{thresh}$ starts at:

$$\tilde{W}_{thresh} = W^*/2 = \min\left[W_{max}-1, \frac{W_{thresh}}{2}\right] = \min\left[2B-1, \frac{cT+B}{4}\right] \tag{2.16}$$

Thus, t_{ss2} is given by:

$$t_{ss2} = T\log_2\tilde{W}_{thresh} = T\log_2\min\left[2B-1, \frac{cT+B}{4}\right] \tag{2.17}$$

and n_{ss2} is given by:

$$n_{ss2} = \min\left[2B-1, \frac{cT+B}{4}\right] \tag{2.18}$$

Hence, the total duration of the entire slow-start phase, t_{ss}, and the total number of packets transmitted during this phase, n_{ss}, are given by:

$$t_{ss} = t_{ss1} + t_{ss2} \tag{2.19}$$

$$n_{ss} = n_{ss1} + n_{ss2} \tag{2.20}$$

In order to converge this analysis we look at the congestion avoidance phase. It is assumed that the starting window for the congestion avoidance is W_{ca} and the congestion avoidance phase will end once W_{ca} reaches W_{max}. Moreover, W_{ca} is equal to the slow-start threshold from the preceding slow-start phase. Hence, using (2.3) and (2.16) we have:

$$W_{ca} = \begin{cases} W_{max}/2 & if \quad B > cT/3 \\ \min[2B-1, (cT+B)/4] & if \quad B < cT/3 \end{cases} \tag{2.21}$$

As opposed to the slow-start phase, where the window size grows exponentially, in the congestion avoidance phase the window size growth is linear and thus better suited for a continuous-time approximation for the window increase. Consequently, a differential equation will be used to describe the growth of the window in the congestion avoidance phase.

Let $a(t)$ be the number of ACKs received by the source after t units of time in the congestion avoidance phase. Further, let $[\mathrm{d}W/\mathrm{d}t]$ be the growth rate of the congestion avoidance window with time, $[\mathrm{d}W/\mathrm{d}a]$ the congestion avoidance window's growth rate with arriving ACKs and $[\mathrm{d}a/\mathrm{d}t]$ the rate of the arriving ACKs. We can then express $[\mathrm{d}W/\mathrm{d}t]$ as:

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \frac{\mathrm{d}W}{\mathrm{d}a}\frac{\mathrm{d}a}{\mathrm{d}t} \tag{2.22}$$

Given that the size of the congestion avoidance window is large enough so that the link is fully utilized, then $[\mathrm{d}a/\mathrm{d}t] = c$. Otherwise $[\mathrm{d}a/\mathrm{d}t] = W/T$ and consequently:

$$\frac{\mathrm{d}a}{\mathrm{d}t} = \min\left[\frac{W}{T}, c\right] \tag{2.23}$$

Moreover, during the congestion avoidance phase, the window size is increased by $1/W$ for each received ACK. Thus

$$\frac{\mathrm{d}W}{\mathrm{d}a} = \frac{1}{W} \tag{2.24}$$

By using (2.23) and (2.24) we obtain:

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \begin{cases} 1/T & if \quad W \leq cT \\ c/W & if \quad W > cT \end{cases} \tag{2.25}$$

As stated in (2.25) the congestion avoidance phase is comprised of two sub-phases that correspond to $W \leq cT$ and $W > cT$, respectively.

1. $W \leq cT$

   During this phase the congestion avoidance window grows as t/T and the duration for this period of growth is given by:

   $$t_{ca1} = T(cT - W_{ca}) \tag{2.26}$$

   since the initial window size is W_{ca} (see (2.21) and, for $\beta < 1, W_{ca} \leq W_{max}/2$ is always less

than cT). The number of packets transmitted during this phase is:

$$\begin{aligned} n_{ca1} &= \int_0^{t_{ca1}} a(t)\mathrm{d}t \\ &= \int_0^{t_{ca1}} \frac{W(t)}{T}\mathrm{d}t \\ &= \int_0^{t_{ca1}} \frac{W_{ca}+t/T}{T}\mathrm{d}t \\ &= \frac{W_{ca}t_{ca1}+t_{ca1}^2/(2T)}{T} \end{aligned} \tag{2.27}$$

2. $W > cT$

   From (2.25) W^2 grows as $2ct$. Hence, for $t \geq t_{ca1}$, $W^2(t) = 2c(t-t_{ca1})+(cT)^2$. This growth period and the cycle ends with buffer overflow when the congestion avoidance window size exceeds W_{max}. The duration for this sub-phase is given by:

$$t_{ca2} = \frac{W_{max}^2-(cT)^2}{2c} \tag{2.28}$$

   The total number of packets transmitted during this sub-phase is

$$n_{ca2} = ct_{ca2} \tag{2.29}$$

   as the link is fully utilized during this period.

In a similar manner as for the slow-start phase, the total duration of the congestion avoidance phase, t_{ca}, and the total number of packets transmitted during this phase, n_{ca}, are given by:

$$t_{ca} = t_{ca1}+t_{ca2} \tag{2.30}$$

$$n_{ca} = n_{ca1}+n_{ca2} \tag{2.31}$$

At this point, we are able to compute the TCP throughput by using (2.19), (2.20), (2.30) and (2.31).

$$\text{TCP throughput} = \frac{n_{ss}+n_{ca}}{t_{ss}+t_{ca}}. \tag{2.32}$$

2.2.3 Rate-based Mechanisms

The TCP's CC protocol depends upon several reliability mechanisms (windows, timeouts, and ACKs) for achieving an effective and robust CC [38]. However, this may result in unfairness and insufficient control over queueing delays in routers due to TCP's dependence on packet loss for congestion detection. This behavior results in that TCP uses buffer resources leading thus to large queues. A solution to reduce queueing delays is in this case to discard packets at intermediate routers forcing therefore TCP to reduce the transmission rate and releasing valuable network resources.

Nevertheless, simple drop schemes such as *drop-tail* may result in burst dropping of packets from all participating TCP connections causing a simultaneous timeout. This may further lead to the underutilization of the link and to global synchronization of multiple TCP sessions due the halving of the `cwnd` for all active TCP connections [26].

However, any analysis of network congestion must also consider the queueing because most network devices contain buffers that are managed by several queueing techniques. Naturally, properly managed queues can minimize the number of discarded packets and implicitly minimize network congestion as well as improve the overall network performance. One of the basic techniques is the FIFO queueing discipline, i.e., packets are processed in the same order in which they arrive at the queue. Furthermore, different priorities may be applied on queues resulting so in a priority queueing scheme, i.e., multiple queues with different priorities in which the packets with the highest priority are served first. Moreover, of crucial importance is to assign different flows to their own queues thus differentiating the flows and facilitating the assignment of priorities. Further, the separation of flows ensure that each queue contains packets from a single source, facilitating in this way the use of a CC scheme.

In addition, window-based flow control does not always perform well in the case of high-speed WANs because the bandwidth-delay products are rather large in these networks. Consequently, this necessitates large window sizes. Additionally, another fundamental reason is that windows do not successfully regulate e2e packet delays and they are unable to guarantee a minimum data rate [8]. Hence, several applications that require a maximum delay and a minimum data rate (e.g., voice, video) in transmission do not perform well in these conditions.

Another approach to CC is the *rate-based flow control* mechanism. Congestion avoidance rate-based flow control techniques are often closely related to Active Queue Management (AQM). AQM is proposed in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2309 and has several advantages [11]:

- Better handling of packet bursts. By allowing the routers to maintain the average queue size small and to actively manage the queues will enhance the router's capability to assimilate packet bursts without discarding excessive packets.
- AQM avoids the "global synchronization problem". Furthermore, TCP handles a single discarded packet better than several discarded packets.
- Large queues often translate into large delay. AQM allows queues to be smaller, which improves throughput.
- AQM avoids *lock-outs*. Tail-drop queuing policies often allow only a few connections to control the available queueing space as a result of synchronization effects or other timing issues (they "lock-out" other connections). The use of AQM mechanisms can easily prevent the lock-out behavior.

However, the queueing management techniques (either simple ones such as drop-tail or active ones such as Random Early Detection (RED)) must address two fundamental issues when using rate-based flow control [30, 8]:

1. *Delay–Throughput trade-off*: Increasing the throughput by allowing too high session rates often leads to buffer overflow and increased delay. Delays occur in the form of retransmission and timeout delays. Large delays have as a consequence lower throughput on a per-source basis. This implies wasted resources for the dropped packets as well as additional resources consumed for the retransmission of these packets.
2. *Fairness*: If session rates need to be reduced in order to serve new clients, this must be done in a fair manner such that the minimum rate required by the already participating sessions is maintained.

Thus, rate-based techniques should reduce the packet discard rate without losing control over congestion and offer better fairness properties and control over queueing delays as well. Hence, network-based solutions hold an advantage over e2e solutions. Accordingly, the IETF proposed several improvements to TCP/IP-based control both at the transport and network layers. We continue this report by presenting a few interesting solutions.

Random Early Detection

The Random Early Detection (RED) AQM technique was designed to break the synchronization among TCP flows, mainly through the use of statistical methods for uncorrelated early packet dropping (i.e., before the queue becomes full) [26, 11]. Consequently, by dropping packets in this way a source slows down the transmission rate to both keep the queue steady and to reduce the number of packets that would be dropped due to queue overflow.

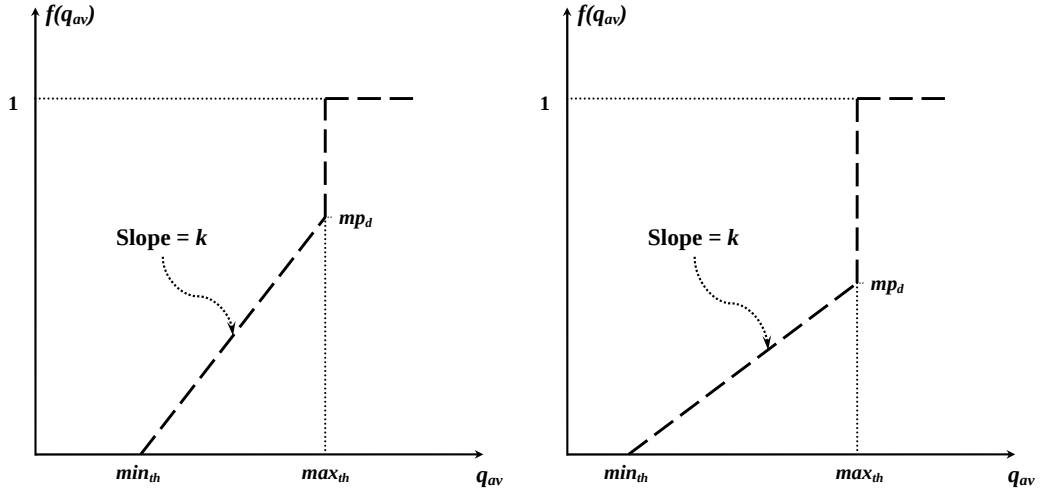
RED makes two major decisions: *i)* when to drop packets, and *ii)* what packets to drop by "marking" or dropping packets with a certain probability that depends on the queue length. For this, RED keeps track of the average queue size and discards packets when the average queue size grows beyond a predefined threshold. Two variables are used for this: *minimum threshold* and *maximum threshold*. These two thresholds regulate the traffic discarding behavior of RED, i.e., no packet drops if traffic is bellow the minimum threshold, selective dropping if traffic is between the minimum and the maximum threshold, and all traffic is discarded if the traffic exceeds the maximum threshold.

RED uses an exponentially-averaged estimate of the queue length and uses this estimate to determine the marking probability. Consequently, a queue managed by the RED mechanism does not react aggressively to sudden traffic bursts, i.e., as long as the average queue length is small RED keeps the traffic dropping probability low. However, if the average queue length is large, RED assumes congestion and starts dropping packets at a higher rate [26].

If we denote q_{av} as being the average queue length, the marking probability in RED is given by [76]:

$$f(q_{av}) = \begin{cases} 0, & if \quad q_{av} \leq min_{th} \\ k(q_{av} - min_{th}), & if \quad min_{th} < q_{av} \leq max_{th} \\ 1, & if \quad q_{av} > max_{th} \end{cases} \tag{2.33}$$

where k is a constant and min_{th} and max_{th} are the minimum and maximum thresholds, respectively, such as the marking probability is equal to 0 if q_{av} is below min_{th} and is equal to 1 if q_{av} is above max_{th}. The RED marking probability is illustrated in Figure 2.2. The constant k depends on min_{th}, max_{th} and the *mark probability denominator* (mp_d) which represents the fraction of packets dropped when $q_{av} = max_{th}$, e.g., when mp_d is 1024, one out of every 1024 packets is dropped if $q_{av} = max_{th}$. The influence of k and mp_d on the behavior of RED's marking probability is illustrated in Figures 2.2(a) and 2.2(b).



(a) RED: Standard Service. (b) RED: Premium Service.

Figure 2.2: *RED Marking Probability.*

The performance of RED is highly dependent on the choice of min_{th}, max_{th} and mp_d. Hence, the min_{th} should be set high enough to maximize link utilization. Meantime, the difference $max_{th} - min_{th}$ must be large enough to avoid global synchronization. If the difference is too small, many packets may be dropped at once, resulting in global synchronization. Further, the exponential weighted moving average of the queue length is given by [76]:

$$q_{av}(t+1) = \left(1 - \frac{1}{w_q}\right) q_{av}(t) + \frac{1}{w_q} q(t) \tag{2.34}$$

where $q(t)$ is the queue length at time t and w_q is the queue weight. RFC 2309 indicates that AQM mechanisms in Internet may produce significant performance advantages and there are no known drawbacks from using RED [11].

Several flavors of RED were later proposed to improve the performance of RED and we only mention some of them. Dynamic RED (D-RED) [4] aims at keeping the queue size around a threshold value by means of a controller that adapts the marking probability as a function of the mean distance of the queue from the specific threshold. Adaptive RED [24] regulates the marking probability based on the past history of the queue size. Weighted RED (W-RED) is a Cisco solution that uses a technique of marking packets based on traffic priority (IP precedence). Finally, Stabilized RED (S-RED) [57] utilizes a marking probability based both on the evaluated number of active flows and the instant queue size.

Explicit Congestion Notification

As mentioned before, congestion is indicated by packet losses as a result of buffer overflow or packet drops as a result of AQM techniques such as RED. In order to reduce or even eliminate packet losses and the inefficiency caused by the retransmission of these packets a more efficient technique have been proposed for congestion indication, namely Explicit Congestion Notification (ECN) [65].

The idea behind ECN is for a router to set a specific bit (*congestion experienced*) in the packet header of ECN-enabled hosts in case of congestion detection (e.g., by using RED). When the destination receives this packet with the ECN bit set, it will inform the source about congestion via the ACK packet. This specific ACK packet is also known as an *ECN-Echo*. When the source receives the ECN-Echo (explicit congestion signal) it then halves the transmission rate, i.e., the response of the source to the ECN bit is equivalent to a single packet loss. Moreover, ECN-capable TCP responds to explicit congestion indications (e.g., packet loss or ECNs) at most once per `cwnd`, i.e., roughly at most once per RTT. Hence, the problem of reacting multiple times to congestion indications within a single RTT (e.g., TCP-Reno) is avoided. It must be noted that ECN is an e2e congestion avoidance mechanism and it also requires modification of the TCP standard implementation, i.e., it uses the last two bits in the `RESERVED`-field of the TCP-header [65].

The major advantage of the ECN mechanism is that it disconnects congestion indications from packet losses. ECN's explicit indication eliminates any uncertainty regarding the cause of a packet loss. ECN develops further the concept of congestion avoidance and it improves network performance. However, the most critical issue with ECN is the need of cooperation between both routers and end systems thus making the practical deployment more difficult.

Network Block Transfer

NETwork BLock Transfer (NETBLT) [18] is a protocol operating at the transport level and it is designed for the fast transfer of large bulks of data between end hosts. NETBLT proposes a reliable and flow controlled transfer solution, and it is designed to provide highest throughput over several types of underlying networks including IP-based networks.

The NETBLT bulk data transfer operates as follows [18, 19]: First, a connection is established between the two NETBLT enabled hosts. In NETBLT hosts can be either *passive* or *active*, where the active host is the host that initiates the connection. During the connection setup, both hosts agree upon the buffer size used for the transfer. The sending application fills the buffer with data and sends it to the NETBLT layer for transmission. Data is divided into packets according to

the maximum allowed size required by the underlying network technology and it is transmitted. The receiver buffers all packets belonging to a bulk transfer and checks if the packets are received correctly.

NETBLT uses Selective ACKs (SACKs) to provide as much information as possible to the sending NETBLT. Consequently, in NETBLT the sender and the receiver synchronize their state either if the transfer of a buffer is successful or if the receiver determines that information is missing from a buffer. Thus, a single SACK message can either confirm the successful reception of all packets contained in a particular buffer or it can notify the sender precisely what packets to retransmit.

When the entire buffer is received correctly, the receiving NETBLT sends the data to the receiving application and the cycle is repeated until all information in the session has been transmitted. Once the bulk data transfer is complete, the sender notifies the receiver and the connection is closed. An illustration for NETBLT is provided in Figure 2.3.

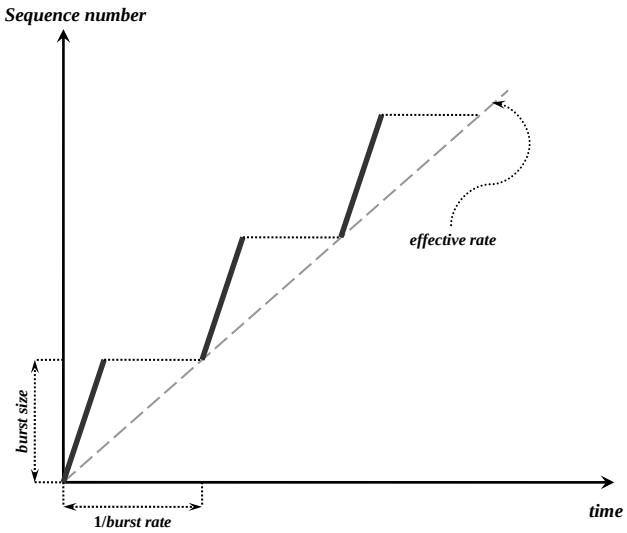


Figure 2.3: *NETBLT Operation.*

An important challenge in NETBLT is how to select the optimum buffer size. Hence, buffers should be as large as possible to improve the performance of NETBLT by minimizing the number of buffer transfers. Furthermore, the maximum size of the NETBLT depends upon the hardware architecture of the NETBLT-enabled hosts.

In NETBLT, a new buffer transfer cannot take place until the preceding buffer is transmitted. However, this can be avoided if multiple buffers are used, allowing thus for several simultaneous buffer transfers and improving the throughput and the performance of NETBLT. The data packets in NETBLT are of the same size except for the last packet. They are called `DATA` packets while the last packet is known as `LDATA` packet. The reason is the need of the receiving NETBLT to identify the last packet in a buffer transfer.

Flow control in NETBLT makes use of two strategies, one internal and one at the client level [18]. Because both the sending and the receiving NETBLT use buffers for data transmission, the client flow control operates at buffer level. Hence, either NETBLT client is able to control the data flow through buffer provisioning. Furthermore, when a NETBLT client starts the transfer of a given buffer it cannot stop the transmission once it is in progress. This may cause several problems, for instance, if the sender is transmitting data faster than the receiver can process it, buffers will be overflowed and packets will be discarded. Moreover, if an intermediate node on the transfer path is slow or congested it may also discard packets. This causes severe problems to NETBLT since the NETBLT buffers are typically quite large.

This problem is solved in NETBLT through the negotiation of the transmission rate at con-

nection setup. Hence, the transfer rate is negotiated as the amount of packets to be transmitted during a given time interval. NETBLT's rate control mechanisms consists of two parts: *burst size* and *burst rate*. The average transmission time per packet is given by [18]:

$$average\ transmission\ time\ per\ packet = \frac{burst\ size}{burst\ rate} \tag{2.35}$$

In NETBLT each flow control parameter (i.e., packet size, buffer size, burst size, and burst rate) is negotiated during the connection setup. Furthermore, the burst size and the burst rate can be renegotiated after buffer transmission allowing thus for adjusting to the performance observed from the previous transfer and adapting to the real network conditions.

TCP Rate Control

TCP rate control is a rate-based technique in which end systems can directly and explicitly adapt their transmissions rate based on feedback from specialized network devices that perform rate control. One of the available commercial products is *PacketShaper* manufactured by Packeteer [58].

