POLITECNICO DI TORINO

Dipartimento di Automatica e Informatica

Dottorato di Ricerca in
Ingegneria Informatica e dei Sistemi

Tesi

Real-time Services over Packet Switching Networks

Candidato:
Mario Baldi

Tutori:
Prof. Silvano Gai
Dr. Yoram Ofek

Coordinatore:
Prof. Pietro Laface

1998

To Marceline

Acknowledgements

Many dear people have either deeply affected my way through these last years of studies, or actively helped in my research work (some of them did both); I would like to thank them.

First of all, I would like to thank my parents who have always supported and encouraged me, even in my decision to spend time abroad - a decision that certainly weighted on them.

I am deeply grateful to my Ph.D. advisors Silvano and Yoram for having helped and supported me throughout the last years with ad vices that went well beyond my project. A special thank goes to Prof. Fabio Neri who introduced me to Dr. Yoram Ofek, thus strongly impacting my research activity.

I would also like to thank Letizia Jaccheri for having given me a first insight into the meaning of research and Gian Pietro Picco for having shown me a lot about research. I also need to thank Prof. Marco Ajmone Marsan, who included me in a great research group, which I enjoyed a lot.

I want to thank Davide, Fulvio, and Dante with whom I work daily, and all the people who contributed to the results of my research, either reported or not in this dissertation: Andrea Bianco, Renato Lo Cigno, Maurizio Munafò, Prasoon Tiwari, Peter Westerink, Simone Martini, Paolo Losi, Luca Fantolino, and Agostino Vailati.

Finally, I would like to thank especially Marceline Saïbou who helped me significantly during the last year; she has also been a “wonderful” English teacher.

Contents

1 Introduction **1**

2 Local and Global Timing for Packet Scheduling **5**

2.1 Packet Scheduling using Local Timing 6

2.1.1 Traffic Shaping with a Leaky Bucket 6

2.1.2 Static Priority 7

2.1.3 Stop-and-Go Queuing 7

2.1.4 (Weighted) Round Robin 8

2.1.5 Generalized Processor Sharing 8

2.1.6 Packet by Packet GPS and Packet Fair Queuing 9

2.1.7 Weighted Fair Queuing Derivatives 10

2.1.8 FIFO+ 11

2.2 Packet Scheduling using Global Timing 12

2.2.1 Network Timing 12

2.2.2 Packet Forwarding 13

3 Using Static Priority for Packet Switching Telephony **15**

3.1 Delay of Voice Transmission over Packet Switched Networks . 17

3.2 Designing a Packet Switched Telephone Network 20

3.3 Numerical Results 22

3.4 Integrated IP and ATM Architectures for Packet Switched Telephone Networks 27

3.4.1 Motivations 28
3.4.2 Integrated Services IP-ATM Models 30
3.5 Discussion 32

4 Efficiency of Packet by Packet Generalized Processor Sharing 33
4.1 The Simulator 34
4.1.1 Call Admission Control 35
4.1.2 Measurements and Statistics 37
4.2 A Case Study: Toll-quality Telephony 38
4.2.1 Call Admission Control 39
4.2.2 Implementation of Sources 39
4.2.3 Uniform Sources 43
4.2.4 Heterogeneous Sources 52
4.3 Discussion 55

5 Adaptive Group Multicast with Time-Driven Priority 57
5.1 Related Work 58
5.2 Principles of Operation 60
5.2.1 Network Model 60
5.2.2 Packet Forwarding 60
5.2.3 Bandwidth Allocation 61
5.2.4 Forwarding on a Virtual Ring 63
5.2.5 Relative Time 64
5.3 Scheduling 65
5.3.1 Feasible Ring-Scheduling Algorithm 67
5.3.2 Optimal Selection of TFs 68
5.4 Adaptive Bandwidth Sharing 72
5.4.1 Rate Adaptation 74
5.4.2 Bandwidth Sharing 75

5.5 Scalable Growth of the Multicast Group 78
5.5.1 Join as a Receiving Node 78
5.5.2 Join as a Transmitting Node 78
5.6 Discussion 80

6 Videoconferencing over Asynchronous and Synchronous Packet Switching Networks 83

6.1 Scope of the Work 85
6.2 The Model 86
6.2.1 Requirements for Interaction 87
6.2.2 Requirements for Visual Quality 88
6.2.3 The System 90
6.3 Transmission of Raw Video 92
6.3.1 Dedicated Link Between Sender and Receiver 92
6.3.2 Circuit Switching 96
6.3.3 Packet Switching with Time-Driven Priority 98
6.3.4 Transmission of Video and Scheduling 99
6.3.5 Asynchronous Packet Switching 105
6.4 Transmission of VBR MPEG Video 110
6.4.1 MPEG Overview 110
6.4.2 Dedicated Link Between Sender and Receiver 116
6.4.3 Packet Switching with Time-Driven Priority 120
6.4.4 Asynchronous Packet Switching 137
6.4.5 Circuit Switching 143
6.5 Transmission of CBR MPEG Video 144
6.5.1 Intra-frame Coding Only 145
6.5.2 Intra-frame and Predictive Coding 159
6.5.3 Experimental Data 168
6.5.4 Dedicated Link between Sender and Receiver 178

6.5.5 Circuit Switching 182
6.5.6 Packet Switching with Time-Driven Priority 183
6.5.7 Asynchronous Packet Switching 189
6.6 Summary 195

7 Conclusions **199**

A The MPEG Encoding Standard **207**
A.1 General Principles 208
A.2 Intra-coded Pictures 209
A.2.1 Discrete Cosine Transformation 209
A.2.2 Quantization 210
A.2.3 Entropy Encoding 211
A.2.4 Controlling I-frame Dimension 212
A.3 Predictive-coded Pictures 214
A.3.1 Motion Estimation 214
A.3.2 Motion Compensation 215
A.3.3 DCT 215
A.3.4 Quantization 216
A.3.5 Entropy Encoding 216
A.3.6 Controlling P-frame Dimension 216
A.4 Signal to Noise Ratio 217
A.5 Packetization and Streaming 217

B List of Acronyms and Symbols **219**

Chapter 1

Introduction

Computers are getting larger diffusion and a more relevant role in everyday life. If the computer is an important and useful tool by itself, when it is augmented with the ability to exchange data with other computers, it increases its usefulness and broaden its scope of application. The natural way in which computer exchange data is through messages, usually referred to as *packets*, and computer networks are naturally based on the *packet switching* paradigm.

Since users sit behind computers which are able to communicate among themselves, they are willing to use such computers to communicate with other users. Communication among people is naturally an audio and visual one; thus, the latest challenge for computer networks is to support telephony and videoconferencing together with a number of other applications (like, for example, video on demand) which share the need for a *real-time* service with a *statistically* or *deterministically guaranteed* quality.

Traditionally computer networks are built to support a *best-effort* service. The sender of a packet hands the packet to the network which does its best to deliver it to the destination. However, the network does not provide any guarantee on the delivery, i.e., on the service it is providing. The higher protocol layers (usually the transport layer, as defined by the ISO/OSI reference model) in the sender and the receiver take care of the success of the communication.

Real-time applications need a *time relationship between the sender and the receiver*. For example, voice transmission is based on the source sampling the

voice signal at constant intervals ($125\mu s$) and then the destination playing back the samples at the same pace. The fastest way to transmit digitalized voice is to transmit each sample as soon as it is produced; if the network keeps a time relationship between sender and receiver, the latter plays each sample as soon as it receives it. Traditionally, voice samples are transmitted over circuit switched networks which provide a real-time service because the *variation* on the delay experienced by each sample during transmission is very small. This ensures a time relationship between sender and receiver and samples get to the destination with the same spacing they have been captured ($125\mu s$).

Nevertheless, transmission of voice, video, and any other delay sensitive data over packet switched networks has the appealing advantage of integrating all these services and possibly non-real-time data transfer services on a single network. Moreover, since circuit switched networks statically allocate resources to flows, when a flow is not using the resources it has been allocated, they cannot be used by other flows and are consequently wasted. Since in packet switching networks resources can be used more dynamically, the cost of the transmission is expected to be lower, especially for the traffic which does not require any guarantees.

If voice samples or any other delay sensitive data must be sent over a packet switched network they need a real-time service that guarantees a constant delay. Actually, delay sensitive data can be transmitted also over a network providing a best-effort service. The higher protocol layers in the destination take care of the spacing between data units by buffering early ones to cancel the delay variations experienced in the network. However, this approach introduces a long delay which is not acceptable for interactive communications.

Thus, the network itself must provide the time relationship between sender and receiver by controlling (1) the (variation of the) delay packets experience in the network and (2) the loss due to congestion in network nodes. The two matters are strictly related since they have to do with management and occupancy of buffers inside network nodes.

The Quality of Service (QoS) can be guaranteed (statistically or deterministically) combining the two following mechanisms:

1. *Reservation of network resources* to real-time flows thus avoiding that too many flows share the same resource. This is accomplished by the

call admission control: it receives from the applications a description of both the their traffic and their QoS requirements, and it checks whether the necessary network resources are still available.

2. Exploitation of specific *packet scheduling algorithms* that use the resources reserved to a flow to provide it with the requested QoS and prevent other flows from disrupting the service.

This work focuses on packet scheduling rather than on the protocols to perform resource reservation; the two topics are anyway strictly related since the applied packet scheduling mechanism determines the amount of resources to be reserved in the network. This work assesses the deployment of various packet scheduling techniques for providing *deterministic* QoS guarantees.

The main techniques are presented in Chapter 2 differentiating between those based on a local time reference and those based on a global time reference. We call *asynchronous* packet switching networks those exploiting the former schemes, as opposed to *synchronous* packet switched networks which exploit the latter.

Chapter 3 contains an analytical study on the application of one of these techniques (namely, static priority) to provide toll quality telephony over a packet switched network which also carries best-effort traffic. Chapter 4 presents a call level simulator aimed at studying the efficiency of employing packet scheduling algorithms based on local timing. In particular, the chapter contains a case study of the application of the Packet by Packet Generalized Processor Sharing technique for the provision toll quality telephony.

The exploitation of a packet scheduling technique based on global timing (namely, Time-Driven Priority) for the provision of adaptive real-time many-to-many communications is considered in Chapter 5. Chapter 6 analyzes the end-to-end delay of various videoconference configurations over asynchronous and synchronous packet switched networks. Conclusions are drawn in Chapter 7.

Chapter 2

Local and Global Timing for Packet Scheduling

A network aiming to provide guaranteed QoS must avoid that different traffic components interfere with each other. Sample components are a call in a connection oriented network, a flow in a connectionless network, or a set of calls/flows showing some common characteristic (e.g., QoS requirements). Segregation of traffic components can be done using a local time reference by (1) queuing them in (virtually) separate buffers and (2) properly selecting packets for transmission from the queues (a.k.a. *packet scheduling*). In this case we talk of *asynchronous packet switching* since nodes operate independently and asynchronously. Many approaches have been proposed so far and many of them have similar properties. Section 2.1 considers the main ones; the application of two of them is studied in Chapter 3 and Chapter 4.

Alternatively, a global time reference can be effectively used by network nodes to regulate packet scheduling; this is the case for *time-driven priority* whose principles of operation and properties are the subject of Section 2.2. Chapter 5 shows how adaptive real-time group communications can be based on this technology. The exploitation of time-driven priority to support MPEG videoconference is considered as a case study in Chapter 6 and compared to the transmission of MPEG video over asynchronous packet switching networks.

2.1 Packet Scheduling using Local Timing

In this section we review and the main buffering and scheduling algorithms and compare their properties. Since the properties of these schemes depend on characteristics of the traffic entering the network, traffic shaping and policing at the network edge is first described.

2.1.1 Traffic Shaping with a Leaky Bucket

Scheduling algorithms used to control queuing delay are designed for dealing with constant bit rate (CBR) traffic. When delaying with variable bit rate (VBR) traffic, the amount of network resources required to guarantee a given delay bound is larger as higher is the burstiness of the traffic flow.

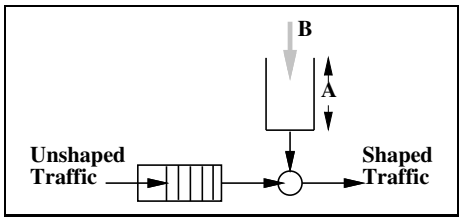

Figure 2.1: Model of a Leaky Bucket.

In order to ensure lower burstiness (and thus lower resource requirements) the source shapes the traffic it is generating by using traffic shaping mechanisms such as a *leaky bucket* [6], as shown in Figure 6.16. The bursty traffic generated by a source is shaped by a leaky bucket into a flow having average rate B and controlled burstiness. A bucket is assumed to contain at most A tokens and it is filled at the rate of B tokens per second. When packets are presented to the traffic shaper, they are sent into the network at link speed[1] as long as the number of tokens in the bucket is larger than the packets dimension (in data units). A token is removed from the bucket for each data unit sent. We assume to exploit a buffered leaky bucket which is modeled as shown in Figure 2.1. When the bucket is empty, packets may be dropped or buffered (this is performed by a so called *buffered leaky bucket*) until enough tokens are present in the bucket to send the packet. This buffer adapts the

[1]There is a variant called dual leaky bucket that sends burst of data at a rate lower than the link speed. Such a traffic shaper can be built by cascading two simple leaky buckets.

source rate to the characteristics of the shaped traffic at the expenses of a network shaping delay. Since, given the video frame rate and the resolution, the traffic characteristics of a raw video sequence are known in advance, the buffer can be dimensioned properly so that it never overflows.

2.1.2 Static Priority

A fixed priority is set among the queues which are served in order of priority. The highest priority queue which has a backlog of packets is served until it is empty. Then, the next backlogged queue in priority order is served.

Each traffic component is not affected by any other components with lower priority, but it is affected by the components at higher priority. Highest priority can be given for example to low bandwidth time-sensitive traffic and the lowest priority to best effort traffic.

Chapter 3 provides a detailed analysis of the delay experienced by voice calls in a network in which two priority classes are defined: voice calls have high priority, while best-effort data traffic has low priority.

2.1.3 Stop-and-Go Queuing

Stop-and-go Queuing proposed by Golestani [22, 23, 24] assumes that the source behaves according to an (r,T)-smooth shaper, i.e., it does not transmit more than r bits in a time frame of T bit times. When a node receives a packet, it does not start forwarding it before the whole frame in which the packet arrived has been received. Assuming that capacity has not been overallocated, i.e., that a node is not required to forward more than T_f bits over an output link during a time frame, the delay experienced by a packet in each node is between T_f and $2 \cdot T_f$.

The stop-and-go queuing algorithm is a simple way of providing a bound on delay and jitter (i.e., maximum delay). The disadvantage of this scheme is that if the network is not underutilized, the delay bound is inversely proportional to the bandwidth of the lowest rate traffic component and directly proportional to the smallest packet size. In fact, in order to fully utilize the network, the reserved amount of bits must be sent in each time frame. The minimum amount of bits that can be sent in a time frame is given by the

minimum packet size, i.e., P_s. Thus, given the minimum bandwidth B to be reserved, the minimum time frame length to avoid underutilization is

$$T_f = \frac{P_s}{B}$$

Stop-and-go has been proven to provide a delay bound larger than the one granted by weighted fair queuing (described in Section 2.1.6) unless network resources are overallocated [46].

2.1.4 (Weighted) Round Robin

The scheme was first proposed in [40] with the objective of isolate different traffic components guaranteeing a fair sharing of the bandwidth of a link. Each traffic component is buffered in a separate queue and backlogged queues are served in a round robin fashion. This scheme is sensitive to arrival patterns since a packet entering a queue immediately after its queue has been served, experiences a long delay because it has to wait until all the backlogged queues be served.

Moreover, the scheme does not allow to differentiate the service provided to traffic components in a controlled manner. *Weighted round-robin* has been proposed to overcome this problem: the service rate is not the same for each queue and it is controlled by a relative weight. Notice that the order in which queues are served remains anyway fixed.

2.1.5 Generalized Processor Sharing

The Generalized Processor Sharing (GPS) [46] operates on traffic flows that have infinitely fine granularity; this is referred to as the *fluid flow* model. Each active flow is inserted in a separate buffer and all the backlogged buffers are served concurrently with a rate which is proportional to a weight ϕ_i. A GPS scheduler have the interesting property of guaranteeing to each traffic flow i a minimum service rate

$$g_i = \frac{\phi_i}{\sum_j \phi_j} \cdot r, \tag{2.1}$$

where r is the rate of the output of the scheduler (i.e., usually the capacity of the output link). The service rate of a traffic component is independent of the rate of the other traffic components.

Moreover, if a traffic flow is compliant to the traffic exiting a leaky bucket with output rate $B_i < g_i$ and token bucket of depth A_i, the GPS guarantees an upper bound on the queuing delay of traffic component i given by

$$Q_{i,M}^{GPS} = \frac{A_i}{g_i} = A_i \cdot frac \sum_j \phi_j \phi_i \cdot r$$

This bound can be intuitively explained by considering that the burst of A_i bits generated by the source is buffered in the access node which fluidly serves the traffic component at minimum rate g_i thus introducing a maximum delay A_i/g_i. All the subsequent nodes on the path to the destination serve the traffic component at the same rate, i.e., without further buffering the flow.

2.1.6 Packet by Packet GPS and Packet Fair Queuing

GPS is based on the fluid flow model, but it should be applied on packet switching networks in which flows have packet granularity. Packet by Packet GPS (PGPS) [46] extends GPS to handle packet flows: arriving packets are served according to their GPS packet service time, i.e., the instant in which the last bit of a packet would have been sent by GPS. Implementing a PGPS scheduler is not trivial and usually computationally heavy.

When operating over a flow which is compliant to a leaky bucket with rate B_i and token bucket depth A_i, PGPS provides a delay bound [47]

$$D_i = \frac{A_i}{g_i} + \frac{(h_i - 1) \cdot A_i}{g_i} + \sum_{m=1}^{h_i} \frac{P_M}{r_m} \tag{2.2}$$

where h_i is the number of hops on the path of flow i, r_m is the service rate of the m^{th} node (usually the capacity of link m, and P_M is the maximum packet size allowed on the network.

The delay bound provided by Equation 2.2 is proportional to the burstiness of the source A_i and inversely proportional weight associated to the source. Thus, when a delay requirement is to be fulfilled for a flow i, the higher

the burstiness of a source, the larger the weight ϕ_i (relatively to the other weights) must be. In other words, as shown by Equation 2.1, the larger the burstiness of a source, the larger must be the share of link capacity (i.e., the amount of resources) reserved to the flow. This suggests that PGPS is best suited to constant bit rate flows[2].

Moreover, Equation 2.2 shows that the delay bound depends on the number of traversed nodes. Thus, given a delay requirement for flow i, the larger the number of hops on its path, the higher amount of link capacity must be reserved to satisfy the delay requirement. Lastly, notice that, as shown by the second addendum, Equation 2.2 is devised assuming that the same minimum service rate is guaranteed in each node. Nevertheless, this is not compulsory and the a variable amount of capacity can be reserved to a flow on the nodes it traversed according to local resource availability. This increases the flexibility in accommodating resource reservations.

Weighted Fair Queuing (WFQ) [11] is an algorithm which implements PGPS scheduling aiming at keeping the same properties with a reasonable, though still high, computational complexity.

2.1.7 Weighted Fair Queuing Derivatives

The computational and implementation complexity of WFQ has led to the development of a set of algorithms which approximate the behavior of PGPS with acceptable complexity.

The *VirtualClock* [67] scheduling algorithm associates to each traffic flow a virtual clock which ticks according to the amount of data served from the queue. Each arriving packet is tagged with its virtual arrival time and the scheduler serves the packet with the older arrival time. VirtualClock queuing segregates the various traffic flows, but since it keeps track of the amount of data served for each flow, it gives preference to flows which have transmitted less. This results in a penalization of the flows which transmit large amount of data during underload periods; this is not acceptable because it is in discordance with the statistical multiplexing idea which is the main rationale for the deployment packet switching.

Self-Clocked Fair Queuing [21] has a complexity on the order $\log(N)$, where N

[2]This same consideration can be extended to the others scheduling algorithms.

is the number of flows on which the scheduler operates. The lower complexity is obtained at the expenses of a looser delay bound.

A family of Hierarchical Packet Fair Queuing [2, 3] algorithms has been proposed to hierarchically reserve link capacity. The capacity of a link is first shared among various sets of traffic flows; then, a scheduler handles packets according to the bandwidth share to which they belong. Hierarchical schemes are deemed particularly suited to the Internet where bandwidth is not centrally managed.

2.1.8 FIFO+

The delay bounds provided by the queuing schemes mentioned above is high compared to the average delay experienced by packets. Moreover, these schemes are generally computationally much harder than simple First In First Out (FIFO) queuing. Being FIFO a work conserving scheme, it provides the same average delay as WFQ; nevertheless, the provided delay is highly variable.

FIFO+[9] aims at keeping the simplicity of FIFO queuing while adding some guarantees on the provided service. Each node keeps track of the average delay experienced by packets in its buffer. Before transmitting a packet the node devises the difference between the average delay experienced by packets in the node's buffer and the delay experienced by the packet at hand. The result is accumulated in a field in the packet header which keeps track hop-by-hop of how much the packet is late or early with respect to the average delay experienced by packets on the path. When a packet arrives to a buffer (which is served with a FIFO policy), it is inserted in the queue as if arrived at the real arrival time plus the offset recorded in the header[3]. As a result late packets are pushed towards the head of the queue, while early packets are kept behind.

Simulation results and an analytical study [10] show that the probabilistic delay bounds offered by FIFO+ are lower than those offered by WFQ. It is worth noticing that FIFO+ performs statistical multiplexing and it is suited to applications that are able to work with a delay bound which can be occasionally violated.

[3]Early packets are tagged with a negative offset.

2.2 Packet Scheduling using Global Timing

Time-Driven Priority (TDP) [36] is a multiplexing scheme aimed at sharing link capacity while guaranteeing sources from uncontrolled delays (or even loss) due to contention in accessing links. TDP allows the network to handle both real-time and best effort traffic. The basic idea is to give priority to real-time traffic over best effort traffic on a periodic manner. This guarantees against the latter using the resources of the former. Network resources are reserved to real-time traffic at predefined time instants and real-time sources send data according to the time in which the network is going to serve them. This enables the network to pride QoS guarantees to periodic real-time traffic.

2.2.1 Network Timing

All the nodes are provided with a global timing structure. The timing information can be obtained from internal sources, as proposed in [43], or external sources like GPS. If, for example, GPS is used for this purpose, each node in the network receives clock ticks from a GPS receiver and uses them to keep its internal clock synchronized. As shown in Figure 2.2, there is a period of time in which the clocks of all the nodes have the same time value [42].

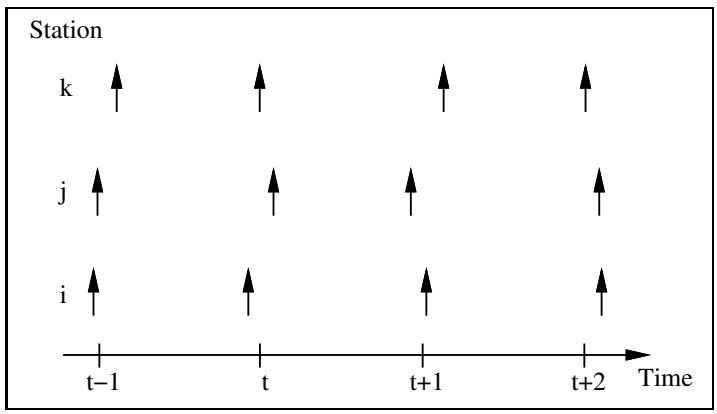

Figure 2.2: Clock Ticks.

The time between two successive ticks is called a *Time Frame* (TF) and has a typical duration of 125 μs or 8kHz. Thanks to this global timing structure all nodes have the same instant knowledge of the *temporal value* of the current

TF. In each TF a fixed amount of bits $T_f \cdot C$ can be sent on a link: data are grouped in packets and a packet is transmitted during a single TF.

2.2.2 Packet Forwarding

One or more packets can be sent during a TF; if there are not enough packets in a queue to be sent for the duration of the whole TF, a node can send best effort traffic during the leftover time.

The network shows the best performance when a packet is forwarded during the TF immediately following the one in which it was received[4]. This is called the *pseudo-isochronous* or *RISC-like* forwarding principle proposed in [42, 35, 36]. The basic idea is that packet hopping in the network is a function of TFs. Precisely, a packet received by a node in TF t, is forwarded by the node so that it reaches the next node along the path in TF t+i (for $i \geq 1$). Note that in wide area internetworking environments i may be large due to the processing and propagation delays. Furthermore, this scheme assumes that the delay between the output port of a node and the output port of the next node along the path is a constant integer number of TFs. If this does not fit actual delays, receiving nodes can introduce extra delay by buffering packets and moving them to the output port at the predefined instant.

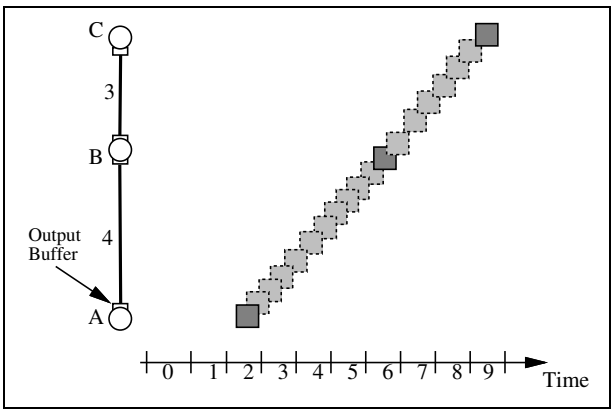

Figure 2.3: Packet Forwarding.

Consider as an example the simple topology depicted in Figure 2.3; links are

[4]A node can happen to require a number of TFs to process the packet; the best TF for the forwarding is the first after this processing time.

labeled with the delay (in TF units) between the output port of the node transmitting on the link, and the output port of the receiving node. Node A has a packet to send on its output port (buffer) at TF 2. This packet will arrive to node B and will be ready to be sent from B's output port at TF 6; the destination node C receives the packet at TF 9. The propagation delay from the source A to the destination C is fixed; the jitter is bound by two TFs and is due to the uncertainty on when node A's transmission begins, during TF 2, and when node C's reception ends, during TF 9.

To guarantee that packet forwarding is always possible, *TFs are reserved* on the links on the path between source and destination and the source transmits data during the TFs reserved on its outgoing link. A TF is reserved for a session when a fraction of the $C \cdot T_f$ bits which can be used during it is kept available for forwarding a packet (belonging to the session) received in the previous TF. The other TFs, and even the non-reserved bits in the reserved TFs, can be used to send data not belonging to the videoconference call. Moreover, if a packet belonging to the videoconference call is not to be forwarded, the allocated bits can be used anyway to send other data.

Since with RISC like forwarding packets are buffered in intermediate nodes for a fixed time, each packet takes a fixed number of TFs to move from the output buffer of an intermediate node to the output buffer of the following one on the path to the destination. As a consequence the time needed to move a packet from source to destination depends only on the physical topology of the path. and has a delay variation or jitter of $2 \cdot T_f$ [36]. This network jitter is fixed and does not depend on traffic load or the path length from source to destination. As a consequence, a packet switched network with time driven priority and resource reservation compiles with Requirement 4 because network delay has a predefined bound. The buffering time in each node is fixed at reservation time: keeping it small enough to avoid overflowing of buffers, the network is not subject to congestion and the related loss.

Chapter 3

Using Static Priority for Packet Switching Telephony

A big deal of emphasis is being put on Internet telephony. Even though it is possible to transmit voice over the Internet, this cannot be regarded as a telephone service because quality is generally low. In fact, depending on the network load along the communication path, quality can vary widely.

A major quality index in *interactive* voice services is the delay perceived by the users. In order for a conversation not to be annoying, the *end-to-end delay* (i.e., the time between when the speaker talks and when the listener hears) must be shorter than 100-200 ms. We call this maximum acceptable delay the *interaction bound*. If the voice service is used for interactive communication, the end-to-end delay is required to be always below the interaction bound.

This work has two main goals:

- study the end-to-end delay of voice transmission over a packet switched network[1] from an analytical point of view, and
- identify some possible architectural solutions for packet switched networks expressly designed to carry telephone traffic. We call such networks *Packet switched Telephone Networks* (PTNs).

Then we analyze the end-to-end delay more deeply to identify the key factors in the design of a PTN providing toll quality telephone services, i.e., for

[1] In the context of this work packet switching is intended in a broad sense, encompassing also cell switching.

possibly replacing traditional telephone networks. This is attractive because unused capacity can be exploited to carry data traffic, and network deployment and maintenance are easier since the network itself can be used for management purposes. We show that by limiting the number of hops in the path between any pair of users and dimensioning link capacity accordingly, the end-to-end delay can be kept below the interaction bound. Thus, packet switching could be exploited to provide a high quality telephone service even though the raw capacity needed is larger than with circuit switching. This is justified by the possibility of carrying best effort traffic on the same network and by the lower installation and management costs.

In this analysis we consider two enabling technologies for a packet switched telephone network: the *Asynchronous Transfer Mode* (ATM) and the *Internet Protocol* (IP). ATM was standardized by the ITU-T as the foundation of the *Broadband - Integrated Services Digital Network* (B-ISDN) [30] and was explicitly designed for the provision of real-time services while taking advantage of statistical multiplexing for an efficient usage of network resources. On the other side, the Internet protocol suite is currently the most widely deployed network architecture. IP was designed as an internetworking protocol for carrying best effort traffic among heterogeneous networks and lots of applications that use its services have been developed. The Internet community is actively working on the *Integrated Services Internet* [5], an Internet architectural evolution aiming at providing real-time services over IP. This will be fundamental to the videoconferencing and telephony applications which are being deployed over the Internet.

We identify the delay components and the main factors in determining the delay bound. This first analysis shows that the current Internet architecture is not suited to the provision of telephone services with end-to-end delay shorter than the interaction bound. Some numerical results are devised by applying the analytical equations to a PTN designed after the Telecom Italia's telephone network (i.e., using the same topology and number of supported calls). Various network configurations, which differ for enabling technology, link capacity, and speech encoding techniques, are compared, showing that some of them guarantee the interaction bound.

Our analytical model shows that ATM outperforms IP as was expected since the former was expressly designed to support low bandwidth real-time traffic. Nevertheless, because of the large number of existing applications based on

IP and its internetworking capabilities, it must be taken into account as a candidate technology for carrying both best effort and high bandwidth real-time multimedia traffic in packet switched telephone networks. We claim that ATM and IP together can be successfully exploited for devising service integration over a PTN. IP is desirable for its broad diffusion and the large number of existing applications. Moreover, due to its internetworking capabilities, it allows systems to communicate through heterogeneous technologies. When deployed together with mechanisms for guaranteeing *Quality of Service* (QoS), it is also suited to real-time applications which require both high bandwidth and low latency like videoconferencing. On the other side, ATM has been designed to provide QoS guarantees, but it requires all the network to be implemented with the same technology and all the existing applications to be rewritten in order to take advantage of its capabilities. Nevertheless, ATM outperforms IP when dealing with telephony, i.e., low bandwidth real-time applications. We propose *integrated services IP/ATM* models to exploit both IP and ATM in the implementation of packet switched networks for telephony. These models are derived from the integrated IP/ATM approaches currently being exploited for an effective operation of IP over ATM networks. Integrated services IP/ATM models provide ATM based services for telephony and IP (over ATM) based services for any other kind of traffic (namely, best effort and multimedia) in order to get the best from the two technologies.

3.1 Delay of Voice Transmission over Packet Switched Networks

The end-to-end delay in the transmission of voice over a packet switched network has three components:

1. The *processing* delay D_{proc} is introduced when processing the audio signal. Digital transmission of voice requires the audio signal to be sampled 8000 times per second. Samples are quantized, encoded, and transmitted to the receiver which plays them at a fixed pace. The *Pulse Code Modulation* (PCM) encoding, which is used on traditional digital telephone networks, uses 8 bits to encode each sample, thus generating a 64 Kb/s flow for each voice connection. The processing

delay introduced by PCM is negligible. The voice signal can be encoded with different techniques in order to produce a bit stream at rates as low as 8 Kb/s with audio quality comparable to PCM [64]. Some of these techniques introduce a delay up to 100 ms [51], thus not being suited to interactive telephone services.

2. The *network* delay D_{net} is given by the time to inject into and propagate through the network the data stream. It has five components:

   (a) The voice encoder produces a bit stream at a rate of R bits per second. Before being transmitted through the network, bits are clustered in packets. In each packet P bits are stuffed in the payload, this clustering introduces a *packetization* delay

   $$D_{pkt} = \frac{P}{R} \tag{3.1}$$

   (b) The *transmission* delay

   $$D_{tr} = \frac{P_S}{C}$$

   is introduced to send a packet of size P_s over a link having capacity C.

   (c) D_{pr} is the delay due to signal *propagation* through the physical links connecting network nodes.

   (d) The *node processing* delay D_{np} is introduced by a network node each time it has to forward a packet.

   (e) The *queuing* delay is the time spent by packets in nodes' buffers while contending for the same output port. It is a relevant component in determining the end-to-end delay of a voice connection. The queuing delay Q_i experienced by a packet in the i^{th} node on the route to the destination, depends on the instantaneous status of the output buffer associated to the link on which the packet must be forwarded. As a consequence the queuing delay is not known a priori; whenever a best effort service is provided (as currently is by the Internet) a bound is given by

   $$0 \leq Q_i \leq \frac{B_i - P_S}{C_i}, \tag{3.2}$$

where B_i is the dimension (in bits) of the buffer and C_i is the capacity of the output link[2].

3. A good quality reconstruction of the voice signal at the receiver requires samples to be played at regular intervals (e.g., every 125 μs). Whenever the two previous delay components are variable in time (i.e., the end-to-end delay presents a *jitter*), different samples experience different end-to-end delays. If they were played as soon as they are available to the receiving application, they would not be uniformly spaced (in time). Thus, the receiving application uses a *replay buffer* to store the samples that experience a delay shorter than the maximum: it retrieves and plays the samples at a regular pace. The *compensation* delay, i.e., the time spent by early samples in the replay buffer, should be between 0 and $\delta_{RB} = \Delta D_{proc} + \Delta D_{net}$, where ΔD_{proc} and ΔD_{net} are the maximum[3] variations of the processing (item 1 above) and network (item 2) delays, respectively. Actually, since the delay experienced by samples when entering the replay buffer is not known (see Section 6.3.5 for details), the compensation delay is bound by $2 \cdot \delta RB$. Thus, the replay buffer also introduces the *excess compensation* delay $E_c = [0,\delta_{RB}]$. Due to the jitter compensation, the delay perceived by the user is

$$D_{ete} = \max(D_{proc}) + \max(D_{net}) + E_c$$

which can be rewritten as

$$D_{ete} = D_{proc} + D_{pkt} + (N+1) \cdot D_{tr} + D_{pr} + N \cdot D_{np} + Q_{max} + E_c \quad (3.3)$$

where N is the number of network nodes on the path between sender and receiver and Q_{max} is the maximum queuing delay given by

$$Q_{max} = \sum_{i=1}^{N} \max Q_i = \sum_{i=1}^{N} \frac{B_i - P_S}{C_i} \quad (3.4)$$

[2]The time P_s/C_i a packet spends in the buffer while being transmitted on the output link has been already taken into account as transmission delay.

[3]Usually, the compensation delay is dimensioned with respect to some percentile of the delay variation, instead of considering its maximum value. As a result, the bound is smaller, but the samples experiencing delay larger than the chosen percentile are discarded. This yields unpredictable QoS because the distribution of the delay over the samples is not known a priori. In this work, the compensation delay is dimensioned according to the maximum delay variation δ_{RB} nevertheless, the obtained results still hold (with proper adaptations) also when the bound is probabilistic.

Equations (3.3) and (3.4) show that the end-to-end delay strongly depends on the number of network nodes on the path between sender and receiver. As it is shown by the numerical example in Section 3.3, queuing delay is the main contribution to the end-to-end delay[4].

The current architecture of the Internet has been designed to carry data traffic with a *best effort* service. In order to limit loss due to congestion in network nodes, buffers are large, thus leading to a large Q_{max}. Moreover, the number of hops on the path between sender and receiver can be very large (even some tens). Many applications for voice transmission over Internet heuristically choose the compensation delay and possibly adapt it to the actual delay experienced by samples. As a result the quality of a voice call is unpredictable: when the nodes on the communication path are lightly loaded the quality is acceptable. Instead, during heavy traffic periods, either the received signal is not intelligible or the end-to-end delay is long (up to 1 second) [20].

Nevertheless, we believe that a voice transmission service with a commercial like guaranteed quality can be provided over a packet switched network provided that (1) it exploits some mechanisms for supplying QoS guarantees, and (2) its topology be designed so that the path of any voice connection (i.e., phone call) encompasses few nodes. In the following section we identify the key parameters for dimensioning such a network.

3.2 Designing a Packet Switched Telephone Network

Equation (3.4) shows that queuing delay depends on the ratio between buffer size and link capacity. Thus, the end-to-end delay can be decreased by increasing link capacity while keeping buffer dimension fixed. Nevertheless, the probability of losses due to congestion in network nodes is high if buffers are small with respect to link capacity. The problem can be overcome by creating separate queues for best effort and voice traffic, the latter being given

[4]Actually, in Equation (3.3) the term Q_{max} accounts twice being $\delta_{RB} = Q_{max}$ because the processing delay is constant, the queuing delay is the only variable component of the network delay, and Q_{max} is the maximum variation of the queuing delay (see Equation (3.2)).

higher *priority* than the former. In this scenario, Equation (3.4) still holds, B_i being the size of the voice queue alone.

The voice queue can be made small and losses avoided by introducing a call admission control mechanism, which limits the maximum number M_i of voice connections routed through a link. This determines the maximum number of packets present in the voice queue to

$$M_i - \left\lceil \frac{M_i}{I_i} \right\rceil \tag{3.5}$$

where I_i is the number of input ports of the i^{th} node. The voice queue is dimensioned so that it can contain the above amount of packets thus leading to a maximum queuing delay. Actually, due to the variation of delay experienced in upstream buffers, more packets than the number expressed by Equation (3.5) can be present in a queue. Nevertheless a maximum total queuing delay Q_{max}, as given by Equation (3.4), is not exceeded.

$$Q_{max} = P_S \cdot \sum_{i=1}^{N} \frac{\left(M_i \left\lceil \frac{M_i}{I_i} \right\rceil\right)}{C_i}. \tag{3.6}$$

Equation (3.6) shows that given the capacity of a link, the less phone calls are allowed on it, the smaller the maximum queuing delay of the connections traversing the link. The overall voice traffic routed on a link accounts only for a fraction of the link capacity, leaving to best effort traffic the remaining capacity. In order to provide an indication of the fraction of link capacity dedicated to voice traffic, we define the *voice allocation factor* α_i on link i, so that

$$\alpha_i \cdot C_i = M_i \frac{P_S}{P/R} \tag{3.7}$$

where P is the number of bits sent in each packet[5] and R is the rate of the voice stream, i.e., P/R is the time elapsed between two subsequent packets of the same voice connection. Since P_s is the size of packets, $(PsR)/P$ is the (gross) bandwidth used by the connection, taking into account the overhead due to packet headers. Assuming that all the phone calls on the network exploit the same coding (same bit rate R) and the same packetization technique (same P_s and P), the second term of Equation (3.7) is the bandwidth used by telephone traffic on link i.

[5]Throughout this work we assume P to be constant during a voice connection.

We define the *packetization efficiency* as

$$\eta_{pkt} = \frac{P}{P_S} \tag{3.8}$$

and use it to rewrite Equation (3.7):

$$\alpha_i \cdot C_i = M_i \frac{R}{\eta_{pkt}} \tag{3.9}$$

The packetization efficiency shows the fraction of link capacity wasted due to packet overheads (i.e., the header).

When dealing with variable size packets having fixed length header (e.g., in IP networks), the packetization efficiency is particularly relevant to the end-to-end delay. Increasing η_{pkt} requires packets to be enlarged, thus increasing some of the delay components. This can be made explicit by writing the packet size P_s in terms of the number OH of overhead bits

$$P_S = \frac{OH}{1 - \eta_{pkt}}$$

and substituting it in Equation (3.6) thus obtaining

$$Q_{max} = \frac{OH}{1 - \eta_{pkt}} \cdot \sum_{i=1}^{N} \frac{\left(M_i \left\lceil \frac{M_i}{I_i} \right\rceil\right)}{C_i}. \tag{3.10}$$

3.3 Numerical Results

In this section we consider the implementation of a PTN and provide some numerical results obtained by applying to it the equations devised in the previous section. The network topology is designed after the Telecom Italia's telephone network exploiting both IP and ATM technology.

In a circuit switched telephone network, telephones are connected through the local loop to local exchange offices which both have concentration functionality and encode analog voice signals into 64 Kb/s PCM flows. Local exchange offices are connected through digital links to local offices that are circuit switching nodes. In metropolitan areas, local offices are connected together and with toll offices which build up a higher layer of switching offices connected to each other, as shown in Figure 3.1.

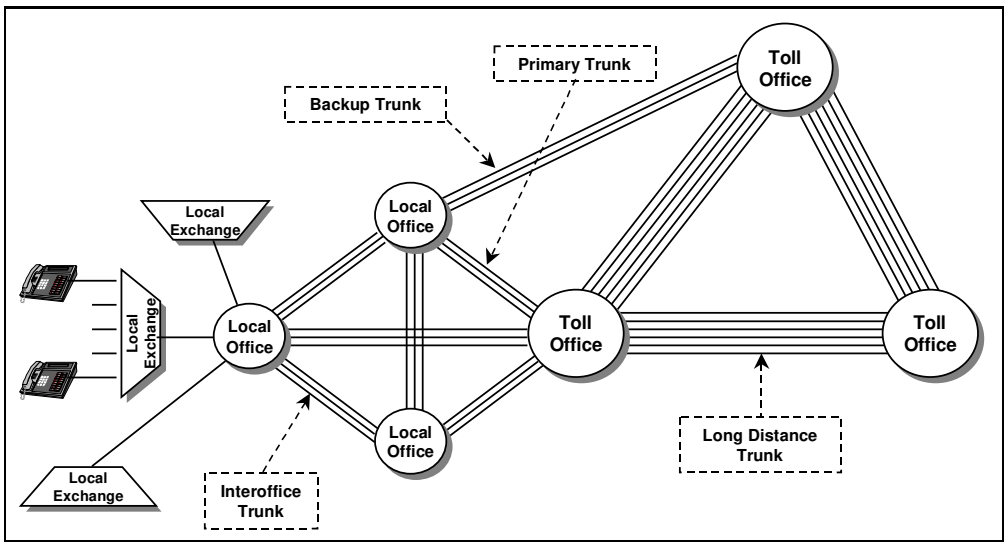

Figure 3.1: Excerpt from the Topology of a Circuit Switched Telephone Network.

The Telecom Italia's telephone network serves about 28 million users connected to about 11,000 local exchange offices which are, in turn, connected to 660 local offices. Local offices are fully meshed within metropolitan areas and each one is connected to at least two toll offices. The higher hierarchical level counts 60 toll offices connected in a full mesh. Finally, 6 international gateways connected to a number of toll offices provide international connectivity. The end-to-end delay of any call on the considered circuit switched telephone network, unless satellite hops are present along the communication path, is well below the interaction bound (usually shorter than 20 ms).

We analyze the delay perceived by users of the PTN obtained by substituting switching offices with packet switching nodes (IP routers or ATM switches). Each local exchange office is connected to a packet switching node. We assume that local exchange offices are the initiators of the call admission control procedures when users dial calls. Moreover, they are responsible for the voice packetization process. The capacity of the link between two nodes is determined according to the number M_i of telephone calls routed on the corresponding link of the real network, i.e.,

$$C_i = M_i \cdot n \cdot 64\,\text{Kb/s}, \tag{3.11}$$

where 64 Kb/s is the bandwidth used for each phone call and n is a factor used to oversize the link capacity in the PTN. Given the meshing of the Telecom

Italia's network and the capacity of the links between nodes as obtained from Equation (3.11), a packet switching node is required to have between 80 and 150 SDH STM-1interfaces[6].

When computing the queuing delay according to Equations (3.6) and (3.100 (for an ATM and IP network, respectively), we introduce an approximation by considering that all the hops on the path give the same contribution (i.e., we assume that $\sum_i Q_i = N \cdot Q$). This approximation is not a significant one since queuing delay basically depends on C_i/M_i which is constant because link capacity is dimensioned according to Equation (3.11)[7]. The value used for M_i is the maximum number of circuits on the link between two toll offices, i.e., M_i = 2,000.

Finally, to devise the results shown in this section, we consider encoding techniques which provide bit streams at 8, 16, 32, and 64 Kb/s. The encoding schemes currently deployed introduce delays ranging from a hundred of ms up to hundreds of ms. Since there are techniques operating at the same bit rate which introduce significantly different delays, the numerical results plotted in the figures of this section have been obtained considering $D_{proc} = 0$.

Figure 3.2 shows the delay (left y-axis) of a long distance call routed through 4 (left column) and 6 (right column) IP routers (star sign) or ATM switches (square sign). The delay is calculated through Equation (3.3) for a range of packetization efficiencies (x-axis). Three cases are considered for the capacity of links: $n = 1$ (dotted line), $n = 2$ (dashed line), and $n = 4$ (continuous line). Four encoding rates for the voice signal are considered (one in each row). On the right y-axis the voice allocation factor is plotted for each configuration. As shown by Equation(3.9), given a value for the packetization efficiency, the voice allocation factor does not depend on the packet characteristics (i.e., it has the same value for IP and ATM).

In the first two rows (voice encoding at 64 Kb/s and 32 Kb/s) the delay is

[6]For this calculation the Synchronous Digital Hierarchy (SDH) is assumed to be used to connect packet switching nodes. Due to the dimension of the problem, 155 Mb/s STM-1 carriers are the candidate links.

[7]Actually, a further approximation is introduced: the granularity of real SDH carriers is not taken into account when dimensioning the links among nodes. When actually building the network, an integer number of SDH carriers is installed between each pair of nodes in order to provide a capacity greater than the value computed using Equation (3.11). This yields a smaller value for α, i.e., actual delays shorter than those presented in this section.

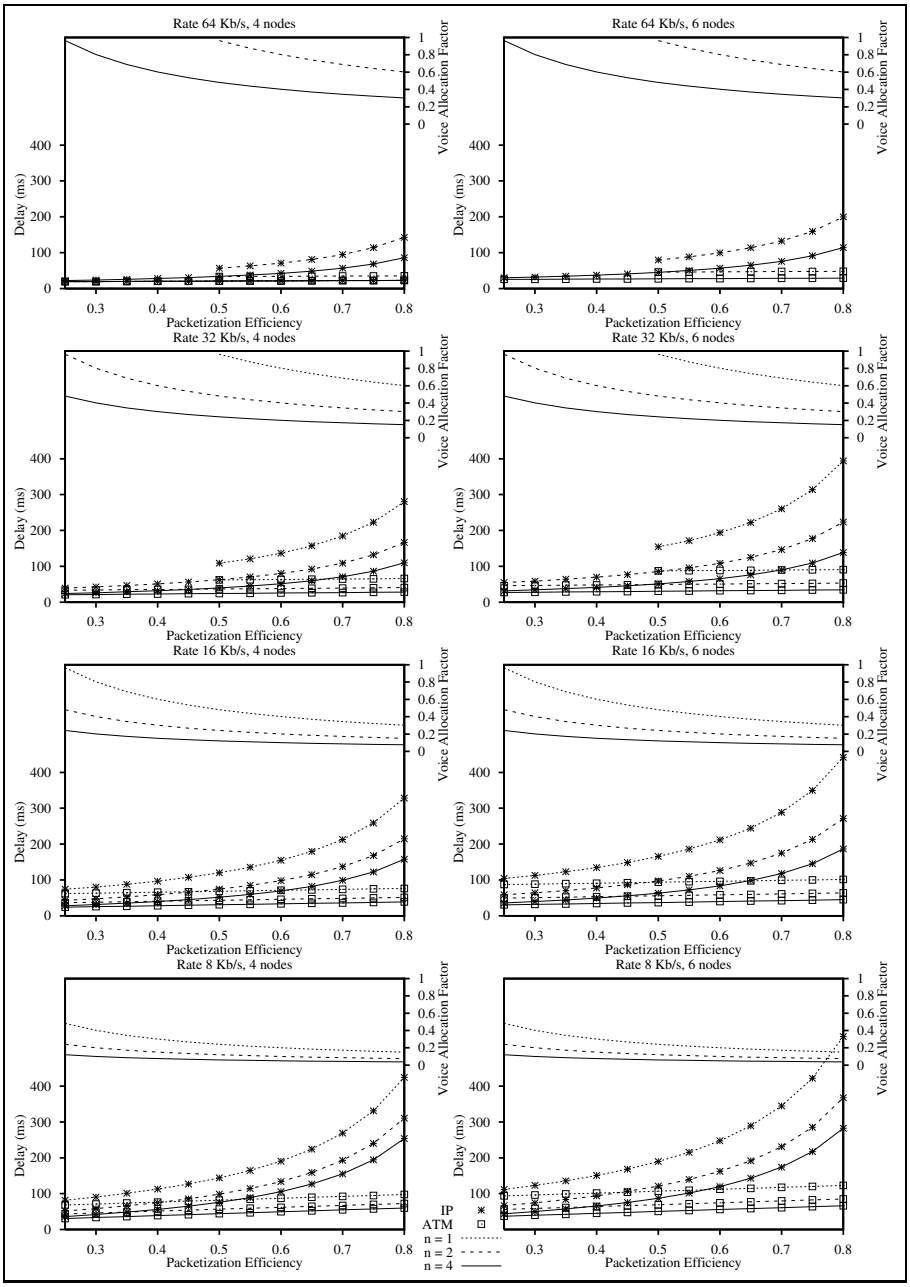

Figure 3.2: End-to-end Delay of a Long Distance Call.

not plotted some of the configurations for all or a range of the values of packetization efficiency. For example, in the upper left picture, for a link capacity double than in the circuit switched network ($n = 2$, dashed line), the delay corresponding to a packetization efficiency smaller than 50 % is not plotted. This happens because when $\eta_{pkt} < 0.5$, more than half of the link capacity is wasted to carry packet headers and thus there is not enough capacity to carry the same number of phone calls as in the original telephone network. In IP networks the delay grows more than in ATM networks as packetization efficiency increases. A cell is assumed to be sent as soon as its payload has been filled. This happens when AAL1 [31] encapsulation is used. Packetization efficiency is varied in ATM networks by filling only part of the cell payload[8]. This slightly varies the packetization delay, but does not change the queuing delay which, according to Equation (3.6), depends on cell dimension. Audio samples are assumed to be carried over IP networks using the protocol architecture shown in Figure 3.3. The *Real-time Transport Protocol* (RTP) [53] provides timing relationship between sender and receiver. The protocol headers result in a fixed per packet overhead. Thus, in IP networks packetization efficiency is increased by enlarging packets which translates in an increase of both packetization delay and queuing delay, as reflected by the steep slope of the curves.

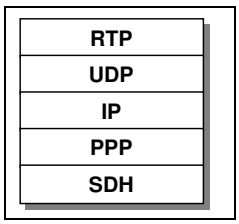

Figure 3.3: Protocol stack used to carry voice over IP networks.

When exploiting ATM technology, the end-to-end delay can be kept below the interaction bound (it is shorter than 100 ms in all the configurations). When IP technology is exploited, the interaction bound is respected only when operating at very low packetization efficiency (less than 50 %). As a consequence, more than half of the links capacity is wasted for transmitting packet headers. The voice allocation factor curves show that when packetization efficiency is high, a large fraction of link capacity is still available to best effort traffic ($1 - \alpha$). Decreasing packetization efficiency reduces the

[8]This feature is currently not supported by AAL1.

amount of best effort traffic that can be carried by the network, even though the number of phone calls supported is the same. Header compression [8] can be exploited to increase packetization efficiency while keeping packets small. This introduces higher storage and processing requirements on IP routers but decreases the average amount of overhead traffic generated on links. The effect of header compression on the end-to-end delay, link utilization (α), and buffer occupancy are out of the scope of this work.

Lastly, speech compression leads to unacceptable delay when exploiting IP technology and packetization efficiency higher than 50%. Using ATM, speech compression slightly increases the end-to-end delay[9], but leaves more capacity in the network for the transmission of best effort traffic (the voice allocation factor decreases).

In any configuration, increasing link capacity substantially improves the performance because queuing delay (inversely proportional to link capacity) is a major component of the end-to-end delay. The extra capacity is not wasted as it can be used by best effort traffic. In addition, over-dimensioning links appears to be sensible since bandwidth is an ever cheaper resource in both links and node ports (when using packet switching technology). If the network carries guaranteed traffic other than telephony, more complex queuing policies (e.g., weighted fair queuing [11]) should be used in nodes and link capacity should be dimensioned differently. General results remain almost the same.

3.4 Integrated IP and ATM Architectures for Packet Switched Telephone Networks

In this section we describe some architectural choices for a PTN based on the integration of IP and ATM.

[9]The increase in the end-to-end delay is due to the packetization delay: the lower the bit rate at the exit of the voice encoder, the longer the time needed to fill the packet payload.

3.4.1 Motivations

In the previous section ATM has been shown to be the best technology for telephony because it offers end-to-end delays lower than IP. Nevertheless, it is strongly desirable for a PTN to be equipped with IP forwarding capabilities for the following reasons:

- IP is the most widely deployed network protocol in both the Internet and many intranets, i.e., lots of applications based on the services provided by the TCP/IP protocol suite are daily deployed on many platforms. Many of them, for example electronic mail, news and the World Wide Web, are nearly ubiquitous.

- Being an internetworking protocol, IP enables communications among networks based on different technologies. On the contrary, native ATM applications require ATM to be exploited along the whole path between communicating entities.

- IP networks have a consolidated and widely supported management framework based on the *Simple Network Management Protocol* (SNMP) and the related definition of *Management Information Bases* (MIBs) for a variety of network devices.

Moreover, no particular advantage is given by the exploitation of ATM for high bandwidth real-time applications (like, for example, videoconferencing and video on demand), with respect to IP with QoS support. In fact, due to the high burstiness of the generated traffic, the gain in statistically multiplexing long burst of small cells over statistically multiplexing large packets is not significant, and queues in ATM switches grow almost as large as in IP routers.

Recently, various proposals for the operation of IP over ATM networks have been made. They aim at providing IP forwarding services while taking advantage of the high speed and, in some cases, the QoS guarantees offered by ATM. Some of these proposals exploit ATM as a data-link layer technology on which IP packets are transferred among routers (e.g., the Classical IP Model). Others, like *Tag Switching* by Cisco Systems Inc. [49], *IP Switching* by Ipsilon Networks Inc. [41], and *Cell Switch Router* by Toshiba Corp. [34]

are good candidate foundations for PTNs since they integrate IP and ATM mechanisms.

These integrated IP-ATM approaches aim at maximizing the exploitation of ATM switching and minimizing the use of IP forwarding. This translates in transporting IP traffic over ATM *Virtual Connections* (VCs) whose endpoints are as close as possible (possibly coinciding) to IP endpoints. The use of ATM signaling and routing protocols is extremely limited, if not avoided at all, i.e., only ATM switching functionality and possibly QoS guarantees are exploited.

The basic ideas common to the various integrated IP-ATM approaches may be summarized as follows:

- Integrated IP-ATM routers are based on ATM switching fabrics so that they can forward ATM cells coming from either IP endpoints (hosts or routers) equipped with ATM interfaces or other integrated IP-ATM routers.
- IP routers somehow identify traffic flows between IP endpoints and autonomously decide whether these flows are best served by classical hop-by-hop IP forwarding (e.g., e-mail, DNS queries, etc.) or ATM cell forwarding, i.e., flows must be carried on dedicated VCs (e.g., multimedia traffic, file transfers, etc.).
- When needed, integrated IP-ATM routers along the communication path create a dedicated VC "on the fly". IP packets are segmented into ATM cells at the source VC endpoint, forwarded cell-by-cell, and reassembled only at the destination endpoint.

In order to setup the dedicated VCs on which IP flows are transported, integrated IP-ATM routers communicate among themselves (and possibly with IP endpoints equipped with ATM interfaces) through simple and efficient service protocols. They allow integrated IP-ATM routers to exchange information about IP flows and the associated VCs. Integrated IP-ATM routers setup and tear down ATM VCs controlling directly their own switching fabric. The various integrated IP-ATM approaches essentially differ in the way these service protocols operate.

If a PTN is based on one of the integrated IP-ATM approaches discussed above, telephony has a quality (in terms of end-to-end delay) close to the

one obtained over an IP network. Over an IP network, voice samples are put into IP packets and forwarded by routers, as shown in Figure 3.4a. Over an integrated IP-ATM network, IP packets containing digital speech samples are segmented into ATM cells which are forwarded by the ATM switching fabrics to their destination, as depicted in Figure 3.4b. The IP like performances stem from the fact that (1) cells are sent into the network in bursts of the dimension of an IP packet, thus affecting buffers like IP packets, and (2) at the receiver IP packets must be reassembled before the contained voice samples can be played back. Thus, speech samples cannot be played as soon as ATM cells arrive, but only when the last cell of each IP packet has arrived. As a result, the delay of the whole packet is that experienced by the last ATM cell.

3.4.2 Integrated Services IP-ATM Models

In order to get better performance from a PTN, we propose to devise *integrated services IP-ATM* models derived from the integrated IP-ATM approaches described above. The basic idea behind integrated services IP/ATM is that once an end-to-end VC is in place, digital speech samples are encapsulated directly into ATM cells (Figure 3.4c), i.e., they are transmitted like over AAL1. At the receiver, voice samples are played back as soon as the ATM cell carrying them arrives. Current integrated IP-ATM approaches use the *Unspecified Bit Rate* (UBR) class of service, i.e., they support only best effort traffic. On the contrary, telephony requires guaranteed QoS. Since most speech encoding schemes produce a constant rate stream, integrated services IP-ATM routers must provide the *Constant Bit Rate* (CBR) class of service.

In order for a phone call to immediately get its dedicated VC, integrated services IP-ATM routers should be able to set up the VC without having to previously identify a traffic flow. Some alternative solutions can be envisioned:

1. A "signaling"IP packet is used to announce the beginning of a phone call; all routers along the communication path should react to it by setting up a VC dedicated to the phone call. When a phone call has terminated, the corresponding VC must be torn down. This can be

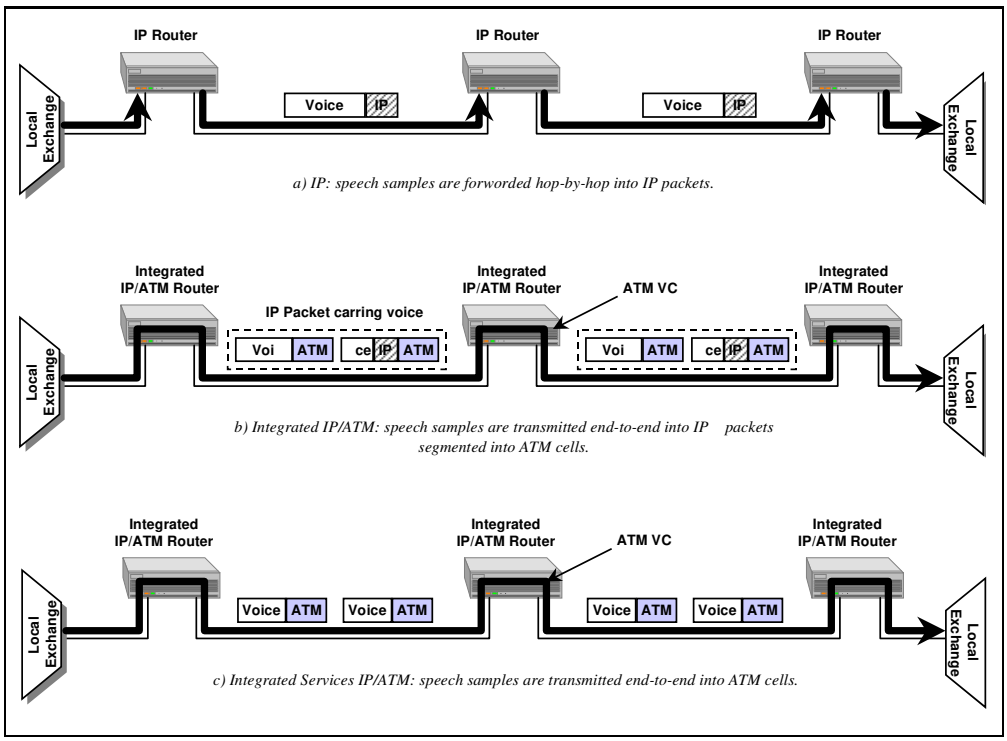

Figure 3.4: Forwarding models.

accomplished by means of a time-out mechanism (already exploited in some of the integrated IP/ATM approaches). This mechanism can be quite reactive since while a call is active, traffic is generated regularly.

2. Telephony applications can be written to exploit *User to Network Interface* (UNI) signaling [17]. Integrated services IP-ATM routers must implement UNI signaling and *Network Node Interface* (e.g., PNNI [14]) routing[10]. ATM signaling allows telephony applications to require the network for the needed QoS, while best effort and multimedia IP traffic is still handled as in integrated IP/ATM approaches. Moreover, native ATM applications (independent of their QoS requirements) are fully supported by the PTN. The main drawback of this solution is that integrated services IP-ATM routers must run a considerably more complex and large software with respect to the previous model.

3. The amount of software running on integrated services IP-ATM routers

[10]These are usually implemented in into ATM switches.

can be reduced by exploiting the *Integrated Private Network Node Interface* (I-PNNI) routing protocol [7] to carry routing information for both IP and ATM. Routers still support both UNI signaling and service protocols exploited in integrated IP/ATM approaches, but IP routing protocols are not run any more.

3.5 Discussion

Voice transmission over packet switched networks is here studied from the point of view of the quality perceived by the user in terms of delay. The results show that the Internet cannot provide good quality, except in particular conditions. Nevertheless, if a packet switched network is expressly designed for telephony by (1) limiting the number of hops in the path between any pair of users and (2) dimensioning link capacity properly, the end-to-end delay is low enough to allow for interaction. Thus, packet switching could be exploited to build a commercial like telephone network even though the raw capacity needed is larger than with circuit switching. This is justified by the possibility of carrying best effort traffic on the same network and by the lower installation and management costs.

The numerical results also show that ATM outperforms IP as was expected since the former was expressly designed to support low bit rate real-time traffic. Nevertheless, because of the large number of existing applications based on IP, it must be taken into account as a candidate technology for carrying both best effort and high bandwidth real-time traffic in packet switched telephone networks.

Thus, we propose that both IP and ATM be employed in the implementation of packet switched networks for telephony based on *integrated services IP-ATM* models. These models are derived from the integrated IP-ATM approaches currently being exploited for an effective operation of IP over ATM networks. Integrated services IP-ATM models provide ATM based services for telephony and IP (over ATM) based services for any other kind of traffic (namely, best effort and high bandwidth real-time) in order to get the best from the two technologies.

Further work shall be done in order to evaluate more carefully the performance of the proposed integrated services models.

Chapter 4

Efficiency of Packet by Packet Generalized Processor Sharing

Packet scheduling algorithms are exploited in asynchronous packet switching networks to provide guarantees on the Quality Of Service (QoS). The application of packet scheduling techniques into nodes is combined with the exploitation of call admission control algorithms that keep track of the number of real-time sessions sharing a resource (e.g., the capacity of a link) in order to keep the utilization low enough to guarantee the required QoS. The objective of this chapter is to evaluate the efficiency of providing *deterministic QoS guarantees* over asynchronous packet switching networks which exploit Packet by Packet Generalized Processor Sharing (PGPS) packet scheduling in their nodes. The utilization of the network is compared to the utilization achievable on a circuit switching network which is traditionally suitable to provide deterministic real-time guarantees.

Chapter 3 takes into consideration the application of static priority to two classes of traffic in order to guarantee a delay bound on the high priority traffic. The implementation of toll quality telephony over a packet switching network is used as a case study. The analysis shows that given the voice encoding algorithm and a bound on the end-to-end delay perceived by users, the network is not able to accept the same number of phone calls that would be accepted on an equivalent circuit switching network. This is due to the fact that the amount of delay spent due to queuing into high priority queues is proportional to the amount of real-time traffic accepted on the network. In other words, packet scheduling into network nodes affects the utilization

of the network by real-time traffic.

In this chapter we extend the analytical results devised in Chapter 3 to the exploitation of PGPS as the packet scheduling mechanism into the network. Similarly to what is done in Chapter 3, the implementation of toll quality telephony on a packet switching network is considered as a case study. A call level simulator is implemented to study the efficiency in the utilization of the network assuming that the number of phone calls accepted on the network is limited by a Call Admission Control (CAC) mechanism which takes into account the packet scheduling algorithm deployed into nodes. The simulator allows the analytical study of Chapter 3 to be extended to a scenario with heterogeneous sources.

Our call-level simulator which generates real-time sessions with pre-definable QoS requirements and duration. A route is found for the session and network resources are considered over the selected route(s). Some CAC mechanism is used to decide whether the amount of available resources is sufficient to provide the session with the required level of QoS. The simulator records the percentage of sessions rejected over the total number of sessions with the aim of devising the average utilization of links.

The simulator is fully configurable to simulate arbitrary network topologies and sessions. Since also packet dimension (in terms of header and payload) is configurable, the simulator is suitable to simulate any packet switching technology (e.g., both IP and ATM). A telephone network implemented using IP technology is considered as a case study in this chapter. This choice stems from the large diffusion of IP networks (namely, the Internet) and the current growing interest transmitting voice on packet switching networks and in particular over the Internet. We move a step further investigating the possibility of providing toll-quality telephony over IP networks.

4.1 The Simulator

The simulator, completely written in C++, allows the description of arbitrary topology networks. Network users are connected to nodes and configured to generate calls with statistically or deterministically predefined destinations and QoS requirements.

When a source generates a call[1], a *routing module* chooses a route for the call. The current routing module takes its decisions according to static routes manually configured prior to starting the simulation run.

All the nodes on the path from source to destination are checked against the amount of resources needed to satisfy the QoS requirements of the call. The amount of resource needed given a QoS requirement is determined according to a Call Admission Control (CAC) rule which is related to the scheduling algorithm used into nodes and to the kind of QoS guarantee that is to be provided.

Depending of whether all the nodes dispose of enough resources, the call can be accepted or blocked. A statistical module records the outcome of each call and statistically evaluates the blocking probability and the average utilization of links.

The sequel of this section describes in more detail the CAC and the statistical module together with the performance indexes it is used to measure.

4.1.1 Call Admission Control

Each call generated by a source is directed to a specific destination, has a certain duration, and explicit QoS requirements. The latter are given in terms of bandwidth of the call and maximum delay introduced by the network. The queuing scheme used in network nodes combined with the proper resource reservation policy, can guarantee a bound on the delay experienced by packets in each node.

Th CAC mechanism has to (1) devise the amount of bandwidth that has to be reserved to a call in each node in order to keep the overall delay below the given bound and (2) check whether each node disposes of the needed amount of bandwidth. If enough resource is still available, the call is accepted, otherwise it is rejected.

The CAC mechanism implemented in this work assumes that the Packetzied Generalized Processor Sharing (PGPS) [46, 47] algorithm for packet scheduling is used into network nodes. As proven in [47], the PGPS provides a bound

[1]If the simulator is used to study a connectionless network, the call can be viewed as a real-time session with the associated destination, starting instant, and duration. The network is asked to reserve resources for the call in order to provide the required QoS.

on the delay which depends on many factors, among which the bandwidth reserved to a session. In particular, the delay introduced by each node in the network is inversely proportional to the bandwidth reserved on the outgoing link.

The end-to-end delay required by a call is first reduced by the time real-time data spend in both the source and destination end systems. This time is due to processing (e.g., compression of audio or video) and packetization[2]. The CAC algorithm considers the residual delay budget equally distributed over all the nodes on the selected path from source to destination. In other words, the minimum amount of bandwidth to be reserved on each link of the path from source to destination is devised through Inequality (39) of [46] which provides the deterministic bound on the delay guaranteed by the PGPS scheduling algorithm:

$$D_i \leq \frac{\sigma_i + 2 \cdot (K-1) \cdot L_i}{g_i} + \sum_{m=1}^{K} \frac{L_{max}}{r^m} \tag{4.1}$$

D_i represents the delay bound for session i, σ_i represents the burstiness of the session (i.e., it is the dimension of the token bucket used by the shaper/policer regulating the access of traffic into the network), K is the number of nodes on the selected path, L_i is the maximum packet dimension used by the session, and r^m is the capacity of the m^{th} link on the route.

The CAC checks whether each node on the call path has an amount of available (not yet reserved) bandwidth larger than $\min(\rho_i, g_i^*)$, where ρ_i is the bandwidth required by the i^{th} call and g_i^* is the minimum g_i value that satisfies Inequality (4.1) when D_i is the net delay budget for the session. If enough bandwidth is available, it is reserved to the call on all the traversed links. When a call is closed, the bandwidth reserved to it is released on all the links on the path between source and destination.

Inequality (4.1) shows that calls which have the same QoS requirements (in terms of bandwidth and delay bound), but are routed along paths with different number of nodes, require a different amount of bandwidth to be reserved. In particular, the higher the number of nodes traversed by a call, the larger the bandwidth which must be reserved.

[2]See Sections 3.1 and 6.2.1 for a detailed analysis of the delay components of telephony and videoconference applications, respectively.

At a first glance the proposed approach seems to work only with connection oriented protocols. Actually, whenever QoS guarantees are required, an application must ask the network for reserving resources through some sort of signaling protocol. This reservation installs into nodes state information about the application's flow, which is equivalent to the state information stored for a call in a connection oriented network. The main difference is that in the connectionless network the reservation is *soft-state*, i.e. it has to be renewed periodically.

4.1.2 Measurements and Statistics

Four indexes are monitored to provide insight in the efficiency of the network:

1. The *effective load* is the net bandwidth required by the application, not taking into account the protocol overhead.
2. The *real load* is the raw link capacity required to transmit user data, i.e., the effective load augmented with the overhead due to the headers of the various protocol layers.
3. The *equivalent load* is the bandwidth that must be reserved to a session in order to meet its QoS requirements.
4. The *call blocking probability* is the ratio between the number of calls rejected and the total number of calls offered to the network.

The first three parameters are directly related to the effectiveness in the usage of bandwidth, while the latter gives information on the quality of the service provided by the network.

Simulations are run for different levels of call load offered to the network, i.e., for increasing average number of calls per second generated by the sources and possibly increasing average duration. The above indexes are measured by gathering data during the simulation. A *statistical module* is embedded in the simulator to ensure that the obtained numerical results have statistical relevance.

Data gathered during the initial part of the simulation should not be considered in the evaluation of the performance indexes. In fact, when a simulation

run starts, the network is idle, i.e., all the resources are available and all the calls are obviously accepted. Then, as far as calls are generated by users, the network utilization increases and reaches a stable level. The performance indexes should reflect the network situation in the stable phase. Nevertheless, data gathered during the initial part of the simulation, called the *transient phase*, distort the information obtained on the network.