The idea behind Packeteer's PacketShaper is that the TCP rate can be controlled by controlling the flow of ACKs. Hence, PacketShaper maintains per-state flow information about individual TCP connections. PacketShaper has access to the TCP headers, which allows it to send feedback via the ACK-stream back to the source, controlling thus the behavior while remaining transparent to both end systems and to routers. The main focus lies on controlling the bursts of packet by smoothing the rate of the transmission of the source and ease in this way traffic management [58]. Generally, most network devices that enforce traffic management and QoS implement some form of TCP rate control mechanism.

2.2.4 Layer-based Mechanisms

Another approach to CC mechanisms is to look at them from the DLL perspective, i.e., a layer 2 perspective on CC. However, unlike the transport layer discussed above, which operates both between end systems and between node-by-node, the layer 2 approach is functional only point-to-point. Hence, in order to avoid being over-explicit, all communication paradigms discussed in this section are assumed to occur at the DLL between two directly connected stations.

For instance, when looking at connection-oriented networks, a *session* is defined as the period of time between a *call set-up* and a *call tear-down*. Therefore, the admission control mechanisms in connection oriented-networks are essentially CC mechanisms when looking at a session. If the admission of a new session (e.g., a new telephone call) degrades the QoS of other sessions already admitted in the network, then the new session should be rejected and it can be considered as another form of CC. Further, when the call is admitted into the network, the network must ensure that the resources required by the new call are also met.

However, in contrast to connection-oriented networks, an inherent property of the packet-switched networks is the possibility that packets belonging to any session might be discarded or may arrive out of order at the destination. Thus, if we look at a session level, in order to provide reliable communication, we must somehow provide the means to identify successive packets in a session. This is predominantly done by numbering them modulo 2^k for some k, i.e., providing a k-bit sequence number. Thereafter, the sequence number is placed in the packet header and enables the reordering or retransmission of lost packets [8].

The DLL conventionally provides two services: *i)* Connectionless services (best-effort), and *ii)* Connection-oriented services (reliable). A connectionless service makes the best effort that the frames sent from the source arrive at the destination. Consequently, the receiver checks if the frames are damaged, i.e., performs error detection, and discards all erroneous frames. Furthermore, the receiver does not demand retransmission of the faulty frames and it is not aware of any missing frames. Hence, the correct sequence of frames is not guaranteed.

A connectionless service does not perform flow control, i.e., if the input buffers of the receiver are full, all incoming frames are discarded. Nevertheless, a connectionless service is simple and

has a very small overhead. This type of service keeps a minimal traffic across the serial link (no retransmissions of damaged frames or out-of order frames). Thus, connectionless services are best suited for communication links that have small error rates, e.g., LANs, Integrated Services Digital Network (ISDN) and ATM. In this case, the correction of errors can be performed at higher protocol layers [8].

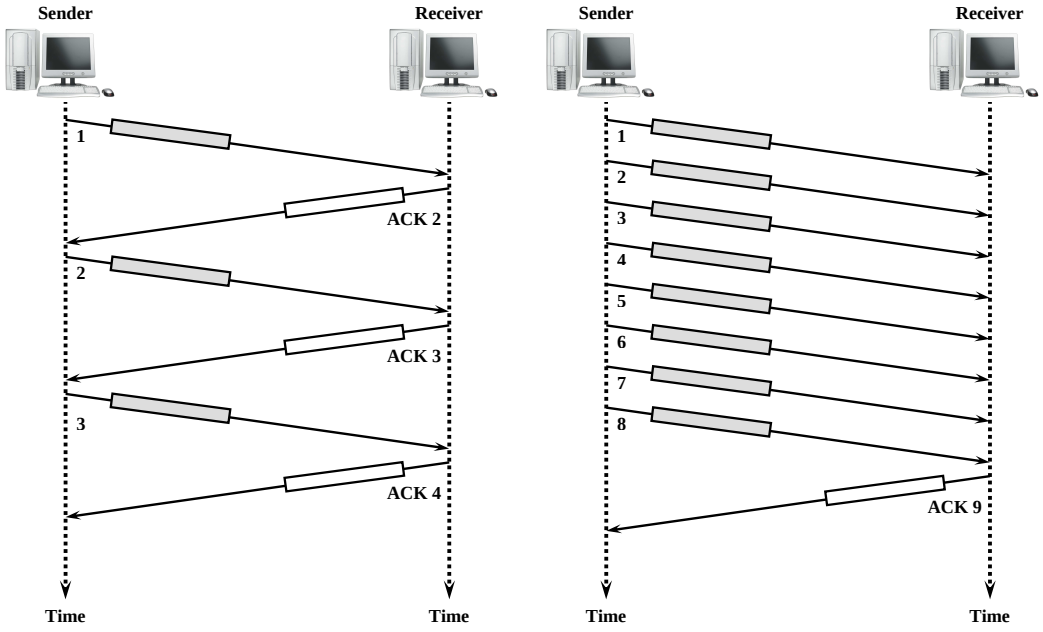
In contrast, connection-oriented services perform error control and error checking. The receiver requests retransmission of damaged or missing frames as well as of frames that are out of sequence. Connection-oriented services also perform flow control, which guarantees that a receiver's input buffer does not overflow. The connection-oriented protocols guarantee that frames arrive at a destination in proper sequence with no missing frames or duplicate frames, regardless of the Bit Error Rate (BER) of the communication link.

Flow Control

The crucial requirement for transmission of data from a sender to a receiver is that, regardless of the processing capability of the sender and the receiver and the available bit rate at the communication link, the buffers at the receiver side must not overflow. Data Link Control (DLC) achieves this through the flow control mechanism. There are two approaches for doing flow control:

1. *Stop-and-Wait* flow control.
2. *Sliding-Window* flow control.

In the Stop-and-Wait flow control the sender transmits a frame and waits for ACK (Figure 2.4(a)). Upon the receipt of the ACK it then sends the next frame. This is a simple mechanism that works well for a few frames. However, if frames are too big, they must be fragmented. In the case of transmission of multiple frames, the stop-and-wait flow control becomes highly inefficient.



(a) Stop-and-Wait. (b) Sliding-Window.

Figure 2.4: *Flow Control Approaches.*

There are several reasons for not having too large frames. An important role is played by the available buffer size, i.e., both the hardware and software resources available in a node are limited.

Further, longer frames exhibit a higher probability of being corrupted during their transmission, i.e., there are more bits that may get corrupted due to the inherent BER on the particular transmission links. Moreover, long frames are more likely to completely monopolize the underlying communication link.

The Sliding-Window mechanisms is based on the idea of pipelining, i.e., several frames can be transmitted without waiting for ACK (Figure 2.4(b)). A single frame is often used to acknowledge several other, i.e., *cumulative* ACKs are used. Furthermore, in order to control the number of frames that can be sent and received, a pair of sliding windows are used by both the sender and the receiver as illustrated by Figure 2.5.

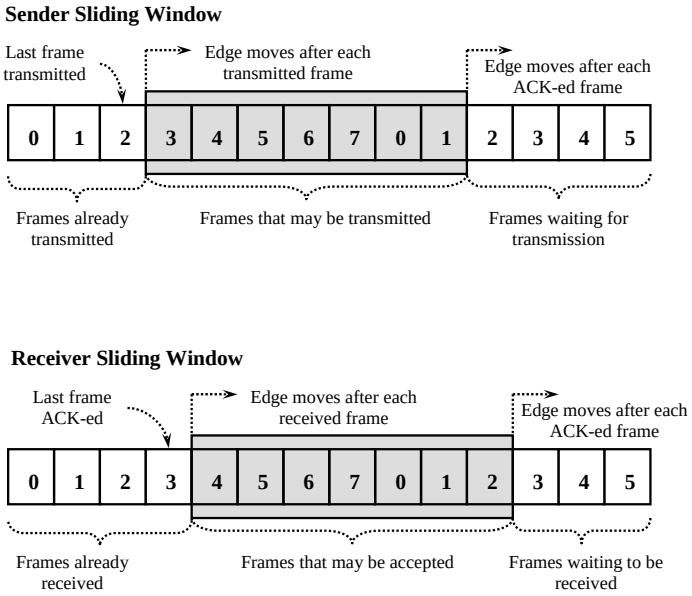


Figure 2.5: *Sliding-Window Flow Control.*

Consequently, flow control is a mechanism whose main goal is to adapt the transmission rate of the sender to that of the receiver according to current network conditions. Hence, flow control ensures that data transmission attains a high enough rate to guarantee good performance, and also protects the network or the receiving host against buffer overflows.

There are several distinctions between flow control and CC mechanisms. Flow control aims at preventing the sender from transmitting too much data such that the receiver gets overflowed. The sliding-window protocol achieves this fairly easy, i.e., ensures that the sender's window is not larger than the available buffer space at the receiver side, so that the receiver is not overflowed.

On the other hand, CC tries to prevent the sender from sending data that ends up by being discarded at an intermediate router on the transmission path. Consequently, CC mechanisms are more complex since packets originating from different sources may converge on the same queue in a network router. Thus, CC attempts to adjust the transmission rate to the available network transmission rate. Hence, CC aims at sharing the network with other flows in order to not overflow the network.

2.2.5 TCP Friendliness

TCP is the most dominant transport protocol in the Internet today. Hence, one important paradigm for CC is the concept of *TCP-friendliness*. Non-TCP traffic flows are considered to be TCP-friendly if the flow behaves in such a way that it achieves a throughput similar to the throughput obtained by a TCP flow under the same conditions, i.e., with the same RTT and the same loss/error rate. The notion of TCP-friendliness was formally introduced by Mahdavi and

Floyd [51].

The TCP CC mechanism depends on the cooperation from all participating users to achieve its goals and hence a non-TCP flow must adapt the throughput T according to [51]:

$$T = \frac{C \cdot MTU}{RTT \cdot \sqrt{loss}} \tag{2.36}$$

C is a constant, MTU is the maximum size of the packets transmitted, RTT is the round trip time and $loss$ is the event loss rate perceived by the connection. Accordingly, a non-TCP flow must measure the Maximum Transmission Unit (MTU), the loss rate and the RTT. MTU can be obtained by using an MTU discovery algorithm while loss rate and RTT can be continuously monitored by maintaining current averages over several RTTs.

The TCP-friendly paradigm can be summarized as follows [51]: If there is no congestion experienced in the network, the non-TCP flow may transmit at its preferred rate. Meanwhile, the non-TCP flow should monitor the overall packet loss and as long as the loss rate is low enough such that a corresponding TCP flow achieves the same throughput, the non-TCP flow may maintain its transmission rate. If the loss rate increases such that a corresponding TCP flow is not able to achieve the same throughput, then the non-TCP flow must reduce the transmission rate by half. If the loss rate still increases, the non-TCP flow must further reduce the transmission rate. Increases in transmission rates are allowed only after monitoring the loss rate for at least one RTT.

The TCP-friendly paradigm depends on the collaboration of all the users. This may not be a valid assumption given the current size of the Internet. Further, TCP-friendliness requires that all applications adopt the same congestion control behavior given in (2.36). This cannot be assumed to be valid for the wide range of emerging new applications being deployed today.

2.3 Error Control Mechanisms

TCP provides a reliable transport protocol to the upper layers, i.e., TCP is responsible for delivering a stream of data to the requesting application. The delivered data is in order, without error and without any segment lost or duplicated. Reliability in TCP is obtained by using error control mechanisms, which may include techniques for detecting corrupted, lost, out-of-order or duplicated segments. Error control in TCP is achieved by using a version of the Automatic Repeat reQuest (ARQ) error control protocols operating at DLL and it involves both error detection and retransmission of lost or corrupted segments. The following subsections provide a brief description of the common ARQ protocols.

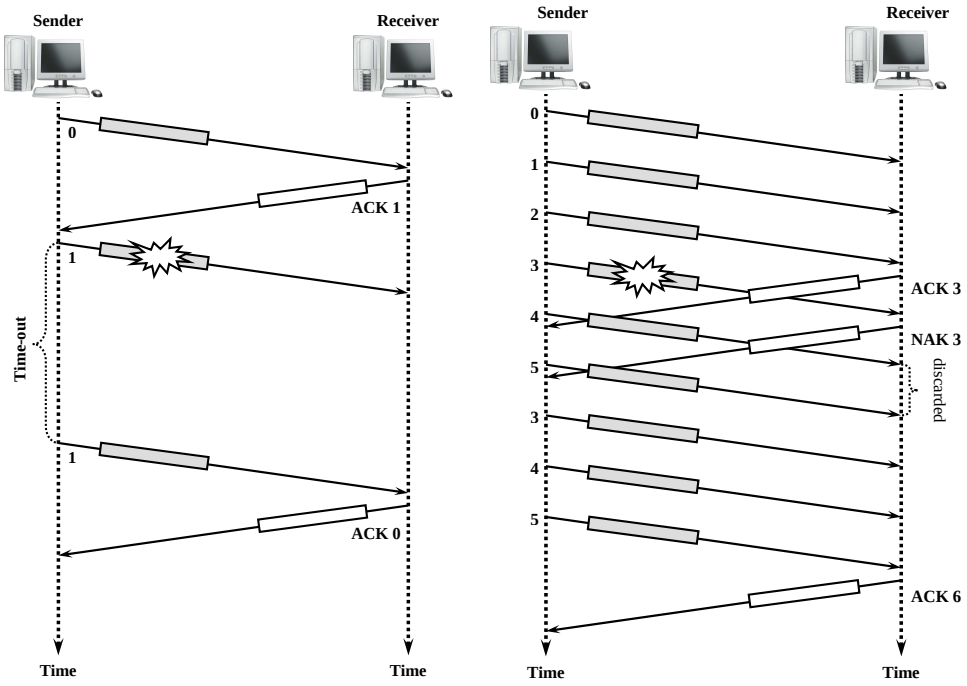
2.3.1 Stop-and-Wait ARQ

The Stop-and-Wait ARQ error control mechanism is based on the Stop-and-Wait flow control. In its simplest form, the sender transmits a frame and it waits for ACK. Until the successful receipt of the ACK, no other data frames are transmitted. The sender must also keep a copy of the transmitted frame until it receives the ACK for the specific frame. The functionality of the Stop-and-Wait ARQ is illustrated in Figure 2.6(a).

The main advantage of the Stop-and-Wait ARQ is given by its simplicity. However, because it must transmit an ACK for each received frame, makes it also an inefficient mechanism in noisy environments such as wireless transmissions.

2.3.2 Go-Back-N ARQ

The Go-Back-N ARQ is also known as the ”continuous transmission error control”. Go-Back-N ARQ is based on the sliding window flow control and it uses both ACK and Negative ACK (NACK) frames. Both of them use sequence numbers, e.g., ACK n, NACK n, where n is the sequence number of the data frame ACK-ed or to be resent. The transmitter keeps all frames that have not been ACKed yet.



(a) Stop-and-Wait ARQ. (b) Go-Back-N ARQ.

Figure 2.6: *ARQ Error Control Mechanisms.*

In the Go-Back-N mechanism the sender may transmit a series of frames sequentially numbered modulo window size. If no errors occur, the receiver acknowledges incoming frames by using cumulative ACKs. However, if an error is detected in a frame, the receiver sends a NACK for the specific frame and discards the erroneous frame and all future incoming frames until the corrupted frame is correctly received. Hence, the transmitter, upon receipt of a NACK must retransmit the corrupted frame and all frames transmitted after. An illustration of the Go-Back-N ARQ is provided in Figure 2.6(b).

2.3.3 Selective-Repeat ARQ

The Selective-Repeat ARQ mechanism does not retransmit a sequence of frames but only the frames that receive a NACK. This is more efficient than Go-Back-N as it minimizes the amount of retransmissions. The drawback is that it requires more sophistication from the sender and the receiver in order to handle out-of-sequence frames. For instance, the receiver must maintain a buffer large enough to buffer frames received after a NACK frame and it must be able to re-insert the frame in the proper sequence.

2.3.4 Error Detection

Possibly the most robust error detection method used in networking is the Cyclic Redundancy Check (CRC). The fundamental advantage of CRC is that it can be completely implemented in hardware. A CRC generator is a unique bit pattern that is normally predetermined in a given protocol. Hence, an identical bit pattern is used both at the transmitter and receiver. Occasionally, the CRC generator is represented in a polynomial form (e.g., `10110101` is represented by the polynomial $p = x^7 + x^5 + x^4 + x^2 + 1$).

Error detection in CRC is performed by comparing a Frame Check Sequence (FCS) computed on the received frame against a FCS value initially computed and stored. Hence, an error is

announced to have occurred if the stored FCS and the computed FCS values are not equal. However, there is a small probability that a frame corruption that alters just enough bits in just the right pattern can occur, leading thus to an undetectable error. Hence, the minimum number of bit inversions required to achieve undetected errors remains a fundamental problem in the proper design of CRC polynomials.

Basically, an error cannot be detected by the FCS check if it is divisible by the CRC generator. With a carefully chosen generator, this probability may become very small. It can be shown that the following frame errors are detectable by the CRC [33]:

- all single-bit errors.
- all double-bit errors if the CRC generator contains at least three 1s.
- any odd number of errors if the CRC generator contains a factor $x+1$.
- any burst error if its length is less or equal than the length of FCS.
- most of larger burst errors.

2.3.5 Error Control

Error control in TCP is based on Go-Back-N ARQ and it is achieved through the use of *checksums*, *ACKs* and *retransmission* timers.

Checksums: Each TCP segment contains a 16-bit checksum field that is used by the receiving TCP to detect errors. The checksum covers both the TCP header and the data. If the checksum fails, the segment is discarded.

Acknowledgments: TCP uses ACKs in order to confirm the receipt of data segments. Most TCP implementations use cumulative ACKs, i.e, the receiver advertises the next byte it expects to receive, ignoring all segments received out-of-order. However, some implementations are using the Selective ACK (SACK). A SACK provides additional information to the sender, e.g., it reports out-of-order and duplicate segments.

Retransmissions: In order to guarantee reliable delivery of data, TCP implements retransmission of lost or duplicate segments. Hence, a segment is retransmitted when a retransmission timer expires or when the sender receives three duplicate ACKs (fast retransmission algorithm).

2.3.6 Forward Error Correction

Forward Error Correction (FEC) is a mechanism of error control that is based on introducing redundant data into the transmitted message thus allowing a receiver to detect and possibly correct errors caused by a noisy transmission channel. The main advantage of FEC is that retransmission of corrupted data can be avoided. However, FEC requires higher bandwidth due to the introduced redundancy. The number of errors that can be corrected by a FEC code is dependent on the code used, which implies that different FEC codes are suitable for different conditions.

In FEC systems, the source sends the message to be transmitted to an *encoder* that inserts redundant bits into the message and outputs a longer sequence of code bits, the so-called *codeword*. The codeword is then transmitted to a receiver, which uses an appropriate decoder for extracting the original message. Two main types of FEC codes are used today: *algebraic coding* (operate on blocks of data), and *convolutional coding* (operate on streams of symbols of arbitrary length).

2.4 Concluding Remarks

This chapter introduced several fundamental concepts for CC and error control schemes used in unicast environments. The operation of TCP, which is the major player in CC for unicast transmissions, was presented in some detail. Further, several CC solutions available for unicast transmissions were also presented. We introduced the notions of window-based and rate-based CC and presented several solutions proposed for this scope.

In addition, we presented the main solutions for error control in unicast environments. We presented important error control techniques operating at the DLL, since all transport protocols that require retransmissions implement one of these traditional techniques. We also presented FEC, an error control scheme that is based on transmission of redundant data allowing for better network utilization due to greatly reduced retransmissions necessary for data recovery.

Chapter 3

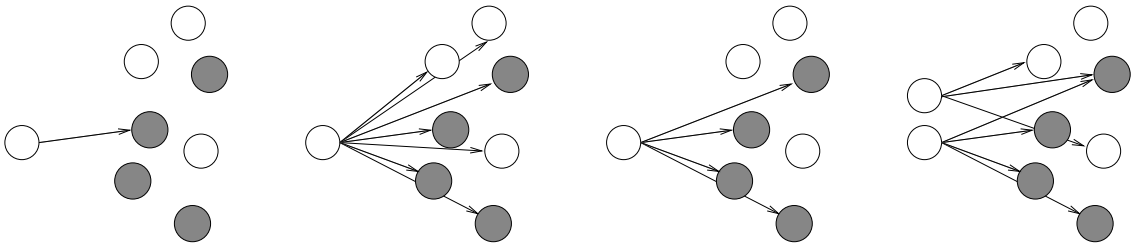
Congestion and Error Control in IP Multicast Environments

3.1 IP Multicast Environments

Group communication as used by Internet users today is taken more or less for granted. Forums and special interest groups abound, and the term "social networking" has become a popular buzzword. These forums are typically formed as virtual meeting points for people with similar interests, that is, they act as focal points for *social groups*. In this section, we discuss the technical aspects of group communication as implemented by IP multicast.