Each gathered data is passed to the statistical module which identifies the end of the transient phase. Given an index under statistical evaluation, the statistical module computes the mean over a predefined number n of samples. The transient phase is considered over if the relative difference between each of the last N means and their average mean is smaller than a predefined threshold ϵ. The parameters that determine the completion of the transient phase must be chosen empirically. When the transient phase is considered over, the statistical module discards all the collected data.

Then, the statistical module determines when the simulation can be stopped since the performance indexes have reached a steady state and are not going to change significantly. This is considered to happen when the *confidence interval* of the samples is smaller than a predefined threshold. The confidence interval is the range in which the average over a fixed number of samples falls with given probability. The confidence interval is computed using the *central limit theorem* which states that the average of n samples of a stochastic variable with mean μ and variance σ^2, has a Gaussian probability distribution with the same mean and with variance σ^2/n.

4.2 A Case Study: Toll-quality Telephony

Telephony has been used as a case study for a real-time application. Telephony and best-effort traffic are carried on the same network and nodes use PGPS to guarantee a deterministic delay bound to each phone call. The results devised in this section hold both in case the network carries only telephony and in case it carries also other traffic (real-time and/or best-effort).

The effective load of each call is given by the voice encoding algorithm while the probability distribution of calls depends on the mix of traffic (residential and business) assumed on the network. The working principle of the sources used in the simulation and the rationale behind them is detailed in

Section 4.2.2. The effective load and the delay requirement of the call are used by the CAC to devise the equivalent load and determine whether the call can be accepted, as explained in Section 4.2.1.

Two different network topologies are taken into account: the first is simpler and features a number of sources generating calls with the same QoS requirements, while the second takes into consideration phone calls with heterogeneous bandwidth requests. The two topologies and the simulation results are described in Section 4.2.3 and Section 4.2.4, respectively.

4.2.1 Call Admission Control

The equivalent load g_i of call i is devised from the following inequality which is derived from Equation 2.2 simply adding the propagation delay P:

$$D_{req} \geq (h_i + 1) \cdot \frac{P_{S,i}}{g_i} \sum_{m=1}^{h_i} \frac{P_M}{r_m} + P \tag{4.2}$$

where h_i is the number of hops traversed by phone call i, $P_{S,i}$ is the packet size used by this phone call, P_M is the maximum packet size deployed over the network, r_m is the overall service rate of the packet scheduler (usually the capacity of the m^{th} link on the path of the phone call), P is the total propagation delay on the $h_i + 1$ links traversed by the call, and D_{req} is the delay requirement of the call. Inequality (4.2) takes into account the packetization delay inside the first addendum; moreover the processing delay to encode and decode the voice signal is neglected.

If we call e_i the real load of call i obtained by increasing the effective load by the protocol overhead, and g_i^* the minimum g_i which satisfies Inequality (4.2), the equivalent load of call i is given by $\min\{e_i, g_i^*\}$. The CAC accepts a call if the sum of the equivalent load of all the calls routed on the links traversed by call i (included call i itself) does not exceed the link capacity.

4.2.2 Implementation of Sources

We consider that access to the network is provided as described in Chapter 3: each telephone is connected to a local exchange through the subscriber loop

and local exchanges are in charge of encoding and packetization. The internal part of the network (among local exchanges) is built with packet switching technology, i.e., local offices and toll offices (shown in Figure 4.1 which has been borrowed from Chapter 3) traditionally built in circuit switching technology in a telephone network are substituted with IP routers or ATM switches.

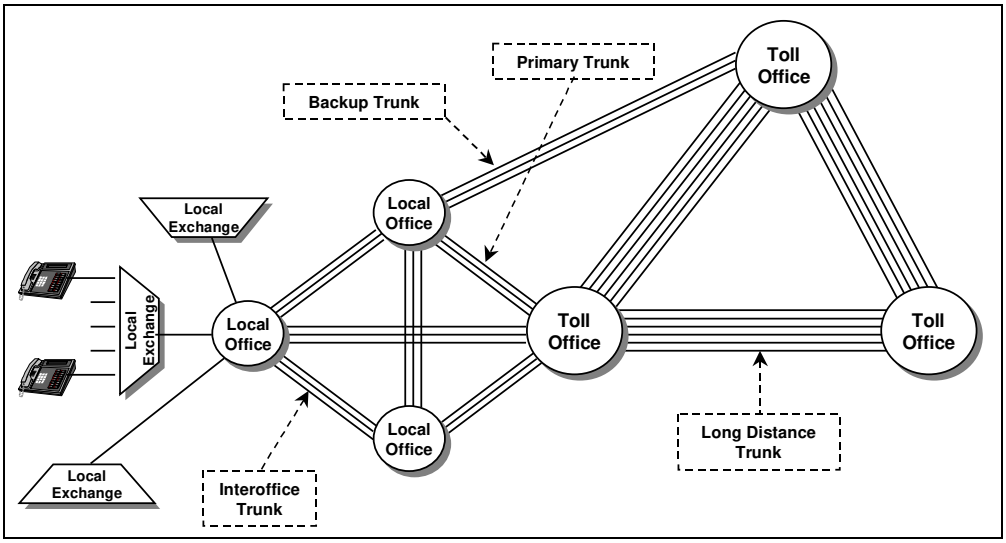

Figure 4.1: Excerpt from the Topology of a Circuit Switched Telephone Network.

Thus, in the following local exchanges, rather than customers' telephones, are considered call sources. A class hierarchy implements users' behavior by simulating the call generation process which would originate from a set of telephones. Each call requires a predefined amount of bandwidth and has a randomly chosen duration; the time between subsequent calls is also randomly determined.

The bandwidth required by a phone conversation depends on the encoding scheme exploited; the simulator implemented for this work encompasses four types of encoding:

1. Pulse Code Modulation (PCM) is the encoding traditionally used in digital telephone networks. It samples the voice signal every $125\mu s$ and digitalizes each sample on 8 bits using a non linear encoding based either on the μ Law or on the A Law. As a result a PCM encoder produces a CBR flow at 64 Kb/s.

2. ADaptive PCM (ADPCM) encoders use *differential encoding* exploiting the temporal redundancy in the voice signal to reduce the bandwidth of the encoded flow. The ITU-T Recommendations G726 and G727 specify the *Embedded ADPCM* encoding for output rates of 40, 32, 24, and 16 Kb/s.

3. Linear Predictive Coding (LPC) aims at devising a set of parameters that represent a segment of a voice signal and allow it to be reconstructed. The voice segment is modeled by a linear system which has in input a predefined signal. The system is characterized by a set of parameters (from 8 to 12) and a gain constant which are periodically devised from the voice signal being encoded. The time period for recalculating the coefficients varies from 10 to 30 ms. The coefficients constitute the encoding of the voice segment and are sent to the receiver that uses them to reconstruct the linear model, feeds it with the predefined signal, and obtains the reconstructed voice segment. The CBR flow generated by the encoder has a bandwidth of 2.4 Kb/s to 9.6 Kb/s.

4. GSM encoders sample the voice signal at 8 kHz and digitalize each sample on 13 bits. Each group of 160 words (containing 20 ms of the voice signal) is independently compressed to 260 bit (i.e., with a compression ration of 1:8) thus producing a 13 Kb/s CBR flow.

5. Code-Excited Linear Predictive (CELP) encoders also devise the parameters of a model of fixed duration segments of the vocal signal. They produce a CBR flow at 4 Kb/s to 16 Kb/s.

Often, for network dimensioning purposes, phone calls are considered to have a duration of 3 minutes and inter-arrival times distributed according to the Poisson probability distribution. This model does not fit to the call pattern present on telephone networks, due mainly to short phone calls. [4] proposes a more accurate model in which the call duration is distributed according to a probability distribution obtained by the weighted composition of 3 functions:

$$F(x) = w_s \cdot F_s(x) + (1 - w_s) \cdot [\alpha \cdot F_1(x) + (1 - \alpha) \cdot F_2(x)]$$

$F_s(x)$, weighted from 1 to 3%, accounts for short calls (shorter than 3 seconds); even though the probability distribution of short calls is quite complex, $F_s(x)$ approximates it with a uniform probability distribution. $F_1(x)$ and $F_2(x)$ are Gaussian logarithmic distributions and account for all the other kinds of calls (taking into consideration both residential and business users). Figure 4.2 shows the probability density of the duration of the calls generated by the simulator according to the described model.

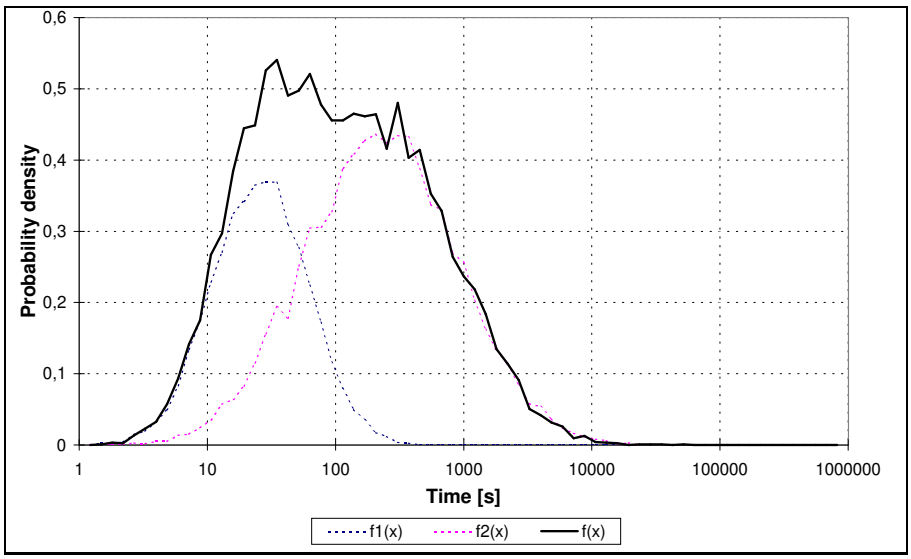

Figure 4.2: Probability density of call duration as generated by the simulator.

When objects modeling sources are instantiated, a set of parameters is specified to characterize the users. The simulator allows to set the average number of calls per hour which determines the average inter-arrival time whose probability distribution which can be set to be either a Poisson or a Gauss one. The codec determines the effective load required by each call; the real load is determined adding the protocol overhead. We consider to implement the network in IP technology through a number of routers connected by point-to-point links running the PPP line protocol. This leads to the protocol stack shown in Figure 4.3; the sum of the headers of all the layers in the protocol model yields a protocol overhead of 384 bits for each voice packet. The number of samples per packet must be a tradeoff between overhead efficiency and packetization delay.

Given a network topology, simulations where run for increasing offered load in terms of calls per hour, with the goal of devising the maximum achievable

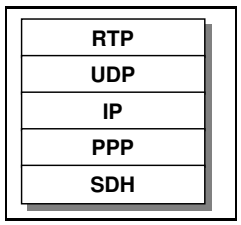

Figure 4.3: Protocol stack used to carry voice over IP networks.

utilization of the network. Since the call load offered to the network depends on call duration besides call frequency, in the results presented in this work the offered load is expressed in *Erlang*, i.e. call frequency times average call duration.

4.2.3 Uniform Sources

Figure 4.4 depicts the topology of the network under study. The two sources `S1` and `S2` generate phone calls towards `D1` and `D2`, respectively. Each phone call requires an effective bandwidth of 32 Kb/s as with ADPCM32. This simple topology is characterized by all the calls traversing the link from `B` to `C`; this link is the one loaded most and thus it is kept under statistical observation.

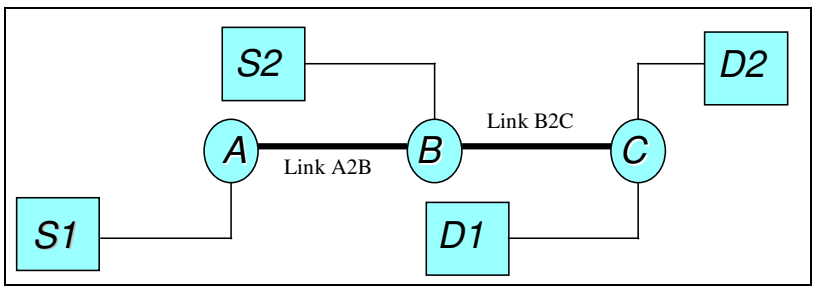

Figure 4.4: Network topology used in the simulations.

All the links[3] have a capacity of 34 Mb/s; access links are characterized by a propagation delay of $5\mu s$ (i.e., local exchanges are 1 Km away from local offices), while backbone links have a propagation delay of $50\mu s$. If this topology is related to a telephone network, it represents a typical scenario for

[3]Notice that all the calls traverse the links in the same direction; thus, links can be considered uni-directional without loss of generality.

local calls since node B can be seen as a toll office to which two local offices (A and C are connected through 10 Km links. The most demanding calls are, of course, long distance ones; the network efficiency in supporting long distance calls can be devised by running simulations on this topology with an end-to-end delay target reduced by the propagation delay over a long distance path. In fact, various simulations results are shown for different end-to-end delay requirements. A delay of 100 ms is generally considered not to be annoying for interaction; simulation runs with shorter delay requirement are take into consideration long distance phone calls which spend most of the end-to-end delay budget in traversing long trunks.

Packetization delay is varied across simulations; the higher the number of samples gathered into each packet, the lower the transmission overhead introduced by the IP header. Nevertheless, gathering a large number of samples in a packet requires part of the end-to-end delay budget to be spent waiting for enough samples be produced by the encoder. As a result, less capacity is wasted in overhead, but more capacity has to be allocated to satisfy the delay bound. The numerical results show a possible tradeoff between packetization overhead and link utilization.

The two sources generate calls with the same encoding (i.e., effective load), the same packetization delay (i.e., the same real load), and the same end-to-end delay bound. Nevertheless, since the calls from different sources span a different distance, the network must reserve different amounts of extra bandwidth to meet the queuing delay budget (i.e., calls from different sources generate a different equivalent load).

For each simulation run two graphs are shown, both expressing the offered call load in Erlang. The first plots the effective, real, and equivalent load on the link from B to C, while the second plots the average number of accepted calls (left Y axis) and the blocking probability (right Y axis) on the same link.

Uniform Link Capacity

The results shown in Figure 4.5 are devised imposing a packetization delay of 16 ms. This implies that the payload of each packet is 512 bits; considering that the header is 384 bits, the packetization overhead is 43%. This is shown by the distance from the effective load and the real load curves

in Figure 4.5. The fact that the equivalent load and the real load are the same, shows that the network allocates to each call its real bandwidth with no need for overallocating bandwidth in order to reduce the queuing delay to meet the delay bound which is set at 80 ms. When no extra bandwidth is reserved to a phone call (i.e. real and equivalent load are the same), the maximum delay experienced by a packet in each node implementing PGPS is its packetization delay. The blocking probability grows higher when the overall bandwidth required by calls (i.e. the aggregated equivalent load) offered to the network require for an overall bandwidth is larger than the capacity of the link between `B` and `C`.

In order to reduce the packetization overhead, the number of voice samples per packet has been increased up to an overall amount of 640 bits introducing a packetization delay of 20 ms. Figure 4.6 shows the results of the simulation runs with this setting, still imposing an end-to-end delay bound of 80 ms. The first graph shows that the distance between the real load curve and the effective load one is reduced with respect to Figure 4.5. Nevertheless, the bandwidth actually reserved to the calls is the one represented by the equivalent load curve and the number of accepted calls is smaller than with a lower packetization efficiency. The capacity represented by the zone between the equivalent load and real load curves is left available to best effort traffic. Summarizing, a lower packetization overhead leaves some capacity to best-effort traffic, but also reduces the total number of phone calls accepted on the network.

The effective load is the fraction of link capacity that a phone call would get on a circuit switching telephone network. Thus, the difference between the equivalent load and the effective load curves provides a comparison of the efficiency of carrying voice over the packet switching network versus carrying it on the traditional telephone network.

High Capacity Backbone and Short End-to-End Delay

The results devised so far refer to local calls; satisfying the delay requirement is more resource consuming for long distance calls for which a large part of the 100 ms delay budget is spent in propagation over links. It is possible to evaluate the efficiency of a long distance call while keeping the same topology, by simply decreasing the call end-to-end delay requirement. In this second set

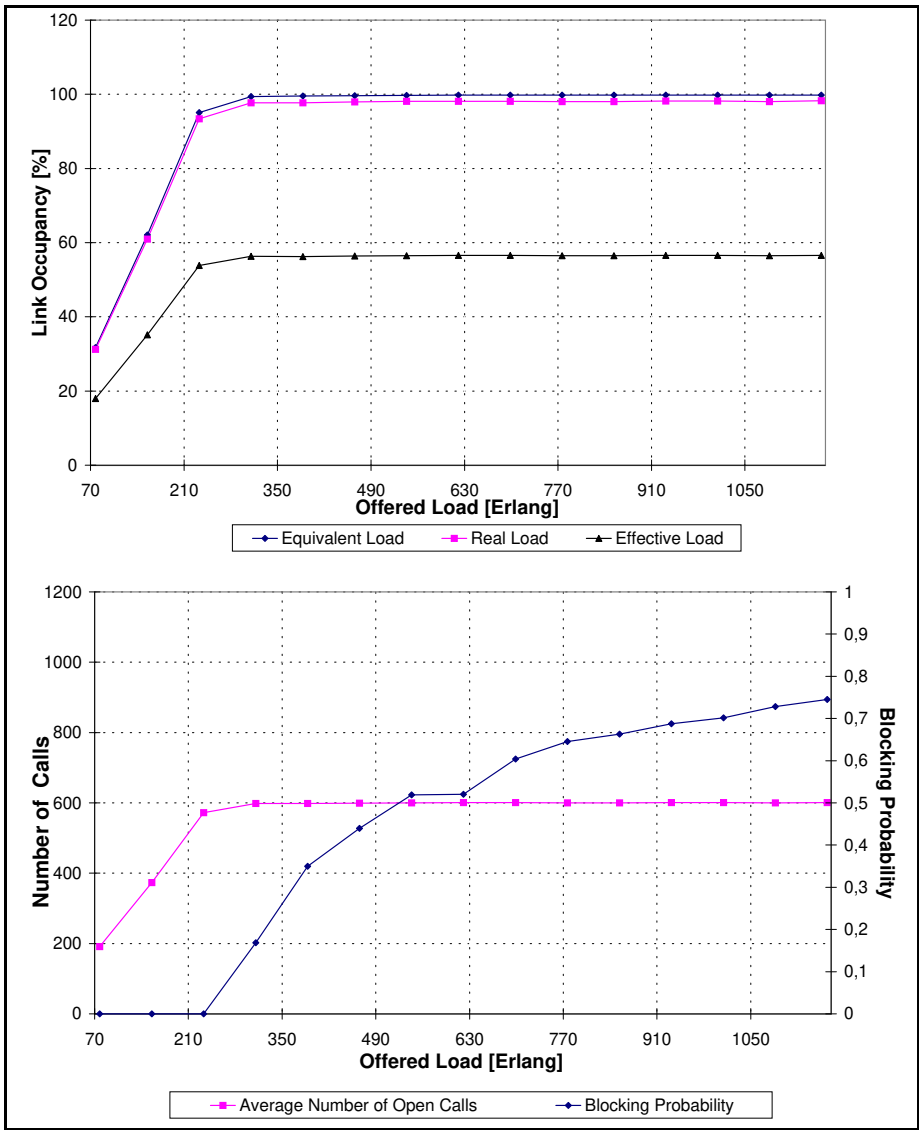

Figure 4.5: Link at 34 Mb/s, packetization delay of 16 ms, and delay bound of 80 ms.

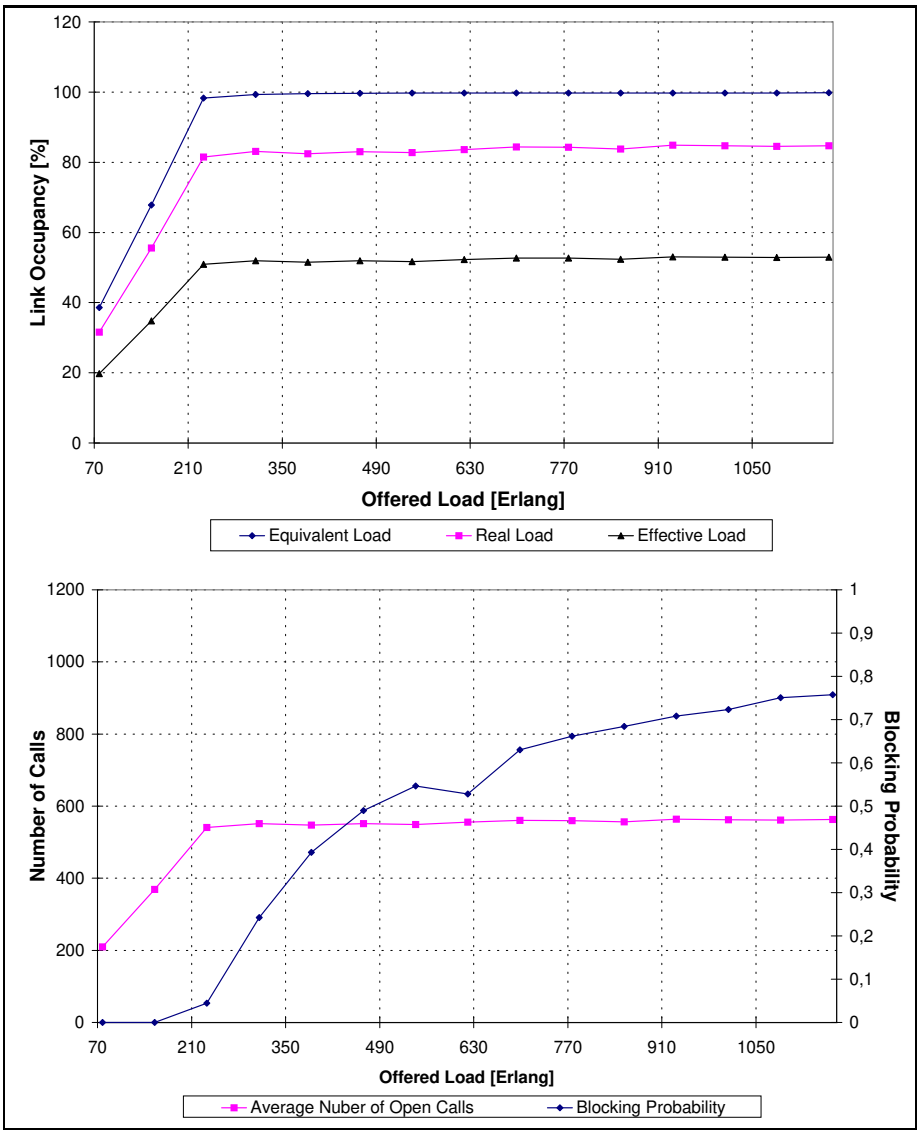

Figure 4.6: Links at 34 Mb/s, packetization delay of 20 ms, and delay bound of 80 ms.

of simulations it has been set to of 10 ms which is roughly the delay budget which is left to packetization and queuing on a transoceanic call, after the propagation delay has been subtracted from the interaction bound of 100 ms. The delay introduced by the links of the simulated topology (which is taken into account by the CAC during the simulation) can be considered the delay spent by packets on the access section of the network, i.e. to reach the long distance carrier.

Requiring such a low delay bound imposes shorter packetization delays which increase the percentage overhead and thus the real load on the network. Thus, in these set of simulations the capacity of backbone links (from `A` to `B` and from `B` to `C`) is increased to 155 Mb/s. Figure 4.7 shows the results with a packetization delay of 4 ms, i.e., a payload of 128 bits and a 75% packetization overhead which is reflected by the difference in between the real load and the effective load curves. Moreover, due to the low end-to-end delay budget, excess bandwidth must be allocated as shown by the equivalent load curve. This implies that the maximum number of calls accepted on the network is small. Notice that less than half of the capacity of the link from `B` to `C` is reserved, but no more calls can be accepted since the whole capacity of the 34 Mb/s access links has been allocated.

The packetization overhead can be reduced by doubling the number of samples carried by each packet, as done in the simulation runs of Figure 4.8, where the distance between the real load and effective load curves is smaller than in Figure 4.7. This leads to a packetization delay of 8 ms which almost accounts by itself alone for the whole end-to-end delay budget. As a consequence a large amount of bandwidth must be reserved for each call in order to guarantee a short service time in each traversed node. This is represented by the huge distance from the equivalent load and real load curves in Figure 4.8, but even more by the number of calls accepted by the network dropping from almost 200 to a little bit more than 30.

Optimal Packet Size

The above results suggest that there should exist an optimal packet dimension which maximizes the number of calls accepted on the network (i.e., minimizes the call blocking probability). Figure 4.9 plots the average number of accepted calls and the call blocking probability versus the packetization

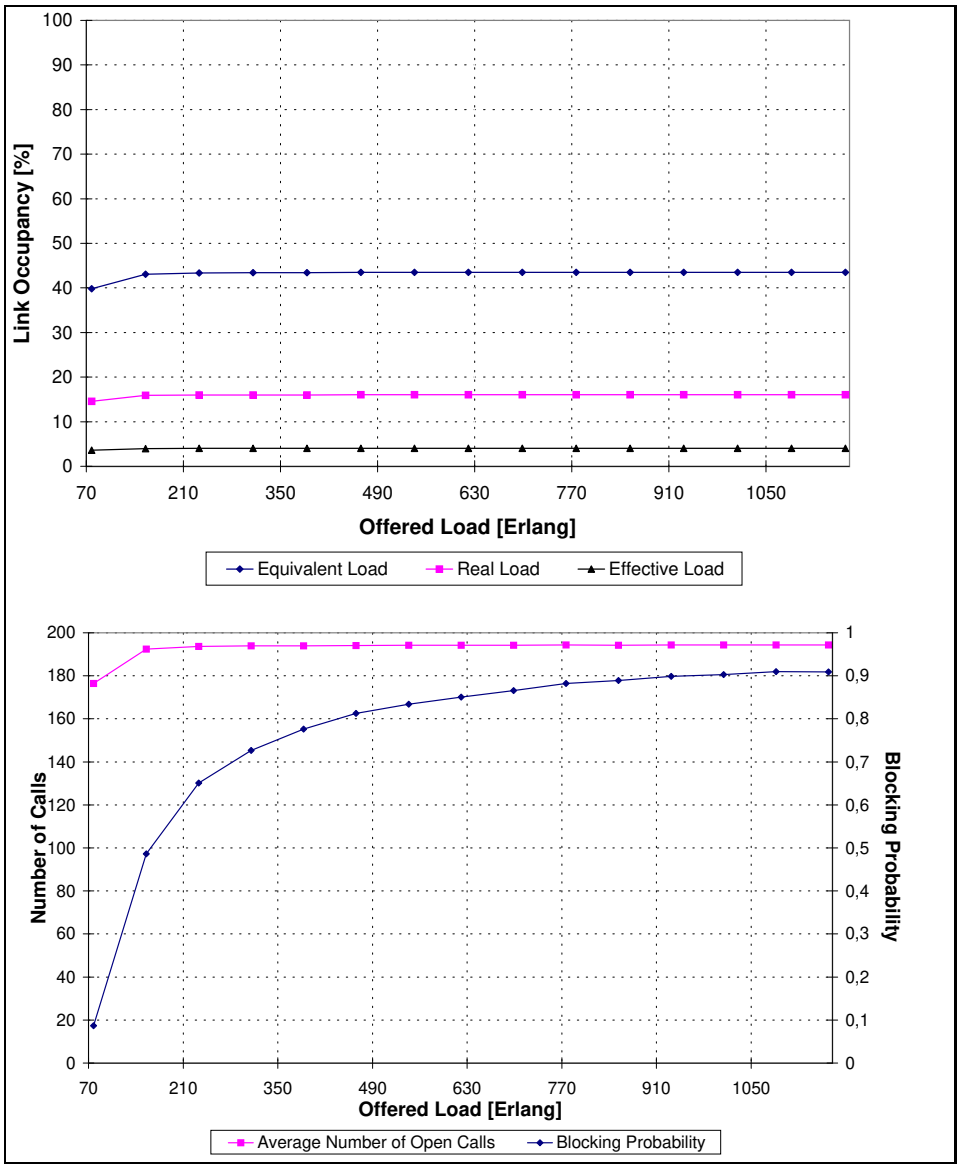

Figure 4.7: Access links at 34 Mb/s, backbone links at 155 Mb/s, packetization delay of 4 ms, and delay bound of 10 ms.

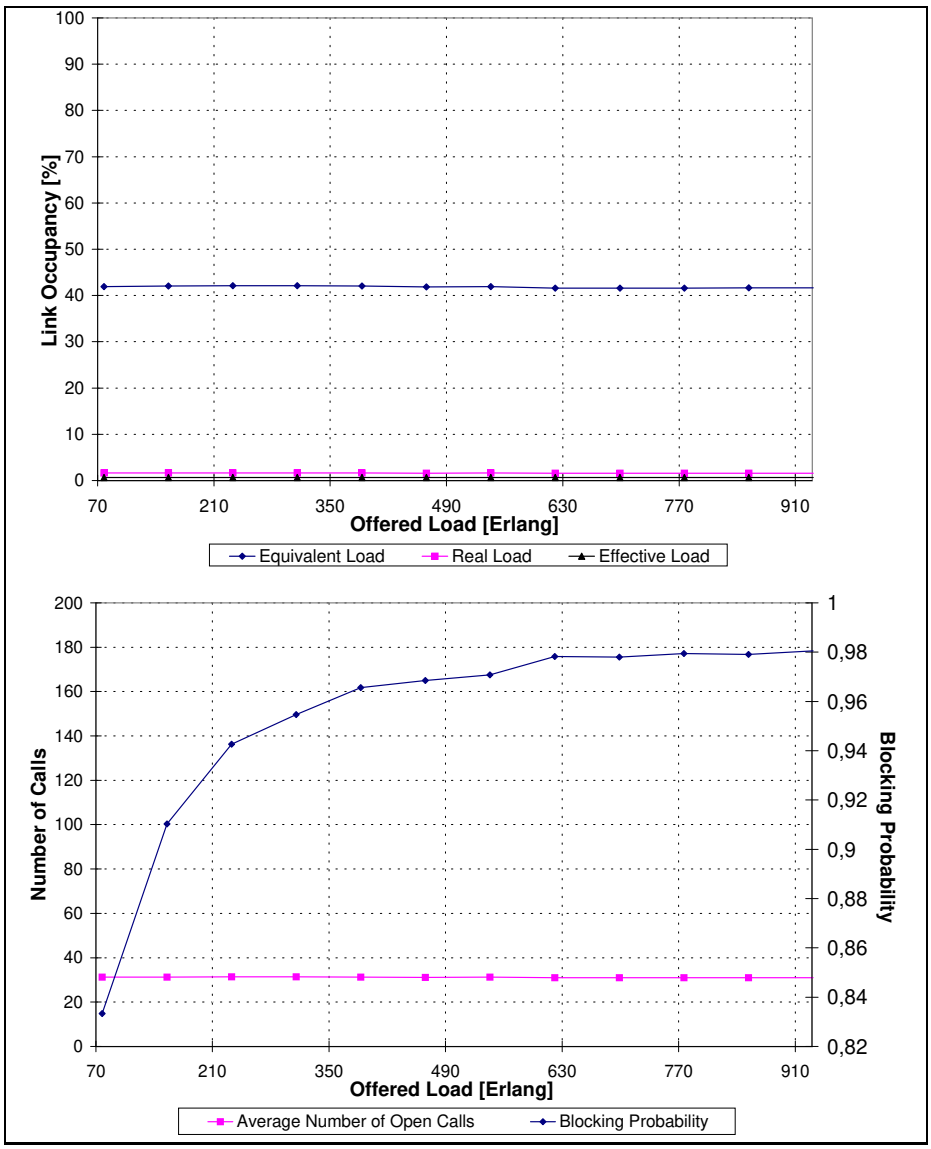

Figure 4.8: Access links at 34 Mb/s, backbone links at 155 Mb/s, packetization delay of 8 ms, and delay bound of 10 ms.

delay which varies from 4 ms to 20 ms. The end-to-end delay requirement is set to 50 ms to simulate calls spanning about 1,000 Km, in addition to length of the access network represented by the simulated topology.

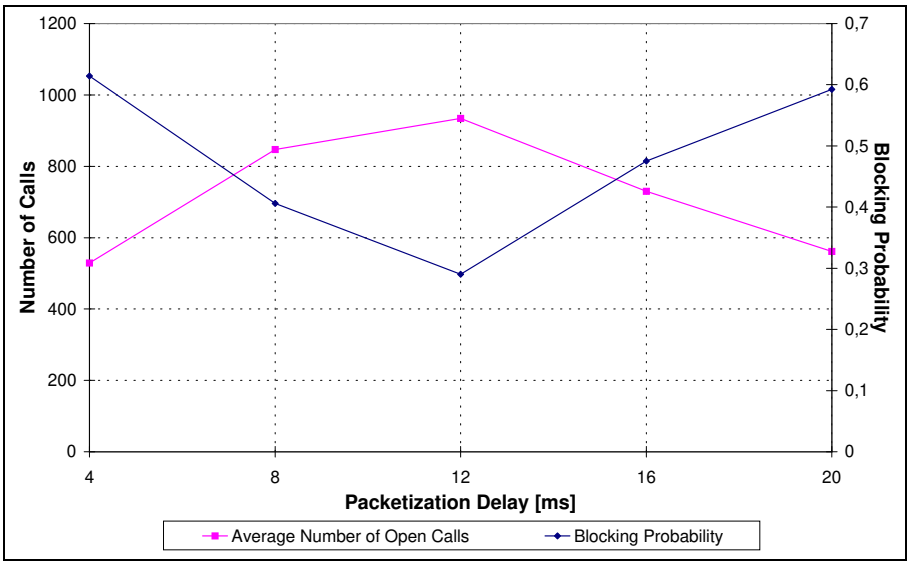

Figure 4.9: Access links at 34 Mb/s, backbone links at 155 Mb/s, 310 Erlang of offered load, and delay bound of 50 ms.

Figure 4.9 shows that the number of calls accepted by the network is maximum for a packetization delay of 12 ms, i.e., with a payload of 284 bits which yields a 50% packetization overhead. Lower packetization delays leave a larger delay budget for queuing, but waste more bandwidth due to the packetization overhead. On the other side, higher packetization delays introduce lower packetization overhead (i.e., each call introduces a lower real load), but the delay budget for queuing is lower and thus the CAC reserves more bandwidth for each call (i.e., the equivalent load of each call grows higher).

The curve in Figure 4.9 is roughly symmetric with respect to the optimum packet size. Nevertheless, there is a significant difference in operating the network in the conditions of the right part of Figure 4.9 or of the left part. In the first case, the small number of accepted calls is due to the large extra bandwidth reserved to each call, i.e. the difference between the equivalent load and the real load, which can be used for transmission of best-effort traffic. In the second case, the small number of accepted calls is due to the high packetization overhead, i.e., to the high real load, and the capacity used

by the headers is wasted.

4.2.4 Heterogeneous Sources

The topology depicted in Figure 4.10 has a larger number of sources and destinations, 34 Mb/s access links, and 155 Mb/s backbone links. Given the number of sources and the capacity of access links, backbone links can be loaded up to 88%.

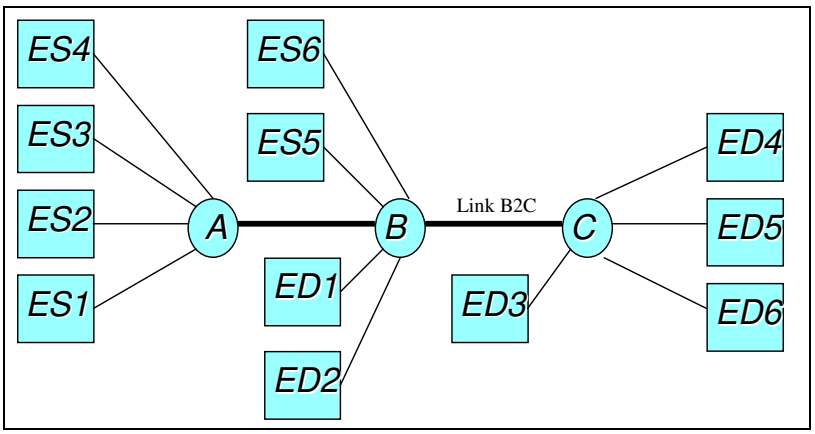

Figure 4.10: Network topology used in the simulations.

A mix of ADPCM32 (which generate a 32 Kb/s flow) and PCM (which generate a 64 Kb/s flow) sources is simulated over the network. The end-to-end delay bound is fixed to 50 ms and the packetization delay is varied from 4 ms to 20 ms in order to identify the packetization delay which maximizes the number of phone calls accepted by the network. Having fixed a packetization delay in each simulation run, sources generate packets with different dimension according to their encoder, i.e. the packetization overhead is not constant over the sources.

Figure 4.11 shows the simulation results for different levels of offered call load. At low call loads (155 and 310 Erlang) the effective load is constant when varying the packetization delay because all the calls are accepted. When the load increases, the effective load has a maximum for mid range values of packetization delay.

Real load and equivalent load are the same for small values of packetization delay. This shows that as far as the packetization delay is low, it is not

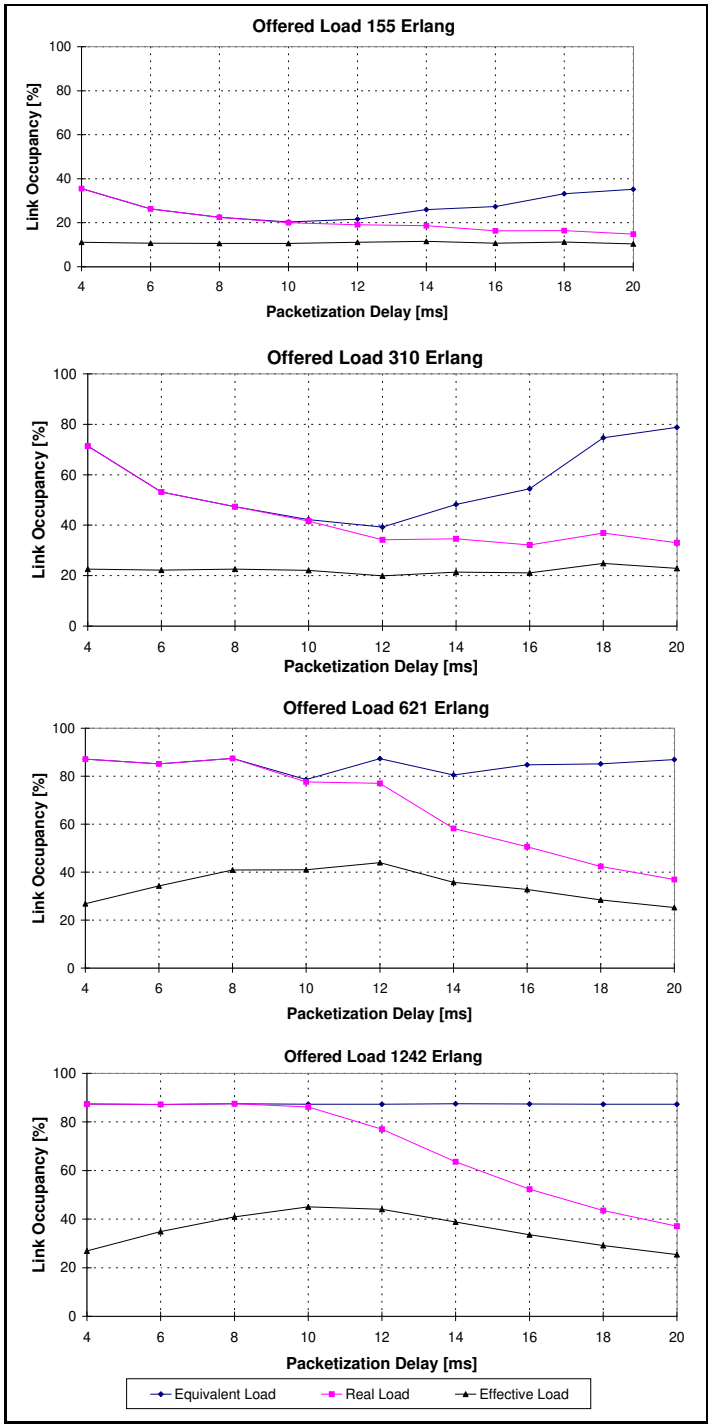

Figure 4.11: Link load for access links at 34 Mb/s, backbone links at 155 Mb/s, and delay bound of 50 ms.

necessary to overallocate bandwidth to guarantee the delay bound. However, as the packetization delay increases, extra bandwidth must be reserved to each call in order to reduce the contribution of queuing delay. This is shown by the equivalent load increasing in the equivalent bandwidth in the plots at low call load (155 and 310 Erlang) and by the real and effective load increasing at high call loads (621 and 1242 Erlang). In fact, at low call load, all the calls are anyway accepted and a large packetization delay decreases the amount of bandwidth still available to be reserved to other real-time traffic. In the mean time, the real load is low since the percentage packetization overhead is low and a good deal of capacity is available to best-effort traffic. Instead, at high call loads all the capacity is reserved on the access links (as shown by the equivalent bandwidth being 88% of the link capacity on the link from `B` to `C`); as the packetization delay increases, the maximum number of calls accepted on the link decreases, as well as the effective and real loads.

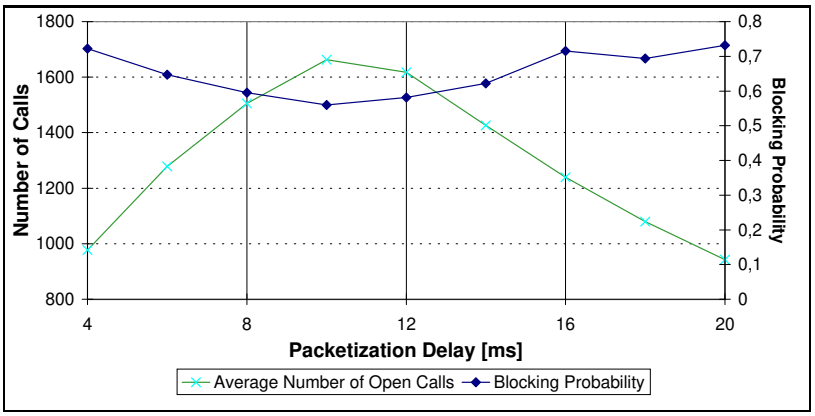

Figure 4.12: Access links at 34 Mb/s, backbone links at 155 Mb/s, delay bound of 50 ms, and offered call load of 1242 Erlang.

Figure 4.12 clearly shows the impact of packetization delay on link utilization. The number of accepted calls and the blocking probability are plotted versus the packetization delay. The number of accepted calls is maximum for a 10 ms packetization delay which corresponds to acceptable packetization overhead while leaving enough delay budget to queuing.

4.3 Discussion

Packet scheduling algorithms enable asynchronous packet switching networks to provide guarantees on the quality of its services. The application of packet scheduling techniques into nodes is combined with the exploitation of call admission control that keeps track of the number of real-time sessions sharing a resource (e.g., the capacity of a link) in order to keep the utilization low enough to guarantee the required QoS.

This chapter extends the results of the static priority analysis reported in Chapter 3 to Packet by Packet Generalized Processor Sharing scheduling algorithms. As done in Chapter 3, the implementation of toll quality telephony on a packet switching network is considered as a case study. A call level simulator is implemented to study the efficiency in the utilization of the network assuming that the number of phone calls accepted on the network is limited by a call admission control mechanism which takes into account the packet scheduling algorithm deployed into nodes. Simulation enables a network with heterogeneous sources to be studied.

Simulations have showed that depending on the span of phone calls, network utilization can be very low for two independent reasons. When queuing delay must be short (because, for example, the propagation delay is long), the equivalent load (i.e., the capacity reserved to the call in order to meet its delay requirement) is high. In this case the capacity given by the difference between the real load (i.e., the capacity required by each call including the packetization overhead) and the equivalent load can be used for the transmission of best-effort traffic.

Alternatively, the utilization can be low because the packetization overhead is large and thus the real load imposed by each call is much higher that its effective load (i.e., the raw bandwidth of the voice encoder). In this case the capacity given by the difference between real load and effective load is wasted in protocol overhead and cannot be used to carry any other traffic.

At a first glance, it seems reasonable to keep the protocol overhead as low as possible in order to reduce the amount of wasted capacity. Nevertheless, this would lead to long packetization delays due to the time spent to gather enough samples to fill up the packet payload. Since packetization delay consumes part of the overall end-to-end delay budget, a shorter delay must be introduced by queuing, i.e., the equivalent load of each phone call is

larger. Even though the extra bandwidth required for phone calls (i.e., the difference between equivalent load and real load) is not wasted and can be used by best-effort traffic, a large equivalent load keeps low the maximum number of phone calls accepted on the network.

Thus, network utilization, as expressed by the average number of open calls and the call blocking probability, has been studied for various packetization delays in order to find out how to maximize it. It seems not possible to devise a general rule since the optimal packetization delay depends on header dimension and even span of the call. The packetization overhead affects the real load, while the call span determines the propagation delay and thus the time packets are allowed to spend in queues.

Further study is needed on the impact of the exploited voice encoding scheme. In fact, low bit rate encoding implies low effective load, but also long packetization delay in order to keep the relative packetization overhead acceptable; this leads to high equivalent load.

Chapter 5

Adaptive Group Multicast with Time-Driven Priority

Group or many-to-many real-time multicast is a demanding service. It is usually exploited by applications which require multiparty communications with guaranteed (sometimes large) bandwidth, small loss, and bounded end-to-end delay and jitter[1]. Given the finite network resources, real-time multicast should be performed in an *efficient* and *adaptive* manner. Efficient means that the communication links can operate near full utilization.

The number and the identity of transmitting nodes in a real-time multicast group changes over time. This has two implications: (i) the resources should be allocated to the group and not individually to each participant, and (ii) the active nodes (i.e., the source nodes) should *adapt* their transmission rate for dynamic sharing of the bandwidth allocated to their multicast group.

The solution presented in this paper exploits a combination of *space* (virtual ring embedding) and *time* (packet forwarding driven by time). Using virtual rings for many-to-many communications is simple and "natural", since a message sent from one member of the group travels around the ring until it comes back to the sender. We simply exploit this property by embedding multiple virtual rings, one for each multicast group. In a previous work it has been shown that the virtual ring can be instrumental in designing a *reliable multicast protocol* from *bursty sources* [44]. In this work the ring circular property is used to adapt, in real-time, the share of capacity each *active*

[1]The jitter is defined as the difference between maximum and minimum delay.

(transmitting) *node* obtains. It is done in a fair manner according to the number of nodes that are trying to transmit to the group at any given time.

Global timing is used to control the forwarding of packets inside the network. This time-driven forwarding is used for:
(i) Minimizing the *queuing delay* bound in the virtual ring in order to provide a guaranteed quality of service (QoS), as we discuss in Section 5.2.
(ii) Sharing the capacity of a link among multiple virtual rings traversing the link in the same direction, as described in Section 5.3.
(iii) Adaptive sharing of bandwidth to ensure that each active node in a virtual ring gets a fair share of the capacity, as discussed in Section 5.4.

Section 5.5 introduces a method for *scalable growth* of the multicast group by adding multiple subtrees to the virtual ring. Adding listeners (receiving only) nodes on subtrees is simple. However, adding active nodes that transmit from the subtree to the multicast group is more complex and may not be adaptive. This complexity helps understanding some of the virtues of using virtual rings for adaptive real-time multicast.

5.1 Related Work

Real-time group multicast with allocation of resources to the group was addressed in depth by the Tenet Group in the framework of the Tenet Project [16]. Multicast communications are implemented through a set of independent trees, each one rooted at a sender [26]. QoS is guaranteed by giving priority to real-time traffic over best-effort traffic, by performing rate control and scheduling in network nodes, and by restricting access to network resources through call admission control. When resources are allocated on a link which belongs to multiple trees used by a multicast group, a shared allocation is performed [27]. Network nodes, based on the traffic characterization provided by each source, allocate enough shared resource to guarantee the required QoS to each flow. When resource sharing is used, rate control and scheduling can be performed per group (instead of per single flow), which is an important scalable property. Applications are not supposed to adapt to the actual resources reserved to the multicast group; rather, at setup each sender hands to the network its request and resources are reserved according to the worst case utilization.

Some other real-time transport approaches are based on asynchronous packet forwarding (dealt with in Section 2.1). Some schemes, such as Packet by packet Generalized Processor Sharing (PGPS) [46, 47] and Weighted Fair Queuing (WFQ) [11] bound the packet delay by using scheduling algorithms that approximate the fluid flow service model. Others, like Jitter-EDD [61] and FIFO+ [9], perform scheduling in each node based on the time packets spent in other nodes. All these schemes guarantee a bound on the queuing delay which is inversely proportional to the connection (or session) rate, proportional to the number of nodes traversed by that connection, and proportional to the packet size.

A similar approach to real-time group multicast is proposed by the Integrated Services Working Group of the Internet Engineering Task Force (IETF) in the framework of the Integrated Services Internet [55]. Multicast communications are implemented through both independent and shared trees. The Integrated Services model encompasses two QoS control models: the *Controlled-Load* [65] service and the *Guaranteed QoS* [54]. The first service model is not relevant to this work, while the second aims at approximating the fluid flow service model in each node by exploiting call admission control, policing at the edge of the network, scheduling of reserved resources, and reshaping within nodes. Applications require the needed QoS using a setup mechanism, e.g., the Resource reSerVation Protocol (RSVP) [66]. According to the Guaranteed QoS control model, the network does not have to provide the requested QoS; rather, it provides an estimate of the QoS it will be able to guarantee and applications must operate with it. Since reservations are soft state, applications must renew their requests periodically and may have to adapt to varying QoS guarantees. Reservation on links used by multiple senders of the same multicast group can be made either individually for each source, or globally. In the latter case, no mechanism is foreseen to coordinate senders and avoid that the instantaneous overall demand exceeds the allocated shared amount. Mechanisms are being devised [53] to allow applications to infer the service actually provided by the network in order to react to its variations [60]. This approach takes into consideration application adaptiveness, but can result in uncontrolled quality.

5.2 Principles of Operation

5.2.1 Network Model

In this work we consider a general topology network with bidirectional links. A virtual ring (VR) is associated with each group multicast by constructing cycles that includes each participant at least once [44]. For example, Figure 5.1 shows two VRs (continuous and dashed lines) embedded onto an arbitrary topology network (bold lines). In this example, the two VRs share link (i,j) that will carry packets for both the VRs.

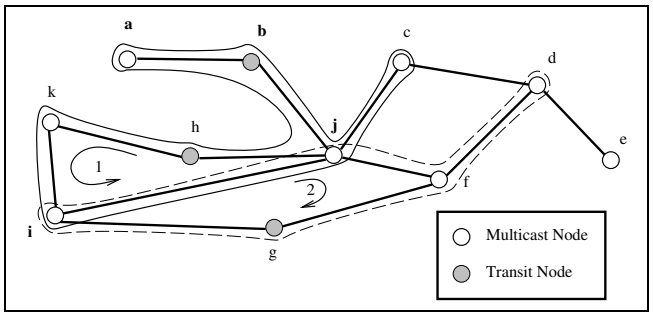

Figure 5.1: Virtual Rings (VRs) Embedded in the Network.

A virtual ring has three types of nodes: (i) active participants (ii) passive participants (e.g., listeners in a video conference), and (iii) the transit nodes that are not the members of the multicast group but they are needed to construct the VR (shaded circles in Figure 5.1).

Although, some nodes are traversed more than once by the packets pertain to the same multicast group (e.g., f and j in Figure 5.1), these packets will be received only the first time they reach these nodes.

5.2.2 Packet Forwarding

Nodes forward packets according to the *pseudo-isochronous* or *RISC-like* forwarding principle (see Section 2.2) proposed in [42, 35, 36]. When a packet is forwarded along the VR, after a fixed number of TFs (say n_r) it is received back by its source node. Note that the integer n_r gives the *length* of the VR in terms of time frames.

5.2.3 Bandwidth Allocation

Some bandwidth is allocated to each multicast group and is shared by the members of the group that are ready to transmit. Section 5.3 shows how bandwidth allocation is performed while Section 5.4 shows how bandwidth is dynamically shared among the active nodes of the group.

We assume that a node in the network can forward from its output port up to $125(\mu s) \cdot B(Mb/s)$ bits during each TF. For example, if the data rate (B) of a link is 100 Mb/s then 12.5 Kb can be transmitted in every TF. The source of the bits can be from this node, from its multicast group, or any other multicast group sharing this link. Note that $125(\mu s) \cdot B(Mb/s)$ is also the output buffer size requirement on every port. It is easy to see that this establishes a *small bound* on the queuing delay of every node along the virtual ring. If the many-to-many group multicast would be done over a tree, the propagation delay would be shorter but the queuing delay bound would be much larger. This is significant, since the delay bound is a critical parameter for real time applications.

The bandwidth is allocated on a link for a VR (i.e., multicast group) by reserving time intervals in some TFs in a periodic manner. (Unused reserved bandwidth can be used for best-effort traffic). For example, allocating a 64 Kb/s for a multicast group means the transmission of 1 byte[2] (i.e., $0.064 \cdot 125 = 8$ bits) during each TF (in each virtual node). A packet size of one byte is obviously not practical. Thus, we have to use a packet size that keeps low the percentage overhead of the packet headers. For example for video-conferencing over IP networks, a 500 bytes payload is a reasonable size in order to keep the overhead due to IP, UDP, and RTP headers under 10%. Therefore, in our example, the 64 Kb/s can be allocated to a multicast group by reserving 500 bytes every 500 TFs.

TFs are grouped into *Time Cycles* (TCs) of fixed length k and are enumerated from 0 to $k-1$, as shown in Figure 5.2. Allocation of bandwidth to a multicast group is done by reserving a fraction of the bits that a node in the VR can send during some of the k TFs in the TC.

[2]In the following examples when dealing with bandwidth we consider the gross bandwidth allocated to a multicast group, not the net bandwidth offered to applications. I.e., the overheads due to protocol encapsulation are not taken into account. This is not a limitation because these overheads can be taken into account by simply having the gross allocated bandwidth larger.

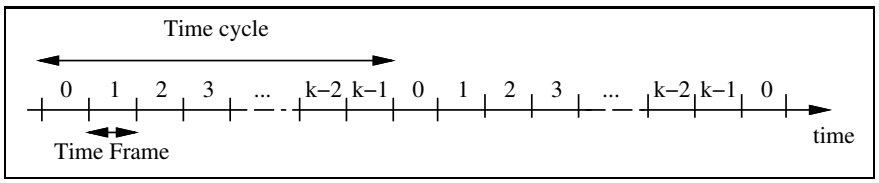

Figure 5.2: Timing Structure and Time Cycle.

Figure 5.3 shows a sample reservation of one packet per TC for a VR. In the figure a rectangle represents the bits reserved during the TF on the virtual ring nodes (i.e., A, B, C and D). Each node on the ring sends or forwards packets of its multicast group during the TFs in which transmission time for packets have been reserved, i.e., it uses always the same TFs in each TC for transmitting packets pertaining to its multicast group.

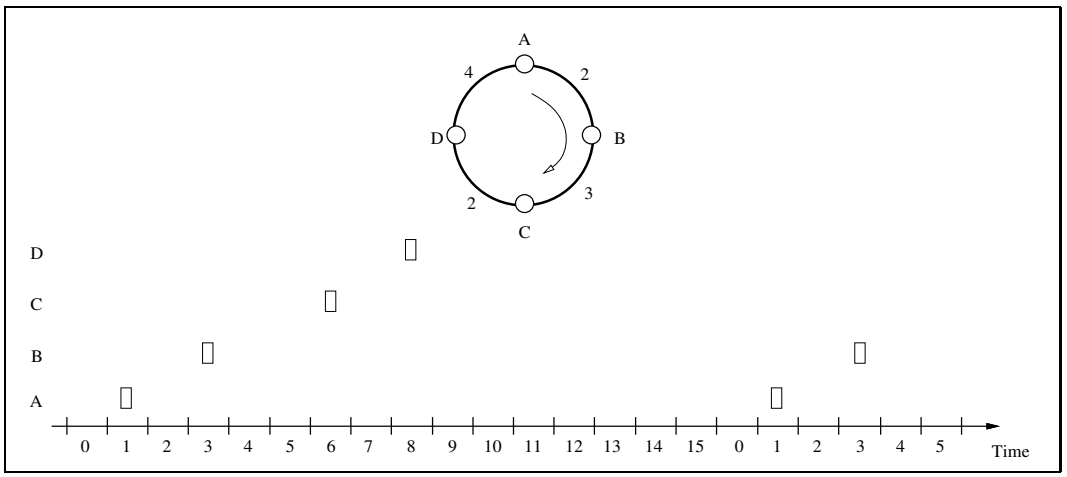

Figure 5.3: Sample Bandwidth Allocation for a Multicast Group.

Assume for simplicity that packets are fixed size of s bytes. If we reserve a packet worth of bytes in p_r TFs for a multicast group r, then the bandwidth allocated to the multicast group is given by

$$MB_r = \frac{p_r \cdot s \cdot 8}{k \cdot 125} \text{ Mb/s}. \tag{5.1}$$

Bandwidth is allocated in multiples of a base quantity which is given by the reservation of 1 packet per TC. In this work we assume $s = 500$ bytes and, as we want the base bandwidth allocation unit to be 64 Kb/s, we fix the number of TFs per TC (length of the TC) to

$$k = \frac{500 \cdot 8}{64 \cdot 10^3 \cdot 125 \cdot 10^{-6}} = 500.$$

Without loss of generality, we assume that only one packet is sent in each TF. As a consequence, bandwidth is allocated by giving to each virtual node on the ring exclusive usage of outgoing links during the whole reserved TFs. Such TFs are said to have been reserved on the node for a specific multicast group or VR.

5.2.4 Forwarding on a Virtual Ring

As an example, consider now the packet forwarding in the sample VR topology with the bandwidth allocation shown in Figure 5.3. If node B has a packet to transmit to the multicast group, it waits until TF 3 and then sends the packet which reaches the output port of C in TF 6, as shown in Figure 5.4. The packet travels on the VR and gets back to node B by TF 3 of the following TC. If node B at this point does not transmit any packet, C will be able to transmit a packet in TF 6 if it is not used for forwarding a packet from some other node.

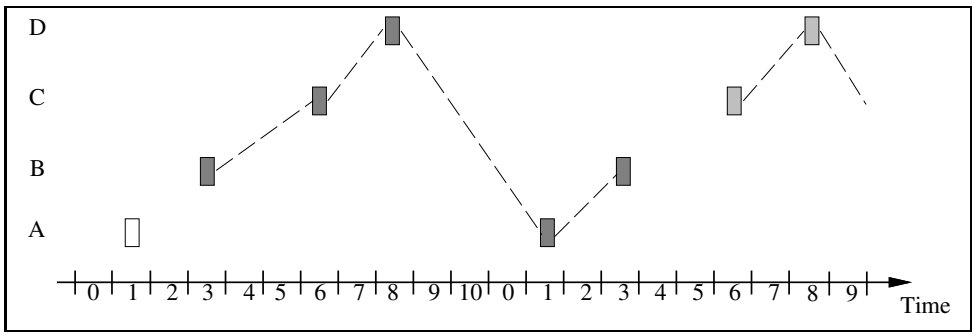

Figure 5.4: Forwarding of a Packet on a Virtual Ring.

Figure 5.5 shows the forwarding of packets on the same VR when $k = 16$. Virtual node A receives B's packet by TF 12, but it has no TF reserved until TF 1 of the following TC. Thus, node A buffers B's packet until the next TF reserved for its VR. Node A is named the *buffering node* of this VR. Each VR must have a buffering node in order to *adapt the length of the VR (virtual ring) to the length of the TC (time cycle).* (Note that due to the presence of a buffering node a node does not necessarily receive its own packet at the same TF, in the next TC, in which it has transmitted it.)

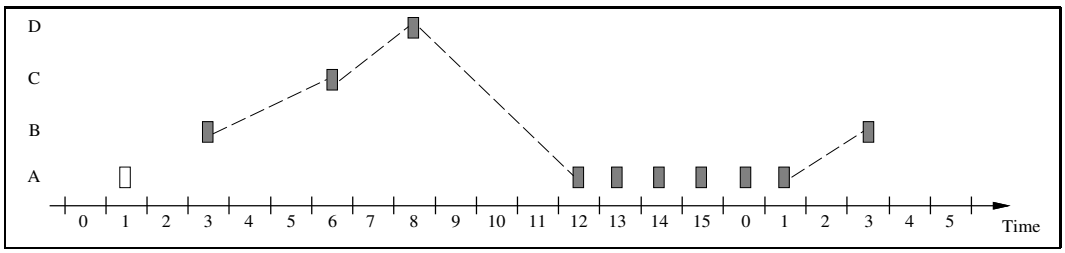

Figure 5.5: Forwarding of a Packet Originated from B on a Virtual Ring with Buffering Node A.

5.2.5 Relative Time

When the timing structure is applied to a ring, the concept of *Relative IDentifier* (RID) of TFs is useful for two purposes:

1. To handle the reservation of TFs in the nodes (Section 5.3) and

2. To dynamically share the allocated TFs among the participants to the multicast group (Section 5.4).

Each node j keeps its *relative time* by assigning a RID to TF i through the expression:

$$r_i(j) = (i - d_{b,j} + k) \bmod k, \tag{5.2}$$

where $r_i(j)$ is the RID given to TF i by node j and $d_{b,j}$ is the delay, expressed in number of TFs, experienced by a packet traveling from node b chosen as the *base* for the relative time, to node j. The base node has the property that

$$r_i(b) = i, \forall i \; 0 \leq i \leq k - 1$$

i.e., the relative time in the base node equals the absolute time.

Figure 5.6 shows the relative time for the nodes of the VR shown in Figure 5.3 when node A is the base node. By comparing the two figures, it is easy to note that the TF reserved to the VR has RID 1 in each node. In general, the relative time has the useful property that each node of a VR identifies the TFs reserved to the multicast group through the same RID.

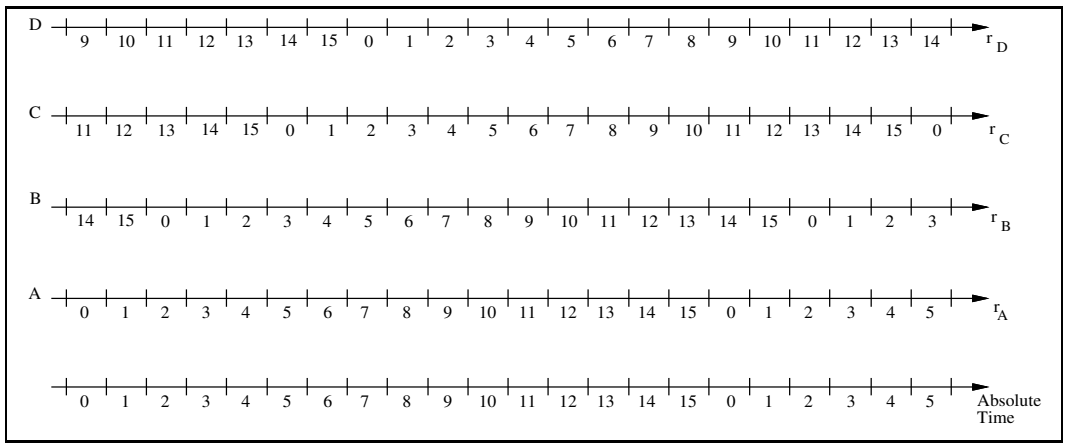

Figure 5.6: Relative Time

5.3 Scheduling

In this section we consider how to allocate resources to a new multicast group M_r with bandwidth request MB_r. As explained in section 5.2.3, the bandwidth request is mapped to a number (p_r) of TFs needed for the new multicast group. Let VR_r be the VR associated with this multicast group and let E be the set of links included in this VR. Since the links of a synchronized network are shared by various types of traffic, some of the TFs (say f) on a link e in E may already be reserved.

It is necessary (*not sufficient*) to reserve p_r unused (available) TFs on each link in E, in order to guarantee MB_r bandwidth for this multicast group. Having p_r available TFs on each link is not sufficient because a TF (on the same link) may need to be allocated to two or more different VRs.

In other words, it is possible that there is available bandwidth (i.e., time frames) for a new multicast group (or to increase the allocation of an existing group), but there is no available *schedule* to assign time frames (of fraction of time frames) such that a time interval in a time frame is *not* allocated to more than one virtual ring, see an example in Figure 5.7. This problem is like the Pigeon Hole Problem where there are enough available TFs on each link, but there is no schedule that can fit them in the right linear (sequential) order, on each of the VRs. The impact of such scheduling conflict is that some of the link bandwidth cannot be allocated to the multicast groups (however, note that this bandwidth can still be used by non-reserved traffic).

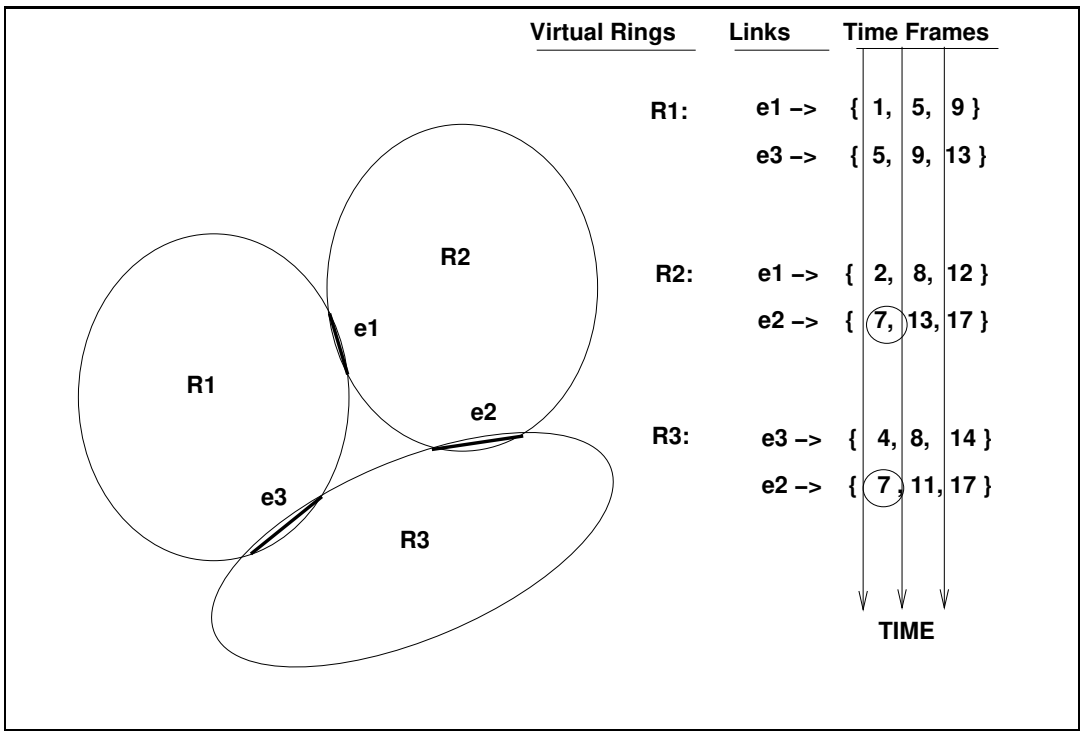

Figure 5.7: Inability to reserve bandwidth to virtual ring due to scheduling conflicts of time frames

The scheduling problem is examined in a hierarchical way. First, we consider a single link used by the multicast group and define the necessary condition for *feasibility* of a link. Next, we consider the whole VR and define *feasible ring scheduling.*

A link of a VR is *feasible* if it has at least p_r available TFs such that p_r satisfies the equation (5.1). Note that if there is even one link in E with less available TFs than p_r (i.e., the link is highly loaded), then this multicast group cannot be allocated the required bandwidth. Next, we introduce the following definitions to be used in this section:

Definition 1 –Feasible Schedule: *Given a virtual ring VR_r, a schedule σ is called feasible schedule for VR_r if σ chooses exactly p_r (available) TFs on each link such that each TF has the same RID at each node.*

Definition 2 –Feasible Ring: *A virtual ring VR_r is called feasible if there exists a* feasible schedule *σ for VR_r.*

Definition 3 –Feasible Network: *Given a network and a set of multicast groups on it, the network is called feasible, if there exists a feasible scheduling for each one of the VRs.*

Proposition 1 *In a feasible network, no TF on a link is used by two different rings.*

5.3.1 Feasible Ring-Scheduling Algorithm

In this section we present a distributed algorithm to find a feasible assignment of TFs to a VR. The algorithm is initiated by a predetermined origin node (e.g., the base node) and executed by each node in a VR. In Figure 5.9 the steps of the algorithm are shown.

In the algorithm a node first assigns a RID to each TF by using equation (5.2). Next, a bit-vector BV_i of size k is constructed at node i. In BV_i a bit is set to 1 (see Figure 5.8.a), if the corresponding relative TF is not available (i.e., it has been previously reserved for another VR or a point-to-point connection).

The origin node on the VR *sends* its bit-vector to its downstream neighbor. Upon receiving a bit-vector, a node performs a logical AND operation between the received and local vector (see Figure 5.8.b). The result is stored as the updated local bit-vector, and sent to the downstream node (see Figure 5.8.c).

When the origin receives a bit-vector from its neighbor (step 1), the first phase of the algorithm is completed. Note that at this point each node knows what all the upstream nodes (from the origin to its upstream neighbor) know about available TFs. Thus, when the origin receives the bit-vector from its neighbor, it learns the ring-wise availability of TFs (global information).