3.1.1 Group Communication

A *group* is defined as a set of zero or more hosts identified by a single *destination address* [21]. We differentiate between four types of group communication, ranging from groups containing only two nodes (one sender and one receiver – *unicast* and *anycast*), to groups containing multiple senders and multiple receivers (*multicast* and *broadcast*).



(a) Unicast. (b) Broadcast. (c) 1-to-m Multicast. (d) n-to-m Multicast.

Figure 3.1: *Group Communication. (Gray circles denote members of the same multicast group)*

Unicast

Unicast is the original Internet communication type. The destination address in the IP header refers to a single host interface, and no group semantics are needed or used. Unicast is thus a 1-to-1 communication scheme (Figure 3.1(a)).

Anycast

In anycast, a destination address refers to a group of hosts, but only one of the hosts in the group receive the datagram, i.e., a 1-to-(1-of-m) communication scheme. That is, an anycast address

addresses a set of host interfaces, and a datagram gets delivered to the nearest interface, with respect to the distance measure of the routing protocol used. There is no guarantee that the same datagram is not delivered to more than one interface. Protocols for joining and leaving the group are needed. The primary use of anycast is load balancing and server selection.

Broadcast

A broadcast address refers to all hosts in a given network or subnetwork. No group join and leave functionality is needed, as all hosts receive all datagrams sent to the broadcast address. Broadcast is a 1-to-m communication scheme as shown in Figure 3.1(b). Broadcast communication is typically used for service discovery.

Multicast

When using multicast addressing, a single destination address refers to a set of host interfaces, typically on different hosts. Multicast group relationships can be categorized as follows [64]:

1-to-m: Also known as "One-to-Many" or 1toM. One host acts as source, sending data to the m recipients making up the multicast group. The source may or may not be a member of the group (Figure 3.1(c)).

n-to-m: Also known as "Many-to-Many" or MtoM. Several sources send to the multicast group. Sources need not be group members. If all group members are both sources and recipients, the relationship is known as *symmetric multicast* (Figure 3.1(d)).

m-to-1: Also known as "Many-to-One" or Mto1. As opposed to the two previous relationships, m-to-1 is not an actual multicast relationship, but rather an artificial classification to differentiate between applications. One can view it as the response path of requests sent in a 1-to-m multicast environment. Wittman and Zitterbart refers to this multicast type as *concast* or *concentration casting* [88].

Table 3.1 summarizes the various group relationships discussed above.

Table 3.1: *Group communication types.*

Senders \ Receivers	1	m
1	Unicast / Anycast	Multicast / Broadcast
n	Concast	Multicast

3.1.2 Multicast Source Types

In the original multicast proposal by Deering [21], hosts wishing to receive data in a given multicast group, G, need only to join the group to start receiving datagrams addressed to the host. The group members need not know anything about the datagram or service sources, and any Internet host (group member or not) can send datagrams to the group address. This model is known as Any-Source Multicast (ASM). Two additional[1] functions that a host wishing to take part in a multicast network needs to implement are:

`Join(G,I)` – join the multicast group G on interface I.

`Leave(G,I)` – leave the multicast group G on interface I.

[1] Additional to the unicast host requirements defined in [10].

Beyond this, the IP forwarding mechanisms work the same as in the unicast case. However, there are several issues regarding the ASM model [7]:

Addressing: The ASM multicast architecture does not provide any mechanism for avoiding address collisions among different multicast applications. There is no guarantee that the multicasted datagram a host receives is actually the one that the host is interested in.

Access Control: In the ASM model, it is not possible for a receiver to specify which sources it wishes to receive datagrams from, as any source can transmit to the group address. This is true even if sources are allocated a specific multicast address. There are no mechanisms for enforcing that no other sources will not send to the same group address. By using appropriate address scoping and allocation schemes, these problems may be made less severe, but this requires more administrative support.

Source Handling: As any host may be a sender (n-to-m relationship) in an ASM network, the route computation algorithm makes use of a *shared tree* mechanism to compute a minimum cost tree within a given domain. The shared tree does not necessarily yield optimal paths from all senders to all receivers, and may incur additional delays.

Source Specific Multicast (SSM) addresses the issues mentioned above by removing the requirement that any host should be able to act as a source [36]. Instead of referring to a multicast group G, SSM uses the abstraction of a *channel*. A channel is comprised of a source, S, and a multicast group G, so that the tuple (S,G) defines a channel. In addition to this, the `Join(G)` and `Leave(G)` functions are extended to:

`Subscribe(s,S,G,I)` – request for datagrams sent on the channel (S,G), to be sent to interface I and socket s, on the requesting host.

`Unsubscribe(s,S,G,I)` – request for datagrams to no longer be received from the channel (S,G) to interface I.

3.1.3 Multicast Addressing

IP multicast addresses are allocated from the pool of class D addresses, i.e., with the high-order nybble set to `1110`. This means that the address range reserved for IP multicast is `224/24`, i.e., `224.0.0.0` – `239.255.255.255`. The `224/8` addresses are reserved for routing and topology discovery protocols, and the `232/8` address block is reserved for SSM. Additionally, the `239/24` range is defined as the *administratively scoped address space*[2] [53]. There are several other allocated ranges [37].

Address allocation

Multicast address allocation is performed in one of three ways [80]:

Statically: Statically allocated addresses are protocol specific and typically permanent, i.e., they do not expire. They are valid in all scopes, and need no protocol support for discovering or allocating addresses. These addresses are used for protocols that need well-known addresses to work.

Scope-relative: For every administrative scope (as defined in [53]), a number of offsets have been defined. Each offset is relative to the current scope, and together with the scope range it defines a complete address. These addresses are used for infrastructure protocols.

[2] An address *scope* refers to the area of a network in which an address is valid.

Dynamically: Dynamically allocated addresses are allocated on-demand, and they are valid for a specific amount of time. It is the recommended way to allocate addresses. To manage the allocation, the multicast address allocation architecture (MALLOC) has been proposed [80]. MALLOC provides three layers of protocols:

Layer 1 – Client–server: Protocols and mechanisms for multicast clients to request multicast addresses from a Multicast Address Allocation Server (MAAS), such as MADCAP [34].

Layer 2 – Intra-domain: Protocols and mechanisms to coordinate address allocations to avoid addressing clashes within a single administrative domain.

Layer 3 – Inter-domain: Protocols and mechanisms to allocate multicast address ranges to Prefix Coordinator in each domain. Individual addresses are then assigned within the domain by MAASs.

3.1.4 Multicast Routing

The major difference between traditional IP routing and IP multicast routing is that datagrams are routed to a *group* of receivers rather than a single receiver. Depending on the application, these groups have dynamic memberships, and it is important to consider this when designing routing protocols for multicast environments.

Multicast Topologies

While IP unicast datagrams are routed along a single path, multicast datagrams are routed in a *distribution tree* or *multicast tree*. A unicast path selected for a datagram is the *shortest* path between sender and receiver. In the multicast case, the problem of finding a shortest path instead becomes the problem of finding a *shortest path tree* (SPT), *minimum spanning tree* (MST) or *Steiner tree*. An SPT minimizes the sum of each source-destination path, while the MST and Steiner trees minimize the *total* tree cost.

Typically, multicast trees come in two varieties: *source-specific* and *group shared* trees. A source-specific multicast tree contains only one sending node, while a group-shared tree allows every participating node to send data. These two trees types correspond to the 1-to-m and n-to-m models in section 3.1.1 respectively. Regardless of which tree type a multicast environment makes use of, a *good* multicast tree should exhibit the following characteristics [69]:

Low Cost: A good multicast tree keeps the total link cost low.

Low Delay: A good multicast tree also tries to minimize the e2e delay for every source–destination pair in the multicast group.

Scalability: A good tree should also be able to handle large multicast groups, and the participating routers should be able to handle a large number of trees.

Dynamic Group Support: Nodes should be able to join and leave the tree seamlessly, and this should not adversely affect the rest of the tree.

Survivability: Related to the previous requirement, a good tree should survive multiple node and link failures.

Fairness: This requirement refers to the ability of a good tree to evenly distribute the datagram duplication effort among participating nodes.

Routing Algorithms

There are several types of routing algorithms available for performing routing in multicast environment. Some of the non-multicast specific include *flooding*, *improved flooding* and *spanning trees*. The flooding algorithms are more akin to pure broadcasting and tend to generate large amounts of network traffic. The spanning tree protocols are typically used in bridged networks, and create distribution trees which ensure that all connected networks are reachable. Datagrams are then broadcasted on this spanning tree. Due to their group-agnostic nature, these algorithms are rarely used in multicast scenarios (with a few exceptions, such as the Distance Vector Multicast Routing Protocol (DVMRP)).

Multicast-specific algorithms include *source-based routing*, *Steiner trees* and *rendezvous point trees* also called *core-based trees*.

Source-based Routing: Source-based routing includes algorithms such as Reverse Path Forwarding (RPF), Reverse Path Broadcasting (RPB), Truncated Reverse Path Broadcasting (TRPB) and Reverse Path Multicasting (RPM). Of these algorithms, only RPM specifically considers group membership when routing. The other algorithms all represent slight incremental improvements of the RPF scheme in that they decrease the amount of datagram duplication in the distribution tree, and avoid sending datagrams to subnetworks where no group members are registered. Examples of source-based protocols are the DVMRP, Multicast Extensions to Open Shortest Path First (MOSPF), Explicitly Requested Single-Source Multicast (EXPRESS) and Protocol Independent Multicast – Dense Mode (PIM-DM) protocols.

Steiner trees: As mentioned previously, the Steiner tree algorithms optimize the total tree cost and is an NP-hard problem, making it computationally expensive, and not very useful for topologies that change frequently. While Steiner trees provide the minimal *global* cost, specific paths may have higher cost than those provided by non-global algorithms. The Steiner tree algorithms are sensitive to change in the network, as the routing tables needs to be recalculated for every change in the group membership or topology. In practice, some form of heuristic, such as the KMB heuristic [42], is used to estimate the Steiner tree for a given multicast scenario.

Rendezvous Point trees: Unlike the two previous algorithms, these algorithms can handle multiple senders and receivers. This is done by appointing one node as a *Rendezvous Point (RP)*, through which all datagrams are routed. A substantial drawback with this approach is that the RP becomes a single point of failure, and that it may be overloaded with traffic if the number of senders is large. Examples of this type of protocol are the Core Based Tree (CBT), Protocol Independent Multicast – Sparse Mode (PIM-SM) and Simple Multicast (SM) protocols.

IP Multicast Routing Protocols

DVMRP: DVMRP [85] was created with the Routing Information Protocol (RIP) for a starting point and uses ideas from both the RIP and the TRPB [20] protocols. As opposed to RIP, however, DVMRP maintains the notion of a receiver–sender path (due to the RPF legacy of TRPB) rather than the sender–receiver path in RIP. DVMRP uses *poison reverse* and *graft/prune* mechanisms to maintain the multicast tree. As a Distance Vector (DV) protocol, DVMRP suffers from the same problems as other DV protocols, e.g., slow convergence and flat network structure. The Hierarchical Distance Vector Multicast Routing Protocol (HDVMRP) [81] and HIP [72] protocols both try to address this issue by introducing hierarchical multicast routing.

MOSPF: MOSPF [54] is based on the Open Shortest Path First (OSPF) link state protocol. It uses Internet Group Management Protocol (IGMP) to monitor and maintain group memberships within the domain and OSPF link state advertisements to maintain a view on the topology within the domain. MOSPF builds a shortest-path tree rooted at the source, and prunes those parts of the tree which have no members of the group.

PIM: Protocol Independent Multicast (PIM) is actually a family of two protocols or operation modes: PIM-SM [25] and PIM-DM [1]. The term *protocol independent* comes from that the PIM protocols are not tied to any specific unicast routing protocol, like DVMRP and MOSPF tied to RIP and OSPF respectively.

PIM-DM refers to a multicast environment in which many nodes are participating in a "dense" manner, i.e., a large part of the available nodes are participating, and that there is large amounts of bandwidth available. Typically, this implies that the nodes are not geographically spread out. Like DVMRP, PIM-DM uses RPF and grafting/pruning, but differs in that it needs a unicast routing protocol for unicast routing information and topology changes. PIM-DM assumes that all nodes in all subnetworks want to receive datagrams, and use explicit pruning for removing uninterested nodes.

In contrast to PIM-DM, PIM-SM initially assumes that *no* nodes are interested in receiving data. Group membership thus requires explicit joins. Each multicast group contains one active RP.

CBT: The CBT [5] protocol is conceptually similar to PIM-SM in that it uses RPs and has a single RP per tree. However, it differs in a few important aspects:

- CBT uses bidirectional links, while PIM-SM uses unidirectional links.
- CBT uses a lower amount of control traffic compared to PIM-SM. However, this comes at the cost of a more complex protocol.

BGMP: The protocols discussed so far are all Interior Gateway Protocols (IGPs). The Border Gateway Multicast Protocol (BGMP) [79] protocol is a proposal to provide inter-domain multicast routing. Like the Border Gateway Protocol (BGP), BGMP uses TCP as a transport protocol for communicating routing information, and it supports both the SSM and ASM multicast models. BGMP is built upon the same concepts as PIM-SM and CBT, with the difference that participating nodes are entire domains instead of individual routers. BGMP builds and maintains group shared trees with a single root domain, and can optionally allow domains to create single-source branches if needed.

3.2 Challenges

An important aspect to take into consideration when designing any communication network, multicast included, is the issue of *scalability*. It is imperative that the system does not "collapse under its own weight" as more nodes join the network. The exact way of handling scalability issues is application and topology-dependent, such as can be seen in the dichotomy of PIM: PIM-DM uses one set of mechanisms for routing and maintaining the topology, while PIM-SM uses a different set. Additionally, if networks are allowed to grow to very large numbers of nodes (in the millions, as with the current Internet), routing tables may grow very large. Typically, scalability issues are addressed by introducing hierarchical constructs to the network.

Related to the scalability issue, there is the issue of being conservative in the control overhead that the protocol incurs. Regarding topology messages, this is more a problem for proactive or table-driven protocols, that continuously transmit and receive routing update messages. On the other hand, reactive protocols pay the penalty in computational overhead, which may be prohibitively large if the rate at which nodes join and leave the multicast group (a.k.a. *churn*) is high.

In addition to keeping topology control overhead low, multicast solutions will also have to consider the group management overhead. Every joining and leaving node will place load on the network, and it is important that rapid joins and leaves do not unnecessarily strain the system. At the same time, both joins and leaves should be performed expediently, i.e., nodes should not have to wait for too long before joining or leaving a group.

Another important issue for RP-based protocols is the selection of an appropriate rendezvous point. As the RP becomes an traffic aggregation point and single point of failure, it is also important to have a mechanism for quickly selecting a replacement RP in the case of failure. This is especially important for systems in which a physical router may act as RP for several groups simultaneously.

While there are many proposals for solutions of the problems and challenges mentioned above, neither of them have been able to address what is perhaps the most important issue: wide scale deployment and use. IP multicast just hasn't taken off the way it was expected to. Whether this is due to a lack of applications that need a working infrastructure or the lack of a working infrastructure for application writers to use is still unclear.

Additional application-specific issues also appear. For instance, when deploying services which are considered "typical" multicast services, such as media broadcasting and Video-on-Demand (VoD). Since IP multicasting operates on the network layer, it is not possible for transit routers to cache a video stream that is broadcasted. If two clients, A and B, try to access the same stream at different times, client A cannot utilize the datagrams already received by B, but will have to wait until retransmission. This waiting time may be on the order of minutes or tens of minutes, depending on what broadcasting scheme is used. Additionally, VCR-like functionality (fast forward, rewind and pause) and other interactive features are difficult to provide.

3.3 Congestion Control

The expansion of multicast services in recent years require also adequate CC mechanisms. The large variety of applications that need multicast transmissions introduces new demands and challenges in the design of multicast CC protocols. Consequently, different features and requirements of such applications necessitate different CC schemes [52, 27].

The vast majority of CC proposals regarding multicast traffic make use of a *rate-based* mechanism in order to control and regulate the transmission rate of the traffic source or the receiving host. The rate-based approach is often preferred because of several problems that arise when a *window-based* CC mechanisms is extended to multicast traffic, the most severe problem being the ACK implosion at the source.

Furthermore, TCP's CC mechanism strives to provide a fair bandwidth sharing among competing TCP sessions. Therefore, another critical issue that must be considered when designing multicast protocols is the TCP-friendly behavior, i.e., the protocols must not drive existent TCP sessions into "bandwidth starvation". CC for IP multicast still remains an intense research topic even after several years since the introduction of IP multicast [21].

The major concern in the design of multicast CC protocols is mainly the handling (in a scalable manner) of highly heterogeneous receivers and how to cope with the highly dynamic network conditions. The goal is to design a multicast CC mechanism with the same behavior of TCP, i.e., to allow both users and applications to share, in a fair manner, the available network resources.

When speaking of multicast CC mechanisms or protocols, different regulation parameters can be considered for controlling the network congestion. They depend upon source, receiver, or both. The parameters used mostly are the transmission/reception rate or the employment of a TCP-like congestion window that controls the data transmission.

Rate-based Congestion Regulation: A rate-based congestion regulation is based upon the transmission or the reception rate of the multicast traffic. When congestion occurs, the source decreases the transmission rate to the multicast group avoiding thus the escalation of congestion.

If the CC mechanisms controls the reception rate, a receiver decreases its rate in case of congestion detection by dropping a layer such as in the case of multi-layer multicast. In a similar manner, the transmission or the reception rate may be increased when network conditions are appropriate, e.g., no packets lost during a predefined time period.

A rate-based multicast CC protocol generally implements a similar AIMD scheme as in TCP. In this case, TCP-friendliness is achieved by forcing the transmission rate to have a throughput

that is TCP "compatible", i.e., as given by the formula derived by Padhye *et al.* [59]:

$$T = MSS \cdot \min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + RTO \cdot \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}\right) \tag{3.1}$$

where T is the TCP-compatible throughput, W_{max} represents the maximum `cwnd`, MSS is the maximum packet size, RTT is the average round trip time between source and receiver, b is the number of packets acknowledged by a received ACK, p is the packet loss probability experienced on the path between source and receiver, and RTO represents the packet retransmission timeout.

Rate-based e2e CC mechanisms can be classified depending on where the CC mechanism is implemented. There are two main categories:

1. *source-based* (source-driven) CC: the source adjusts the transmission rate based upon the information received from the multicast receivers. The level of network congestion is also estimated by the source from the collected information obtained from the receivers.

2. *receiver-based* (receiver-driven) CC: this is generally used together with layered encoded transmissions, i.e., the receiver subscribes to a number of layers based upon its capabilities and the network congestion load.

However, depending upon the granularity of the desired congestion adaptation several trade-offs must be considered such as fairness, level of feedback, bandwidth utilization or scalability.

When both the source and the receivers collaborate in the CC mechanism, a third approach is said to be employed, i.e., a *hybrid* CC mechanism. In this case, both the source and the receivers participate in the CC mechanism by reducing the transmission rate and the layer subscription level, based upon the current network conditions.

Window-based Congestion Regulation: When the multicast network congestion is regulated by a TCP-like window mechanism, the CC is said to be window-based. The regulating congestion window may be implemented either at the source or at the receiver. Similar to TCP, a packet may be transmitted by the source if the congestion window allows it.

However, the major problem when implementing a window-based mechanism for multicast transmissions is related to the ACK implosion phenomenon at the source, i.e., the source may become overwhelmed by the number of ACKs received for each transmitted packet, especially in the case of large multicast groups.

The remaining of this chapter is dedicated to a brief overview for some of the most used mechanisms for e2e CC in IP multicast communications.

3.3.1 Source-based Congestion Control

The main idea behind a sender-based CC approach is that the source multicasts a single data stream whose quality is adjusted (i.e., transmission rate) based upon the feedback information collected from the multicast group members.

Pragmatic General Multicast Congestion Control

The Pragmatic General Multicast Congestion Control (PGMCC) protocol is a multicast CC protocol designed to be fair with the competing TCP flows [66]. PGMCC implements a similar algorithm as in TCP. PGMCC is a single rate protocol and it operates between the source and a representative of the multicast group, the so-called "*acker*". The "acker" is chosen to be the receiver with the slowest throughput (rate) in the multicast group.

Hence, in PGMCC the transmission rate of the source is adapted to the worst throughput existent in the multicast group at any time. Furthermore, the "acker" selection is a continuous process

since group members are continuously leaving and joining the multicast group. The selection of the "acker" in PGMCC is illustrated in Figure 3.2 [66].

As each receiver in a given multicast group may have different capabilities, one problem may arise in PGMCC, namely when trying to achieve an optimal bandwidth usage for the multicast group. This is heavily dependent on the application used, its reliability, and the network impact.