If the number of zeros in the received bit vector BV is less than p_r, the origin concludes (step 1.2.1) that the VR is *not feasible*. If BV contains at least p_r zeroes, then any choice of p_r TFs will result in a *feasible scheduling*. The origin calls the procedure $MinMax_Link_Schedule(VR,BV,p_r)$ (see algorithm in Figure 5.11), to choose p_r TFs for optimal scheduling (step 1.2.2). The bit-vector returned by this procedure indicates the TFs reserved for this multicast group. This bit-vector becomes the bit-vector of the reservation

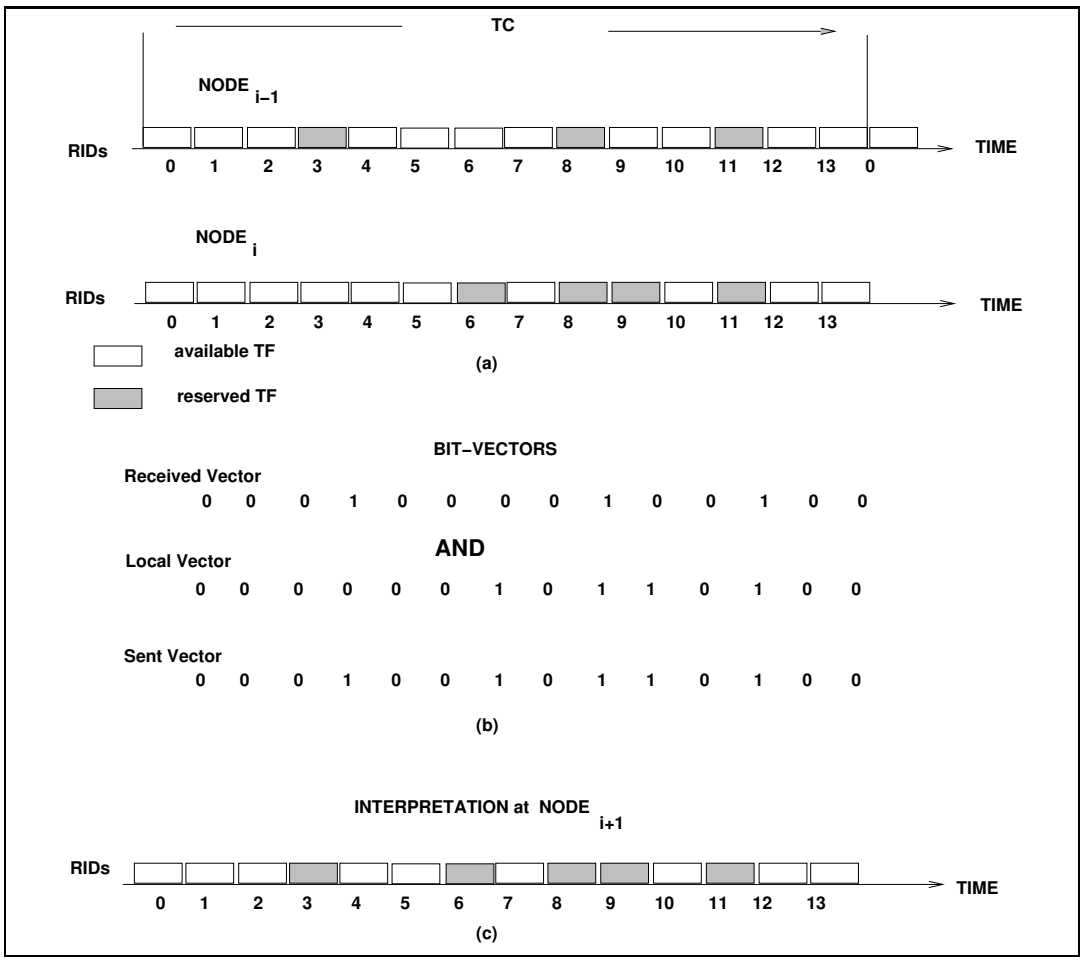

Figure 5.8: Example: Processing and Interpretation of Bit-Vectors to Determine a Feasible Schedule

for this ring (RBV). The origin sends RBV to its downstream node and it is propagated on the ring to inform all the nodes. When the origin receives RBV (step 1.1), the algorithm terminates and each node on the ring has the same RID for the newly reserved TFs.

5.3.2 Optimal Selection of TFs

In this section we consider the problem of optimally choosing p_r TFs for a VR. First we introduce an objective function then an algorithm to construct a solution.

Algorithm: *Feasible_Ring_Schedule* performed at node i
Input: VR_r; origin node o_r; p_r
Preprocessing: for each TF on this link
compute its RID (for example, o_r can be used as base)
construct BV_i to indicate the status of TFs
Begin
RECEIVE a bit-vector BV_{i-1}
1 **if** node $i = o_r$ then
1.1 **if** $BV_{i-1} == RBV$ then STOP
1.2 **else**
1.2.1 **if** number of 0's in RBV $< p_r$ then
print INFEASIBLE
1.2.2 **else**
RBV = $MinMax_Link_Schedule(VR,RBV,p_r)$
SEND RBV to downstream node $i+1$
2 **else**
2.1 **if** $BV_{i-1} == RBV$ then $BV_i = RBV$
2.2 **else** $BV_i = BV_{i-1}$ AND BV_i
2.3 SEND BV_i to downstream node $i+1$
End

Figure 5.9: Ring Scheduling Algorithm to Ensure Ring Feasibility.

Objective Function to Minimize Jitter

Delay jitter determines buffer size at nodes and performance of real-time protocols. The larger the jitter the larger the buffers needed to compensate for it. A uniform distribution of TFs in the TC is beneficial in reducing the end-to-end delay, its variation (jitter), and the size of buffers in the buffering node. Ideally, if reserved TFs are uniformly distributed, the buffering node receives the packets at a (almost) regular pace and forwards them at almost the same regular pace. As a consequence

1. packets are buffered shortly in the buffering node. Thus, the buffer must not be very large and the delay experienced by packets is small;

2. subsequent packets experience similar delays thus yielding small jitter.

Thus, one objective function for optimal schedule is to ensure equal *time distance* among TFs allocated in a TC. Let FS denote the set of all feasible schedules; we consider as optimal schedule σ the one which satisfies

$$\min_{\sigma \in FS} \left\{ \max_{1 \le i \le p_r} \left| [(t(\sigma)_{i \mod p_r+1} - t(\sigma)_i + k) \bmod k] - \frac{k}{f} \right| \right\} \tag{5.3}$$

where $t(\sigma)_i$ denotes the i^{th} TF reserved for the multicast group according to the schedule σ.

Note that if all the TFs were available, the best spacing (timewise) between the p_r TFs would be: $\lfloor \frac{k}{f} \rfloor$. Thus, equation (5.3) compares the maximum difference between the spacing imposed by each feasible schedule to the best possible spacing and chooses the schedule which minimizes this difference.

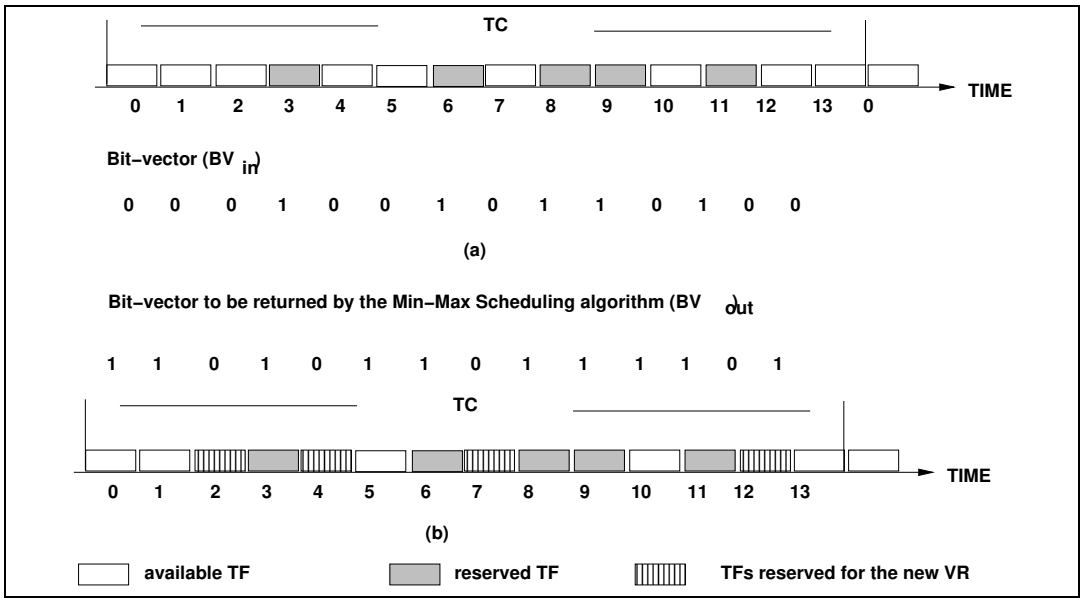

Figure 5.10: Example: Time Cycle with 9 Available TFs.

Complexity of Link Scheduling

The size of the feasible scheduling set FS is determined by two parameters: p_r and $k-f$: enumeration can be obtained by $C(k-f,p_r)$ (i.e., number of ways to pick a subset of p_r out of $k-f$ distinct objects). Since the enumeration involves factorial operation, if p_r and $k-f$ are close to each other, it is possible to choose the best allocation by enumeration. However, for large factorial computations FS can be too large to enumerate; we propose a min-max algorithm to solve this problem with acceptable complexity.

Min-Max Algorithm

The input of the algorithm is a bit vector BV_{in} (Figure 5.10.a) which represents at least $k - f \geq p_r$ available TFs for a VR. The algorithm returns a bit vector BV_{out} (Figure 5.10.b) that indicates which TFs should be reserved for the multicast group.

The algorithm shown in Figure 5.11 builds a Line-Graph (LG) from the information carried by BV_{in}. Define a *median* of a LG as a node to which maximum *distance* from any other node in the graph is minimum. The *distance* between two nodes is defined as the sum of the link *costs* on the path between them as explained in Figure 5.11.

The algorithm takes p_r steps and at each step finds a median of an updated (reduced) graph as shown in Figure 5.12. Each median node corresponds to a TF selected by the algorithm for optimal scheduling. The location of median nodes in BV_{out} is identified by zero while all the other bits are set to 1. Each time a median node is chosen the LG is divided into two connected components (called segments). A new median is chosen from the segment with maximum total cost computed during the selection of the previous median node (see Figure 5.12). The selected segment is called the *active segment*; initially the entire LG is considered a single active segment.

Objective Function for Load Balancing

In this section we relax the assumption that a TF is used to transmit a single packet. In this case different amounts of bits can be reserved during TFs. Balancing the load over the TFs allows a larger number of VRs to be allocated on the network since it decreases the blocking probability. Thus, we introduce a new objective function which aims at balancing load when scheduling TFs for a VR. This objective function minimizes the differences among the amounts of non reserved bits in each of the k TFs of a TC. The optimal schedule is the one which satisfies

$$\min_{\sigma \in FS} \left\{ \sum_{i=1}^{p(\sigma)_r} \left(\beta(\sigma)_i - \overline{\beta(\sigma)} \right)^2 \right\} \tag{5.4}$$

where $\beta(\sigma)_i$ is the number of non reserved bits during the i^{th} TF, *after* the schedule σ has been applied. The variable $\overline{\beta(\sigma)}$ is the average amount of non

Algorithm: *MinMax_Link_Schedule*
Input: VR_r; BV_{in}; p_r
Output: BV_{out} with exactly p_r zeros to indicate
which TFs will be used by VR_r
Preprocessing: construct a weighted line-graph such that
nodes represent available TFs
a link connects two consequent TFs
the weight (cost) on a link is the difference between
RIDs of the end point TFs
mark the line-graph as "active" segment
Begin
1 Set all the bits to 1 in BV_{out}
Begin FOR $i \leq p_r$ DO
2 Choose the median of the active segment and reset the bit
in BV_{out} identified
by the median's index
3 Compute the total cost of each "active" segment
4 Store the costs and end points of the segments in list
5 Choose the segment with maximum total cost from this
list and make this segment "active"
End
6 **Return**(BV_{out})
End

Figure 5.11: A TF Choice Algorithm to Minimize Jitter in a VR

reserved bits over the whole TC. The two objective functions (5.3) and (5.4) can be in conflict and thus a tradeoff between them must be found.

5.4 Adaptive Bandwidth Sharing

The adaptive bandwidth sharing concerns a single VR and does not affect any other VR embedded in the network. Active stations actually share the allocated bandwidth by sending their own packets during some of these TFs. Both passive stations and transit nodes just forward packets during the reserved TFs. The set of active stations changes dynamically as, for example, in a video-conference call the set of active speakers changes. The stations participating in a multicast group run a signaling protocol that allows each of them to know the actual set of active stations[3].

[3]This signaling protocol is out of the scope of this paper; some possible candidate protocols are available for the purpose.

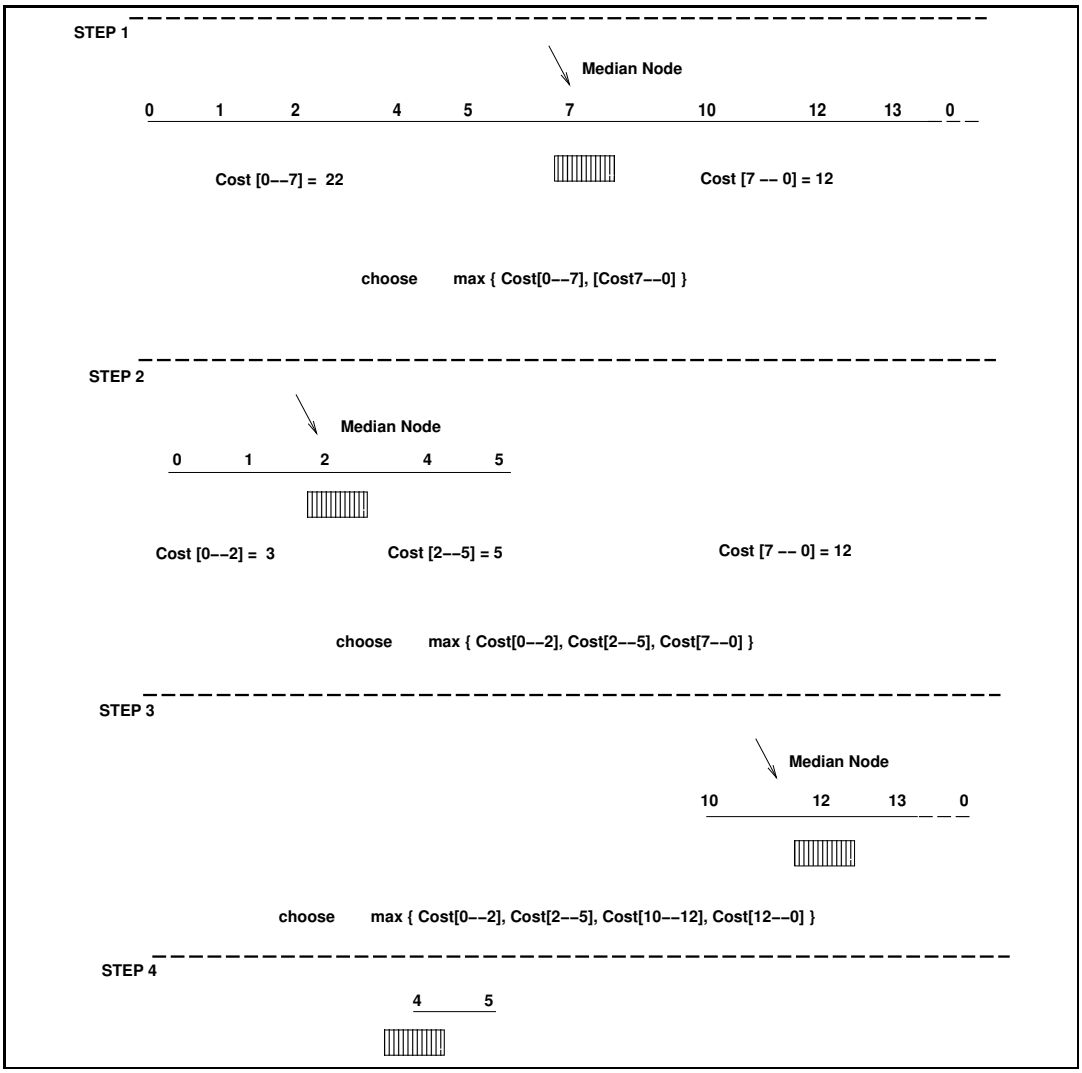

Figure 5.12: Example: Execution of the Algorithm to Choose 4 TFs.

As the number a_r of active stations changes, the share of bandwidth available to each station changes and they have to (1) adapt their transmission rate to it (Section 5.4.1) and (2) agree on which TF each station is using (Section 5.4.2).

5.4.1 Rate Adaptation

Through one of the algorithms described in Section 5.4.2, each active station knows the amount of bandwidth it can use for transmission. When the bandwidth available to an active station changes, the application is notified in order to change the rate at which data is produced.

Some applications require some time in order to make this change. For example, in video-conferencing with compression data is produced through a process starting with capturing video frames, going through a number of processing steps, and ending with quantization and run-length encoding of the video information. The compressed data is buffered and retrieved from the buffer at a constant bit rate. Changing the amount of data produced by this process implies tuning the parameters that regulate some of these processing steps. The earlier the involved processing step, the longer the time for having the change take effect at the network interface.

When a station becomes passive, it notifies other stations and immediately stops transmitting. Each active station realizes that its share of the bandwidth is larger and notifies the application to increase the data generation rate. Even though the increment of the bandwidth share is immediately available to the active nodes, it takes some time before the change in the application generation rate reaches the network interface; i.e., the transmission rate of active nodes does not increase instantly.

When a station becomes active, it notifies other stations. However, it does not start transmitting immediately. All the previously active stations immediately notify their application to lower the data generation rate. Since it takes some time before the actual transmission rate of all the active stations decreases, the new active station waits for its bandwidth share to be available before starting transmitting. We assume that newly active stations wait a fixed time (in the order of one second) before transmitting. This is needed in order to assure that no packet is dropped in either the network interface of active stations or inside the network, due to unavailability of communications

resources.

This waiting time is acceptable as the time scale in which stations change their activity status is usually long. For example, in a video-conferencing application a station becoming active implies that somebody is asking the right to speak. Thus, waiting should not be of concern since the new speaker will usually have to wait before getting the permission to speak.

5.4.2 Bandwidth Sharing

Given the set of the p_r TFs reserved to a multicast group r, the bandwidth sharing problem consists in having each active station using a subset of them to send its own packets[4]. The subset used by each active station should be built satisfying the following criteria:

1. each node gets a fair share of the bandwidth MB_r allocated to the multicast group;
2. the end-to-end jitter experienced by packets is minimum;
3. the unused bandwidth (i.e., the TFs during which no multicast data is transmitted) is minimized[5].

Round Robin Allocation

The allocated bandwidth can be shared fairly among the active stations by having each active station using each TF in a round robin fashion. Active stations transmit their packets in any *free* or unused TF. When a station receives a packet originated by itself, it must not use the next TF to send another packet of its own. This allows downstream stations to use the TF, in a round robin manner, for their own transmissions. The fraction of unused bandwidth due to this phenomenon is:

$$\frac{1}{a_r + 1}$$

This unused bandwidth is analogous to the unused bandwidth due to the token walk time in token ring.

[4]Whenever dealing with TFs in this section, we mean TFs reserved to the VR.

[5]Even though unused bandwidth can be used to transmit best-effort traffic, the VR should try and use as much as possible of the bandwidth allocated to it.

Fixed Allocation

A fixed allocation scheme can be used to reduce the amount of bandwidth unused due to the bandwidth sharing procedure. According to the fixed allocation scheme, active stations come to an agreement on which TFs each of them is using[6]. When a station receives back a packet sent by itself, it can use the next TF for sending its data. Figure 5.13 shows an example in which 4 TFs (RID 1, 4, 8, 12) have been reserved for the VR depicted in Figure 5.3 and nodes B and C are active. Each active node is using two TFs to transmit its data.

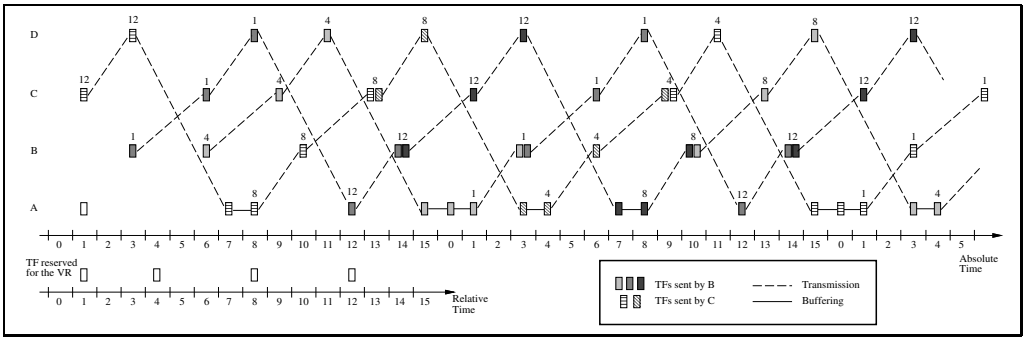

Figure 5.13: Fixed Allocation of TFs

Note that the RIDs of the two TFs each node is using change over time because the buffering node A buffers and forwards packets in a different TF. Moreover, some bandwidth is still unused because in each moment only 3 TFs out of the 4 reserved are used. This is due to the behavior of the buffering node, the distribution of the TFs in the TC, and the relationship between the lengths of VR and TC. This situation can be accepted because with this behavior the buffering node delays packets for a short time as shown in the example of Figure 5.13.

In the example shown in Figure 5.13 the phenomenon is particularly harmful as the unused TF is always one of the 2 assigned to station C which is getting only half of the fair bandwidth share.

This unfairness becomes smaller as the number of TFs assigned to each station becomes larger. This unfairness can be avoided by applying more complex bandwidth sharing schemes or buffering policies in the buffering node.

[6]This can be done through some signaling mechanism that is out of the scope of this paper.

Note that when dealing with unused bandwidth in the following discussion, the contribution given by the foregoing phenomenon is not taken into account.

The fixed allocation of TFs to active stations is possible only if $a_r \leq p_r$, otherwise a different scheme (e.g., round robin allocation) must be applied.

Nevertheless, one of the main fields of application for real-time group multicasting is large scale video-conferencing, where it is possible to have a large number of listeners (i.e., large p_r) and a small number of active speakers (i.e., small a_r). Only the active speakers can transmit to the group.

When $p_r \bmod a_r = 0$, fixed allocation of TFs lets each active station use $\frac{p_r}{a_r}$ TFs per TC, i.e., active stations share the bandwidth fairly and do not leave unused any fraction of it. The jitter introduced depends on the distribution of TFs in the TC and the allocation of TFs to active stations.

When $p_r \bmod a_r \neq 0$, p_r can be written as

$$p_r = I \cdot a_r + N, \text{ where } I = \lfloor \frac{p_r}{a_r} \rfloor \text{ and } N = p_r \bmod a_r$$

Each active node uses the fixed allocation criteria on I TFs; for the remaining N TFs several approaches can be followed.

1. The N TFs are allocated to the active nodes in a round robin fashion. As a result, with this approach some bandwidth is not used and a large jitter is introduced because usually a node sends I packets per TC and seldom it sends $I+1$ packets. This exceptional packet received between two regular packets can sharply reduce the packet inter-arrival time, thus yielding a large variation of the end-to-end delay (increase of the delay jitter).

2. N of the a_r active nodes fixedly use $I+1$ TFs for sending their packets. The bandwidth is not shared fairly, but all the allocated bandwidth is used and the bandwidth sharing scheme does not contribute to the jitter.

3. The N TFs are left unused; the multicast group does not use some of the bandwidth allocated to it, but the bandwidth allocation scheme does not contribute to jitter and it is very simple.

5.5 Scalable Growth of the Multicast Group

This section discusses how new node can join the multicast group in a scalable manner. The solution extends the virtual ring topology by including one or more *trees*, as shown in Figure 5.14. Exploitation of trees is particularly suited to include in the multicast group remote and isolated nodes. The discussion is divided into: (i) join as a receiving node, and (ii) join as a transmitting (active) node.

5.5.1 Join as a Receiving Node

Suppose node I wants to joint the multicast group as a listener, namely, it wants to receive all the information transmitted over the virtual ring. This can be done by the following steps:
Step 1: Find a path to a node on the virtual ring, say node J, as shown in Figure 5.14, or to a node on an existing path that is already connected to the virtual ring. Note that this node can be part of the group or a transit node (as was discussed in Section 2). **Step 2:** Find for each link on this path p_r available TFs, such that, there are feasible schedules for the p_r TFs on this path (as was discussed in Section 3).
Step 3: If no such schedule is feasible, then go back to Step 1.

We note that the path connecting node I to the multicast group has two simple properties: (i) it may become a path of a subtree rooted at J, and (ii) the capacity from the virtual ring down the subtree to all receiving nodes on its leaves is the same as the virtual ring capacity.

Thus, adding receiving nodes is simple and scalable. For example, assume there is a virtual ring for a multicast group in North America and a user in Europe wants to listen to the group it can do it, as described before, and without affecting the existing group operation.

5.5.2 Join as a Transmitting Node

Let's assume that node I, shown in Figure 5.14, is already a receiving node. However, the user at node I after listening to the group for a while wants to "voice" her or his opinion. Next we discuss the steps needed in order to enable this user to transmit to the group.

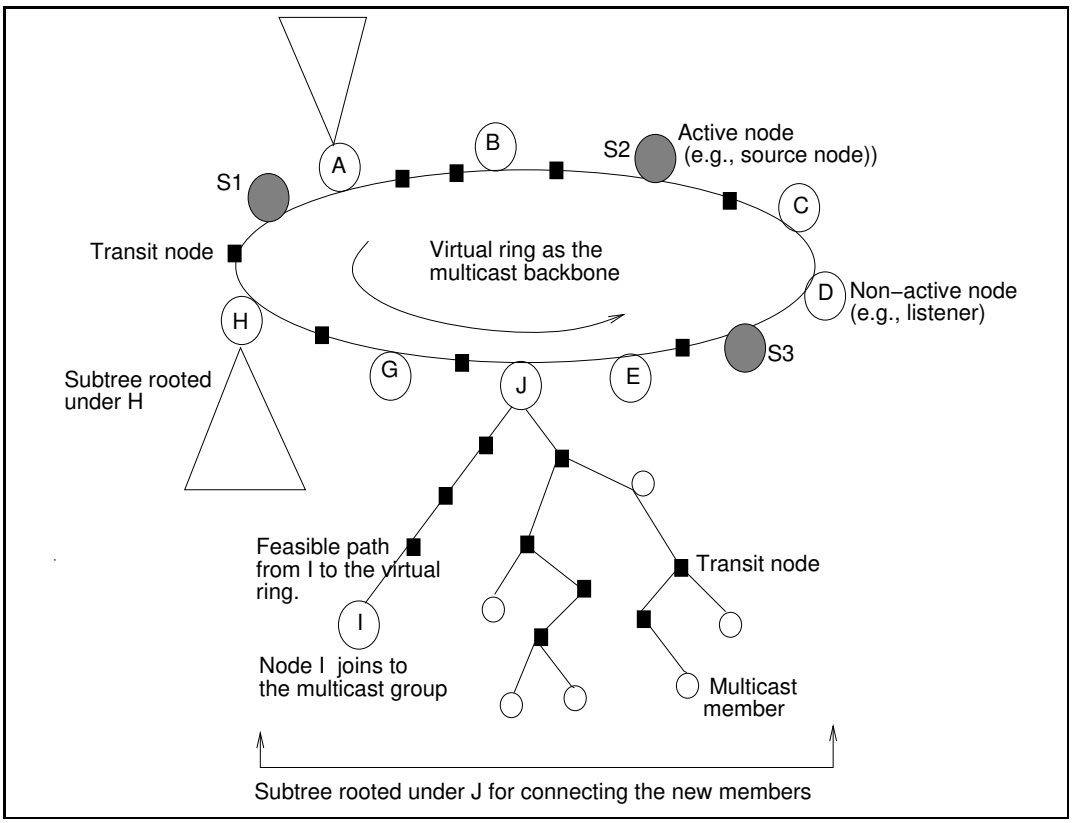

Figure 5.14: Example: Scalable join operation to include new members into a multicast group

Step 1: Check that the subtree is connected to a node on the virtual ring that is a member in the group and not a transit node. If it is a transit node then make this node a part of the multicast group by making it run the real-time multicast protocol.
Step 2: Assume, as shown in Figure 5.14, that node I is connected to node J on the virtual ring. Node I will signal node J that it wants to become active and node J will signal to the group. Since capacity has not been allocated upward in the subtree, node I should allocate on the path to node J the same capacity reserved to the multicast group on the virtual ring. **Step 3:** When it is possible, node J will signal back to node I: (i) that it can start transmitting, and (ii) what fraction of the virtual ring capacity it can use.
Node I uses a fraction of capacity reserved upward on the subtree rooted by node J

The above join and transmitting operation has the following potential problems that do not exist when the transmitting node resides on the virtual ring:

1. Node J should buffer the packets it receives from node I before it can transmit them into the virtual ring. This can add up to one virtual ring round trip delay to every packet.

2. Multiple active nodes on the same subtree are possible. However, a feasible schedule which ensures allocation of enough capacity for each path connecting an active node to the virtual ring is needed. We note that this is a difficult problem similar to finding a feasible schedule for multiple rings.

The above problems are currently under detailed investigation, to quantify the differences between adaptive real-time group multicast over a virtual ring versus over a tree. This study quantifies the buffering and bandwidth requirements with the delay and jitter bounds that can be obtained.

5.6 Discussion

The combination of virtual rings and timing information is interesting and intriguing. In this work we have shown how useful this combination can be for *adaptive real-time multicast*, while in a previous work we have shown that it is also useful for *reliable multicast* for bursty data processing applications [44].

The proposed approach to adaptive real-time multicast is characterized by having a *deterministic behavior*. This is important for real-time applications in two ways: (1) it minimizes the *bound* on the delay variation across each virtual ring, and (2) it ensures no loss due to congestion. The no loss property is important for real-time video applications using compression. In a compressed video stream a loss of a single packet can result in the inability to properly display multiple consecutive pictures, which can annoy the viewer.

There are several intriguing issues which require more research. The scheduling and allocation of time frames to the various multicast groups (virtual

rings) can be improved by further investigating how to reduce the blocking probability. Blocking is defined as the impossibility of allocating enough bandwidth for a new multicast group while some capacity is still available, but not in the proper order of time frames. Another issue concerns the unused capacity assigned to a virtual ring due to the bandwidth adaptation schemes proposed in this work.

Chapter 6

Videoconferencing over Asynchronous and Synchronous Packet Switching Networks

Videoconferencing applications require from the network guaranteed bandwidth, small loss probability, and bounded delay. In order for the participants in a videoconference call to interact naturally, the end-to-end delay should be below human perception - about 100 ms. We identify the components of the end-to-end delay in various configurations with the objective of understanding how it can be controlled and reduced.

The first contribution to the end-to-end delay comes from the processing performed on pictures before sending and after receiving them; we call this processing delay. The second contribution is the network delay. Since pictures must be displayed at the same pace at which they had been captured, any variation in the processing and network delay must be compensated before pictures are displayed. This compensation is done by adding resynchronization delay, which is the third component of the end-to-end delay.

We devise these bounds, and hence the related end-to-end delay, going step-by-step through a number of configurations. We study the transmission of both raw video and Moving Picture Expert Group (MPEG) video (VBR and CBR) over: (i) dedicated links, (ii) circuit switching, (iii) packet switching with time-driven priority, and (iv) asynchronous packet switching. The study

shows that a common time reference used with time-driven priority can provide adequate results, for video conferencing with MPEG, independent of the network load.

The MPEG encoding was designed for storage and one-way (playback) transmission of quality video where end-to-end delay is not a major concern. The main objective has been to reduce the communication resources requirements while maintaining high video quality.

Using MPEG for videoconferencing is not new. However, a systematic study of various quality of service (QoS) parameters has not been done. Such a study is needed in order to avoid various pitfalls, as shown in this report.

We define the following requirements for interactive ("face-to-face") real-time audio and video communications:
(i) for hearing the end-to-end (sender to receiver) delay should be below 100 ms,
(ii) the video stream should be synchronized with the audio stream (a.k.a. lip-synchronization), which implies that the video end-to-end delay should be kept below 100 ms as well, and
(iii) the media (audio and video) *replay* at the receiving side should be *continuous* (which implies that the replay buffer is never overflowed or underflowed).

The main consequence of the above requirements is that we can "close our eyes" and think that the person on the other side is next to us. If we had very high quality 3D (dimension) video, then with our eyes open we could think that the person on the other side is actually next to us - this is also known as *tele-presence or virtual reality*. So the results presented in this work can have further consequences on the design of virtual reality systems.

We identify the components of the delay and study their nature with the goal of reducing each of them thus decreasing the end-to-end delay. While reducing the end-to-end delay it is important to maintain the video quality by avoiding packet loss and by maintaining the quality of each picture. We concentrate on the minimization of the end-to-end processing delay, i.e., we consider the delay introduced by the combination of encoding/decoding and how it is influenced by the transmission and network queuing delay.

6.1 Scope of the Work

In order to have an in-depth understanding of the videoconferencing problem we divide our investigation into several steps. In each step we increase the complexity of the system we analyze. By doing so we can isolate and understand the various delay components.

We consider two variables: the network type and the video type. There are four network types:

1. Dedicated link, which allows the video stream to use the entire capacity of the link, possibly in a bursty manner. Thus, in this case the link capacity can be much higher than average video stream rate.

2. Circuit switching, which allocates a fraction of the capacity to each video stream. In this case the assumption is that the circuit capacity equals the average video stream rate.

3. Time-driven priority [36], which gives the real-time traffic periodic priority over other traffic (both real-time and non real-time). Since all the switching elements are using a common time reference, time-driven priority facilitates pipeline forwarding inside the network. As a result, the real-time traffic cannot be adversely affected by other (real-time or non real-time) traffic, i.e., the real-time traffic is protected by a "fire-wall", and the delay jitter can be bounded by a constant.

4. Asynchronous packet switching, which does not have common time reference and even if priority is used congestion can occur. As a result, the delay jitter can be large and packet loss can occur.

We also investigate three types of video streams in order to better understand the actual effect of the video processing on the end-to-end delay.

1. Raw video. In this case the processing in minimal: pictures are captured and sent directly to the network interface.

2. Variable bit rate (VBR) MPEG. Each picture is encoded and immediately sent to the network interface. In this case the video encoder includes small or no buffer.

3. Constant bit rate (CBR) MPEG. The video encoder includes a buffer and rate controller in order to ensure that the bit rate provided to the network interface is constant. In this case an encoded picture may be delayed for several picture periods before it is forwarded to the network interface.

The various configurations are shown in Table 6.1. Each configuration is characterized by both the video stream and network types. The entries of Table 6.1 show the structure of this work by identifying the Section in which the corresponding configuration is studied. In Section 6.6 we summarize the obtained results.

	Dedicated Link	Circuit Switching	Time-Driven Priority	Asynchronous Packet Switching
Raw Video (Sec 6.3)	Sec 6.3.1	Sec 6.3.2	Sec 6.3.3	Sec 6.3.5
VBR MPEG (Sec 6.4)	Sec 6.4.2	Sec 6.4.5	Sec 6.4.3	Sec 6.4.4
CBR MPEG (Sec 6.5)	Sec 6.5.4	Sec 6.5.5	Sec 6.5.6	Sec 6.5.7

Table 6.1: Configurations Considered in this Work.

6.2 The Model

In this section we introduce the key elements of the system dealt with in this work. We focus on quality of service (QoS) issues in a digital videoconferencing system. In particular we are concern with the delay and jitter, the sender-receiver synchronization, and packet loss. Dealing with the quality of video and audio encoding is beyond the scope of this work. In fact, we

model the delay as being end-to-end, such that, it includes the jitter compensation and media synchronization. In general, synchronization is needed for continuous play of audio and video streams at the receiving side.

6.2.1 Requirements for Interaction

People participating in a videoconference should feel like being face-to-face in the same meeting room. A necessary requirement in order to achieve this goal is to keep the *end-to-end delay* below human perception. The human ear is sensitive to the delay response in a conversation (just think of how annoying are phone calls routed through satellite links); the assumption is that the limit below which the delay is not perceptible is *100 ms* [33].

Of course, the delay requirement is not sufficient, there are other requirements like picture quality, 3D, etc.. However, the end-to-end delay is, in our view, the most essential requirement for interactive real-time applications. The delay requirement is also the least obvious to solve, since it is subject to physical limitation, such as, the speed of light...

Requirement 1 *Video and audio streams are synchronized.*

Since the video stream should be synchronized with the audio the same end-to-end delay constraints applies to video as well. This synchronization is essential since people are used to see even before hearing (consider, for example, a spectator who is attending a concert but is far from the stage). We focus on the end-to-end delay in transmission of video because it is the harder to keep below the 100 ms target.

Requirement 2 *The end-to-end delay of the audio and video streams should be kept below 100 ms.*

The end-to-end delay has three main components that we want to study separately with the aim of reducing them and keep the end-to-end delay below the 100 ms bound:

1. *Processing*: it is needed to transform the audio and video signals in a format suitable for transmission on a digital network. Processing may include also the compression and decompression of the audio and video signals.