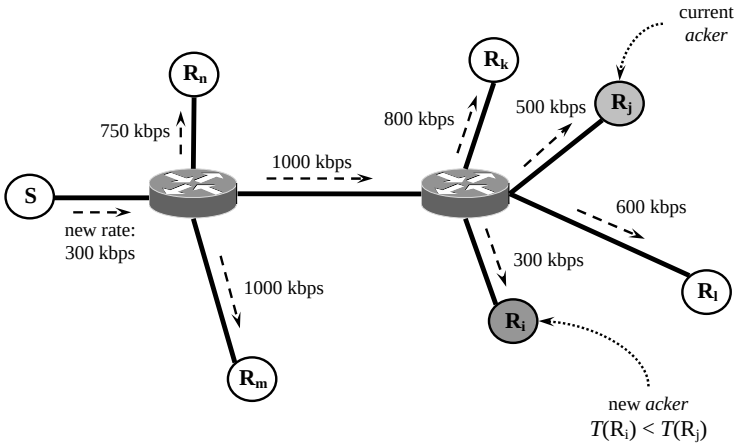


Figure 3.2: *PGMCC Operation: Selection of group representative.*

In addition to "acker" selection, a tight control loop is permanently performed in PGMCC between the source and the group representative (the "acker") [66]. In PGMCC, the source continuously monitors the receiver reports, which are embedded in NACKs (sent by group members other than the "acker") or ACKs (sent only by the "acker"). The source adjust the transmission rate for the multicast group and decides if the "acker" must be reselected [66]. The source also computes the loss rate and the current throughput of each group member. The throughput calculation for the sending rate for each receiver uses a simplified version of (2.36), given by:

$$T(\,) \propto \frac{1}{RTT \cdot \sqrt{loss}} \tag{3.2}$$

where $\propto$ denotes *proportional to*. Furthermore, given a set $\{R\}$ of multicast receivers, a new "acker" is selected if its throughput is less than the throughput of the currently selected "acker":

$$i : T(R_i) \leq T(R_j) \quad \forall j \in \{R\} \tag{3.3}$$

Equation (3.2) embodies a model of a window-based rate controller similar to the AIMD mechanism implemented by TCP. Equation (3.3) defines the "acker" selection process, which is also illustrated in Figure 3.2.

To avoid the problem of NACK implosion at the source, PGMCC allows the network devices to participate in the suppression of NACKs, i.e., they keep track of the NACKs forwarded to the source and report upstream the NACK containing the "worst" conditions [66].

One major impact on the PGMCC's performance is given by the appropriate selection of the "acker" when network conditions change. However, PGMCC implements a TCP-fair mechanism as the transmission rate of the source is derived from a TCP equation.

Scalable Feedback Control

Another source-based CC mechanism for multicast transmissions was proposed by Bolot *et al.* and represented by the Scalable Feedback Control (SFC) protocol [9]. The SFC is primarily intended for real-time applications that use multicast transmissions and where the source adjusts the transmission rate based upon the feedback received from the multicast receivers. SFC employs

probabilistic probing of the multicast group members together with an increasing search scope (defined as an *epoch*) and randomly delayed feedback scheme received from the group members such as to avoid the feedback implosion phenomenon at the source [9].

The information gathered from the receivers is used for adjusting the encoding parameters at the source and thus implicitly the transmission rate. Consequently, when a certain percentage of the receivers report a congestion state, the source halves the maximum encoding rate.

On the other hand, if the feedback received shows that all multicast group members have the capacity to receive at a higher rate, then the source can incrementally increase the encoding rate, which results in a higher transmission rate. The threshold defined in [9] makes use of three network state variables, namely `UNLOADED`, `LOADED` and `CONGESTED`. It allows thus for more flexibility when implementing the SFC mechanism since an ISP may use own local decisions for measuring the congestion state of the local network.

The main drawback of the SFC is the necessity of careful tuning of the parameters employed by the protocol since both the probabilistic probing of the multicast group and the randomly delayed feedback required from the receivers have a major impact on the ability of SFC to react in case of network congestion.

TCP Friendly Multicast Congestion Control

A further development of the equation-based CC mechanism presented by Floyd *et al.* [29], and extended for multicast applications, is represented by the TCP Friendly Multicast Congestion Control (TFMCC) protocol [87]. In TFMCC the source receives continuously feedback from the multicast receivers and when a receiver has a lower throughput than the current transmission rate, the source immediately reduces the transmission rate to accommodate the slower receiver.

Furthermore, in order to eliminate a potentially large number of unnecessary feedback messages, the receiver whose rate is higher than the current transmission rate is not allowed to send feedback messages to the source [87]. In order to also provide the possibility of increasing the transmission rate, TFMCC introduces the notion of *Current Limiting Receiver (CLR)*, which denotes the slowest receiving member of the multicast group (corresponds to the "acker" in PGMCC).

The CLR is allowed to send immediate feedback to the source without any suppression allowing thus for an increase of the transmission rate. Consequently, the transmission rate is permanently adjusted to the rate of the CLR. However, if another receiver reports a lower rate than the current rate of the CLR, the source adjusts the transmission rate accordingly.

The transmission rate in TFMCC is computed according to (3.1) ensuring thus a fair behavior of the TFMCC protocol [59]. The loss rate is calculated by the receivers as weighted averages over m most recent consecutive RTTs, denoted as *loss intervals* [87, 29]. Furthermore, the RTT estimations at the receivers are computed through an exponentially weighted moving average, to smooth the influence on the transmission rate of rare occurrences of high RTT estimates.

The authors of TFMCC reveal however that one critical part in this protocol is represented by the start-up phase since it may take some time until a relevant number of receivers estimate their RTT and notify the source limiting thus the practical use of this protocol to mostly long(er)-lived multicast transmissions.

Scalable RTCP

Another approach for source-based CC is the enhancement of the Real-time Transport Control Protocol (RTCP), namely the Scalable RTCP (S-RTCP) [23]. The idea of S-RTCP emerged from several limitations of the RTCP for very large dynamic groups, e.g., inefficient bandwidth usage, increasing storage state at each group member, feedback delay. Consequently, S-RTCP aims at solving several of the shortcomings of RTCP by employing aggregation of feedback and limiting the bandwidth allocated for the control traffic to a predefined fraction of the total session bandwidth for very large groups.

S-RTCP organizes dynamically the multicast group members into local regions. Every region has an *aggregator* that is responsible for the gathering of the *Receiver Reports (RRs)* from its local

members. The aggregator extracts significant information from the received feedbacks, derives relevant statistics and then forwards this information to a *manager*. The manager further analyzes this information and estimates what network regions suffer from congestion. Finally, the manager informs the source of the congestion state of the network and thus the source will be able to adjust the transmission rate based upon the network congestion status [23].

S-RTCP solution is considered to alleviate the problem of feedback delay, increased storage state and inefficient bandwidth usage of the RTCP protocol providing thus for a more scalable solution for IP multicast CC [23].

Source Adaptive Multi-layered Multicast

Another solution intended for CC in the case of multicast video transmission is the Source Adaptive Multi-layered Multicast (SAMM) algorithm [3]. In SAMM, the source adapts the encoding parameters used for video transmission including the number of generated video layers with their associated encoding rates as a reaction to a continuous congestion feedback from the receivers.

Special feedback servers are used for this purpose. These servers are distributed throughout the physical network and are organized in an overlay topology. The main function of the feedback servers is the aggregation of the congestion feedback flows received from the group members preventing thus the feedback implosion at the source. The architecture of the SAMM algorithm is illustrated in Figure 3.3 [3].

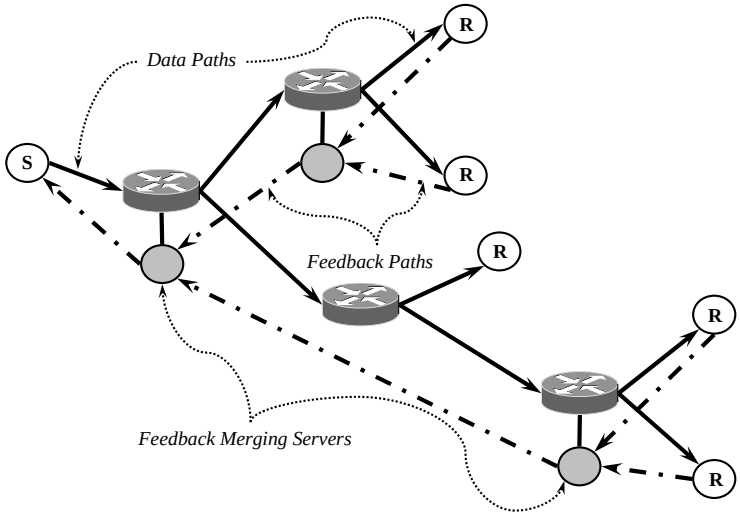


Figure 3.3: *SAMM Architecture.*

In SAMM, the network nodes (i.e., routers and switches) are not required to perform any novel or complex functions. The source generates video segmented into multiple layers, each layer with its own priority. The most important portions of the video stream are assigned to the layer with the highest priority, called the *base layer*. The other layers, called *enhancement layers* have progressive lower priority and are used to encode packets that can be used by the receivers to further enhance the quality of the base layer video stream.

Because the SAMM algorithm is dependent on the priority differentiation of different encoded video layers, it relies on the underlying network to be able to implement a differentiation dropping scheme and flow isolation for the lower priority layers [3]. This is done either through class-based priority forwarding or through the employment of a queueing management scheme such as Weighted Fair Queueing (WFQ).

The SAMM feedback messages contain two vectors: a rate vector listing the rates requested by the receivers and a counter vector listing the number of receivers requesting a certain rate. The feedback servers collect the information received from the receivers by storing the most up-to-date

feedback arrived on each downstream branch merging them if packets are received from the same connection. The merged feedback message is then transmitted to the source [3]. Hence, the rate values contained in the feedback messages are used by the source to adapt the transmission rate for each encoded video layer.

3.3.2 Receiver-based Congestion Control

The receiver-based CC approach is generally based upon a layered coding scheme such that the multicast source is able to transmit several layers to different multicast groups. The receivers subscribe to one or several layers based on their local capabilities.

The receiver-based approach is characterized by the fact that the control is done only at the receiver side. A receiver decides autonomously if it wants to subscribe or to drop an enhancement layer based upon the available resources.

Receiver-driven Layered Control

A receiver-driven CC algorithm designed to achieve TCP fairness was proposed by Vicisano *et al.* in [83]. The algorithm, Receiver-driven Layered Control (RLC) is TCP-friendly and it is intended to be used in the Multicast Backbone (MBone) for the transmission of a continuous stream of data (e.g., video or audio) or reliable bulk data transfers (e.g., file transfer). Data transmission is organized in layers and the MBone multicast receivers may choose, based on the available resources, to join one or several multicast groups.

RLC does not require support from a network node (a router) and does not need per-receiver status maintenance at the source. RLC is designed to be completely receiver driver and does not require any explicit protocol for synchronization of the multicast receivers or special management of multicast group membership. Hence, in RLC algorithm the receivers join the different layers completely autonomously. The *join* and *leave* strategies in RLC are designed to emulate the behavior of TCP and the throughput for a given layer is computed in a similar manner as described by (2.36) [51].

Furthermore, in order to avoid inefficient uncoordinated behavior of receivers located behind a common bottleneck, RLC employs an implicit coordination by using *Synchronization Points (SPs)* in the form of special packets contained within the data stream. A receiver is only allowed to join immediately after a SP and can make its decisions based only on the history of events since the last SP [83]. The source periodically generates short bursts of packets just prior to sending a SP in order to probe the available bandwidth in the multicast tree followed by a period when no packets are transmitted.

The time gap between consequent SPs is proportional to the bandwidth allocated for a given subscription level and the SPs are located always at the end of the packet burst. If the probing does not lead to congestion then there is a certain assurance that following join attempts may be successful. Moreover, the use of SP allows a receiver with a lower subscription level to increase more rapidly their reception rate since SP are more infrequent in higher layers. Consequently, RLC allows a receiver to increase the subscription level (at a SP) if no loss occurred during the previous burst period and to decrease it if the receiver experienced loss during normal transmission.

Receiver-driven Layered Multicast

Similar to RLC, the Receiver-driven Layered Multicast (RLM) protocol can be described as a CC scheme where the source is not required to have an active role in the transmission of multicast data [52]. The CC mechanism of the RLM is performed at the receiver side, where conceptually each receiver is required to perform a simple control loop, to drop a layer in case of congestion (*leave*) or to add a layer (*join*) when capacity is available. By performing the control loop, a receiver is, in essence, searching for an optimal subscription level for the multicast transmission. The source in the RLM protocol encodes the data into several layers and each layer is transmitted on a separate multicast group. Hence, a receiver may subscribe to several layers until congestion

occurs at which point he must back-off to a previous operating point. The RLM protocol operation is illustrated in Figure 3.4 [52].

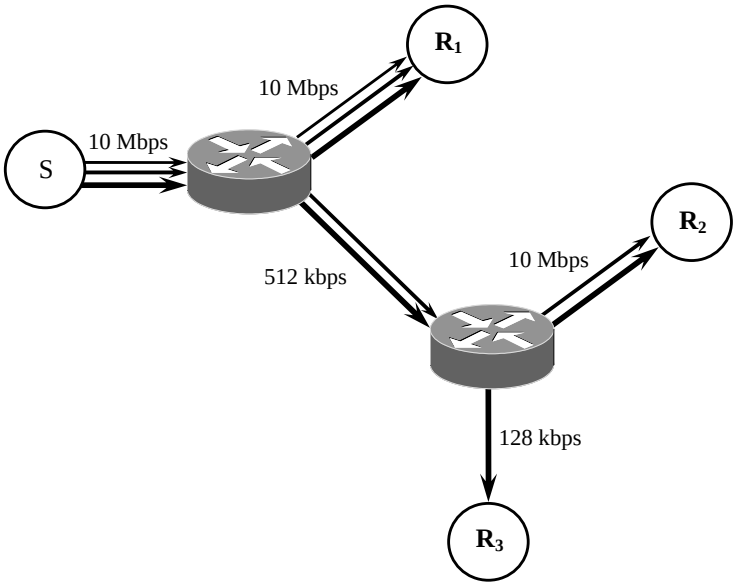


Figure 3.4: *RLM Protocol Operation.*

The source transmits the data encoded in several layers to the multicast tree. If a path between the source and a receiver has high capacity, the receivers belonging to respective path may subscribe to all layers (e.g., R_1 in Figure 3.4) receiving thus the highest quality. If a bottleneck path exists between the source and any subset of receivers belonging to the multicast tree, then the receivers behind the bottleneck must drop one or several layers to accommodate for the bottleneck capacity (e.g., in Figure 3.4, although R_2 may subscribe to all layers he is restricted by the bottleneck link of 512 kbps). Hence, the main effect is that the multicast distribution trees for each encoded layer are implicitly defined as an effect of the receiver adaptation to congestion.

RLM implements an exponential back-off for layers resulting in multiple failed join attempts from the receivers. This is done in order to alleviate the transient congestion state caused by this behavior. At the same time neighboring receivers make use of a *shared learning* algorithm to allow them to learn from each other. The shared learning algorithm gives the possibility to the neighboring receivers to listen to the ongoing join attempts in the multicast group. Consequently, before a receiver performs a join, it notifies the entire group by multicasting a message, which identifies the given layer. This allows that all receivers learn together that a certain layer is problematic instead of each receiver running individual join attempts [52].

However, the shared learning algorithm relies on that the network signals congestion by discarding packets from all participant layers. Consequently, a priority discarding scheme employed in the network impedes the receivers subscribed to high priority layers to detect congestion limiting thus the scalability of the RLM protocol.

Packet-pair Layered Multicast

The Packet-pair Layered Multicast (PLM) protocol introduced in [46] is aimed at eliminating the instability and periodic losses inherent in both RLC and RLM. This is due to the bandwidth inference in these protocols caused by the join experiments or the packet bursts. The PLM uses instead *packet-pair* for inferring the available bandwidth at the bottleneck link and allowing the receivers to decide what layers to join.

PLM provides a receiver driven CC mechanisms and it assumes that the source is able to transmit data via layers and for each layer all packets are transmitted in pairs, i.e., two packets

are transmitted back-to-back. Furthermore, PLM requires that the network nodes provide a fair scheduling queueing mechanisms, i.e., PLM assumes a *fair scheduler* network [46].

A receiver joins the base layer and waits for the first packet pair of this layer. If no packets are received before a predefined timeout period, the receiver is assumed not to have enough capacity to join the multicast session. If the packet pairs can be received then the multicast session can be joined. Furthermore, the packet pair is used for inferring the available bandwidth at the time of reception of the packet pair. Consequently, a receiver drops a layer if the inferred bandwidth given by the packet pair is lower than the current layer subscription level, and it adds a layer if all bandwidth inferred by the packet pairs for a given period of time is higher than the current subscription level during the same time period [46].

Since PLM is designed based on the fair scheduling paradigm, the theoretical properties of an ideal CC protocol such as stability, efficiency, scalability, robustness and feasibility are guaranteed in PLM.

Layered Video Multicast Retransmission

The Layered Video Multicast Retransmission (LVMR) protocol employs a layered MPEG-encoded multicast transmission together with a hierarchy of agents distributed at different levels in the multicast tree in order to perform the CC [47]. Beside the use of a hierarchical approach to the CC for the receivers, LVMR implements the use of retransmissions in a layered environment and addresses application, application-control, and transport layers.

LVMR is considered hierarchical because the receivers do not provide their feedback directly to the source but instead the feedback is transmitted to the *agents* distributed throughout the network. An agent can be either a normal or a specialized receiver and its main responsibility is the collection of feedback information from the receivers and at the same time distributing information back to the receivers. The agents are also responsible for coordinating the join and leave decisions of the receivers [47].

Consequently, the LVMR protocol architecture suggests the use of multiple domains within a multicast group with an Intermediate Agent (IA) for each domain. Here, a domain may be either a physical region (e.g., geographical area) or a logical region (e.g., corporate intranet). Each domain is divided into one or more subnets and each subnet has a Subnet Agent (SA). The SAs collect information regarding the status of their subnets from the receivers while the IAs achieve the same function for their domains. Thus, the feedback from the receivers propagate all the way back to the source.

At the same time, an IA compiles the information received from all SAs belonging to its domain and forwards this information back to the SAs which, in turn, multicasts this information to the receivers. The retransmissions are performed by designated local receivers reducing thus the congestion recovery time. The architecture of the LVMR protocol is depicted in Figure 3.5 [47]. It must be mentioned that SAs and IAs are regarded as being only logically separated from the receivers, though the LVMR protocol does not require that these should be separate entities but instead they may be physically co-located with the receivers.

The authors in [47] report on several improvements of the LVMR over RLM. One drawback of LVMR is however the overhead induced by the use of control agents. On the other hand, the source is totally released from the burden of performing CC. Another advantage is that the control agents can be easily implemented at the application level making it thus a useful choice for a CC mechanism in overlay multicast environments.

Fine-grained Layered Increase/Decrease Dynamic Layering

The Fine-grained Layered Increase/Decrease Dynamic Layering (FLID-DL) reported by Byers *et al.* [14] supplements the RLC protocol [83] by providing it with a generalization by means of a dynamic layering scheme. This is done to avoid the IGMP response bottleneck caused by the leave

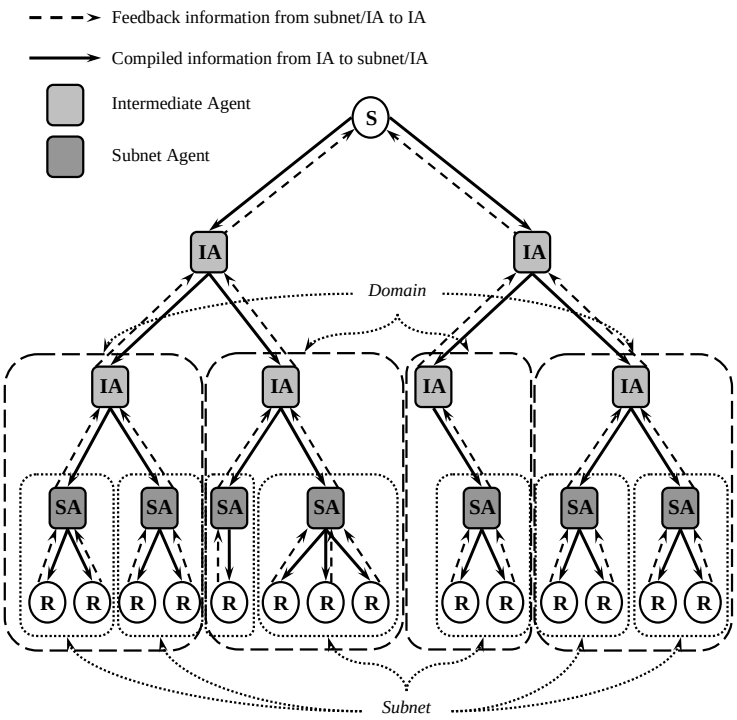


Figure 3.5: *LVMR Protocol Architecture.*

latency[1]. Similar to RLC, the FLID-DL protocol is a scalable CC scheme, where the receivers join an enhancement layer transmitted by the source at predefined synchronization points and drop a layer when they experience congestion. The implementation of the FLID-DL uses a *digital fountain encoding*, which is an FEC technique (Section 3.4.6). Thus, FLID-DL allows a receiver to recover the original data, even if some packets may be lost during transit, when it receives a fixed number of separate packets irrespective of the number of layers it subscribed over time.

In order to limit the leave latency and implicitly the congestion responsiveness, FLID-DL implements *dynamic layers*, i.e., the transmission rate of a layer changes over time. The dynamic layers behavior in FLID-DL is described by the following paradigm: the multicast source decreases its transmission rate of each layer over time allowing thus a receiver to decrease fast its reception rate by not joining any enhancement layers.

The reception rate of the receivers can be maintained by periodically joining layers. The reception rate may be increased by joining additional layers (beyond those required for maintaining a given rate). Consequently, this approach eliminates the congestion responsiveness caused by slow leave operations [14].

Similar to RLC, FLID-DL make use of special flags set by the source in the multicast packets and the join/leave behavior of the receivers is strictly imposed by these packets. Consequently, FLID-DL does not require any feedback from the receivers and the join/leave operations of the receivers take place according to the current network conditions existent on the path between the source and the receivers. Furthermore, time is divided into time slots by the source and a receiver may join an additional layer (i.e., increase its reception rate) only at the beginning of the next time slot, given that the receiver has not experienced any packet loss during the current time slot.

As a result, FLID-DL can scale to a very large number of receivers and, at the same time, receivers with lower capabilities (e.g., lower available bandwidth) have no impact on the receivers

[1]The IGMP join (and its subsequent *graft*) takes approximately one RTT while the IGMP leave (and the associated *prune*) is in the order of seconds. Thus, spontaneous join operations may create transient congestion that lasts until the successful completion of the leave procedure.

Dealing with leave latency is a key issue in the design of most layered multicast CC protocols.

with better capability.

3.3.3 Hybrid Congestion Control

The hybrid approach to CC mechanisms for IP multicast are defined as a combination of both source-based and receiver-based CC solutions, i.e., both the source and the receivers cooperate in the CC mechanism.

Hybrid Adaptation Layered Multicast

The Hybrid Adaptation Layered Multicast (HALM) protocol provides a solution to reduce the drawbacks of the statically assigned transmission rates for layered multicast transmissions and the highly heterogeneous and dynamic rate requirements from the receivers [50]. HALM is effective on top of the Real-time Transport Protocol (RTP) and it assumes a group-oriented IP multicast capable network [70]. In addition, HALM provides e2e control and all functionalities are implemented at the end systems (source or receiver).

In HALM the source dynamically adjusts the number of layers and their corresponding rates according to the bandwidth distribution of the receivers. The source periodically transmits *Sender Report (SR)* packets that include the cumulative rate vector, a timestamp, the RTP synchronization source identifier and a list with the receivers requests [70]. The receivers answer the SR with a *Receiver Report (RR)* packet and the cumulative rate vector is adjusted periodically by the source, based upon the bandwidth distribution reported the RRs. In order to ensure TCP friendliness behavior of the protocol a receiver is required to directly use a TCP throughput function for calculating its expected bandwidth (e.g., as provided in [51] or [59]).

Because HALM implements a cumulative layer subscription policy, the subscription level of a receiver depends on its expected bandwidth and the set of cumulative layer rates. The protocol introduces a *fairness index* $F(.)$ defined as [50]:

$$F(r, \rho_i) = \frac{\Gamma(r, \rho_i)}{r} \tag{3.4}$$

where r is the expected bandwidth of the receiver, ρ_i denotes the cumulative rate vector, and $\Gamma(r, \rho_i)$ is defined as the maximum rate that a receiver with an expected bandwidth r is able to receive. In other words, the fairness index is the ratio between a receiver's reception rate and its requested rate.

Thus, the rate adjustment translates into an optimization problem whose objective is to maximize the expected fairness index $F(r, \rho_i)$ for all multicast receivers by choosing an optimal rate vector. Further, in HALM, each receiver computes its cumulative packet loss rate (for all subscription layers) as well as the RTT of the path to the source. The estimation of the RTT integrates a dual approach in the form of a closed-loop estimation through implementation of a low frequency RR with an open-loop estimation by tracking the one-way trip time when it receives a SR. Moreover, the receiver should adapt their subscription level on SR reception.

Since HALM operates on top of RTP/RTCP, the control periodicity is not scalable with large multicast group sizes. To improve the scalability issue, the source implements feedback sampling and makes its decision based upon a controlled number of RRs such as to maximize the group satisfaction with a confidence level within a predefined confidence interval. As a result, the control period cannot be too small to avoid oscillations. Thus, a trade-off must be made between the responsiveness of the protocol, its scalability, and rate stability. One drawback of the HALM protocol is given by the IGMP join and leave load that may result when a receiver adjust its subscription level on the receipt of each SR.

Multicast Loss-Delay Adaptation

The Multicast Loss-Delay Adaptation (MLDA) [74] approach to CC for multicast environments is a general TCP-friendly solution inspired by the Enhanced Loss-Delay Adaptation (LDA+)

algorithm [73]. LDA+ uses RTP to collect receiver statistics, which are later used by the source to regulate the transmission behavior in accordance with the network congestion state. LDA+ is an AIMD CC mechanism, where the increase and decrease transmission rate values are determined dynamically and inferred by the bottleneck link. Furthermore, LDA+'s additive increase allows a flow with lower bandwidth to increase its rate faster than flows with higher bandwidth with the restriction that the bottleneck bandwidth is not exceeded.

Thus, in the MLDA CC mechanism, the receivers collect information about the path characteristics that connects them the source (e.g., packet loss, packet delay and bottleneck bandwidths). The information is later used to determine a bandwidth share that a TCP flow would have on the same path sharing the same conditions. MLDA is employing layered based data transmission and the source collects periodically information about the bandwidth shares at the receivers and accordingly adjusts the layer transmission rates. It also informs the receivers about the sizes and rates of the different layers.

In order to avoid the feedback implosion at the source, MLDA implements truncated exponentially distributed timers. The receivers can independently determine the number of layers they subscribe to. Furthermore, MLDA is able to avoid multicast group members leave latencies and reacts faster to congestion since it adjusts the transmission rates of the different layers dynamically instead of waiting for the receivers behind a bottleneck link to leave the corresponding multicast group.

Source-channel Adaptive Rate Multicast

Another CC mechanisms that involves both the source and the receiver and uses the RTCP is Source-channel Adaptive Rate Control (SARC) [84]. SARC also deals with the problem of feedback reports by employing a hierarchical aggregation. Furthermore, SARC implements an e2e FEC and source rate control based on the distributed feedback aggregation mechanism, which is coupled with a video layered coding system. The transmission parameters of the source (e.g., number of layers, layer rates, type of FEC) are dynamically adapted to the aggregated feedback collected from the receivers.

At the beginning of a session, the source announces the range of transmission rates according to the rate-distortion characteristics of the source. The range interval is then quantized into subintervals that correspond to allowed values for the layer rates. Furthermore, SARC divides the time into *feedback rounds* and at the beginning of each feedback round, the number of layers and corresponding rates are advertised to the receivers via SRs.

Based upon the estimated network parameters on the path toward the source, a receiver chooses an appropriate level of subscription/unsubscription. The receiver estimated bandwidth and the layer rates are transmitted to the source in RRs. In order to avoid the oscillations caused by the IGMP leave latency, the source has enough time for the layer adaptation rates, while receivers implement a delayed drop of an enhancement layer.

Aggregation agents are placed at strategic positions within the network and are responsible for the feedback aggregation from the receivers. The agents are divided into *local regions* and are also responsible for classifying the receivers within their region based upon the reception behavior. The aggregator agent receives the feedback and sends the derived statistics through point-to-point communication to its parent aggregator called *merger*. The root of the aggregator hierarchy is denoted as the *manager* [84]. The SARC hierarchical aggregation architecture is illustrated in Figure 3.6 [84].

The clustering depends on several variables, namely packet loss rate and TCP-compliant throughput. SARC uses the throughput calculation presented in [59] and exemplified in (3.1). The last level aggregators (those closest to the receiver) also serve as intermediaries in the receivers estimation of the RTT by sending *probe-RTT* packets [84].

Further, the receiver feedback mechanism provides the loss rate, the bottleneck bandwidth and the number of multicast receivers in each cluster. This information is used by the source to dynamically adapt the number of layers and their rates such as to maximize the quality perceived by different regions. The source in SARC uses a *fine grain scalable* encoding system whose param-

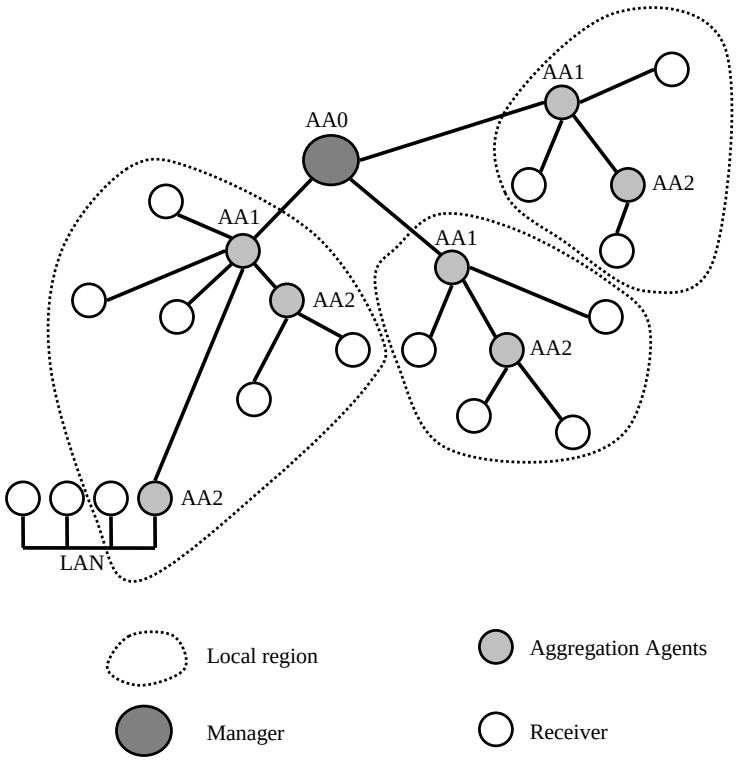


Figure 3.6: *SARC Hierarchy of Aggregators.*

eters are optimized for the number of layers and their rates, according to the receiver feedback. The goal is to maximize the overall *peak signal to noise ratio* by choosing the appropriate layer rates and level of protection, subject to a maximum number of layers.

The authors showed through experimental results that SARC is able to track fluctuations of the available bandwidth in the multicast tree, and has also the capacity to control fluctuating packet loss rates.

Smooth Multirate Congestion Control

The Smooth Multirate Congestion Control (SMCC) solution introduced in [43] is a multiple rate equation-based CC mechanism that employs adaptive layers for multicast transmissions. Each layer in SMCC uses the TFMCC [87] CC approach combining thus the advantages of TFMCC with the flexibility and scalability of using multiple rates. Since each layer in SMCC is independently controlled by the TFMCC protocol, the inherent properties of TFMCC hold for any given layer.

Similar to TFMCC, the receivers subscribed to a given layer do not send feedback to the source unless their calculated rate is less than the current sending rate for the specific layer, avoiding thus the feedback implosion at the source. The CLRs are permitted to send immediate feedback without any form of suppression. In this way, the source can adapt the transmission rate for each layer. Furthermore, each layer of SMCC is transmitted at a rate corresponding to a designated interval according to different participants of the specific layer.

A receiver can initiate a join operation if its estimated throughput is higher than the maximum sending rate corresponding to its current subscription level. The join operation uses the concept of *additive increase layers* implemented as *binary counting layers*. The rate transmitted by the binary counting layers implements an `ON/OFF` functionality with a sending rate of 2^i packets per RTT for each `ON` period, where i denotes the binary counting layer, and the duration of each `ON` and `OFF` period is equal to $RTT \cdot 2^i$ [43].

Consequently, the receiver slowly increases the reception rate by one packet per RTT and when the rate reaches the rate of the next enhancement layer it joins this layer and drops the additive increase layers. The join operation is interrupted if a packet loss occurs during the additive transition period. Furthermore, the receiver must drop a layer if the estimated throughput is lower than the minimum allowed rate of its current subscription level.

One appealing property of SMCC is given by the implementation of additive increase layers instead of directly joining the next enhancement layer. This mechanism reduces considerably the manifestation of transient congestion behavior caused by failed join operations. The modular design of SMCC allows also for the improvement of the CC mechanism if alternative or improved equation-based rate control schemes need to be used.

3.4 Error Control

Generally, the main functionality of transport protocols is geared toward the provisioning of e2e reliable communication and flow control. However, since the basic network service offered by today's Internet is best-effort, packet loss may occur at any node in the network creating thus a need for recovery, i.e., error control mechanisms. Traditionally, error control schemes employ several techniques, and the most used approaches are the following:

1. various ARQ schemes, which use ACKs, timers and retransmissions.
2. different FEC schemes (e.g., digital fountain), which enable packet loss recovery if a given number of packets are received correctly.
3. the use of error resilient source coding to conceal possible errors at the receiver.

Multicast transmissions are often utilized for multimedia applications that have strict real-time constraints in term of bandwidth requirements and an upper bounded delay while, at the same time, the multicast protocols are required to conserve bandwidth. This is especially valid when the number of receivers subscribed to the multicast service grows large. While ARQ error control schemes can be quite challenging in multicast environments, the other two types of schemes are not affected by multicast.

The main purpose of the error control mechanisms in multicast environments is to make use of the available transmission rate to minimize the undesired effect given by packet loss. Consequently, error control is often achieved by the transmission of a given amount of redundant information in the packets trying thus to compensate for the loss of packets.

In the case of reliable multicast, most error control schemes are ARQ based. The avoidance of the retransmission requests implosion at the source is achieved through the use of *hierarchical retransmitters* distributed throughout the network or local error recovery mechanisms [12, 27, 86]. Furthermore, hybrid techniques may also be employed for error control such as the hybrid FEC/ARQ proposed by Nonemacher *et al.* [56]. The remainder of this sections briefly examines some of the error control techniques employed by some reliable multicast protocols.

3.4.1 Scalable Reliable Multicast

The Scalable Reliable Multicast (SRM) framework aims at achieving reliable data delivery among the different receivers [27]. Loss recovery is achieved in SRM through special NACKs packets, the so-called *repair request*, sent by the receivers who experienced a loss. Upon loss detection, the receiver waits a random time before multicasting their request. This is done in order to suppress requests from other group members that may share the same loss.

However, the repair request differ from a traditional NACK in the sense that they are not addressed to a specific sender and request data by its unique persistent name. Consequently, the repair requests can be answered by any other group member that has the particular information. In order to avoid the unnecessary transmission of multiple packets containing the same information,

a receiver suppresses its retransmission of data in response to a request in case it detects it from another member of the group.

Further, SRM uses session messages to allow join and leave operations. The session messages are also used in the RTT computation which, in turn, is used for the initiation of the suppression timers [27].

3.4.2 Reliable Multicast Protocol

Reliable Multicast Protocol (RMP) [86] is a totally ordered and reliable multicast transport protocol acting on top of an unreliable multicast service. RMP implements Van Jacobson's algorithm for flow and CC as well as group membership, name service and updates for group memberships [38].

The source of a multicast group uses a sliding window, which is regulated by retransmission timers, ACKs and NACKs. Thus, a source uses the slow-start algorithm to increase its congestion window while the window size is exponentially decreased when congestion occurs (e.g., signaled by a retransmission timer's timeout).

RMP uses the following entities: RMP *processes*, *token rings*, and *token lists*. The basic entity in the RMP communication is the RMP process. A token ring refers to the basic unit for group communication and RMP message ordering and correspond to a set of processes receiving packets on a given IP multicast address and port. Thus, a token ring identifies a group of members and not a single receiver. Moreover, the membership of a token ring may change over time and a given list of members of a token ring is referred to as a token list [86].

Error control in RMP is done by employment of both ACKs and NACKs messages. All data packets are uniquely identified by stamping them with the tuple: *<RMP process ID, process sequence number, QoS level>*. The data packets are multicasted to the members of the token ring and are handled by a primary receiver called the *token site*. When a receiver detects a packet loss it multicasts a NACK to the multicast group instead to unicast it to the token site. When the source receives a NACK it treats it as sign of congestion (similar to a retransmission timeout) and reduces its transmission window exponentially [86].

3.4.3 Reliable Adaptive Multicast Protocol

Reliable Adaptive Multicast Protocol (RAMP) was initially introduced in RFC 1458 [12] and was later developed for operation in high-speed, all-optical circuit-switched gigabit networks [41].

RAMP acts on top of a multicast network layer protocol and combines receiver initiated NACKs together with source-based unicast retransmission as error control mechanism. In RAMP, a source starts by transmitting a *connect* message to an IP multicast address and starts sending immediately after it receives at least one *accept* message. Data packets contain sequence numbers in ascending order and a receiver unicasts a NACK back to the source as soon as it detects a gap in the sequence numbers.

The source can choose to unicast the retransmission to the receiver or to multicast it to the group. However, since some applications may not require in-order delivery of data, RAMP is also able to allow out-of-order packet delivery to a receiver. The source in RAMP aggregates the retransmission requests by implementing a retransmission hold timer, which controls if the source multicasts or unicasts the retransmission requests [12].

3.4.4 Xpress Transport Protocol

The Xpress Transport Protocol (XTP) provides a reliable, real-time, lightweight transport protocol, designed to support a wide range of applications [12, 89]. XTP is conceived to provide an explicit multicast group management protocol, which allows the control of group membership.

The error control in XTP is based on checksums (over the header or the entire packet) and timers. Reliable delivery in XTP is similar to TCP and when a receiver detects out-of-order packets, it transmits a NACK back to the source. Furthermore, in multicast environments, the retransmissions are multicasted to the group while NACKs are unicasted to the source. This

implies that XTP does not provide any feedback mechanism. However, XTP does offer both a window-based and a rate-based flow control.

3.4.5 Hybrid FEC/ARQ

Nonnenmacher *et al.* investigate in [56] a hybrid error control approach for achieving a scalable and reliable multicast transmission, namely the combination of FEC techniques with ARQ. The authors considered using FEC as a transparent layer beneath a reliable multicast that implements ARQ and the integration of these two techniques into a single layer using the retransmission of the redundant data to recover from packet losses.

Although it has several interesting properties, FEC alone cannot provide full reliability for a transmission. However, when it is combined with the ARQ, the authors in [56] showed that it can provide inherently reliable multicast transmission even to large multicast groups. Even if FEC is decoupled from ARQ in the sense that it operates at a layer beneath the operating level of ARQ, it still substantially reduces the packet losses improving thus the overall bandwidth utilization [56].

Thus, employing a hybrid FEC/ARQ error mechanism can dramatically reduce the network traffic for the retransmission of erroneous packets by answering multiple NACKs from several receivers with a single parity (redundant data) packet.

3.4.6 Digital Fountain FEC

Byers *et al.* introduce another ingenious error recovery scheme in the form of an ideal, fully scalable protocol they called a *digital fountain* [13]. The digital fountain solution employs a pure FEC mechanism for reliable multicast with no retransmissions even in the case of high loss rates. Instead, the digital fountain continuously multicasts a revolving stream of FEC parity packets. This allows any number of receivers to receive bulk data with optimal efficiency [13].

The digital fountain implements a new class of erasure codes that are shown through performance measurements to be several orders of magnitude faster than standard erasure codes. This resourceful approach to error control attained its name from the following metaphorical paradigm: the stream of FEC packets act as a fountain for thirsty travelers – interested receivers join the multicast group in order to ”drink” from this digital fountain for as long as they receive enough parity packets that enables them to recover the original data [13].

3.5 Concluding Remarks

This chapter discussed several contributions to CC and error control schemes used in the case of IP multicast environments. The main observation is that fairness is better achieved through multirate transmissions, i.e., when data is divided into several layers [68]. Careful consideration must be given to management aspects such that join and leave operations does not disrupt the multicast session. Layering allows also for dynamic rate adjustments.

Multicast communications may involve thousands of group members with heterogeneous receiving capacity and network conditions. Therefore, a multicast CC mechanism must deal with these conditions, i.e., the data distribution rate should adapt to the prevailing network conditions. Equally important is the scalability of the protocol whose performance can be severely affected by feedback mechanisms.