2. *Network*: time taken to move data containing the audio and video information from the source to the other participant(s). It also encompasses protocol processing in both sender and receiver(s).

3. *Resynchronization*: digitalization and playing of both audio and video require strict time synchronization between capture of the signal and replaying it. Being the processing and network delays variable, they introduce asynchrony that has to be compensated by adding further delay; this is described more in detail in the next Section.

6.2.2 Requirements for Visual Quality

Digital transmission of video requires a video-camera capture pictures at a fixed pace (*video frame rate*) and a frame grabber board digitalize them; the time between two subsequent captures is called *video frame period*. The produced bits are sent to the receiver(s) where they are converted back to pictures and displayed on a monitor at the same regular pace at which they have been captured. If the receiver does not display pictures at this same video frame rate, the reconstructed video scene results can be annoying. This is a sort of quality degradation which is not acceptable in this work, i.e.,

Requirement 3 *The receiver continuously display pictures at the average rate they have been captured.*

In order to have continuous playing of the video frames on the receiver side, the end-to-end delay of each picture must be constant, i.e., sender and receiver must be *synchronized*.

Since the network delay component is variable, the receiver has to compensate for this variability (a.k.a. *network delay jitter*) by introducing a *network resynchronization delay*. This is achieved by possibly buffering the received bits and delaying the presentation of the corresponding picture. As shown in Figure 6.1, the buffer used to compensate the network jitter is usually called *replay buffer*. The overall effect of the replay buffer, is that each picture takes the same time to transit from the sender (point `A` in Figure 6.1) to the output of the buffer (point `B` in Figure 6.1); the shorter the network delay experienced, the longer the time the picture spends in the replay buffer (i.e., the network resynchronization delay).

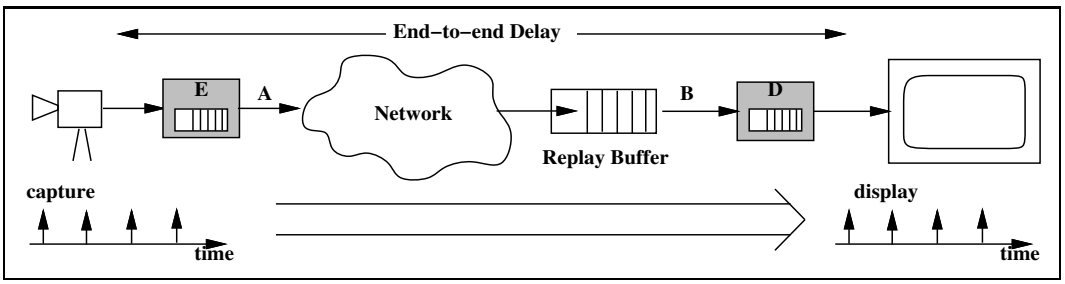

Figure 6.1: Synchronization in Video Transmission.

After the system (in particular the replay buffer) is dimensioned, the time a picture spends in moving from the point `A` to the point `B` in Figure 6.1 is fixed. If a picture experiences a network delay larger than this time, Requirement 3 is not satisfied. If only part of a picture experiences a too large delay, it cannot be used when displaying a picture. This is not acceptable, especially when the video is compressed by eliminating the temporal redundancy between pictures because the resulting image quality degradation is propagated to the following video frames. and thus it is not acceptable. Hence,

Requirement 4 *The network delay is bounded.*

If this requirement is satisfied, the replay buffer can be dimensioned large enough to compensate for the network jitter and guarantee synchronization.

Since the amount of bits obtained by digitalizing a picture is very large and their transmission very expensive, compression is performed before transmission. This takes a variable amount of time as well as their decompression; the encoder/decoder system introduces a *processing resynchronization delay* that compensates for this variability. In other words, the sum of the time each picture spends in block `E` and in block `D` depicted in Figure 6.1 is constant; this time is clearly larger than the maximum processing delay experienced by a picture.

Knowledge of the resynchronization delay is the basis for dimensioning the resynchronization components of the system. Since the aim of this work is keeping the end-to-end delay below a perceptible bound, in the following we do not consider resynchronization delays per se. In fact, a wise resynchronization makes the end-to-end delay be the sum of the maximum processing delay and the maximum network delay experienced by any packet. Thus, we focus on these two maximum values in various different scenarios.

Buffering Issues

Figure 6.1 depicts the replay buffer as a separate function. Actually, network delay variation can be compensated either in the network interface or in the decoder itself, i.e., the replay buffer can be merged with the decoder buffer. In the former case, the compensation can be more efficient (i.e., the network resynchronization delay be minimum) because it can exploit timing information about the network. Instead, merging the two buffers provide more robustness against misdimensioning of the buffers because the variations of network and processing delay can sometimes compensate each other.

In principle, synchronization can be performed by a single buffer inserted just before the display, as shown in Figure 6.2. Buffers can be present in the network interface (e.g., if a packet switched network is used incoming bits must be assembled in packets before the header is processed) and in the decoder to store arriving data until there is enough data to be processed and while the decoder is busy. Anyway this buffers are not used for compensating variations in the processing and network delays.

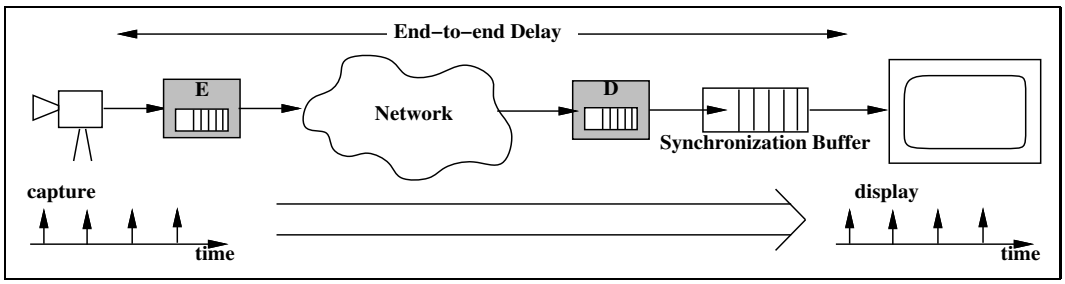

Figure 6.2: Synchronization Just Before Displaying.

Encoded pictures are decoded as soon as they are received, buffered in the format suitable for displaying, and tagged with the displaying time; when this time has come, the picture is displayed. The synchronization buffer is larger that the buffers used for synchronization before decoding the pictures.

6.2.3 The System

In our view, a videoconferencing system is composed of the functional blocks depicted in Figure 6.3. Not all of them are present in any configuration taken

into consideration in this work. For example, when transmission of raw video is considered, both the encoder and decoder functionalities are not included in the system.

In this work each of these functional blocks is considered from the viewpoint of the delay it introduces. In this Section only capture card and display are discussed; in the following Sections the other blocks are described as they are first introduced.

A capture card, or frame grabber, takes around 3 ms to digitalize each picture captured by the video camera; this time can be neglected with respect to the objective 100 ms end-to-end delay bound.

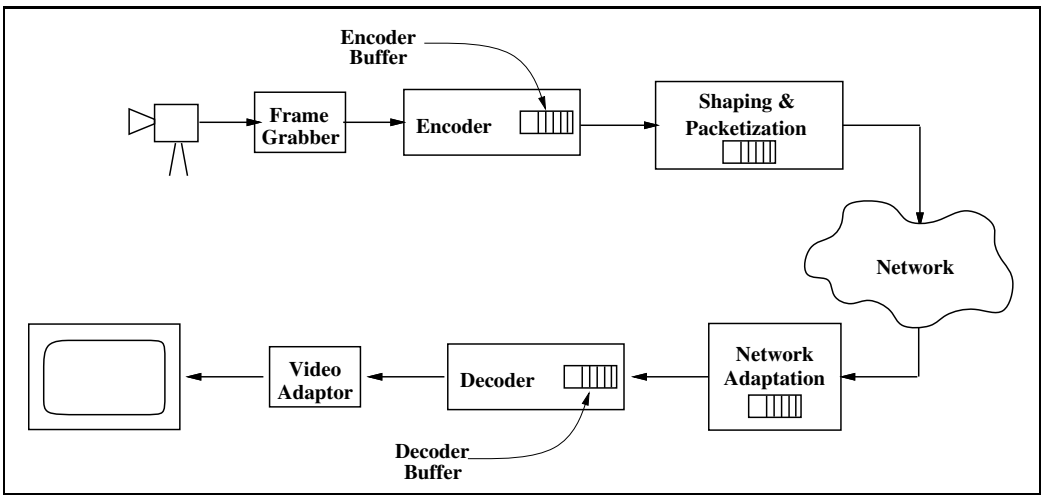

Figure 6.3: High Level Model of a Videoconferencing System.

When a picture is available on the receiver side in a format suitable for displaying, it is inserted into the video frame buffer. The video adaptor periodically scans the video frame buffer and traces the image on the screen according to the information stored for each pixel. The delay introduced by the display depends on the video refreshing frequency. Current monitors work at refreshing frequencies R_f between 60 Hz and 100 Hz, thus introducing a *presentation delay*

$$P_d \in \left[0, \frac{1}{R_f}\right]$$

The bound on the presentation delay is 17 ms for $R_f = 60$ Hz. $P_d = 0$ is obtained by synchronizing the video adaptor with other components of the

receiver: the component to be synchronized will be identified for each of the system configurations presented throughout this work. When no synchronization is provided, the value of P_d varies in the given interval according to the instantaneous time relationship between when a picture is inserted in the video frame buffer and when the buffer is scanned for refreshing the display.

6.3 Transmission of Raw Video

In order to better analyze each component of the end-to-end delay and show in which component of the system it is introduced, an incremental approach is used. We start from the configuration of the videoconferencing system which requires the smallest number of functional blocks and has the smallest number of components in the end-to-end delay. Then, in Sections 4 and 5 we add functional blocks to the configuration and show how the end-to-end delay is affected.

In this first approach we consider the transmission of raw video over a number of different network configurations. Although video is not transmitted in a raw format due to its high bandwidth requirements, it is instructive to study this case. Since in this case we can study in isolation the network delay and the presentation delay[1].

6.3.1 Dedicated Link Between Sender and Receiver

Figure 6.4 shows the block diagram of a videoconferencing system in which sender and receiver are directly connected by a link having capacity C. The F_r bits of a picture are made available to the network interface, from the capture card, in a few milliseconds. The picture is then sent without any additional processing at the link speed, as shown in Figure 6.5. We call *picture dimension* the number of bits encoding a picture.

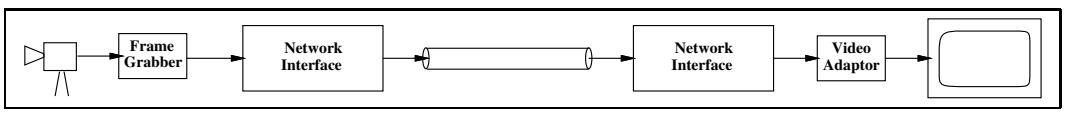

Figure 6.4: Raw Video over a Dedicated Link

[1]The time required for digitalizing (capture) a picture is considered relatively small.

A buffer is needed on the sender side to match the rate at which bits are produced by the frame grabber, to the link speed C. A buffer is also needed on the receiver side to gather the bits while they are being received from the network; as soon as the last bit of a picture is received, the picture can be moved into the video frame buffer for display. Thus, each bit experiences in the sender and receiver buffers an overall delay which is equal to the *transmission delay* of F_r bits (dimension of a raw picture) over a link of capacity C, i.e., F_r/C: the first bit spends the time only in the receiver buffer, while the last one only in the sender buffer; intermediate bits spend time partly in the sender buffer and partly in the receiver one.

If the time needed for digitizing pictures is neglected, the end-to-end delay is given by

$$\Delta_{Raw}^{Ded} = \frac{F_r}{C} + P + P_d,$$

where F_r is the dimension of a raw picture, P is the *propagation delay*, and P_d is the presentation delay which depends on the refreshing frequency of the monitor and the synchronization relationship between capture card and video adaptor. P_d is null if the video adaptor on the receiver is synchronized with the capture card of the sender.

The propagation delay exists in all the configurations presented in this work, even though in some of them it is concealed inside some other component of the end-to-end delay. The propagation delay is a physical lower bound on the end-to-end delay of the system; if sender and receiver(s) are so far away such that $P > 100$ ms[2], the bound on the end-to-end delay cannot be satisfied (i.e., face-to-face-like interaction is not possible), independently of the configuration used for the videoconferencing system.

F_r depends on the resolution of the pictures (i.e., the number of pixel per picture) and the number of bits used to encode each pixel. For example, a QCIF (Quarter Common Image Format) picture has a definition of 176 x 144 pixels; since on the average each pixel requires 12 bits, $F_r = 297$ Kb. Instead a High Definition TeleVision (HDTV) video frame is digitalized using 1920 x 1080 pixels, i.e., 16200 Kb. Thus, for example, the transmission delay of a QCIF picture on a 100 Mb/s link is 2.97 ms: even considering the worst

[2]Propagation delay of 100 ms in fiber is about 20,000 kilometers or half of the Earth circumference.

case for the presentation delay, 80 ms can be spent in propagation delay over the link, while keeping the end-to-end delay under the 100 ms bound. Instead, an HDTV picture takes 162 ms to be transmitted over a 100 Mb/s link; this is above the 100 ms bound required for interaction and moreover it does not allow for real-time video, as shown in the following.

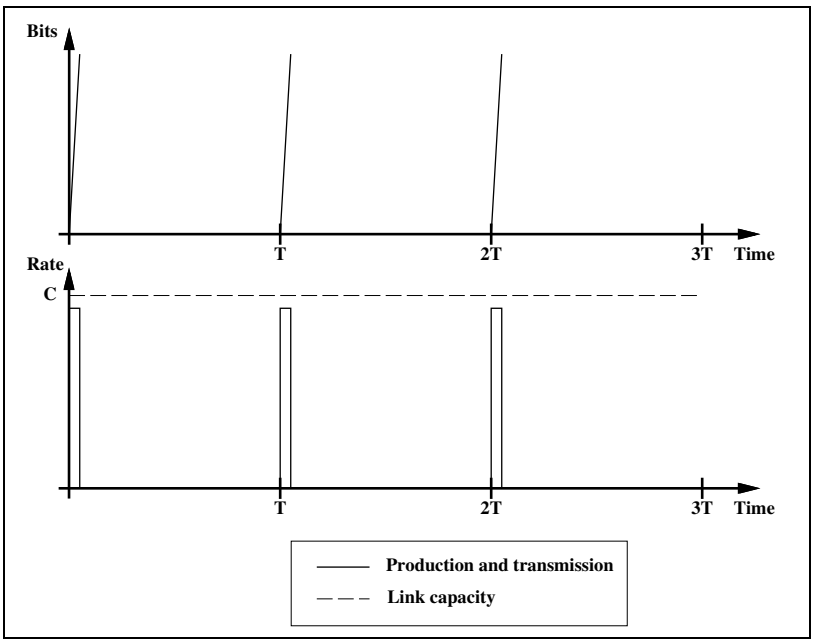

Figure 6.5: Generation and Transmission of Raw Pictures over a Dedicated Link.

Real-time video transmission requires that

Requirement 5 *Each picture is transmitted within the video frame period* T*,*

i.e., the transmission delay must be smaller than T; thus, given the resolution of pictures and the video frame rate, the capacity of the link must be large enough to satisfy

$$C \geq \frac{F_r}{T}. \tag{6.1}$$

For example, the minimum link capacity needed to transmit video sequences at HDTV definition is 486 Mb/s since the HDTV frame rate is 30 pictures

per second. When a minimum capacity link is exploited, the generation and transmission of pictures shows the behavior depicted in Figure 6.6.

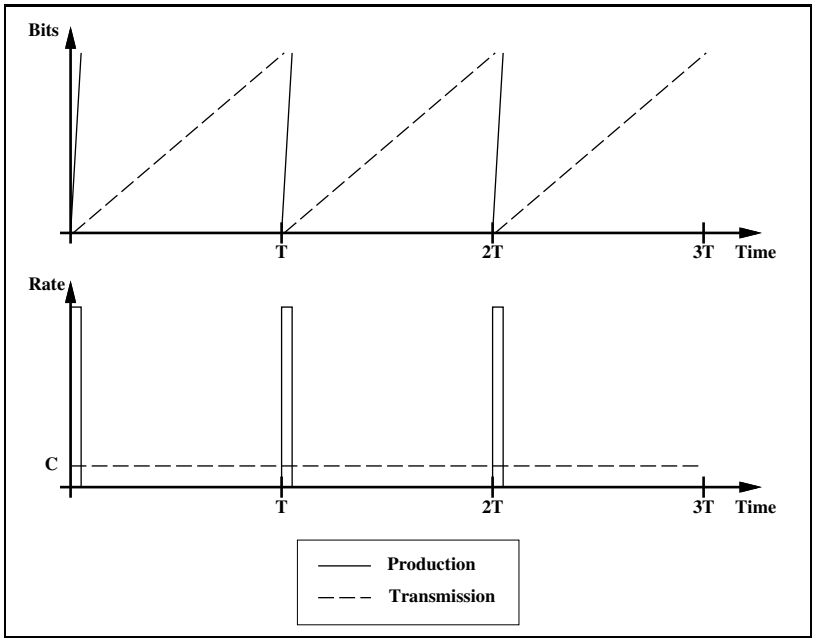

Figure 6.6: Generation and Transmission of Raw Pictures over a Slow Dedicated Link.

Thus, in a videoconferencing system that transmits raw video over a dedicated link, the main parameter in determining the end-to-end delay is the link speed. The higher the picture definition and the video frame rate, the faster the links. If the video frame rate is low or the propagation delay is large, the lower bound on link speed is not given by (6.1), but by the bound on the end-to-end delay. This leads to high cost for the videoconferencing system, especially when high definition is required and sender and receiver are far (that is the situation in which videoconference is usually needed).

High speed links are now available also on long distances, but they are very expensive. Thus, dedicating the whole capacity of a link to a videoconference call is not cost effective, particularly over long distances; nevertheless, given the speed of links, the configuration described in this Section provides a lower bound in the end-to-end delay for transmission of raw video. According to other transmission schemes, links are shared among various calls or types of traffic; in the following other schemes are studied with respect to the results shown in this Section.

6.3.2 Circuit Switching

In circuit switching link capacity is shared among different calls. A fixed amount of capacity is assigned to each connection using time division multiplexing. The architecture of the videoconferencing system is depicted in Figure 6.7; sender and receiver communicate as they were connected by a dedicated link, even though its bandwidth is usually smaller than the physical speed of the links on which data actually flow.

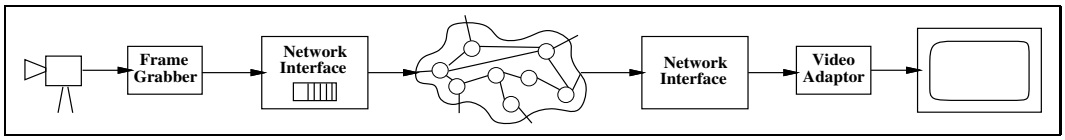

Figure 6.7: Model of a Videoconferencing System Exploiting a Circuit Switched Network.

A videoconference call is allocated a circuit with bandwidth B by reserving the corresponding fraction of the capacity of each link on the path between sender and receiver. The bits encoding each picture are sent through the circuit at rate B as shown in Figure 6.8. This introduces a network shaping delay given by:

$$S_n = \frac{F_r}{B} \tag{6.2}$$

The end-to-end delay is given by

$$\Delta_{Raw}^{CS} = S_n + P + Sw + P_d, \tag{6.3}$$

where P is the propagation delay and Sw is the *switching delay*. This end-to-end delay is (almost) constant because B, P, and Sw are almost constant; thus, network resynchronization is not needed in the receiver.

According to Requirement 5 for videoconferencing application $S_n \leq T$, i.e.,

$$B \geq \frac{F_r}{T}$$

.

On one hand, the larger B, the smaller the end to end delay; on the other hand, the larger B, the larger the amount of bandwidth is wasted because

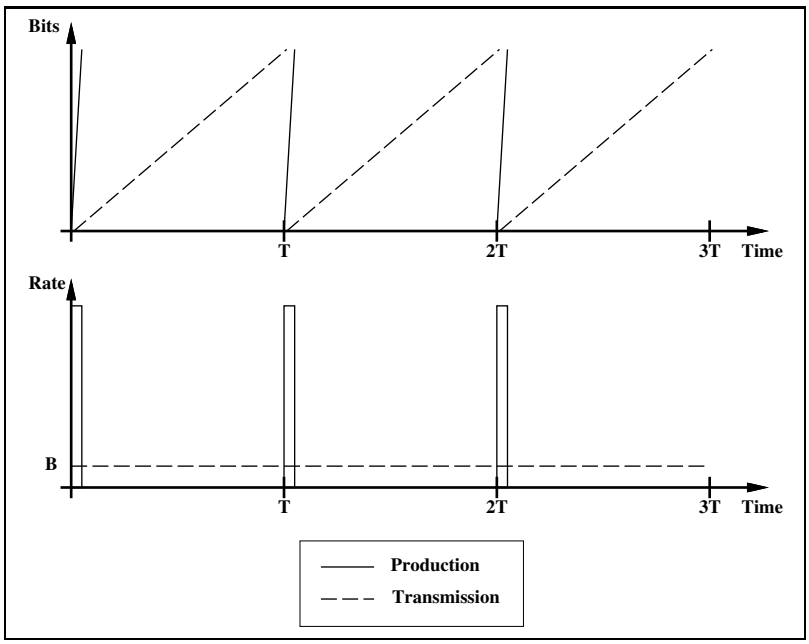

Figure 6.8: Amount of Bits and Rate in Raw Video Encoding and Transmission over a Circuit Switched Connection.

the circuit is busy only for a time S_n in each video frame period T. During the remaining time $T - S_n$, no other connection or class of traffic can exploit this unused fraction of links capacity.

If minimum bandwidth is allocated to the videoconference call, the end-to-end delay is larger than the video frame period; the lower the video frame rate, the larger the end-to-end delay. For example, the minimum bandwidth required to send QCIF pictures at 15 fps, is 4.5 Mb/s, but the resulting shaping delay is 67 ms. Nevertheless, if more bandwidth is allocated to decrease the network shaping delay to 30 ms, 50% of the allocated bandwidth is wasted because the circuit is idle for half of the video frame period. On of the motivations for packet switching is to share link capacity in a flexible and efficient manner. However, some packet switching schemes may require to maintain low bandwidth utilization when trying to satisfy real-time traffic requirements.

6.3.3 Packet Switching with Time-Driven Priority

Figure 6.5 shows that the transmission of a video frame can require use of the link for a short time: the higher the link speed, the longer the idle time. In this Section we show how the link can be used for transmission of other classes of traffic (either requiring guarantees on service quality or "best effort") during its idle periods.

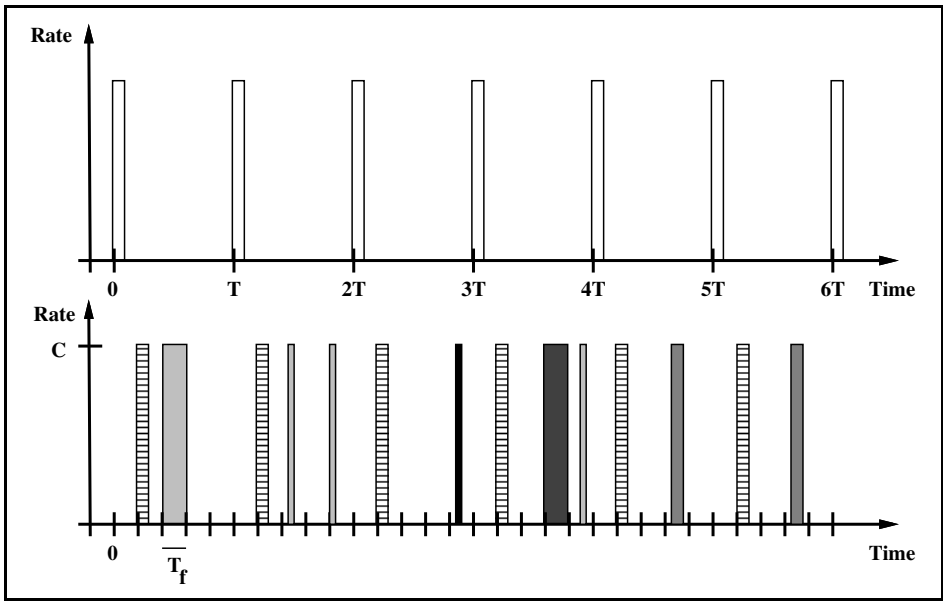

Figure 6.9: Generation of Raw Pictures and Multiplexing with Time-Driven Priority.

A picture is divided into one or more parts which are inserted into packets. Each packet is sent in a single TF. For the sake of simplicity, in this work we do not consider the processing required for packetization in either the sender or the receiver and the transmission overhead introduced by the packet headers. In the TFs between the transmission of two subsequent pictures of the same session (and in the remaining time in the same TF), other data can be transmitted (see Figure 6.9), i.e., traffic multiplexing is performed on the links.

Sender and receiver are not directly connected through a dedicated link; instead, they are connected to intermediate systems, which in turn are interconnected in a general topology, as shown in Figure 6.10. The sender sends its packets to the intermediate node to which it is directly connected; the

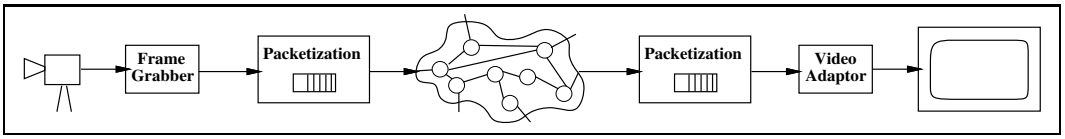

Figure 6.10: Model of a Videoconferencing System Exploiting a Packet Switched Network.

packet header identifies the receiver so that the intermediate nodes can route it to the destination.

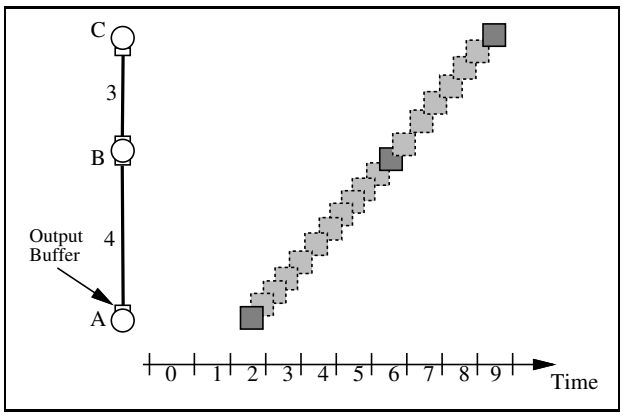

Figure 6.11: RISC-like Packet Forwarding.

6.3.4 Transmission of Video and Scheduling

Reservation of TFs allows nodes to perform *RISC like forwarding* [35] (see Section 2.2). Given the path through the network, a packet takes a time $L \cdot T_f$ to travel from source to destination: the end-to-end delay in the videoconferencing system is

$$\Delta_{Raw}^{TDP} = L \cdot T_f + P_d$$

where $P_d \in [0,1/R_f]$ is the presentation delay and R_f is the refreshing rate of the display. P_d can be eliminated by synchronizing the video adaptor and the receiver network interface so that the display is refreshed just after the TF in which the picture is received.

The variation of the network delay is small enough to be tolerated, notwithstanding the continuous playing Requirement 3. Hence, there is no need for

introducing a network resynchronization delay, i.e., no replay buffer is needed on the receiver side. L depends on the number of hops on the path (it is at least equal to the number of hops), the processing delay inside each node, and the propagation delay between each pair of intermediate systems.

Resource reservation is based on the definition of a *time cycle* which encompasses a fixed number of TFs: all the nodes share the same knowledge of the ordinal position of the current TF inside the time cycle. Bandwidth is allocated to a sender/receiver pair, by reserving a proper number of TFs per time cycle on each link on the path between sender and receiver. In order to have intermediate nodes performing RISC-like forwarding, the TFs on a link must be chosen according to the TFs reserved on the upstream link and the time needed for a packet to transit from the output buffer of the upstream node to the output buffer of the considered node.

Since raw video has a natural pacing driven by the picture rate, the time cycle duration must be an integer multiple of the video frame period. For the sake of simplicity we consider the time cycle being equal to the video frame period; the obtained results can be extended to the general case by taking into the considerations made in Section 6.3.4. A TF in the time cycle is reserved to the videoconference call, as shown in the lower part of Figure 6.12. The choice of the TF to be reserved within the time cycle on each involved link is called *scheduling* and it must impacts the maximum number of real-time connections concurrently supported by the network [35]. In the following the impact of scheduling on the end-to-end delay of the videoconference call is presented; anyway, as scheduling is out of the scope of this work and it will not be addressed in details in this work.

The lower part of Figure 6.12 depicts the TFs allocated on the link connecting the sender to the network and the upper part shows the timing of the frame grabber digitalizing pictures. If the frame grabber is not synchronized to the network interface, the bits produced by the former are buffered for a time $S_n^{AS} \in [0,T]$ while waiting to be transmitted; this is the *application synchronization* component of the *network shaping delay* since it is used to adapt the timing characteristics of the application to the ones of the network. If the clock that drives the pace at which pictures are captured is synchronized to the one used by the network interface to delineate TFs, S_n^{AS} is constant during a videoconference call. If there is a drift between the two, S_n^{AS} slowly

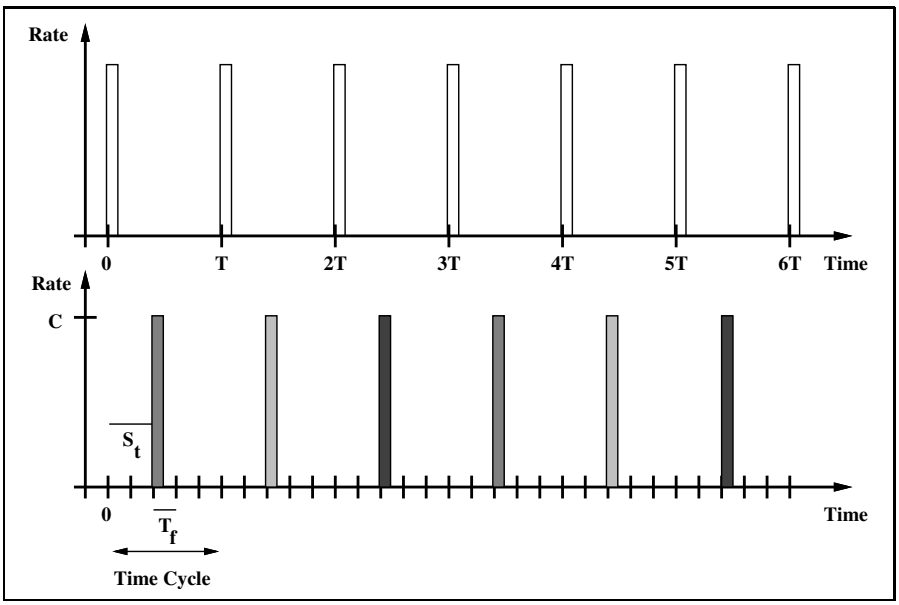

Figure 6.12: Transmission with Time-Driven Priority.

increases or decreases; when S_n^{AS} reaches the boundary of its variation interval and wraps around, a TF can be left unused (when S_n^{AS} changes from 0 to T) or a picture overwritten by a new one (when S_n^{AS} changes from T to 0), and hence it is not transmitted.

The end to end delay is given by:

$$\Delta_{Raw}^{TDP} = S_n^{AS} + L \cdot T_f + P_d \tag{6.4}$$

The given expression holds when a picture can be included in a single packet, i.e., $F_r \leq C \cdot T_f$. This can be satisfied when the links are fast enough so that

$$C \geq \frac{F_r}{T_f}$$

Even with QCIF format pictures, capacity of links must be larger than 2.3 Gb/s in order to send a whole picture during a single TF (HDTV definition requires link speed larger than 130 Gb/s. Where such links are not available, a picture can be sent during N_s TFs, where

$$N_s = \left\lceil \frac{F_r}{T_f \cdot C} \right\rceil$$

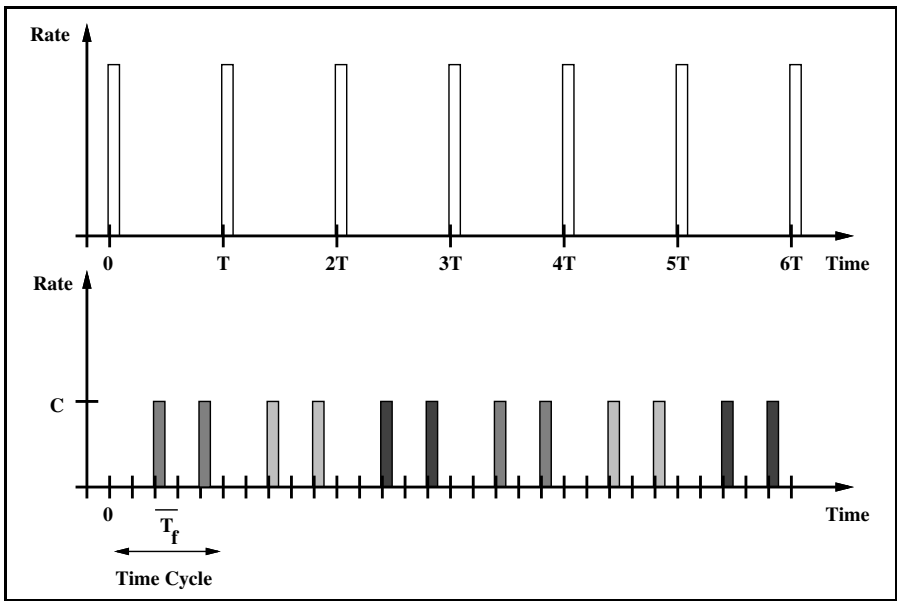

Figure 6.13: Transmission with Time-Driven Priority.

(e.g., in Figure 6.13 $N_s = 2$). Thus, the end-to-end delay in a videoconferencing system sending raw video frames using packet switching with time-driven priority is given by:

$$\Delta_{Raw}^{TDP} = S_n^{AS} + (N_r - 1) \cdot T_f + L \cdot T_f + P_d$$

where $N_r \geq N_s$ is the number of TFs between the first and the last TF reserved for the transmission of a picture. In fact, it can happen that N_s consecutive available TFs cannot be found on each link in the proper position inside the time cycle; thus, non consecutive TFs are reserved and the end-to-end delay is increased.

The end-to-end delay can then be written as[3]

$$\Delta_{Raw}^{TDP} = S_n + L \cdot T_f + P_d \tag{6.5}$$

The component $L \cdot T_f$ depends on the topological and physical characteristics of the system. We call *network shaping delay*

[3]If the time cycle does not coincide with the video frame period and an optimal schedule is not found, the problems presented in Section 6.3.4 arise.

$$S_n = S_n^{AS} + (N_r - 1) \cdot T_f$$

the delay introduced in order to adapt the rate at which the bits encoding each picture are generated, to network and transmission scheme exploited to transmit them. S_n depends on the link speed, the definition of pictures, and the scheduling.

Using QCIF format at 15 pictures per second, $N_s = 25$ and S_n can vary from 3 ms (when capture card and network interface are synchronized and an optimal schedule is found) to 68.8 ms. Considering a scenario in which the path from sender to receiver includes 3 hops (with negligible propagation and processing delay, i.e., $L = 3$), $S_n^{AS} = 0$ and $P_d = 0$, the end-to-end delay is 3.375 ms. In the configuration with direct link in similar conditions ($C = 100$ Mb/s and P is negligible) the end to end delay is 2.97 ms, as was shown in Section 6.3.1. Thus, the end-to-end delay is comparable in the two cases, but 97% of the dedicated link capacity is left unused.