Another issue that multicast protocols must consider is that of fairness since such transmissions must avoid other competing flows from going into resource starvation and eventually congestion collapse. Further, the mechanism must also strive for good bandwidth utilization such that is able to attain a stable transmission throughput in the multicast session. In addition, the targeted application plays a major role in the design of a reliable multicast protocol. Interactive and data dissemination applications differ in their timely delivery (e.g., delay, delay jitter) and reliability requirements. Consequently, data dissemination multicast applications may tolerate higher de-

lays but require highly reliable delivery while interactive applications (e.g., teleconferencing) may tolerate some packet losses but require time-bounded data delivery (real-time).

The CC mechanisms presented here have been classified based on where the control is performed, i.e., the source, the receiver or both. The main drawback of the source-based approach is the requirement of feedback from receivers and the feedback implosion phenomenon that must be seriously taken into consideration when designing such protocol. In this context, one difficulty regarding the design of an efficient source-based CC mechanism is represented by the trade-off between the responsiveness of a given mechanism and the amount of feedback that particular mechanism requires for achieving it.

For instance, PGMCC and TFMCC use feedback suppression relative to the acker or the CLR while SFC employs a probabilistic feedback mechanism for their operation. Although these mechanisms are able to avoid the feedback implosion at the source, they suffer from poor performance during the start up phase as well as in large heterogeneous multicast groups where network conditions can rapidly change. On the other hand, S-RTCP and SAMM adopt another strategy, namely feedback aggregation although these solutions require added protocol complexity.

Because a multicast source cannot meet conflicting requirements of a heterogeneous group of receivers without feedback, another approach to multicast CC is represented by the receiver-based mechanism. In the receiver-based approach, the receiver itself chooses the level of subscription during a multicast session (layering). Further, layer-based (i.e., receiver-based) multicast sessions can achieve better fairness when compared to single-rate sessions [68]. The layering technique eliminates the feedback implosion phenomenon but it still needs to cope with the IGMP join and leave operations that often creates transient congestion states in a multicast session. In addition, the number of layers and their transmission rates may be impossible to control in these protocols due to application-specific requirements or due to the lack of feedback from the group receivers.

On the other hand, a hybrid CC mechanism has the main advantage that it can adapt better to rate mismatches between transmission rates and the dynamic rates that are required at receivers (granularity of the adaptation). Since in hybrid mechanisms, a part of CC is performed at the source, these protocols are able to receive feedback from receivers. Consequently, the transmission rate adaptation can help reducing the IGMP leave latency while the receivers can still subscribe to several layers. This can prevent receivers to leave some layers resulting thus in a decrease of network load and, implicitly, better network utilization. As an example, SMCC uses an additive increase layer join procedure instead of directly joining the next layer, reducing thus the occurrence of transient congestion due to failed join attempts.

Further, we presented here several solutions to error control for IP multicast. Error control is mainly performed by the transport protocol used in the multicast session. The techniques used for this are the traditional ARQ-based schemes that rely on the use of retransmissions (e.g., ACKs and NACKs) as well as various timers. One ingenious technique used for error control is the digital fountain FEC-scheme, which continuously multicasts a revolving stream of FEC parity packets allowing thus for optimal receiver efficiency.

However, both CC and error control mechanisms must address the strict constraints imposed by the multicast applications and must be able to transparently recover from packet losses and adverse network conditions. In addition, the main drawback of all IP-based multicast protocols is the required support from existing network nodes, i.e., the routers need to maintain group state as well as the need of (often) new network infrastructure – investments that most network operators do not make easily unless a clear demand for such services arise.

Chapter 4

Congestion and Error Control in Multicast Overlay Networks

4.1 Overlay Networks

By an overlay network we mean a network built on top of another network, as shown in Figure 4.1. The idea is that the underlying network provides basic communication services, which are used to offer advanced services in the overlay network. For example, in the beginning, the Internet was built on top of the telephone network, i.e., it used the communication services provided by the telephone network. IntServ and Peer-to-Peer (P2P) file-sharing networks also span some form of overlay network that use the best-effort service provided by IP.

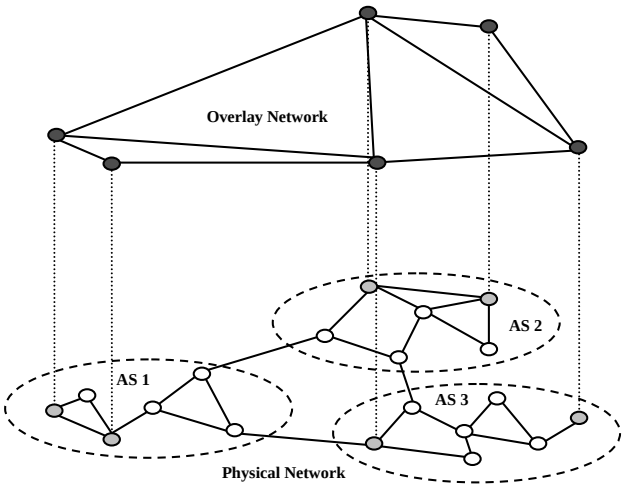


Figure 4.1: *Overlay Network.*

A direct connection between two overlay nodes is typically called a *link*. The *path* between two overlay nodes consists of one or several contiguous links. A link, which is one hop in the overlay, may involve several hops in the underlying network. This is the typical case where overlay paths are in general longer than those in the underlay.

4.2 QoS Routing in Overlay Networks

An interesting subject of Internet research has been the extension of the current best-effort architecture to adopt a framework for guaranteed QoS. Concerted efforts from industry and academia

have resulted in two QoS frameworks: IntServ and DiffServ. Both IntServ and DiffServ require that all intermediate routers support QoS [78]. Lack of interest on the part of ISPs has resulted in failure to adopt any architecture on a global scale. Consequently, the Internet consists of small islands of IntServ or DiffServ enabled networks in a sea of best-effort service as provided by IP.

New efforts to provide QoS have focused instead on use of overlay networks. The key advantage of these networks is the ability to offer some form of QoS routing on top of the shortest-path routing provided by IP. This implies choosing a path in the overlay network with specific constraints, e.g., packet loss ratio, delay, average bitrate. Two examples of overlay networks that provide QoS routing are Resilient Overlay Network (RON) and QRON [2, 48].

A RON denotes an application-layer overlay network that enhances path reliability. The overlay network consists of N fully connected nodes (i.e., the network contains $N(N-1)$ User Datagram Protocol (UDP) connections). Each RON node monitors its $N-1$ connections through a combination of passive observation and active probing. By default, the following metrics are considered: latency, packet loss rate, and throughput. The architecture allows also user-defined metrics. The nodes use a link-state routing protocol to disseminate topology and performance information. When data enters the RON, the entry node will consult its routing table and forward the packet to the "best" RON node according to the policy set for destination. During a 72-hour system evaluation, RON was able to detect and route around all path outages occurring in the overlay network. In addition, the RON improved loss rate and throughput.

The architecture of QRON assumes the presence of one or several Overlay Brokers (OBs) in each Internet autonomous system. An OB is similar to a RON node, providing services such as resource management and overlay routing. The OBs interconnect thus forming an overlay network. Additionally, they perform active measurements to discover available bandwidth, loss rate and delay for overlay connections. This information is periodically broadcasted. To enable the overlay network to scale to a large number of nodes, the OBs are hierarchically organized into clusters. When a route request arrives at the overlay, QRON attempts to find a path that corresponds to the QoS constraints of the request while simultaneously balancing the traffic load among the overlay links. If the overlay is unable to maintain the QoS constraints along the current path, QRON is able to detect the problem and find an alternate path. Simulation has shown that the path-selection algorithms used by QRON perform much better than shortest-path algorithm.

4.3 Multicast Overlay Networks

Although IP multicast was introduced more than a decade ago, it has not yet achieved its desired deployment [22]. However, in recent years a new concept has emerged, which is known as *overlay multicast* also known as *Application Layer Multicast (ALM)*.

The main idea behind an overlay multicast network is to implement all necessary multicast functionality at the end-hosts instead of network routers as is the case with traditional IP multicast. Thus, the participating end-hosts form a multicast overlay network, acting on top of the underlying physical network. Packets are replicated and distributed by the multicast application running at the end-hosts as opposed to being replicated and forwarded inside the IP network. Consequently, in order to have a proper operation, the overlay network must establish and maintain an efficient *data plane* and an associated *control plane*. It must be mentioned that even if the multicast service is implemented and run at the application layer, the overlay network uses the physical network for the actual data transmission.

Since overlay multicast is operating at the application level, there are two main disadvantages, i.e., a higher delay and less efficient utilization of the available bandwidth. This is mainly because a packet in an overlay network may cross the same physical link several times before reaching the intended recipient. In this context, two parameters (metrics) are widely employed in order to assess the quality of data delivery for an overlay path. These are the *stress* of a link belonging to the data plane (i.e., the physical underlying network), and the *stretch* of an overlay path between the source and a given destination.

The stretch of a path is defined as the ratio between the delay of the path in the overlay

network and the minimum delay achievable at the network layer for the same path. The stress of a link is defined as the total number of identical packets transmitted over the same link due to the overlay network. The *efficiency* of an overlay multicast tree, based on the stress metric, is often used in order to evaluate its efficiency in real networks [15]:

$$\delta = 1 - \frac{L_m}{L_u} \tag{4.1}$$

where L_m is the defined as the total number of links traversed in the overlay multicast distribution tree and L_u is the total number of all unicast links traversed to reach all multicast group members. Consequently, δ represents the percentage gain of using multicast instead of unicast transmission. Thus, a value of δ close to 0 means that employing multicast instead of unicast provides little advantage (i.e., bandwidth efficiency) as is the case of widely distributed multicast groups. In a similar manner, a value of δ approaching 1 means that all receivers are sharing a single multicast overlay path resulting thus in maximum bandwidth efficiency in the overlay distribution tree [15].

In order to manage the overlay network, the participating overlay nodes must exchange *refresh* messages with the neighboring overlay peers, resulting in overhead control traffic experienced by an overlay network. This aspect must be seriously taken into consideration to attain a scalable overlay.

4.4 Challenges

Several attributes are required from an overlay multicast so as to provide a feasible solution over traditional IP multicast. At the same time, each attribute generates several challenges that must be addressed. Consequently, an overlay multicast must be *scalable* since this is one of the major drawbacks of conventional IP multicasting and one of the main reasons behind the overlay multicast solution. In addition, an overlay multicast solution should provide a *performance* comparable or at least not much worse than conventional IP multicast as otherwise it will not provide the viable solution it was intended to be.

Another challenge that researchers and multicast service providers must keep in mind is that an overlay multicast is required to quickly *adapt* to changes affecting the underlying network and to *react* in an appropriate manner so as to provide an uninterrupted application communication between its intended recipients. Moreover, the overlay multicast must be *robust* with regard to multicast group dynamics and multicast tree topology changes created for instance by node failures or churn behavior of the overlay nodes.

Further, the overlay multicast should also supply extra services or features that traditional IP multicasting does not provide such as for instance *security* of data transmission. Finally, in order to get a widespread acceptance by both end-users and diverse service providers, the overlay multicast must provide *reliable* communication regardless of the application that employs it.

4.5 Congestion Control

In general, traditional IP multicast requires a rather intricate support from the network devices and the multicast protocols to provide the control framework necessary in the handling of large multicast member groups. IP routers must maintain separate states for the different multicast groups (which violates the stateless concept on which Internet was built) resulting thus in the limited deployment of IGMP and IP multicast.

An increasing number of emerging applications such as web cache replication, private chat rooms or multi-party on-line games often contain a small number of group members with often dynamic group creation and do not benefit from conventional IP multicast. The control complexity required for group management severely limits the application performance. In this context the employment of an application-based multicast approach emerged as a practical solution. Overlay

multicast solutions employ *middleware* implementations, which allow for a faster deployment of such applications and allows for more flexibility regarding application specific requirements such as data coding, scheduling, error and flow control, or specific security demands.

Consequently, application layer multicast has emerged as a feasible solution for providing multicast services despite the fact that it may not attain the efficiency of network bandwidth utilization compared to traditional IP routing. The employment of an overlay multicast solution for data distribution also greatly reduces the multicast CC problems of IP multicast. The requirement of an e2e CC mechanism between the multicast source and the receivers can be reduced to a unicast CC performed between neighboring overlay nodes.

In the following paragraphs, several solutions for overlay multicasting CC mechanisms are presented.

4.5.1 Overcast

The Overcast solution introduced by Jannotti *et al.* [39] is an application-layer multicast scheme using TCP connections between Overcast nodes by using the HyperText Transfer Protocol (HTTP) protocol. This implies that no changes must be made to an existing IP network, except the introduction of supplementary Overcast servers. The multicast groups are represented in Overcast through HTTP URLs, where the hostname represents the root of an Overcast network and the path represents a distinct multicast group [39]. The use of HTTP allows for a greater flexibility in Overcast since hierarchical addressing can be employed.

Overcast is comprised of a multicast source, any number of Overcast nodes and standard HTTP clients. The multicast tree in Overcast is rooted at the source and adapts to changes in the underlying network. Consequently, Overcast allows for scalable and reliable multicast overlay multicast. Furthermore, the multicast source may be replicated in Overcast allowing thus for higher fault tolerance through redundancy. When multiple sources are used, they are replicated at the highest level so to minimize the risk of single node failure. The Overcast distribution topology is depicted in Figure 4.2.

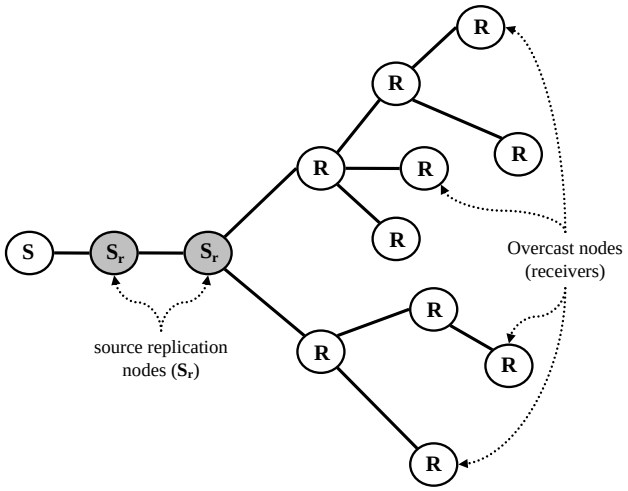


Figure 4.2: *Overcast Distribution Network.*

The multicast tree is created and rebuilt through an "up/down" protocol [39]. The "up/down" protocol attach first new nodes directly to the root of the multicast tree and then periodically moves them down or up from their current position so as to minimize the hop count and the latency. Each Overcast node, including the root, maintains information about all nodes situated at a lower level as well as a log of all tree changes. In this manner, the root has the most up-to-date information about the tree hierarchy.

The reliability for data transmission in Overcast is achieved through TCP (hop-by-hop). The e2e reliability and loss recovery due to tree partitions or node failures is provided through a log-based mechanism. Each Overcast node is required to maintain a log of the data it received for a predefined time period. When tree partitions or node failures occur, the remaining Overcast nodes detect the losses and retransmit to its downstream tree branch. Since Overcast is a TCP-based solution, the TCP-friendly behavior is automatically attained.

4.5.2 Reliable Multicast proXy

A hybrid approach to multicast communications was introduced by Chawathe *et al.* [16] in the form of the Reliable Multicast proXy (RMX) solution. The RMX combines the unicast reliability provided by TCP with the efficiency of multi-point communication given by IP multicast. The authors employ a "divide-and-conquer" paradigm to attain e2e reliability by dividing large heterogeneous multicast groups into homogeneous *data groups* proximally co-located. Data groups are coordinated into a multicast spanning-tree of overlay TCP connections by a set of application-aware agents denoted as Reliable Multicast proXys (RMXs) [16].

The multicast sources transmit the data to the RMX belonging to its local group, and the RMX then forwards it to the rest of the data group. The RMXs have detailed knowledge about the application semantics allowing for an adaptation to the heterogeneity requirements of the data group. Thus, a receiver that has high capabilities is able to maintain its reception rate.

The RMXs also communicate with other RMXs from other data groups and are able to forward data to/from their local groups to other local groups/RMXs. The architecture of RMX is based on *scattercast*, which employs the partitioning of a multicast session into several smaller groups connected by TCP unicast links. The scattercast concept is depicted in Figure 4.3.

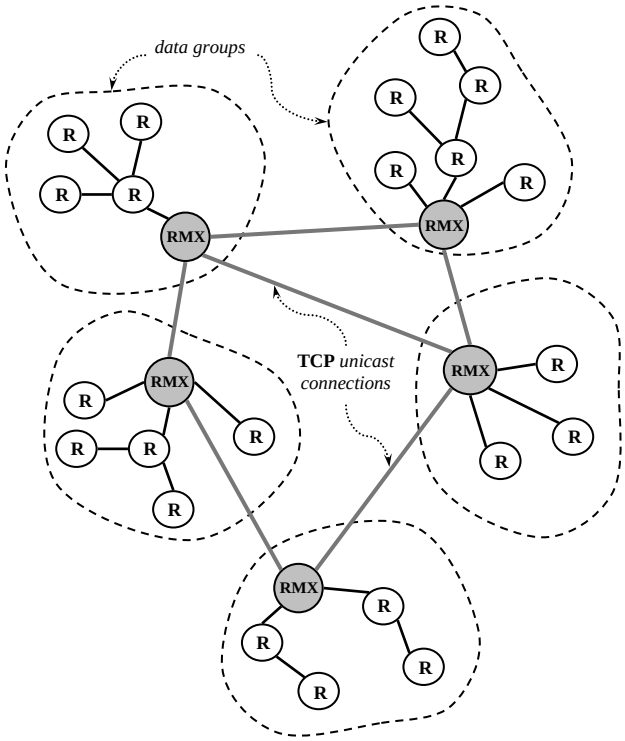


Figure 4.3: *RMX Scattercast Architecture.*

Scattercast is a delivery-based data forwarding service for multi-point communication and uses a *mesh-first* routing algorithm. RMX participates both in the multicast communication of its local data group and it maintains connection at the same with other RMXs. Within the local

group IP multicast can be employed for an increased communication efficiency, or an application multicast solution may be applied if the underlying network of the local group does not support IP multicast [16].

Reliability within the RMX framework is achieved through the use of TCP (inter-RMX) and SRM [27] (intra-RMX) as underlying transport protocols. Moreover, RMX allows the applications to define own requirement semantics providing thus for a great flexibility. This is achieved through the use of the so-called *asmods* (Application-Specific MODules), which are mainly dynamic code libraries loaded at RMX run-time. This allow for the use of application specific e2e reliability requirements.

For example, if a receiver requires retransmission, the request is first multicasted to the data group by the RMX. If the requested data is not available within the local group, the RMX forwards the request up the hierarchy toward the multicast source. Thus, when an RMX receives data on a link, it forwards the data to all remaining links including its local data group through a spanning-tree flooding. Consequently, the data eventually propagates to the entire multicast session. In order to limit the scope of data retransmission, RMX employs the Pragmatic General Multicast (PGM) protocol [16, 75].

The RMX is allowed to transform the data before it forwards it to other nodes. Nevertheless, in order for the receivers to distinguish between different data, the authors implement a Transport-Independent Naming System (TINS). TINS identifies the transmitted data, regardless of the transport protocol responsible for its delivery, allowing thus for an application to implement own semantics regarding data delivery or reliability requirements. Hence, an application is able to identify its data and can employ its own measures against data loss [16].

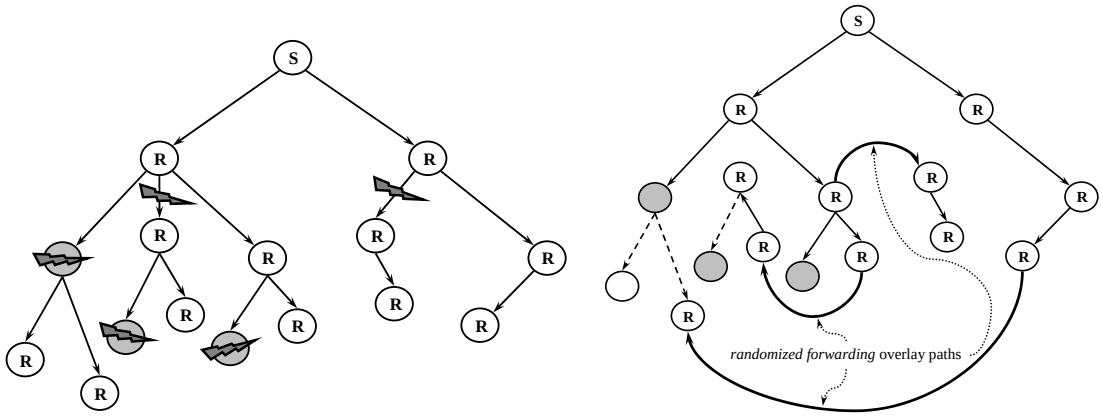
4.5.3 Probabilistic Resilient Multicast

One major challenge in the design of a reliable ALM protocol is the provisioning of fast recovery from overlay node failures (or departures), which often partition the multicast delivery tree. The overlay node failures cause data loss for the downstream nodes, especially if the failed overlay node is not a leaf node in the overlay multicast tree. Furthermore, recovery latency induced by loss recovery mechanisms based on ACK/NACK schemes may render an application unusable. This is even more important in the case of an ALM solution since node failure at the overlay level take, in general, longer time to be detected than normal network packet losses.