As equipment for videoconferencing applications will evolve, synchronization between capture card and network interface will be more likely and the S_n^{AS} component will be null because a picture is sent immediately after it has been digitalized (provided that an optimal schedule does exist). The lack of this synchronization can have a worse impact on the end-to-end delay the lower is the video frame rate. Even though capture card and network interface are synchronized, a large network shaping delay can be due to the impossibility of finding an optimal schedule. Thus, if the route from sender to receiver is characterized by a small value of L, a sub-optimal schedule can be accepted provided that it compiles with the 100 ms end-to-end delay bound. If L is large, an optimal schedule is required to the network when the conference call is submitted; this increases the probability for the call to be blocked.

Multiple Videoconference Calls on the Same Network

In Section 6.3.4 is shown how bandwidth is allocated to a videoconference call by reserving on TF in a time cycle whose duration equals the video frame period T. If more than one videoconferencing system is exploiting the same network and they do not use the same video frame rate, the time cycle is

chosen as the minimum common multiplier of all the video frame periods in use.

From the point of view of each of the videoconference calls, the time cycle is an integer multiple M of its own video frame period T. Assuming that each picture is sent within a single TF, bandwidth is allocated by reserving M TFs within the time cycle. If the TFs are evenly distributed within the time cycle, i.e., the time distance between two subsequent TF is T, everything has been stated about the case $M = 1$, still holds.

If the reserved TFs are not evenly distributed, their relative time distance is not T any more and the receiver does not receive pictures at a constant pace. This is equivalent to a variation of the network delay; the less evenly distributed are the TFs, the larger the resulting variation of the picture inter-arrival time. If this variation is too large according to the continuous playing Requirement 3, the receiver compensates it through a replay buffer and the end-to-end delay is accordingly increased with respect to the one given in (6.4). If more than one TF is used to send each picture, the choice of the TFs to be reserved impacts the end-to-end delay; a detailed analysis of the yielded end-to-end delay is outside the scope of this work.

In general, scheduling is important to both network performance and end-to-end delay of videoconferencing applications. The two sometimes impose conflicting optimality criteria for scheduling and thus a trade-off must be found among them [1]. A videoconference application requiring a bandwidth reservation to the network, can ask for a schedule that minimizes the end-to-end delay and does not require a resynchronization capable receiver, at the risk of getting the request refused. Alternatively, if the receiver is equipped with a replay buffer to compensate the variation of the network delay introduced by scheduling, it can accept a schedule that increases the end-to-end delay.

The larger M, the harder is for the scheduling to satisfy the optimality criteria. In order to keep M small, videoconferencing application should choose video frame rate that are integer multiples one of the other. The largest video frame period can be chosen as time cycle in the network.

6.3.5 Asynchronous Packet Switching

The videoconferencing system can be built on a traditional asynchronous packet switching network which statistically multiplex data on the links. The F_r bits encoding a picture are inserted in packets of (maximum) dimension P_s which are sent into the network at the full link speed. In the network packets experience fixed delay due to transmission (F_r/C) and propagation (P), and variable delay due to queuing.

Queuing introduces a network jitter that must be compensated by a replay buffer as described in Section 6.2.2. According to Requirement 4, the network delay must be bound, i.e.,

Assumption 1 *A bound Q_M on the queuing delay does exist.*

Packets are queued in network nodes while they are contending for a busy link; thus, the queuing delay depends on the load in the network and it does not have a deterministic bound. A bound can be guessed but to have a low probability of breaking it, it must be large. Moreover, the queuing delay can be kept smaller by avoiding bursts of packets that increase queue dimension; for this purpose traffic is shaped at the boundaries of the network.

In this Section we analyze the contribution of network resynchronization and traffic shaping on the end-to-end delay.

Network Resynchronization Delay

Given the bound Q_M on the queuing delay, each packet that has experienced a shorter queuing delay is delayed in the replay buffer. This is shown in Figure 6.14: the upper diagram shows the arrival time of pictures to the replay buffer, while the lower shows the exit time. The resulting end-to-end delay of pictures is

$$\Delta_{Raw}^{Async} = \frac{F_r}{C} + P + Q_M + P_d. \tag{6.6}$$

If the network interfaces of sender and receiver are not synchronized, the latter is not able to determine the queuing delay experienced by a packet. Thus, the first packet received for the videoconference call is buffered for a

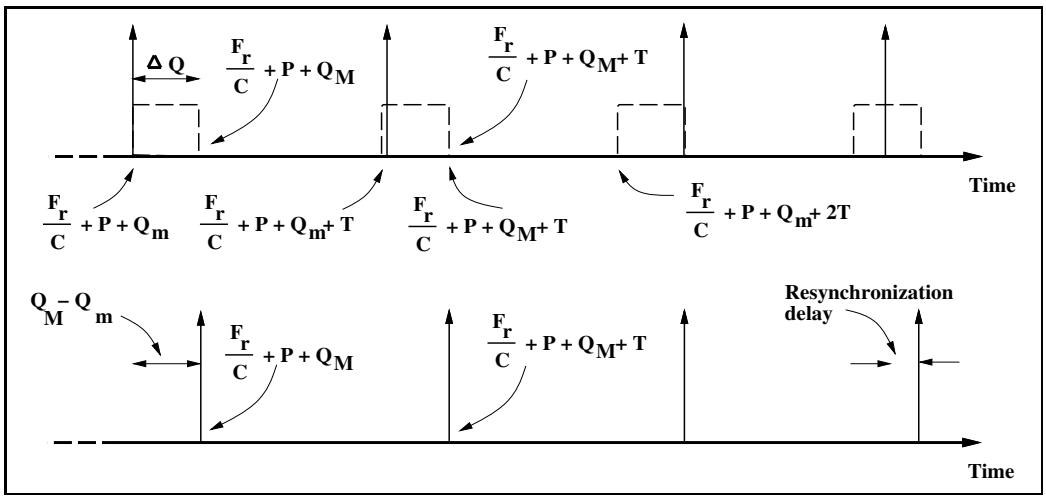

Figure 6.14: Network Resynchronization Delay to Compensate Queuing Delay Variation.

time that allows resynchronization in the worst case: it is assumed that the packet has experienced the minimum queuing delay Q_m and is buffered for a time

$$\Delta Q = Q_M - Q_m \tag{6.7}$$

The following packets are resynchronized accordingly, as shown in Figure 6.14, because they are retrieved from the replay buffer at the constant pace at which pictures are displayed.

The upper diagram of Figure 6.15 shows a resynchronization example when the first packet experiences an actual queuing delay Q_M. The middle diagram shows the timing of packets exiting in case the receiver does not know the actual queuing delay experienced by the first packet, delays it by ΔQ according to (6.7). As a consequence, the overall delay experienced by following packets due to queuing and resynchronization is between Q_M and $Q_M + \Delta Q$. If the receiver knows the actual delay experienced by the first packet (e.g., sender and receiver have a common time reference and the packet contains a time stamp indicating when it was sent), does not delay it at all and the exiting time from the replay buffer are the ones depicted in the lower diagram in Figure 6.15.

In summary, the end-to-end delay is given by

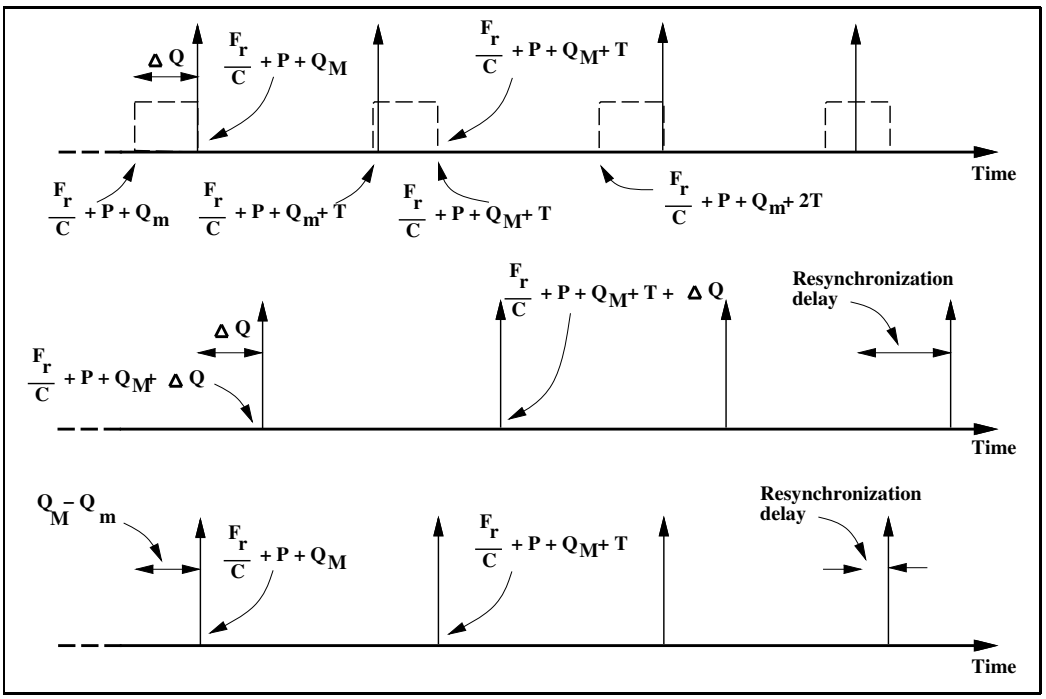

Figure 6.15: Network Resynchronization Delay to Compensate Queuing Delay Variation.

$$\Delta_{Raw}^{Async} = \frac{F_r}{C} + P + Q_M + E_r + P_d$$

where $E_r \in [0, \Delta Q]$ is the *excess resynchronization delay*; it is constant over a videoconference call and depends on the queuing delay experienced by the first packet received, $E_r = 0$ when sender and receiver network interfaces are synchronized.

Traffic Shaping at Network Boundaries

Resources must be allocated into the network in order to guarantee the bound Q_M on the queuing delay (and possibly limit the amount of data lost in the network due to congestion). Since transmission of large amounts of data at link speed (*bursts*) causes sudden growth of queues, large amount of resources is reserved for bursty sources: limits queues growth by keeping link utilization low.

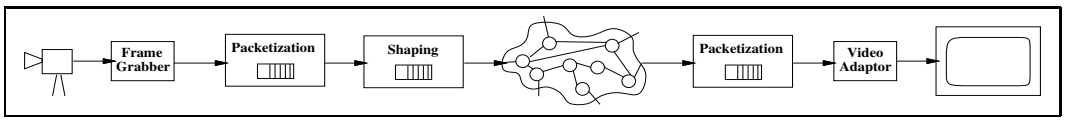

Figure 6.16: Model of a Videoconferencing System Exploiting a Packet Switched Network with Traffic Shaping at Network Boundaries.

Assumption 2 *Traffic shaping is used at the network boundary to provide an average bandwidth between sender and receiver while keeping the burstiness below a predefined level.*

The source shapes the traffic it is generating by using a traffic shaping mechanisms such as a *leaky bucket* [6] (see Section2.1.1). The traffic generated by the source is shaped into a flow having average rate B and maximum burst length (equivalent to the size of the token bucket) A. The choice of the parameters of the traffic shaper determines its effects on the end-to-end delay. According to Requirement 5, as we are dealing with real-time video transmission, the time taken to refill the token bucket must be smaller than the video frame period, i.e.,

$$B \geq \frac{F_r}{T}. \tag{6.8}$$

If $A \geq F_r$ the whole picture is inserted in packets which are sent at the link speed as soon as it is captured, as shown in Figure 6.17 where $A = P_s$. Traffic shaping does not have any effect and the average rate is F_r/T, as determined by the video frame rate $(1/T)$ and dimension (F_r), and the end-to-end delay is given by (6.6).

If $A < F_r$, A/P_s packets of dimension P_s are sent as soon as the picture is digitalized, as shown in Figure 6.18 where $A = P_s$; afterwards, the time needed to generate P_s tokens (i.e., P_s/B) is waited. Thus, the network shaping delay S_n is given by the time needed to generate $F_r - A$ tokens (since, thanks to (6.8) the token bucket is full before starting transmitting a frame):

$$S_n = \frac{F_r - A}{B} \tag{6.9}$$

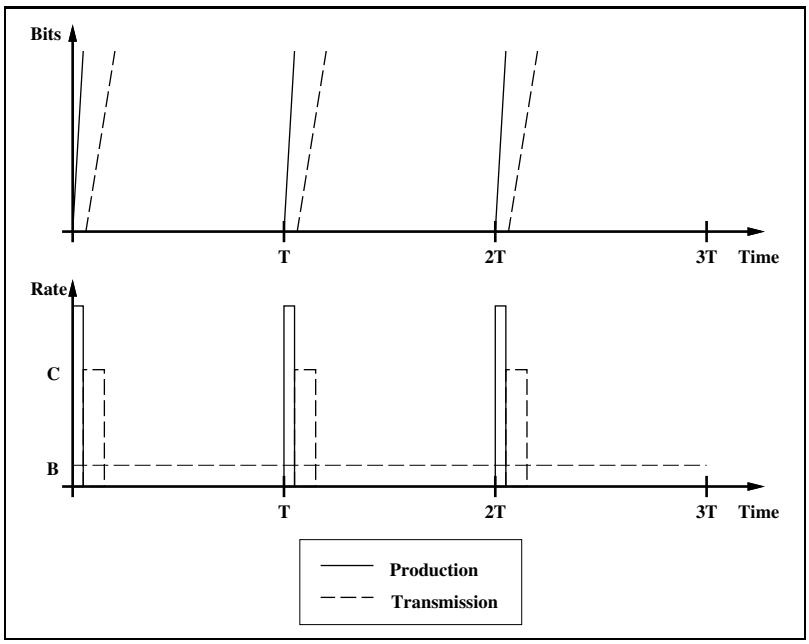

Figure 6.17: High Burstiness Transmission of Raw Video over Packet Switched Networks.

The time needed to transmit a picture being the network shaping delay S_n plus the transmission delay of a packet (P_s/C), the end-to-end delay is given by:

$$\Delta_{Raw}^{Async-Sh} = S_n + \frac{P_s}{C} + P + Q_M + E_r + P_d \tag{6.10}$$

where P is the propagation delay on the links on the path between sender and receiver, Q_M is the maximum queuing delay, and $E_r \in [0,\Delta Q]$ is the excess resynchronization delay.

Comparing (6.6) and (6.10) the main difference is that in the former, where the burstiness is maximum, the network shaping delay is not present. Nevertheless, the lack of traffic shaping at network boundaries results is lower link utilization or larger maximum queuing delay Q_M.

The maximum queuing delay delivered by the network depends on the reservation and queuing policy used by nodes. For example, if network nodes are performing weighted fair queuing, the bound is inversely proportional to the allocated bandwidth [46]:

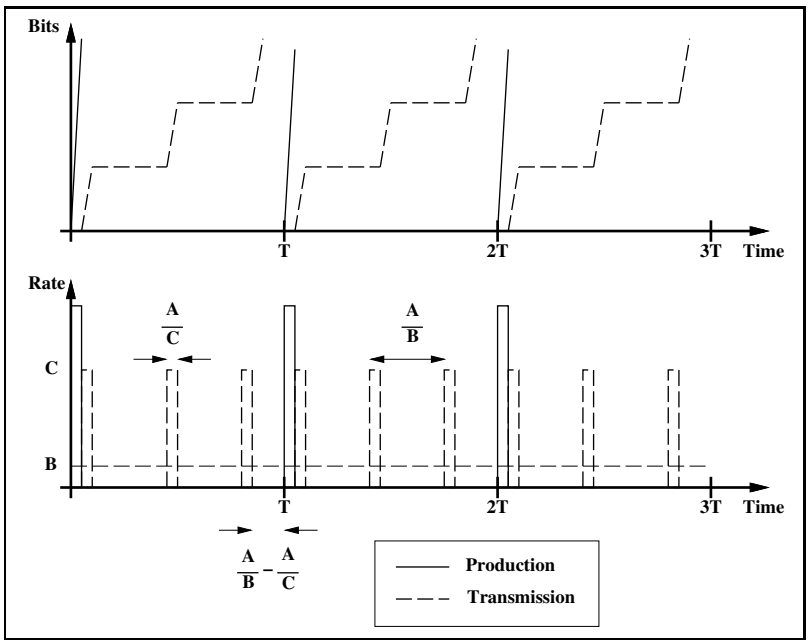

Figure 6.18: Low Burstiness Transmission of Raw Video over Packet Switched Networks.

$$Q_M = o\left(\frac{1}{B}\right)$$

6.4 Transmission of VBR MPEG Video

Transmission of raw video requires large amount of bandwidth, particularly if the end-to-end delay has to be kept under the 100 ms bound. Thus, compression is used to reduce the amount of bits needed to encode each picture by exploiting spatial and possibly temporal redundancy present inside pictures and between subsequent pictures, respectively. The compressed pictures are sent over the network to the receiver where they are decoded and display.

6.4.1 MPEG Overview

The Motion Picture Expert Group (MPEG) [29, 58, 19] video encoding standard was designed for digital storage of quality video for later re-playing. A digitalized uncompressed image is composed of one matrix of nxm pixels of

luminance information and two matrices of $(n/2)$x$(m/2)$ pixels of chrominance information. The encoder receives in input a sequence of digitalized pictures and encodes each of them in one of two different ways[4].

Intra-frame Coding

Spatial redundancy inside the picture is exploited to gain compression. The picture (the luminance and the two chrominance components) is divided in 8x8 pixel blocks. On each block a Discrete Cosine Transform (DCT) is performed; it yields an 8x8 matrix with the coefficients of the spatial frequencies present in the block.

Each coefficient is *quantized* by integer dividing it by an integer called quantization stepsize. Actually, a different quantization stepsize is used for each coefficient because human eye is more sensitive to high spatial frequencies [59]: high frequency coefficients are quantized more coarsely than low frequency ones, but the perceived visual quality does not degrade. Thus, each coefficient is multiplied by the corresponding element of an 8x8 quantization matrix.

The result of the quantization is run-length encoded in order to minimize the number of bits needed to represent it thanks to the fact that many coefficients (particularly the high frequency ones) are zero. Finally, the run-length encoded symbol sequence is Huffman Encoded. The picture encoded in this fashion is called *I-frame*.

Predictive Coding

Temporal redundancy is eliminated to reduce the number of bits needed to encode the picture, with respect to I-frames. The picture is divided into *Macro Blocks* (MBs), each composed of a 16x16 pixel matrix of luminance information and the two corresponding 8x8 pixel blocks of the two chrominance components. Given a MB to encode, *motion estimation* is performed, i.e., the previous picture is searched for a "similar" MB. If this MB is found,

[4]Actually, a third type of encoding, called bidirectional predictive coding, exists. In order for a picture to be coded, a reference subsequent picture must be captured and coded. This introduces a delay of some frame periods that we consider not acceptable if the end-to-end delay be kept below 100 ms. Thus, this compression technique is not considered.

the pixel by pixel difference between the actual MB and the reference one is calculated and coded by performing DCT, quantization, run-length encoding, and Huffman encoding. If a "similar" MB is not found, each block of the MB is encoded like a block in an I-frame.

Pictures coded in this fashion are called *P-frames* and their dimension is from 2 to 4 times smaller than I-frames. The smaller the difference between the MB being coded and the reference one, the higher the compression gained through DCT and run-length/Huffman encoding. The more similar two subsequent pictures, the higher probability of finding a MB "similar" to the one being coded and thus yielding better compression; subsequent pictures are similar if the scene is slow moving because if does not change much from a video frame period to the other.

Bit Generation Rate and Quantization

A video sequence is compressed by encoding a picture out of N as I-frame, and the remaining $N - 1$ as P-frames; the sequence starting with one I-frame and terminating with the last consecutive P-frames is called a *Group Of Pictures* (GOP). The larger N, the smaller the overall amount of bits used to encode a scene. Nevertheless, if the network introduces an error in an encoded picture, it propagates in the following ones; the next I-frame is the first picture not affected by the error and stops its propagation, i.e., the larger N the more sensitive to errors the stream.

The rate of the bit stream produced by the encoder has high variability: each picture is encoded within the video frame period, but a P-frame is from 2 to 4 times smaller than an I-frame. Moreover, even video frames of the same type have different dimension: the dimension of an I-frame depends on the spatial redundancy present in the image, while the P-frame one depends also on the amount of temporal redundancy. Figure 6.19 shows the amount of bits produced by the software MPEG encoder `dvdenc` [38] during the encoding of four video sequences with a resolution of 720x480 pixels; an image from each sequence is shown in Figure 6.20. The overall number of bits encoding each sequence and the way it is distributed among the pictures are different due to the different amount of spatial (detail) and temporal (motion) redundancy in each scene.

A different quantization matrix is used for a MB, depending on the type of

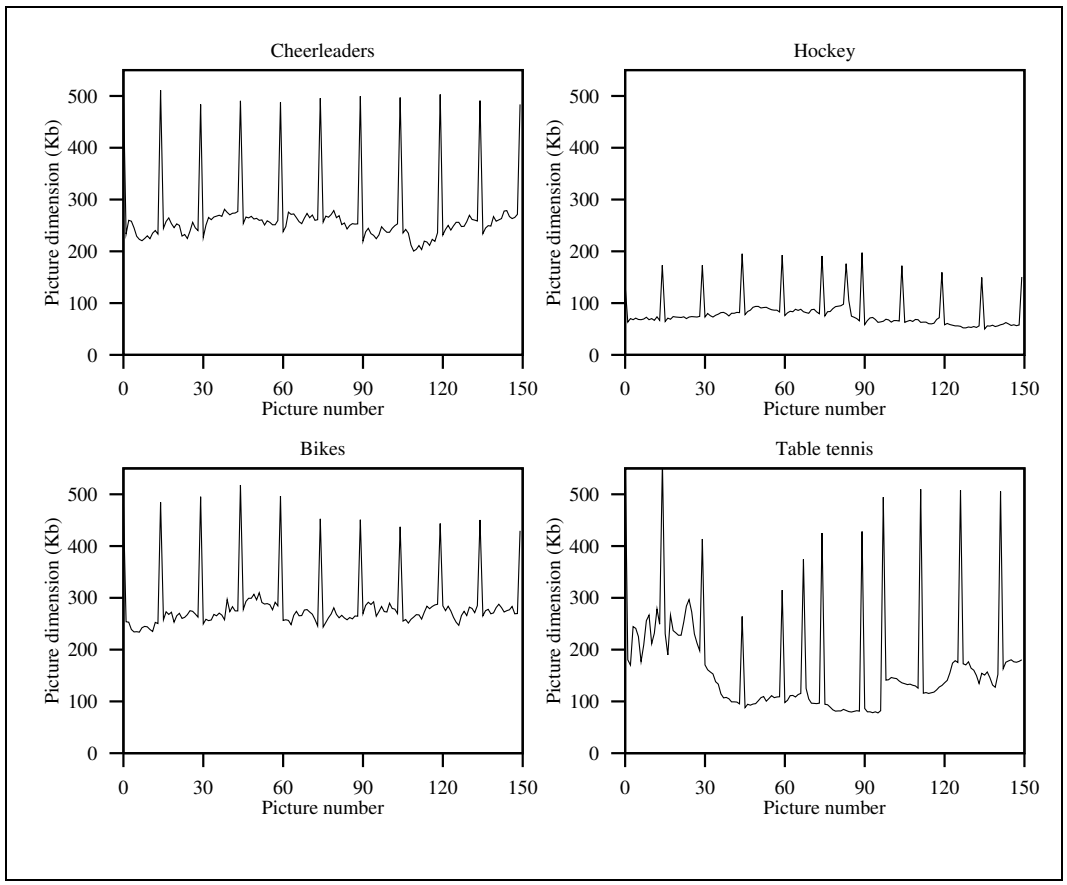

Figure 6.19: Natural Dimension of MPEG Encoded Pictures.

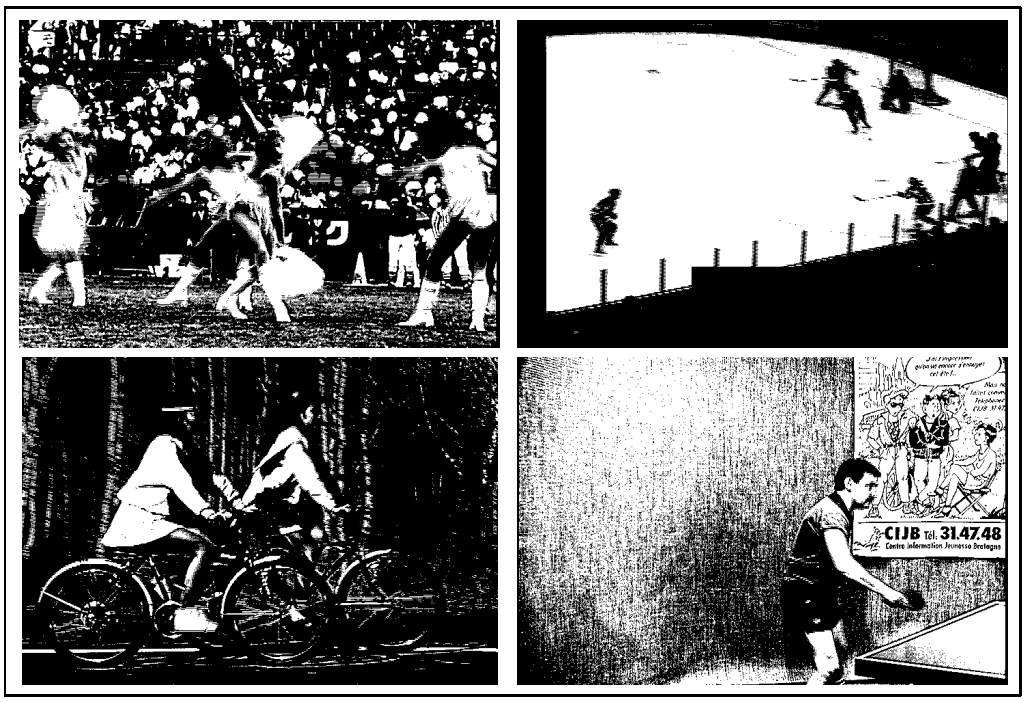

Figure 6.20: Images from the Four Video Sequences Used in the Experiments.

coding (intra-frame or predictive) being performed on it. Intra-frame coded MBs typically contain energy at all frequencies and coarse quantization of low frequencies is more noticeable to human eye than high frequencies. Instead, predictive coded MBs have higher energy at high frequencies and human eye is less sensitive to coarse quantization of predictive coded MBs.

Moreover, human eye is more sensitive to quantization error in flat areas, than very detailed zones [59]. Thus, the actual quantization step size is obtained by multiplying the quantization matrix by a *MB local activity* factor that changes on a MB by MB basis with the goal of keeping constant the visual quality throughout the picture while minimizing the number of bits produced. The way in which the MB local activity factor is calculated is dependent on the implementation of the encoder and is based on the estimation of the "activity" level of the MB [25] [59].

Moreover, a *global distortion level* factor that possibly changes on a picture by picture basis, provides overall control on picture dimension. The product of the MB local activity factor p_{mb} and the global distortion level factor G gives the *quantization parameter* Q_{mb} for MB mb, i.e.,

$$Q_{mb} = p_{mb} \cdot G$$

In summary, quantization is performed according to the following formula:

$$\hat{c}_{i,j}^{mb} = \frac{c_{i,j}^{mb}}{q_{i,j}^{\{I|P\}} \cdot Q_{mb}}$$

where $c_{i,j}^{mb}$ is the (i,j) DCT coefficient of MB mb, $\hat{c}_{i,j}^{mb}$ is its quantized value, Q_{mb} is the for MB mb, and $q_{i,j}^{\{I|P\}}$ is element (i,j) of the quantization matrix used for I-frame or P-frame coding, according to the type of encoding being performed on the current MB.

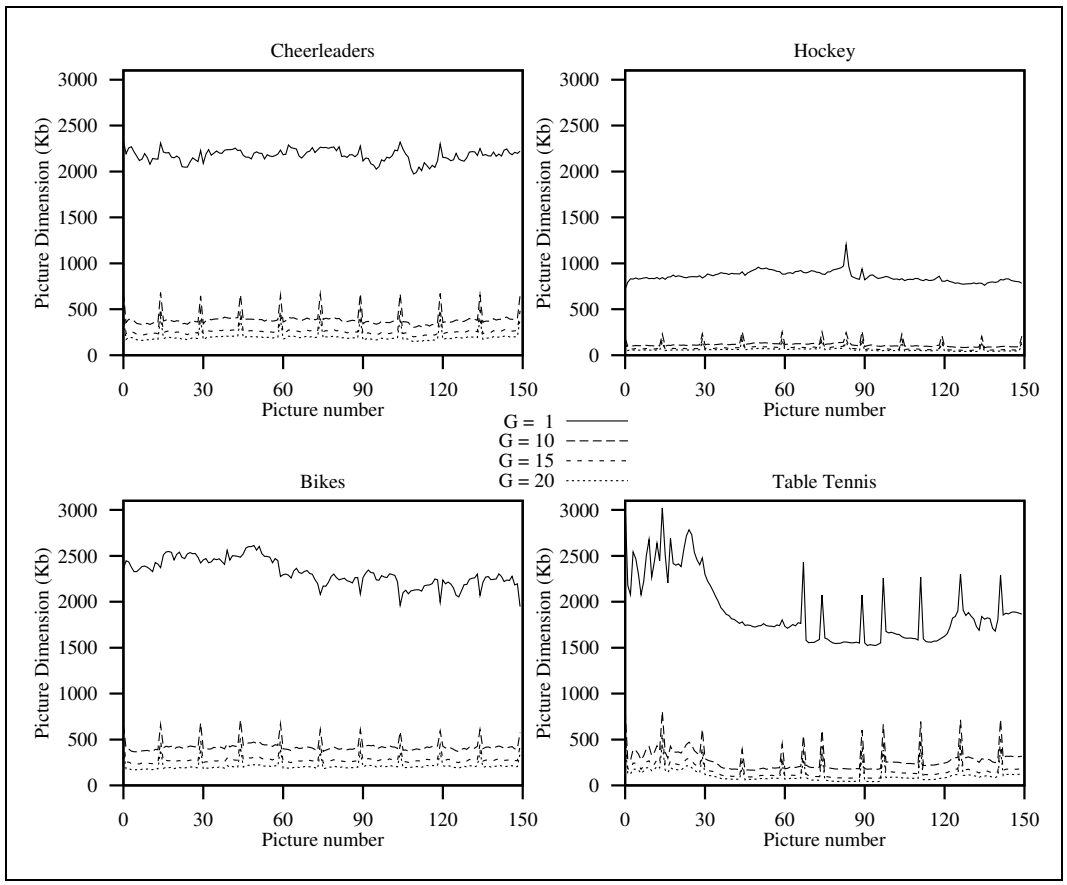

Figure 6.21: Dimension of Pictures Coded according to MPEG Standard.

Four sequences were encoded using four values for the global distortion level G, which was kept constant over the whole video sequence. The dimension

yielded for the pictures is plotted in Figure 6.21: the larger G, the larger the pictures. For small values of G (finer quantization), there is a small difference between the dimension of I-frames and P-frames. In fact, on one side, P-frames dimension grows because the small DCT coefficients of differentially encoded MBs are not reduced to zero by the fine quantization. On the other side the high frequency DCT coefficients of intra-frame coded MBs are coarsely quantized by the corresponding elements of the quantization matrix. This is confirmed by the plots in Figure 6.22 which show that for $G = 1$ the Signal to Noise Ratio (SNR) due to the encoding[5] is smaller for I-frames than for P-frames. Figure 6.23 plots the average quantization parameter used on each picture; even though it changes due to the characteristics of the pictures (i.e., because b_{mb} changes), it is almost constant. The variation is larger for larger values of G because G amplifies b_{mb} variations.

In the following we study the end-to-end delay in a videoconferencing system that uses VBR MPEG compression; the impact of different transmission schemes on the delay is taken into account.

6.4.2 Dedicated Link Between Sender and Receiver

Minimum delay can be obtained if sender and receiver are directly connected by a dedicated link, as shown in Figure 6.24. The encoder processes pictures and produces the bits encoding them as I-frames or as P-frames with the timing shown in Figure 6.25: the time taken to encode a picture is not constant as well as the amount of bits produced.

In principle, if the link speed is higher than the natural instantaneous rate of the encoder[6], bits can be sent over the dedicated link as soon as they are produced. After a propagation delay P, bits get to the decoder which decodes the stream generating pictures in the format suitable for the video adaptor.

The time needed to encode a picture is not constant; also decoding time

[5]The SNR is calculated according to the formula reported in Section A.4. The SNR is used to provide a mathematical measure of the quality of a picture, but it does not exactly correspond to the perceived visual quality because it does not take into account the characteristics of the human visual system. Thus, even though coarse quantization of high frequency components introduces an error, it does not necessarily degrade the perceived visual quality.

[6]This is likely because the peak rate of the encoder is of the order of few tens of Mb/s.

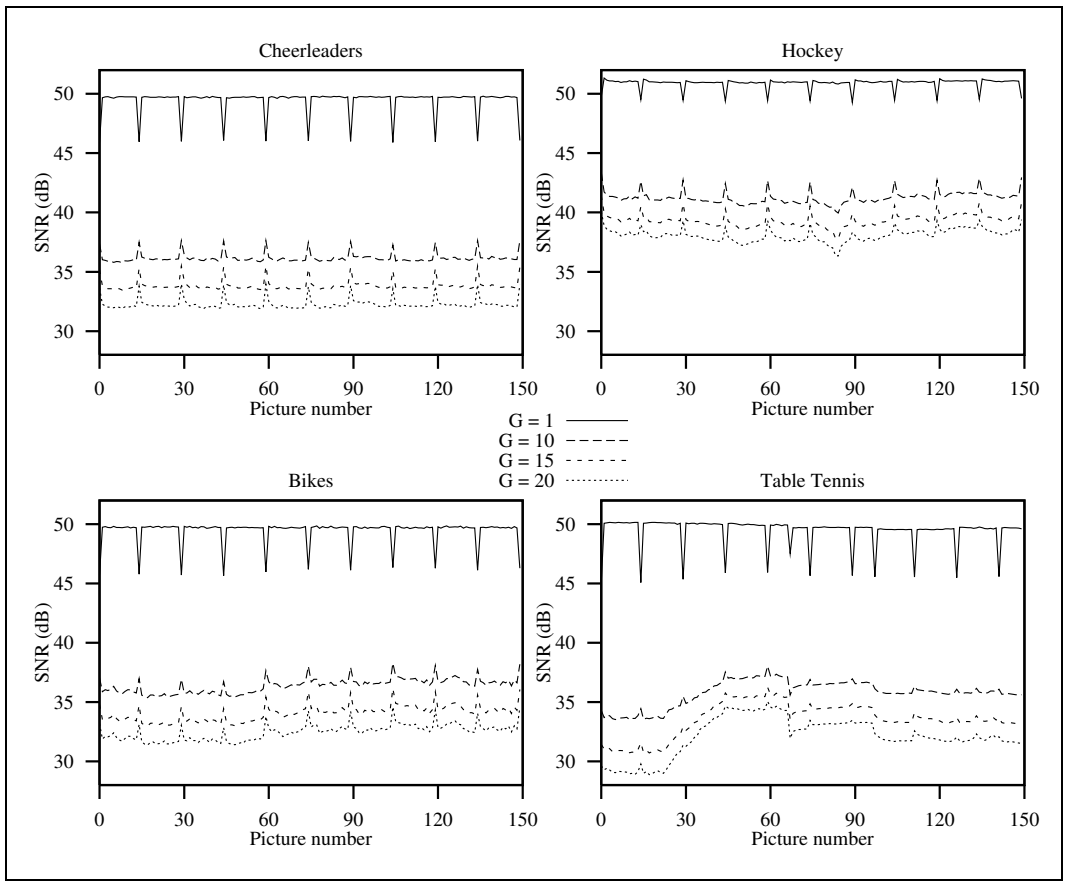

Figure 6.22: SNR of VBR MPEG Encoded Video Sequences.