The Probabilistic Resilient Multicast (PRM) provides an overlay multicast data recovery scheme that employs both a proactive and a reactive component so as to provide a resilient solution that maintains a low e2e latency [6]. The proactive component in PRM is represented by a *randomized forwarding* that allows the overlay nodes to choose uniformly randomly a fixed number of other overlay nodes and forward data to them with a low probability. The randomized forwarding is employed along with the usual forwarding. Consequently, some duplicate packets are created by this mechanism but the duplicates are detected and removed through the employment of sequence numbers. The randomized forwarding, although it introduces small overhead into the ALM, it allows for high packet delivery ratios even for high overlay node failure rates.

The proactive component implemented by PRM is provided by *triggered NACKs* that allow for packet loss detection and recovery. The proactive component is primarily intended for loss recovery caused by link errors and network congestion. Implementation of the proactive component is provided through the use of piggybacked bit-masks within the transmitted packets. Hence, packet loss is detected through gaps in the bit-mask. This allows for information exchange both between neighboring nodes in the multicast tree as well as between nodes involved in the randomized forwarding. The randomized forwarding and the recovery from node/link failures is illustrated in Figure 4.4.

In PRM, each overlay node employs a periodical discovering mechanism, based on a *random-walk* on the multicast tree, of other overlay nodes for randomized forwarding. Consequently, in case of a node failure, although a node that received packets through randomized forwarding does not have the complete set of data, is able to evaluate the loss based on the sequence numbers, which, in turn, allows for a faster recovery through the triggered NACKs.



(a) PRM: Node/Link Failures. (b) PRM: Randomized Forwarding Recovery.

Figure 4.4: *PRM Randomized Forwarding Recovery Scheme.*

The authors of [6] also proposed some extensions for the PRM in the form of *loss correlation* and *ephemeral guaranteed forwarding*. Loss correlation provides lower path correlation with the root of the multicast tree between two nodes involved in randomized forwarding. The ephemeral guaranteed forwarding is intended as an extension of the triggered NACK mechanism. A node that detects a gap in the sequence number in the packet received through randomized forwarding, can temporarily request the forwarding node to increase its forwarding probability towards itself for a shorter period of time.

4.5.4 Application Level Multicast Infrastructure

Another solution for reliable overlay multicast was proposed by Pendarakis *et al.* [61], namely the Application Level Multicast Infrastructure (ALMI). In ALMI, the overlay links may use either UDP or TCP as transport protocol. If the multicast application requires it, TCP may be used to provide point-to-point reliability between the overlay nodes. The use of TCP eliminates packet losses caused by network congestion or transmission errors. Nevertheless, an e2e reliability mechanism is still needed such that ALMI can recover from overlay node failures that impact on the multicast distribution tree.

ALMI provides an implicit control of implosion and retransmission exposure, without the need of router support from the underlying network. When a multicast group member detects a loss, it transmits a NACK on the same interface that received the data. The requests are aggregated at each hop and only one request is forwarded from the multicast sub-tree that experienced packet loss.

The application may provide buffering capabilities and in this case, if data is available at the parent node, the retransmission is handled locally. Otherwise, the parent node transmits a `NODATA` message to the requesting node, which in turn initiates an out-of-band connection directly to the multicast source and requests the retransmission of the missing data. The retransmission takes place over this connection. As soon as the retransmission is completed, the out-of-band connection is closed. Regardless of retransmission procedure (local or out-of-band), the overlay node forwards the retransmitted data to other nodes downstream that requested the same data [61].

Furthermore, ALMI employs the use of ACKs to synchronize the data reception state at the multicast group members. This is done to meet the requirements of applications that need e2e reliability but cannot provide the overlay nodes with enough buffering space. In these cases, the overlay nodes must ensure that the packets available in the buffer are correctly received by all group members. The ACK messages in ALMI comprise a <*source, sequence number*> pair that indicates the highest, contiguous sequence number received from the source.

The transmission of ACK messages is initiated by the leaf nodes of the multicast tree and are aggregated at each upstream branch node. When the upstream node receives ACKs from all children it forwards upstream the minimum sequence number for each source. When the source receives the ACK message, it multicasts downstream the message containing the common minimum and upon the receipt of this message, a node is allowed to remove the packets that have been received by all group members. The transmission of ACK messages is regulated by both the data rate of the transmission as well as the buffer size available [61].

4.5.5 Reliable Overlay Multicast Architecture

An approach based on loosely coupled TCP connections between overlay nodes intended primarily at high speed bulk transfers for reliable transfers in Content Distribution Networks (CDNs) is represented by Reliable Overlay Multicast Architecture (ROMA) [44]. ROMA implements FEC techniques, which allows for a control mechanism decoupled from reliability. ROMA is also able to handle asynchronous joins of multicast group members, difference in download speeds as well as to dynamically adapt to tree topology changes.

The loosely coupled TCP connections between the nodes in the overlay network are realized in ROMA through a departure from the classical *store-and-forward* paradigm, where an intermediate node forwards all received packets to the downstream node, and employing instead a *forward-when-feasible* approach characterized by the fact that an intermediate node immediately forwards those packets that can be written into the downstream TCP socket [44]. Consequently, an upstream TCP connection may possibly impact on the performance of a downstream TCP connection. However, a downstream TCP connection does not impact at all on the performance of its upstream TCP connections. The data lost when packets cannot be written to the TCP socket is recovered through the use of erasure resilient codes [44].

The forward-when-feasible paradigm implemented by ROMA combined with erasure-resilient codes (i.e., FEC) reduces the need for large buffers at the application level. Further, the application maintains a buffer managed as a circular queue together with a small overflow buffer for each outgoing TCP connection, as illustrated in Figure 4.5.

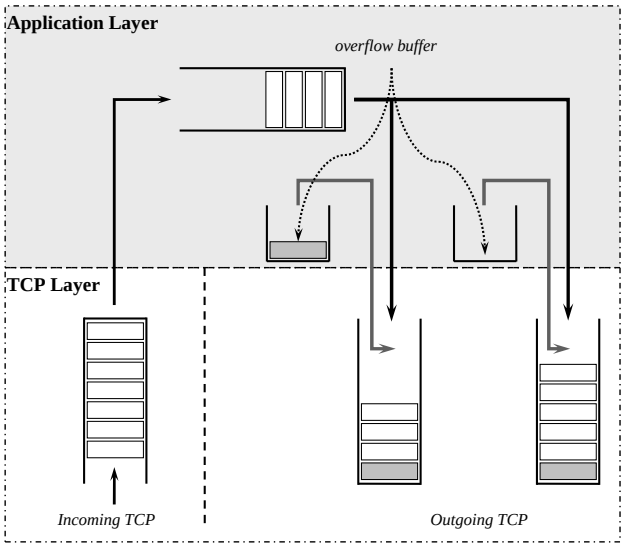


Figure 4.5: *ROMA: Overlay Node Implementation.*

Data packets are encoded before transmission and decoded at each receiver. The process does not introduce a large decoding overhead. The overflow buffers store the bytes that could not be written to the outgoing TCP socket and the contents of these buffers are written first. This guarantees that encoded symbols are contiguous before transmission. Slow downstream links are

compensated by allowing a node to drop packets and instead to send additional encoded packets to the receiver.

When the original content is completely received at a node, the node can leave the ROMA multicast group by closing its TCP connections. If the node still remains within the group it may choose to either dispatch source encoded content to its downstream connections, or it can close its upstream TCP connection, generate its own encoded symbols from the full (already available) content and transmitting them along its downstream TCP source (acts as a pseudo-source) [44].

4.5.6 Overlay MCC

The Overlay Multicast Congestion Control (MCC) solution employs the use of a fixed-size window-based CC protocol through the use of unicast TCP overlay links among neighboring nodes combined with a window-based flow control at each overlay node [82]. Similar to ROMA, the Overlay MCC implements an outgoing buffer for each outgoing TCP connection realizing thus a loosely coupled mechanism.

The decoupling is not complete so as allowing the source to still apply rate control in the case of link congestion. Consequently, a *back-pressure* mechanism is implemented at each node through a fixed-window control mechanisms acting between the input and output buffers in the overlay node. The size of the back-pressure mechanism, denoted as W_f^0, prevents a node to forward additional packets until all packets from the window have been written to the output buffers. Essentially, this is just flow control between the input and the output buffers at a given overlay node.

The coupling between a receiver and the source is realized by limiting the size of the input buffer W_c^0 and transmission of ACKs messages to parent nodes for each packet that can be read into the input buffer. Hence, a packet is read into the input buffer only if all output buffers can store it. When the input buffers start to fills up, no ACKs are transmitted to the upstream node, which senses the back-pressure and this propagates all the way back to the source. Each node in the Overlay MCC must be able to buffer W_c^0 packets to avoid unnecessary packet loss because this is the total number of packets that may be in transit between two adjacent nodes. The implementation of an Overlay MCC node is depicted in Figure 4.6.

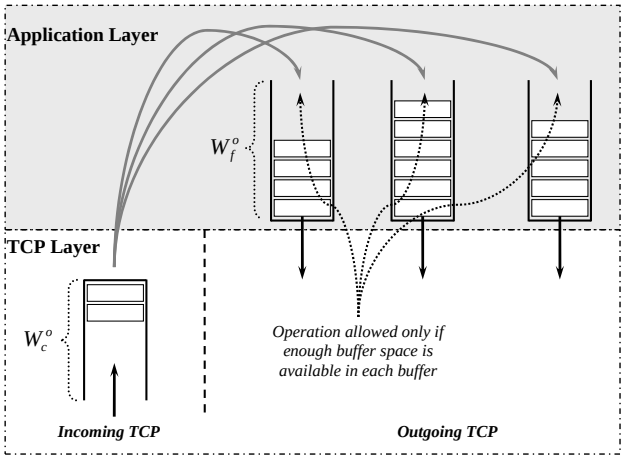


Figure 4.6: *Overlay MCC: Node Implementation.*

The Overlay MCC employs a similar paradigm to divide-and-conquer by dividing the e2e CC mechanism into local control loops performed at each node with additional control loop per multicast tree branch. In essence, the size of the window W_f^0 regulates the amount of control coupling and consequently the speed of the backpressure propagation within the system. An infinite value of W_f^0 results in a complete decoupled system.

The authors of [82] showed through formalized max-plus algebra computations and numerical results obtained through simulations that there is no significant throughput degradation with

an increasing number of receivers for the Overlay MCC model. These are encouraging results, however the applicability of this model in the case of large, geographically scattered, overlay multicast groups was left for further research and study.

4.6 Error Control

Similar to the error control performed in the case of traditional IP multicast (described in Section 2.3), overlay multicast solutions employ both ARQ error control mechanisms and error-resilient coding provided by FEC techniques. FEC schemes are preferred in the case of overlays since the overlay nodes are mainly application processes run on conventional end-hosts allowing thus for a greater flexibility in the requirements for the application. This is because the end-hosts have an increasing capability in the form of large storage capacity, available memory and processing power.

Moreover, newer and improved FEC codes [49], often allow for an unbounded number of encoding symbols to be generated on demand. Such codes provide also good probabilistic decoding guarantees together with low overhead.

4.6.1 Joint Source-Network Coding

Along with the error control schemes described so far, an interesting FEC technique was introduced by Shan *et al.* in their proposed scheme intended for video streaming, namely the Joint Source-Network Coding (JSNC) [71].

JSNC provides a scalable, efficient and near optimal adjustment of both encoded video and error control coding, to heterogeneous receivers. JSNC introduces two new concepts, namely *integrated video coding* (IVC) and *fine granular adaptive*-FEC (FGA-FEC). The video coding functions are dispersed through IVC across both source and network, which allows for heterogeneous receiver adaptation while the FGA-FEC coding scheme works in cooperation with IVC to adapt the coded stream to the heterogeneity of the receivers. Further, JSNC does not require decoding/recoding at the intermediate overlay nodes.

The authors show through simulation results that the adaptation quality of JSNC is comparable with pure source coding techniques while the employment of the FGA-FEC error recovery technique adapts the FEC codes at intermediary overlay nodes by only adjusting the packet size.

4.7 Concluding Remarks

The main advantage of an ALM approach is represented by its design flexibility and the capability of implementing locally an efficient CC algorithm. This is because of the capability of an end-host, which often has more computing resources than a network device, to perform resource-demanding tasks locally.

ALM can be efficiently used as an effective multicast service that can be deployed directly although it does not achieve the same network utilization as traditional IP multicast. Related to this issue, an ALM solution must deal with the overhead caused by soft-state exchanges and to limit the unnecessary packet replication inherent in all ALM approaches.

Consequently, the main challenge faced by the overlay multicast is its necessary awareness of the underlying network topology such that a performance comparable with IP multicast is attained. Further, feedback aggregation is more easily implemented in ALM and does not require special features from the available network devices. The aggregation is done at end-hosts but of critical importance is a correct feedback compression that must be achieved by the clustering algorithm such as to reduce the amount of information transmitted back to the source. At the same time, the aggregation must be still able to reveal changing network conditions such that the source can adapt the transmission rate to prevailing congestion conditions.

Several performance metrics that impact on the multicast overlay must be taken into consideration, such as link stress, link stretch, efficient bandwidth utilization and protocol robustness.

Further, overlay topology discovery and recovery, overlay routing, protocol performance as well as security related issues are some of the concerns that an overlay solution must carefully consider.

The multicast overlays emerged as viable solutions to group communication while their adaptability to changing network conditions and robustness to group members join and leave operations allow for higher scalability. A major advantage of overlay multicast is that these solutions do not require any special support from network routers and can therefore be quickly deployed as application-specific libraries with no requirements for special standardization.

In this chapter, we presented several contributions to CC and error control schemes used in the case of multicast overlay environments. The solutions presented have both advantages and disadvantages and their performance is often dependent on the particular type of multicast application employed. Error control is performed in multicast overlay networks in a similar manner to traditional IP multicast, i.e., ARQ-based and FEC-based mechanisms. The highlights of the overlay multicast solutions discussed in this chapter are presented below.

Overcast Reliability in Overcast is attained through the built-in overlay links, which are TCP-based. The multicast group in Overcast is represented by a HTTP URL and because the Overcast topology is based on a single-source multicast tree, this allows for easier overlay topology optimization. Although the root of the tree may provide a single point of failure in Overcast, this is replicated for increased protocol reliability. The root is also allowed to forward data originating from other sources. In addition, Overcast provides tree adaptation and fault tolerance in the case of overlay node failures or leave operations through the use of a "up/down" protocol [39].

However, an Overcast network suffers from delay and jitter induced by the hop-by-hop TCP connectivity making thus an improper choice for real-time multimedia applications that may require strict upper bounded delay. Moreover, Overcast does not provide e2e flow control due to its decoupled nature and may also have a high stress of the overlay links.

RMX RMX can be viewed as a hybrid solution between application-level and IP multicast. An interesting approach of RMX is that it splits a heterogeneous multicast session in several homogeneous data groups based on proximal location. Consequently, the data groups may implement own CC and multicast reliability mechanisms, which can be either IP-based if the underlying network infrastructure allows it, or overlay-based otherwise.

The reliability among data groups is achieved through a spanning tree of TCP links between the RMXs. Moreover, special application modules allow an overlay node to define application-specific reliability requirements while overlay routing is done through scattercast [16].

Although RMX provides a lot of flexibility, one drawback of this system is that it necessitates an infrastructure of carefully placed proxies (the RMXs), which may limit its use in case of large-scale multicast applications spanning several network providers.

PRM The PRM can be implemented on top of any ALM protocol and while it cannot provide perfect reliability, it does provide high data delivery rates. PRM implements both a proactive reliability solution through its randomized forwarding, and a reactive reliability scheme through the use of NACKs, which are triggered by a gap in the received sequence number at an overlay node [6]. Moreover, a loss correlation extension of PRM allows for higher efficiency of the protocol, while the ephemeral guaranteed forwarding extension allows protocol adaptation in case of high loss rates. Among the main advantages of PRM are its high data delivery ratios and the guaranteed low latency bounds.

In addition, PRM provides a scalable solution for multicast overlays since it is able to increase the data delivery ratio in the ALM environment while keeping a low overhead and low e2e latency. Further, PRM adapts quickly to changes in the multicast group membership and it is able to handle adverse network conditions that incur high packet losses.

ALMI The ALMI architecture proposed in [61] provides point-to-point reliability at the overlay level when the overlay links use TCP as transport protocol. If UDP is used, an e2e reliability

mechanism must be employed instead. ALMI implements hop-by-hop aggregation of retransmission requests (i.e., NACKs) and local recovery is first attempted at the upstream node. If this is not possible, ALMI nodes use a temporary out-of-band direct link to the multicast source.

ALMI also uses ACKs for synchronizing the reception state of the group members. The ACK transmission rate in ALMI depends on the data rate of the multicast group and the available buffer space at the slowest receiver. The main drawback of ALMI is related to its scalability since this solution is intended for the support of relatively small multicast groups (several tens of members), which may limit its deployability.

ROMA An interesting solution for a multicast overlay architecture is proposed in ROMA [44]. Reliability in ROMA is achieved through the use of loosely-coupled TCP links at the overlay level and the use of the digital fountain FEC-scheme. In addition, ROMA adopts a forward-when-feasible data transmission paradigm, eliminating thus the need of large buffers. ROMA models the multicast overlay as a chain of TCP links and the overall group throughput is maximized by finding overlay routes that provide a better TCP throughput than the default IP link. Multirate multicast transmissions are also possible in ROMA.

The main advantage of ROMA stems from its use of FEC, which allows it to have small buffers at the overlay nodes as well as to easily adapt to topology changes. At the same time, because it uses FEC, ROMA may suffer from performance degradations.

Overlay MCC Another solution that provides reliable multicast in an overlay network is Overlay MCC [82]. Similar to ROMA, reliability in Overlay MCC is attained through loosely-coupled TCP overlay links. Further, Overlay MCC implements a back-pressure mechanism in order to allow rate adaptation in case of congestion. The protocol uses a window-based flow control and the feedback loop is separated into multiple control loops (one loop per branch of multicast tree). These control loops are chained and both flow and CC are coupled through the use of buffers and controlled write operations at TCP sockets.

Overlay MCC shows good throughput performance even with larger numbers of multicast group members and the use of TCP overlay links ensures also its effectiveness and fairness. However, since this solution is only applicable for TCP-based overlay links, it can suffer from scalability issues when UDP is used since flow and CC is decoupled in this case.

Chapter 5

Conclusions and Future Work

This report has presented a brief overview of the main issues and challenges regarding conventional IP multicast with the main focus on CC solutions and its related protocols. Further, overlay multicast or ALM was introduced as a feasible solution to IP multicast due to the deployment issues faced by the latter. The solutions and protocols presented for ALM focused mainly on the reliability and CC mechanisms they propose.

One major advantage of the ALM solutions over traditional IP multicast is their ability to implement CC mechanism on a hop-by-hop basis which allows for greater flexibility compared to IP-based approaches. However, in order to achieve acceptance from both end-users and service providers, ALM must solve several issues, namely: an ALM-based solutions must have performance comparable to that provided by IP-based implementations. The solution must be robust to handle ungracious join/leave operations of the group members that partition the multicast distribution tree and they must adapt well to the traffic conditions of the underlying network such as not to starve communication streams that share the same links as ALM.

In this context, several IP and overlay multicast protocols and implementations were discussed so as to provide a better and clearer view of these solutions with regard to reliability, CC and error control mechanisms they use.

The overview of the available solutions presented in this report lead us to the conclusion that each protocol or mechanism introduced so far try to solve a particular problem (or is designed for a specific application) and none of them can be regarded a a general solution. This fact was not unexpected and it supports our belief that a solution must solve a particular need and different services require different demands from the network providing these services.

5.1 Future Work

Our future research will focus on routing in overlay networks with the aim to find mechanisms that provide soft QoS guarantees for different applications that use overlay networks as communication paradigm. We plan to implement a virtual testbed that may be employed as a research framework that will address several issues of overlay networks:

- Identify traffic invariants in overlay networks via traffic measurements to establish adequate and credible traffic models that capture and characterize traffic burstiness and correlation.
- Use these traffic models to implement and validate mechanisms and solutions that improve diverse overlay application requirements such as provisioning of CC, optimal overlay route selection, multicast tree adaptation and QoS-aware overlay routing.

In addition, this report will form the ground for further research work on reliable and QoS-aware multicast overlay networks, which will culminate with three doctoral dissertations at the Dept. of Telecommunication Systems at BTH. Consequently, our planned future work is to develop a dedicated middleware environment, which will be used to develop new protocols for

multimedia distribution over IP and to offer soft QoS guarantees for specific applications in a multicast environment. We are also planning to develop analytical and simulation models to validate our results.

Further, we plan to join PlanetLab [62] to test our proposed solutions in a real network environment allowing thus for the validation under real conditions of our CC algorithms and QoS-aware overlay routing mechanisms.

Appendix A

Acronyms

ACK	Acknowledgment
AIMD	Additive Increase Multiplicative Decrease
ALM	Application Layer Multicast
ALMI	Application Level Multicast Infrastructure
AQM	Active Queue Management
ARQ	Automatic Repeat reQuest
ASM	Any-Source Multicast
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
BGMP	Border Gateway Multicast Protocol
BGP	Border Gateway Protocol
BTH	Blekinge Institute of Technology
CBT	Core Based Tree
CC	Congestion Control
CDN	Content Distribution Network
CLR	Current Limiting Receiver
CPN	Cognitive Packet Network
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	CSMA with Collision Avoidance
CSMA/CD	CSMA with Collision Detection
DLC	Data Link Control
DLL	Data Link Layer
DiffServ	Differentiated Services
DV	Distance Vector
DVMRP	Distance Vector Multicast Routing Protocol
e2e	end-to-end
ECN	Explicit Congestion Notification
EXPRESS	Explicitly Requested Single-Source Multicast
FCFS	First Come First Served
FCS	Frame Check Sequence
FEC	Forward Error Correction
FIFO	First In First Out
FLID-DL	Fine-grained Layered Increase/Decrease Dynamic Layering
HALM	Hybrid Adaptation Layered Multicast
HDVMRP	Hierarchical Distance Vector Multicast Routing Protocol
HTTP	HyperText Transfer Protocol
IA	Intermediate Agent
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
IntServ	Integrated Services
JSNC	Joint Source-Network Coding
LAN	Local Area Network

LDA+	Enhanced Loss-Delay Adaptation
LVMR	Layered Video Multicast Retransmission
MAC	Media Access Control
MAAS	Multicast Address Allocation Server
MBone	Multicast Backbone
MCC	Multicast Congestion Control
MLDA	Multicast Loss-Delay Adaptation
MOSPF	Multicast Extensions to Open Shortest Path First
MPLS	Multi Protocol Label Switching
MSM	Multiple Source Multicast
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NACK	Negative ACK
NETBLT	NETwork BLock Transfer
OB	Overlay Broker
OSPF	Open Shortest Path First
P2P	Peer-to-Peer
PIM	Protocol Independent Multicast
PIM-DM	Protocol Independent Multicast – Dense Mode
PIM-SM	Protocol Independent Multicast – Sparse Mode
PGM	Pragmatic General Multicast
PGMCC	Pragmatic General Multicast Congestion Control
PLM	Packet-pair Layered Multicast
PRM	Probabilistic Resilient Multicast
QoS	Quality of Service
RAMP	Reliable Adaptive Multicast Protocol
RED	Random Early Detection
RFC	Request For Comments
RIP	Routing Information Protocol
RLC	Receiver-driven Layered Control
RLM	Receiver-driven Layered Multicast
RMP	Reliable Multicast Protocol
RMX	Reliable Multicast proXy
ROMA	Reliable Overlay Multicast Architecture
RON	Resilient Overlay Network
RP	Rendezvous Point
RPB	Reverse Path Broadcasting
RPF	Reverse Path Forwarding
RPM	Reverse Path Multicasting
RR	Receiver Report
RSV	Resource ReSerVation
RTCP	Real-time Transport Control Protocol
RTO	Retransmission TimeOut
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SA	Subnet Agent
SACK	Selective ACK
SAMM	Source Adaptive Multi-layered Multicast
SARC	Source-channel Adaptive Rate Control
SDH	Synchronous Digital Hierarchy
SFC	Scalable Feedback Control
SM	Simple Multicast
SMCC	Smooth Multirate Congestion Control
SP	Synchronization Point
SR	Sender Report
SRM	Scalable Reliable Multicast
S-RTCP	Scalable RTCP
SSM	Source Specific Multicast
TCP	Transport Control Protocol
TFMCC	TCP Friendly Multicast Congestion Control
TRPB	Truncated Reverse Path Broadcasting
TTL	Time to Live
UDP	User Datagram Protocol

URL	Uniform Resource Locator
VC	Virtual Circuit
VoD	Video-on-Demand
WAN	Wide Area Network
WFQ	Weighted Fair Queueing
WRR	Weighted Round-Robin
XTP	Xpress Transport Protocol

Bibliography

[1] Adams A., Nicholas J. and Siadak W., "Protocol Independent Multicast – Dense Mode (PIM-DM): Protocol Specification (Revised)", *IETF RFC 3973*, 2005.

[2] Andersen D., "Resilient Overlay Networks", *Tech. Report*, Massachusetts Institute of Technology, 2001.

[3] Albuquerque C., Vickers B. and Suda T., "An End-to-End Source Adaptive Multi-Layered Multicast (SAMM) Algorithm", *Tech. Report*, University of California, ICS-TR 98-31, 1998.

[4] Aweya J., Ouellette M. and Montuno D. Y., "A Control Theoretic Approach to Active Queue Management", *Computer Networks*, Vol. 36, No. 2-3, pp. 203–235, 2001.

[5] Ballardie A., "Core Based Trees (CBT) Multicast Routing Architecture", *IETF RFC 2201*, 1999.

[6] Banerjee S., Lee S., Bhattacharjee B. and Srinivasan A., "Resilient Multicast using Overlays", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 31, No. 1, pp. 102–113, 2003.

[7] Bhattacharyya S., "An Overview of Source-Specific Multicast (SSM)", *IETF RFC 3569*, 2003.

[8] Bertsekas D. and Gallager R., *Data Networks*, 2^{nd} *Edition*, Prentice Hall, 1992.

[9] Bolot J-C., Turletti T. and Wakeman I., "Scalable Feedback Control for Multicast Video Distribution in the Internet", *Proceedings of ACM SIGCOMM 1994*, pp. 58–67, 1994.

[10] Braden R., "Requirements for Internet Hosts – Communication Layers", *IETF RFC 1122*, 1989.

[11] Braden B., Clark D., Crowcroft J., Davie B., Deering S., Estrin D., Floyd S., Jacobson V., Minshall G., Partridge C., Peterson L., Ramakrishnan K., Shenker S., Wroclawski J. and Zhang L., "Recommendations on Queue Management and Congestion Avoidance in the Internet", *IETF RFC 2309*, 1998.

[12] Braudes R. and Zabele S., "Requirements for Multicast Protocols", *IETF RFC 1458*, 1993.

[13] Byers J. W., Luby M., Mitzenmacher M. and Rege A., "A Digital Fountain Approach to Reliable Distribution of Bulk Data", *Proceedings of ACM SIGCOMM 1998*, pp. 56–67, 1998.

[14] Byers J. W., Horn G., Luby M., Mitzenmacher M. and Shaver W., "FLID-DL: Congestion Control for Layered Multicast", *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8, pp. 1558–1570, 1987.

[15] Chalmers R. C. and Almeroth K. C., "Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees", *Proceedings of IEEE INFOCOM 2001*, pp. 449–458, 2001.

[16] Chawathe Y., McCanne S. and Brewer E. A., "RMX: Reliable Multicast for Heterogeneous Networks", *Proceedings of IEEE INFOCOM 2000*, pp. 795–804, 2000.

[17] Chen S. and Nahrstedt K., "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing", *Proceedings of IEEE LCN'98*, pp. 80–89, 1998.

[18] Clark D., Lambert M. and Zhang L., "NETBLT: A Bulk Data Transfer Protocol", *IETF RFC 998*, 1987.

[19] Clark D. D., Lambert M. L. and L. Zhang L., "NETBLT: A High Throughput Transport Protocol", *ACM SIGCOMM Computer Communication Review*, Vol. 17, No. 5, pp. 353–359, 1987.

[20] Deering S., "Multicasting in Internetworks and Extended LANs", *SIGCOMM '88: Symposium proceedings on Communications Architectures and Protocols*, pp. 55–64, 1988.

[21] Deering S., "Host Extensions for IP Multicasting" *IETF RFC 1112*, 1989.

[22] Diot C., Levine B. N., Lyles B., Kassem H. and Balensiefen D., "Deployment Issues for the IP Multicast Service and Architecture", *IEEE Network*, Vol. 14, No. 1., pp. 78–88, 2000.

[23] El-Marakby R. and Hutchison D., "A Scalability Scheme for the Real-time Control Protocol", *Proceedings of* 8^{th} *International Conference on High Performance Networking*, pp. 153–168, 1998.

[24] Feng W., Kandlur D. D., Saha D. and Shin K. S., "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks", *Tech. Report*, University of Michigan, CSE-TR-349-97, 1997.

[25] Fenner B., Handley M., Holbrook H. and Kouvelas I., "Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification (Revised)", *IETF RFC 3973*, 2005.

[26] Floyd S. and Jacobson V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp. 397–413, 1993.

[27] Floyd S., Jacobson V., Liu C. G., McCanne S. and Zhang L., "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 784–803, 1997.

[28] Floyd S. and Fall K., "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 4, pp. 458–472, 1999.

[29] Floyd S., Handley M., Padhye J. and Widmer J., "Equation-Based Congestion Control for Unicast Applications", *Proceedings of ACM SIGCOMM 2000*, pp. 43–56, 2000.

[30] Floyd S., "A Report on Recent Developments in TCP Congestion Control", *IEEE Communications Magazine*, Vol. 39, No. 4, pp. 84–90, 2001.

[31] Forouzan B. A., *TCP/IP Protocol Suite*, 3^{rd} *Edition*, McGraw-Hill, 2004.

[32] Gevros P., Crowcroft J., Kirstein P. and Bhatti S., "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network*, Vol. 15, No. 3, pp. 16–25, 2001.

[33] Hall J. I., "Notes on Coding Theory", Chapter 8 – Cyclic Codes. Available from: `http://www.mth.msu.edu/~jhall/classes/codenotes/coding-notes.html` (URL verified: 2007.01.23).

[34] Hanna S., Patel B. and Shah M., "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", *IETF RFC 2730*, 1999.

[35] Hardjono T. and Tsudik G., "IP Multicast Security: Issues and Directions", *Annales de Télécomunications*, Vol. 55, pp. 324–340, 2000.

[36] Holbrook H. and Cain B., "Source-Specific Multicast for IP", *IETF RFC 4607*, 2006.

[37] Internet Assigned Numbers Authority, "Internet Protocol v4 Multicast Address Assignments",
`http://www.iana.org/assignments/multicast-addresses` (URL verified: 2007.01.23).

[38] Jacobson V., "Congestion Avoidance and Control", *Proceedings of ACM SIGCOMM 1988*, pp. 314–329, 1988.

[39] Jannotti J., Gifford D. K., Johnson K. L., Kaashoek M. F. and O'Toole J. W. Jr., "Overcast: Reliable Multicasting with an Overlay Network", *Proceedings of the* 4^{th} *Symposium on Operating System Design & Implementation (OSDI) 2000*, pp. 197–212, 2000.

[40] Kelly F., "Charging and Rate Control for Elastic Traffic", *European Transactions on Telecommunications*, Vol. 8, pp. 33–37, 1997.

[41] Koifman A. and Zabele S., "RAMP: A Reliable Adaptive Multicast Protocol", *Proceedings of IEEE INFOCOM 1996*, Vol. 6, pp. 1442–1451, 1996.

[42] Kou L., Markowsky G. and Berman L., "A Fast Algorithm for Steiner Trees", *Acta Informatica*, Vol. 15, pp. 141–145, 1981.

[43] Kwon G. and Byers J. W., "Smooth Multirate Multicast Congestion Control", *Proceedings of IEEE INFOCOM 2003*, Vol. 2, pp. 1022–1032, 2003.

[44] Kwon G. and Byers J. W., "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections", *Proceedings of IEEE INFOCOM 2004*, Vol. 1, pp. 385–395, 2004.

[45] Lakshman T. and Madhow U., "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, pp. 336–350, 1997.

[46] Legout A. and Biersack E. W., "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", *Proceedings of ACM SIGMETRICS 2000*, pp. 13–22, 2000.

[47] Li X., Paul S. and Ammar M. H., "Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical Rate Control", *Proceedings of IEEE INFOCOM 1998*, pp. 1062–1072, 1998.

[48] Li Z. and Mohapatra P., "QRON: QoS-Aware Routing in Overlay Networks", *IEEE Journal on Selected Areas in Communications*, pp. 29–40, 2004.

[49] Luby M., "LT Codes", *Proceedings of the* 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science*, pp. 271–282, 2002.

[50] Lui J., Li B. and Zhang Y-Q., "A Hybrid Adaptation Protocol for TCP-Friendly Layered Multicast and its Optimal Rate Allocation", *Proceedings of IEEE INFOCOM 2002*, pp. 1520–1529, 2002.

[51] Mahdavi J. and Floyd S., "TCP-Friendly Unicast Rate-Based Flow Control", *Technical note sent to the end2end-interest mailing list*, January 8, 1997.

[52] McCanne S., Jacobson V. and Vetterli M., "Receiver-driven Layered Multicast", *Proceedings of ACM SIGCOMM 1996*, Vol. 26, No. 4, pp. 117–130, 1996.

[53] Meyer D., "Administratively Scoped IP Multicast", *IETF RFC 2365*, 1998.

[54] Moy J., "Multicast Extensions to OSPF", *IETF RFC 1584*, 1994.

[55] Nagle J., "Congestion Control in IP/TCP Internetworks", *IETF RFC 896*, 1984.

[56] Nonnenmacher J., Biersack E. W. and Towsley D., "Parity-based Loss Recovery for Reliable Multicast Transmission", *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4., pp. 349–361, 1998.

[57] Ott T. J., Lakshman T. V. and Wong L. H., "SRED: Stabilized RED", *Proceedings of IEEE INFOCOM 1999*, Vol. 3, pp. 1346–1355, 1999.

[58] Packeteer, "PacketShaper®", `http://www.packeteer.com/products/packetshaper` (URL verified: 2007.01.23).

[59] Padhye J., Firoiu V., Towsley D. F. and Kurose J. F., "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation", *IEEE/ACM Transactions on Networking*, Vol. 8, No. 2, pp. 133–145, 2000.

[60] Paxson V., "Measurements and Analysis of End-to-End Internet Dynamics", *PhD Thesis*, University of California at Berkeley, 1997.

[61] Pendarakis D., Shi S., Verma D. and Waldvogel M., "ALMI: An Application Level Multicast Infrastructure", *Proceedings of* 3^{rd} *USENIX Symposium on Internet Technologies and Systems (USITS) 2001*, pp. 49–60, 2001.

[62] PlanetLab, "PLANETLAB – An open platform for developing, deploying, and accessing planetary-scale services", `http://www.planet-lab.org` (URL verified: 2007.01.23).

[63] Postel J., "Transmission Control Protocol", *IETF RFC 793*, 1981.

[64] Quinn B. and Almeroth K., "IP Multicast Applications: Challenges and Solutions", *IETF RFC 3170*, 2001.

[65] Ramakrishnan K. and Floyd S., "A Proposal to add Explicit Congestion Notification (ECN) to IP", *IETF RFC 2481*, 1999.

[66] Rizzo L., "PGMCC: A TCP-friendly Single-rate Multicast", *Proceedings of ACM SIGCOMM 2000*, pp. 17–28, 2000.

[67] Roughan M., and Erramilli A. and Veitch D., "Network Performance for TCP Networks Part I: Persistent Sources", *Proceedings of the* 17^{th} *International Teletraffic Congress, ITC–17*, pp. 24–28, 2001.

[68] Rubenstein D., Kurose J. F. and Towsley D. F., "The Impact of Multicast Layering on Network Fairness", *IEEE/ACM Transactions on Networking*, Vol. 10, No. 2, pp. 169–182, 2002.

[69] Sahasrabuddhe L. H. and Mukherjee B., "Multicast Routing Algorithms and Protocols: A Tutorial", *IEEE Network Magazine*, Vol. 14, No. 1., pp. 90–102, 2000.

[70] Schulzrinne H., Casner S., R. Frederick R. and Jacobson V., "RTP: A Transport Protocol for Real-Time Applications", *IETF RFC 3550*, 2003.

[71] Shan Y., Kalyanaraman S., Woods J. W. and Bajic I. V., "Joint Source-Network Error Control Coding for Scalable Overlay Video streaming", *Proceedings of IEEE International Conference on Image Processing (ICIP) 2005*, pp. 177–180, 2005.

[72] Shields C. and Garcia-Luna-Aceves J. J., "The HIP Protocol for Hierarchical Multicast Routing", *Proceedings of* 17^{th} *Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pp. 257–266, 1998.

[73] Sisalem S. and Wolisz A., "LDA+: A TCP-Friendly Adaptation Scheme for Multimedia Communication", *Proceedings of IEEE International Conference on Multimedia and Expo (III) 2000*, pp. 1619–1622, 2000.

[74] Sisalem D. and Wolisz A., "MLDA: A TCP-friendly Congestion Control Framework for Heterogeneous Multicast Environments", *Proceedings of* 8^{th} *International Workshop on Quality of Service (IWQoS 2000)*, 2000.

[75] Speakman T., Crowcroft J., Gemmell J., Farinacci D., Lin S., Leshchiner D., Luby M., Montgomery T., Rizzo L., Tweedly A., Bhaskar N., Edmonstone R., Sumanasekera R. and Vicisano L., "PGM Reliable Transport Protocol Specification", *IETF RFC 3208*, 2001.

[76] Srikant R., *The Mathematics of Internet Congestion Control*, ser. "Systems & Control: Foundations & Applications", Birkhäuser, 2004.

[77] Stevens W. R., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *IETF RFC 2001*, 1997.

[78] Subramanian L., Stoica I., Balakrishnan H. and Katz R., "OverQoS: An Overlay based Architecture for Enhancing Internet QoS", *Proceedings of* 1^{st} *Symposium on Networked Systems Design and Implementation*, 2004.

[79] Thaler D., "Border Gateway Multicast Protocol (BGMP): Protocol Specification", *IETF RFC 3913*, 2004.

[80] Thaler D. and Handley M. and Estrin D., "The Internet Multicast Address Allocation Architecture", *IETF RFC 2908*, 2000.

[81] Thyagarajan A. and Deering S., "Hierarchical Distance-Vector Multicast Routing for the Mbone", *Proceedings of the ACM SIGCOMM 1995*, pp. 60–66, 1995.

[82] Urvoy-Keller G. and Biersack E., "A Congestion Control Model for Multicast Overlay Networks and its Performance", *Proceedings of* 4^{th} *International Workshop on Networked Group Communication (NGC)*, 2002.

[83] Vicisano L., Rizzo L. and Crowcroft J., "TCP-Like Congestion Control for Layered Multicast Data Transfer", *Proceedings of IEEE INFOCOM 1998*, pp. 996–1003, 1998.

[84] Viéron J., Turletti T., Salamatian K. and Guillemot C., "Source and Channel Adaptive Rate Control for Multicast Layered Video Transmission Based on a Clustering Algorithm", *EURASIP Journal on Applied Signal Processing*, Vol. 2004, No. 2, pp. 158–175, 2004.

[85] Waitzman D., Partridge C. and Deering S., "Distance Vector Multicast Routing Protocol", *IETF RFC 1075*, 1988.

[86] Whetten B., Montgomery T. and Kaplan S. M., "A High Performance Totally Ordered Multicast Protocol", *Proceedings of Selected Papers from the International Workshop on Theory and Practice in Distributed Systems*, pp. 35–57, Springer-Verlag, 1995.

[87] Widmer J. and Handley M., "Extending Equation-Based Congestion Control to Multicast Applications", *Proceedings of ACM SIGCOMM 2001*, pp. 275–286, 2001.

[88] Wittmann R. and Zitterbart M., *Multicast Communication*, Morgan Kaufmann, 2001.

[89] *Xpress Transport Protocol Specification*, XTP Revision 4.0b, July 1998. Available from: `http://www.packeteer.com/resources/prod-sol/XTP.pdf` (URL verified: 2007.01.23).