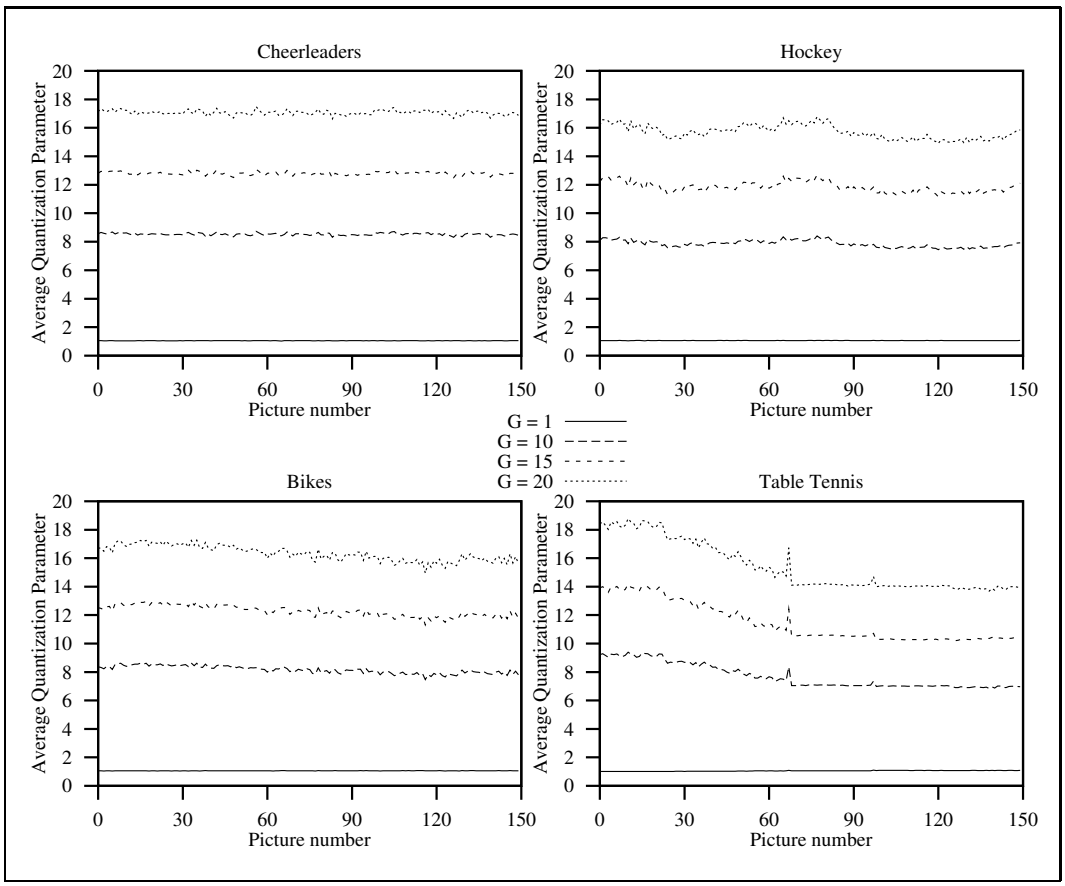

Figure 6.23: Average Quantization Parameter Used for VBR Encoding according to MPEG Standard.

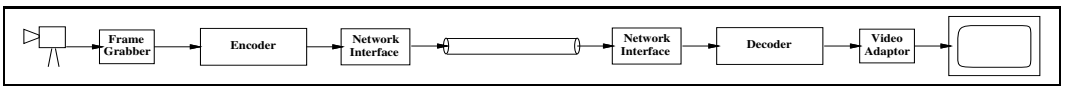

Figure 6.24: Videoconferencing System Configuration with Dedicated Link between Sender and Receiver.

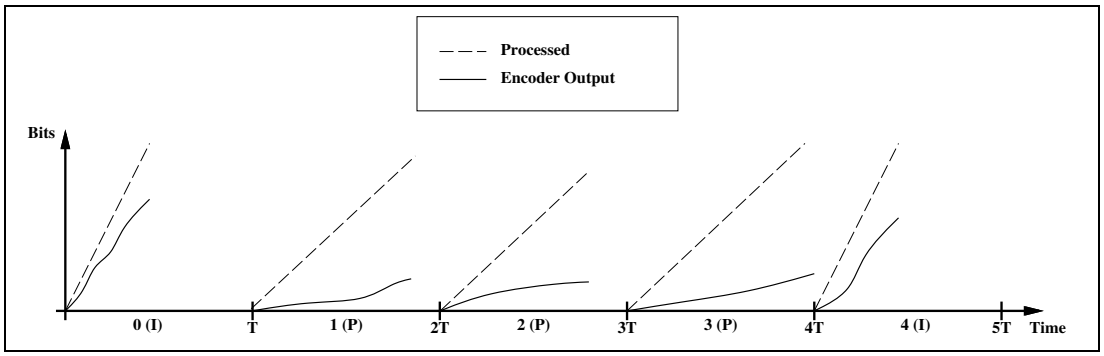

Figure 6.25: Bit Production of a Natural VBR MPEG Encoder.

is not constant even though it is more predictable and its variations are smaller. The decoder keeps constant the delay synchronization between the input of the encoder and its own output (continuous playing) by buffering the encoded or decoded pictures. This compensates for the variability of the encoding and decoding time and introduces the processing resynchronization delay.

The decoding time strongly depends on the decoder implementation: i.e., both on the availability of special purpose hardware, and on how early the decoder starts processing the bits it is receiving. For the purpose of this work we consider constant[7] the *decoding delay* D.

The end-to-end delay is given by

$$\Delta_{VBR}^{Ded} = C_M + P + D + P_d \tag{6.11}$$

where C_M is the maximum *coding delay*; when a frame takes less than C_M to be encoded, it is delayed by the remaining time in the decoder buffer.

The faster the encoder and decoder, the smaller the end-to-end delay; in fact, since we are dealing with real-time video

[7]If the decoding delay is not constant the decoder compensates its variations and (6.11) holds with $\max D$ instead of D.

Requirement 6 *A picture should be encoded within the video frame period,*

i.e., $C_M \leq T$. The maximum encoding delay is usually the video frame period corresponding to the highest frame rate the encoder supports. Thus, if a scene is captured at 30 frames per second and the propagation delay between sender and receiver is 20 ms (e.g., they are connected through a transoceanic link), the end-to-end delay is around 60 ms, i.e., acceptable to guarantee interactivity between the participants in a videoconference.

Motion estimation is the most time consuming part of the encoding process: C_M (and consequently the end-to-end delay) can be reduced by using only intra-encoding (I-frames). The yielded compression is lower and a larger number of bits is needed to encode a video sequence.

The system configuration discussed in this Section provides a lower bound on the end-to-end delay obtained in a videoconferencing system exploiting MPEG compression. Nevertheless, it is not efficient given the expensiveness of bandwidth: in fact, the capacity of the link connecting sender and receiver is only partially used. In the following Sections we study how the end-to-end delay is affected by the usage of multiplexing schemes to more efficiently exploit bandwidth of links.

6.4.3 Packet Switching with Time-Driven Priority

A packet switched network with time-driven priority can be exploited for transmission of encoded pictures, as shown in Figure 6.26. As soon as all the bits encoding a picture are produced by the encoder, they are inserted into a packets and sent at the full speed of the ingress link, as shown in Figure 6.27. The end-to-end delay of the system is given by

$$\Delta_{VBR}^{TDP} = C_M + L \cdot T_f + D + P_d \tag{6.12}$$

where L is the number of TFs a packet takes to travel from sender to receiver and P_d is the presentation delay. For the sake of simplicity, in this work we consider that the processing required for packetization does not introduce delay. Expression (6.12) is the actual end-to-end delay only if the nodes on

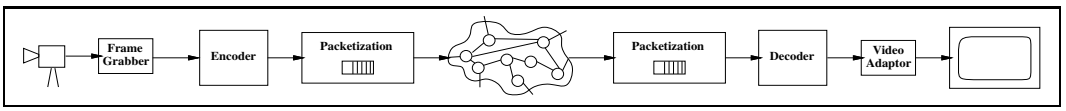

Figure 6.26: Videoconferencing System Configuration with Packet Switched Network between Sender and Receiver.

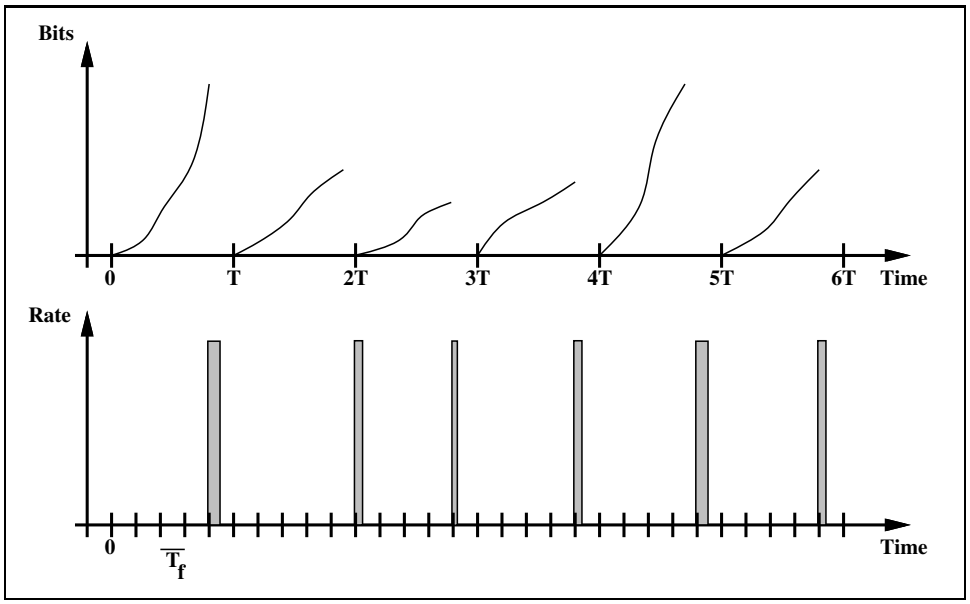

Figure 6.27: Time-Driven Priority without Reservation.

the path from sender to receiver are able to perform RISC-like forwarding of packets (see Section 2.2).

To guarantee the fixed bound $L \cdot T_f$ on the network delay and loss free delivery, resources must be allocated in the network and video frames sent during reserved TFs. To reserve resources in packet switched networks with time-driven priority, the amount of data to be sent and their timing must be known at reservation time so that the proper fraction of link capacity can be reserved during the proper TFs.

The amount of bits reserved should be larger than (and as close as possible to) the dimension of the picture being sent. As was shown in Figure 6.19, the number of bits encoding a picture depends on the type of coding used for the corresponding picture, and the spatial and possibly temporal redundancy present in the image. Since the amount of redundancy depends on the scene being encoded, picture dimension is not known in advance and resource

reservation cannot be accurate. If during a TF a user sends more bits than the amount reserved to it, the network does not provide any guarantee on the delivery of the excess bits. If, on the other hand, the videoconferencing application uses only a fraction of the reserved capacity, the leftover bandwidth can be used by best effort traffic and is not wasted. Even though this is acceptable from the network point of view, the solution is not optimal for the user that is possibly paying for the allocated bandwidth and would like to use it all by himself.

In the following of this Section we show how to determine the amount of bits to be reserved for both types of pictures given the bandwidth to be allocated to a videoconference call. Then, we show how the encoding process can be tuned in order to control picture dimension so that the videoconferencing system never uses more bandwidth than the amount allocated and exploits as much of it as possible. Lastly, we analyze how the synchronization between encoder and network interface, and the scheduling (i.e., the choice of the TFs to be reserved) impact the end-to-end delay. Even though some configurations can deliver unacceptable end-to-end delay, if the system is adequately equipped and operated, its performance is actually given by (6.12).

Choosing a Bound on Picture Dimension

As explained in Section 6.4.1, the slower the motion in the scene being encoded, the larger the dimension of I-frames with respect to P-frames. Since videoconferences are expected to be slow moving scenes, we propose to reserve different amounts of bits for transmission of I-frames and P-frames. These amounts determine the bandwidth reserved to the videoconference call as

$$B^{tdp} = \frac{F^I + (N-1) \cdot F^P}{N \cdot T} \tag{6.13}$$

where F^I and F^P are the amount of bits reserved for I-frames and P-frames, respectively, N is the number of pictures per GOP, and $N \cdot T$ is the GOP duration.

As discussed in Section 6.4.1, the relative dimension of I-frames and P-frames yielded by a natural MPEG encode depends on the amount of motion in the scene. The *picture ratio*

$$\alpha = \frac{F^I}{F^P} \tag{6.14}$$

must then be chosen wisely depending to the amount of motion expected in the scene to be encoded and transmitted. This is in general a difficult task, but in the particular case of videoconferencing application, scenes are likely to be slow.

Combining Equations (6.13) and (6.14), the amount of bits to be reserved to each frame can be expressed as function of the bandwidth to be allocated as

$$\begin{cases} F^P = \frac{B \cdot N \cdot T}{N + \alpha - 1} \\ F^I = \alpha \cdot F^P \end{cases} \tag{6.15}$$

Controlling Dimension of Encoded Pictures

The reserved bandwidth is used more efficiently if the encoding process is controlled and the number of bits encoding each picture is kept below the amount of bits reserved in the TF during which the picture is sent. This amounts are upper bound picture dimension and can be used to devise a target dimension for each type of video frame. As shown in Figure 6.28, the encoder is extended with a *rate control function*. It tunes the parameters of the basic MPEG coding process so that the dimension of each picture is best fitted to the target associated to its type.

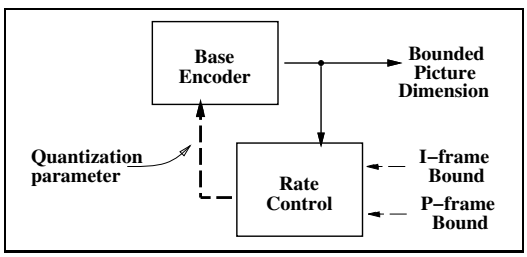

Figure 6.28: MPEG Encoder for Controlling Dimension of Frames.

Among the parameters of the MPEG encoding process the quantization parameter is the most suited to this purpose. For example, Figure 6.21 shows that the dimension of both I-frames and P-frames is strongly dependent on the value of the global distortion level G (which is one of the components of the quantization parameter): the larger, the smaller the encoded pictures.

Nevertheless, G is not suitable for our purpose of controlling picture dimension because its value must be chosen before starting the encoding of the picture. There exist proposals to predict the amount of bits an encoding process is going to produce based on the raw scene [52] or on the portion of stream already produced [32]. However, to get service guarantees from the network, the target must never be exceeded and an adaptive approach based on feedback from the output of the encoder must be exploited.

Other authors propose iterative approaches [63, 37] to determine a suitable value of G. This can lead to coding times not acceptable for real-time encoding as required in videoconferencing applications (Requirement 6). [13] proposes to exploit a rate-quantization model to choose G. The model is tuned according to the characteristics of the encoded stream already produced. The rate control algorithm also proposes how to re-quantize the picture if the yielded dimension is not compliant with the target.

Alternatively, the *MB level rate control* factor b_{mb} is used for calculating the quantization parameter for each MB as

$$Q_{mb} = G \cdot p_{mb} \cdot b_{mb}$$

b_{mb} can be changed on a MB by MB basis in order to adaptively tune the quantization according to the target and the feedback obtained by monitoring the stream being produced [25, 38].

Varying the granularity of quantization throughout the picture delivers non uniform visual quality. Many approaches have been proposed in the literature for uniformly choosing the quantization parameter [28] possibly taking into account the characteristics of the human visual system [59].

The foregoing approaches have been proposed and analyzed in scenarios different from ours. Thus, we have performed some experiments to prove that picture dimension can be controlled as the proposed network requires. The software encoder `dvdenc` has been exploited; it has a rate control function which implements the algorithm shown in Figure 6.29.

The function which calculates the MB level rate control factor b_{mb} is provided with a target F_t and a tolerance on it (in terms of maximum and minimum acceptable dimensions). b_{mb} is incremented (decremented) with respect to b_{mb-1} if the amount of bits F produced so far exceeds (is below) the expected number of bits. This is calculated according to the target F_t and the total

```
according to the type of the picture to be encoded, define:
    a target dimension F_t (e.g., 5 % below reserved number of bits)
    a maximum dimension F_M (reserved number of bits)
    a minimum dimension F_m (e.g., 10 % below reserved number of bits)
b_1 = 1
choose G^{I|P} according to the type (I-frame or P-frame) of coding
F = number of bits encoding MB 1 with quant. par. Q_1 = p_1 · G^{I|P} · b_1
for mb = 2 to total number of MBs Mb
    if F > (F_t/Mb) · mb then
        if F − (F_t/Mb) · mb > F_M − F_t then
            b_mb = MAXb
        else
            b_mb =decr(F,mb,F_M,F_t,Mb)
    else if F < (F_t/Mb) · mb then
        if (F_t/Mb) · mb − F > F_t − F_m then
            b_mb = MINb
        else
            b_mb =incr(F,mb,F_m,F_t,Mb)
    else
        b_mb = b_{mb−1}
    Q_mb = p_mb · G^{I|P} · b_mb
    F = F+ number of bits encoding MB mb with Q_mb
G^{I|P} = (G^{I|P}/Mb) · Σ_{mb=1}^{Mb} b_mb according to the type of the encoded picture
```

Figure 6.29: Video Frame Dimension Control Algorithm.

number of MBs Mb. The increment (decrement) is proportional to the excess (lack) of bits and the tolerance on the target. The average of the MB level control factor over the whole picture is used to calculate the global distortion level for the next picture of the same type. The experimental data shown in Figures 6.30 - 6.35 are obtained using $G = 15$ for both the first I-frame and the first P-frame. This algorithm is not wise from the point of view of yielding uniform visual quality throughout pictures. Nevertheless, the objective here is to show that the dimension of frames can be kept below (and very close to) a predefined bound, not to devise an optimal method to control the dimension of frames.

Figures 6.30 and 6.31 plot the picture dimension obtained encoding four different scenes with the shown bounds (F_M in the algorithm in Figure 6.29): the bounds are never exceeded. The bounds are determined according to Equation (6.15) assuming three bandwidth amounts and picture ratios (one in each row of the Figures).

As shown by Figures 6.32 and 6.33, the average quantization parameter has

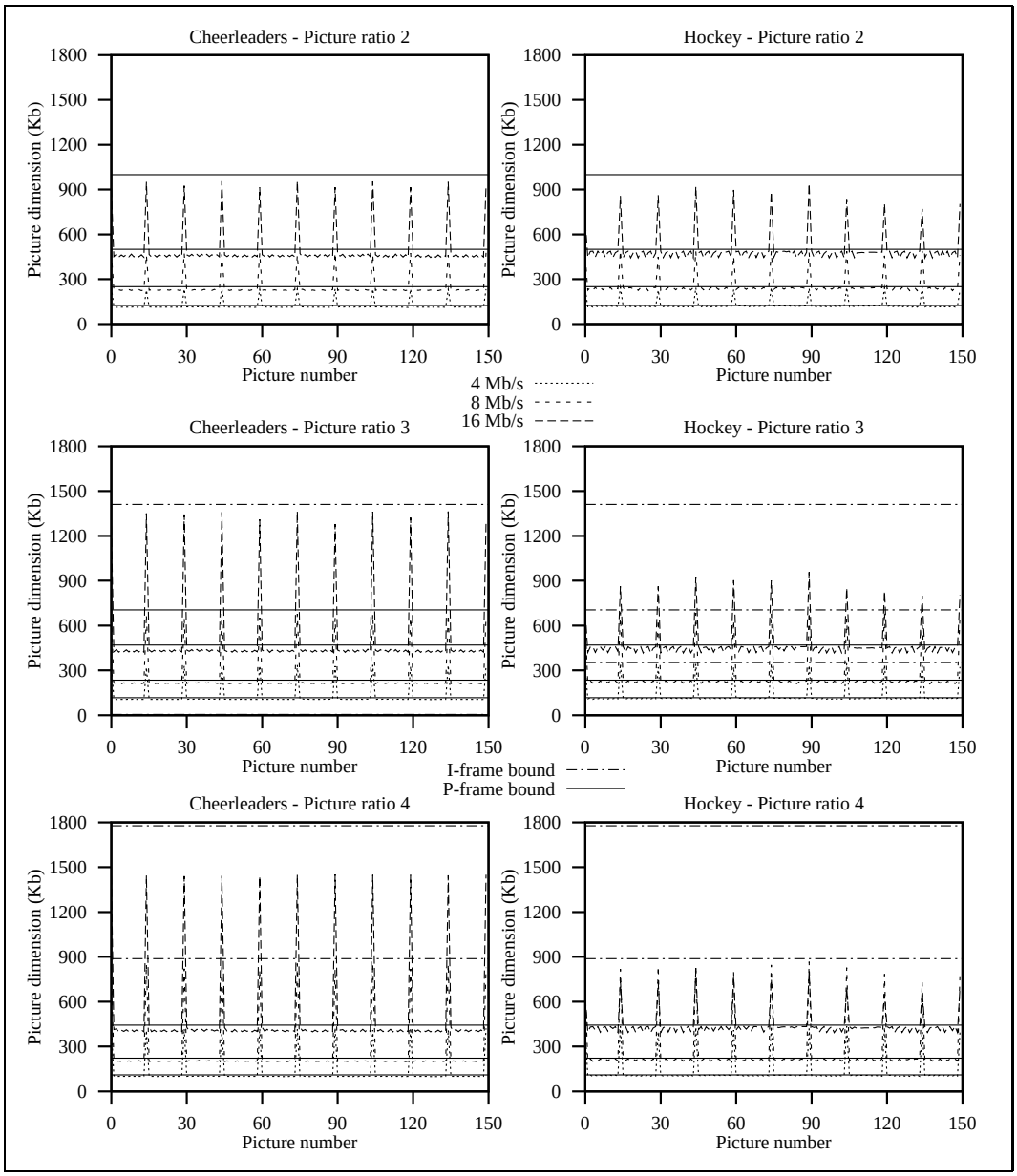

Figure 6.30: Dimension of Pictures in Controlled MPEG Encoding.

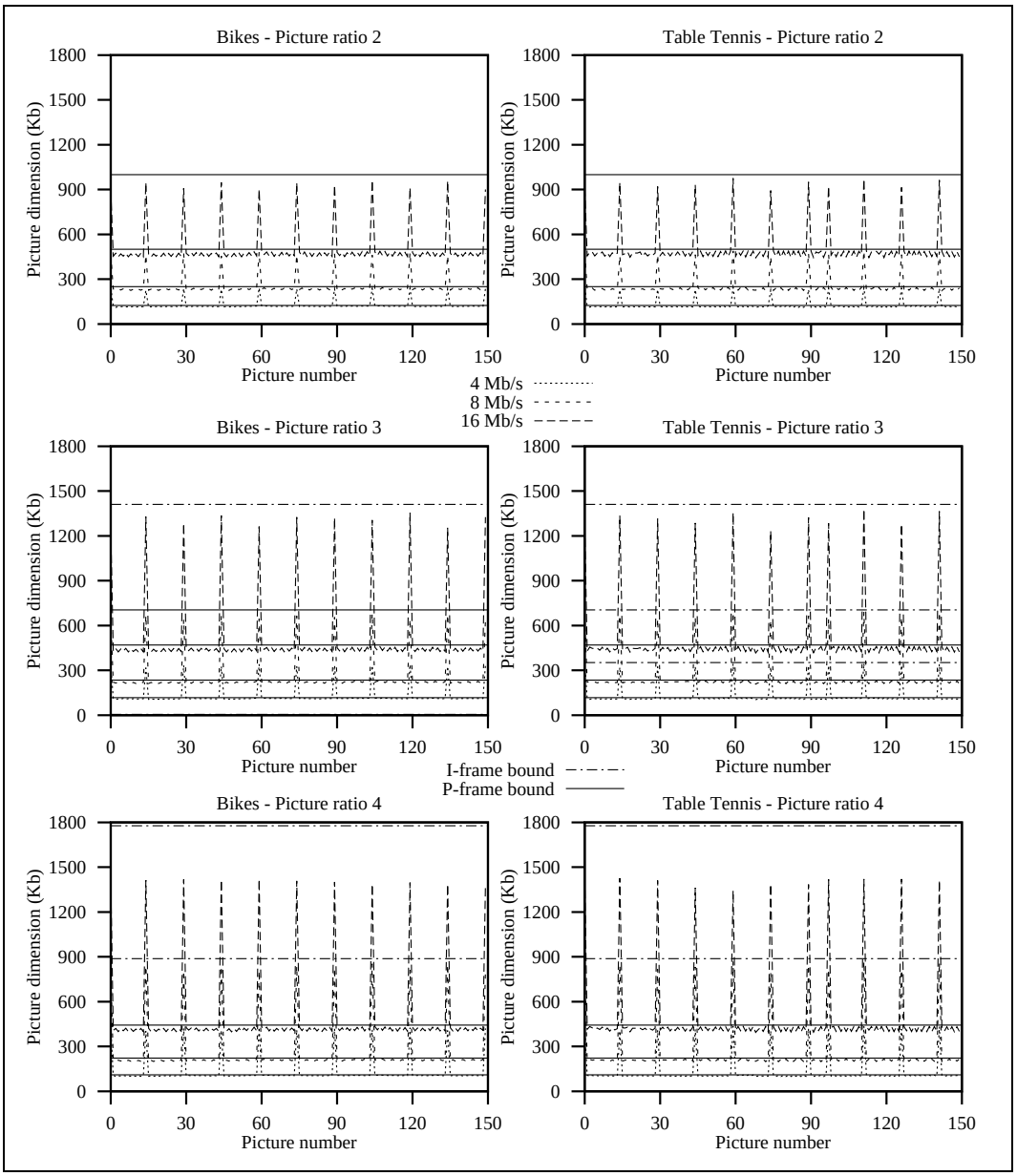

Figure 6.31: Dimension of Pictures in Controlled MPEG Encoding.

high variation for two complementary reasons:

1. The variation of picture dimension is limited by the parameters provided to the rate control function in the experiments: the target (F_t in the algorithm shown in Figure 6.29) has been set 5% below the upper bound (F_M) and the lower bound of picture dimension (F_m) 10% smaller than the target.

2. The rate control function does not have the objective of delivering uniform quality throughout the scene.

As a result the quality of pictures is not uniform throughout each scene, as shown by the SNR plotted in Figures 6.34 and 6.35. The rate function control is not designed to keep the quality uniform inside pictures as well. As a consequence, the MB level rate control factor has high variability within each frame.

The scenes used in these experiments have pictures more detailed and a higher amount of motion than typical videoconferencing head and shoulder scenes. The same simple control function would deliver more uniform quality when dealing with with typical videoconferencing scenes. Moreover, enhancing the rate control function to reduce the quality variation in this scenario is not hard.

Having the rate control function bounding picture dimension, bandwidth is efficiently allocated by reserving fixed amount of bits for the transmission of pictures of the same type. The amount of bits is calculated using Equation (6.15). In the following we analyze how the choice of the TFs in which the bits are reserved affects the end-to-end delay; the conditions under which it can be reduced to Expression (6.12) are also presented.

Synchronization between Encoder and Network

save Scheduling, i.e., the choice of the TFs to be reserved to a videoconference call, is simplified by considering the nature of the application generating traffic. According to Requirement 6, videoconferencing systems must exploit real-time encoders that guarantee the delivery of a picture every T seconds.

To better understand the implications of scheduling on the end-to-end delay, we first consider that only intra-frame coding is performed. Under this

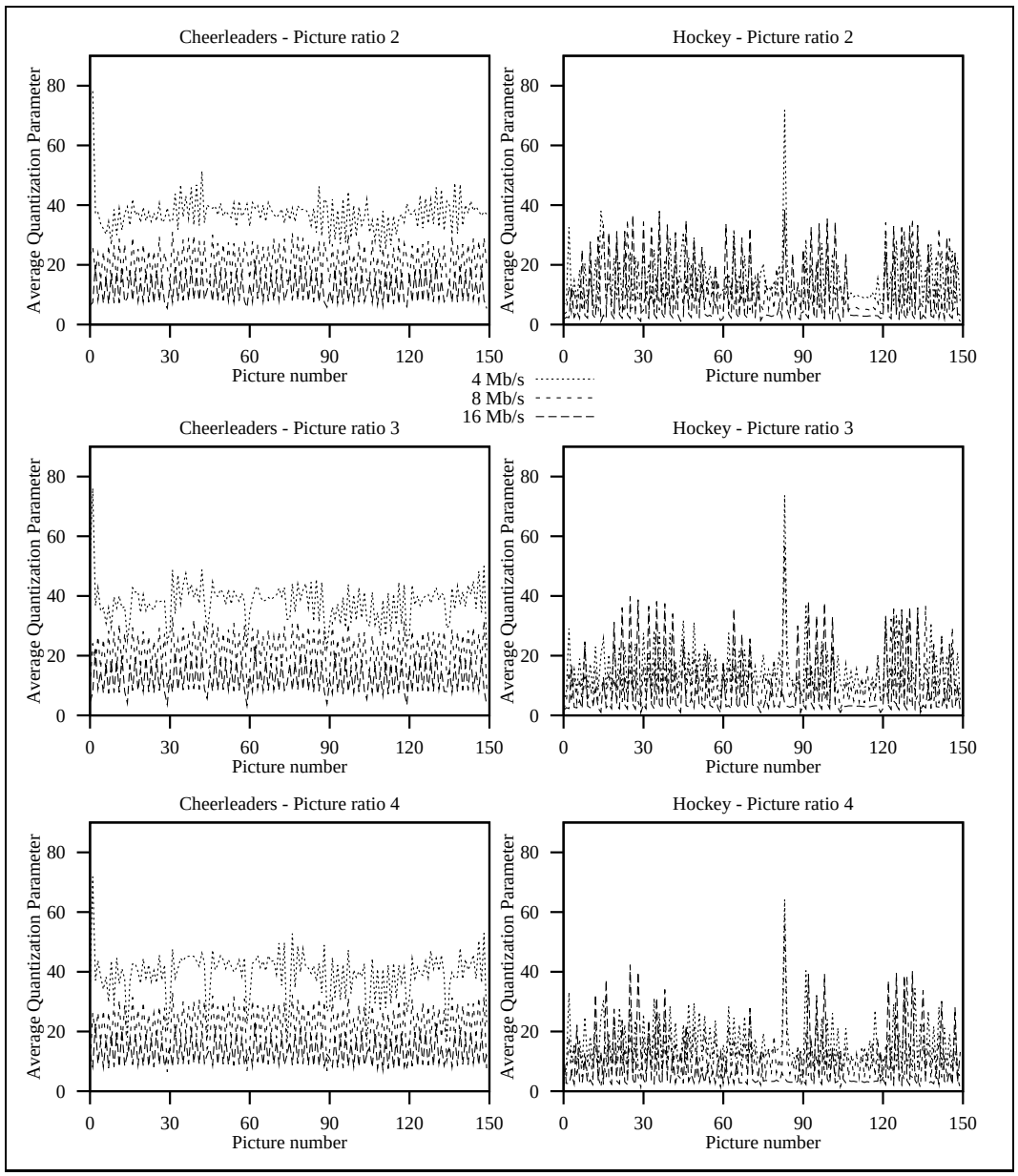

Figure 6.32: Average Quantization Parameter in Controlled MPEG Encoding.

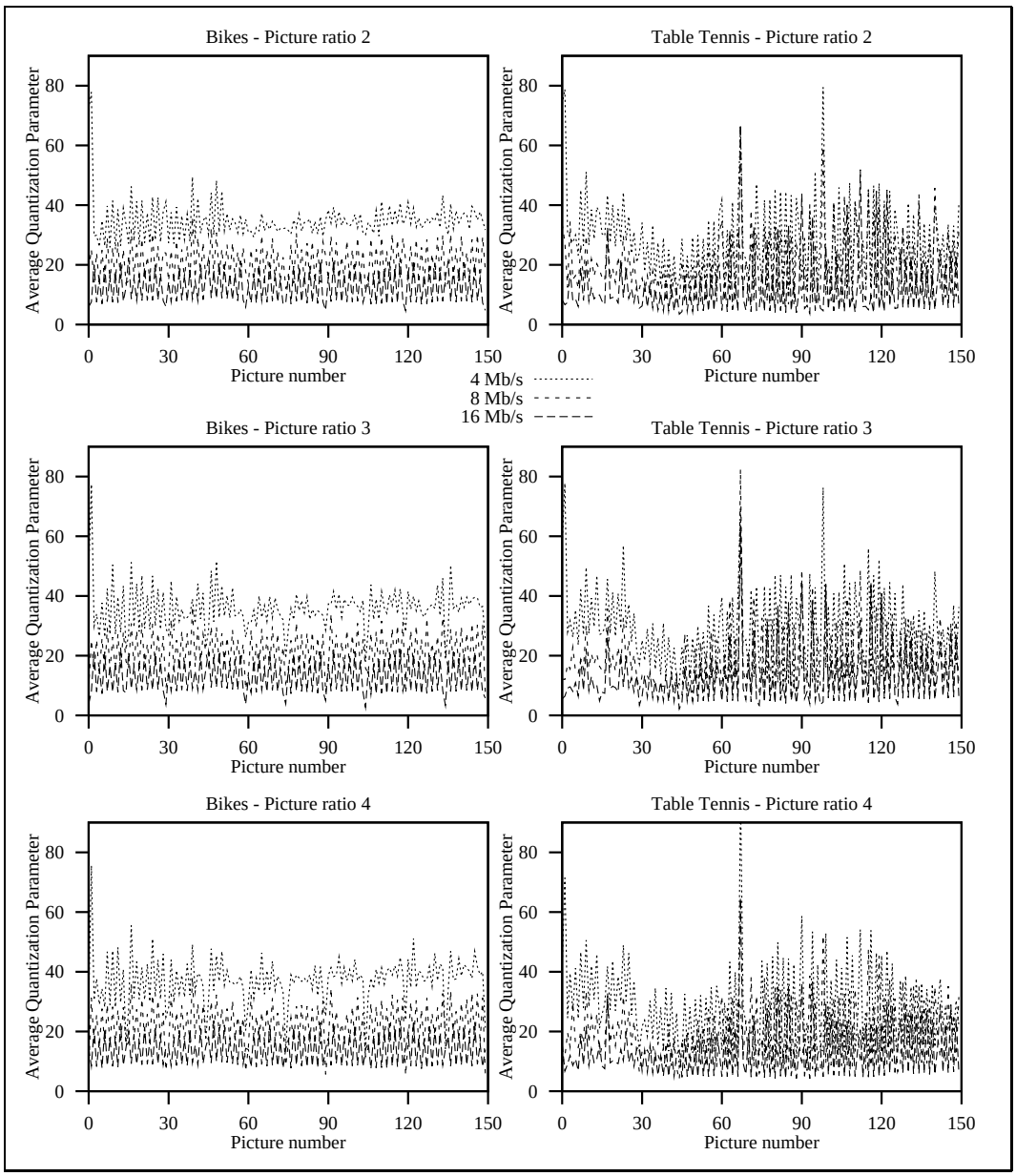

Figure 6.33: Average Quantization Parameter in Controlled MPEG Encoding.

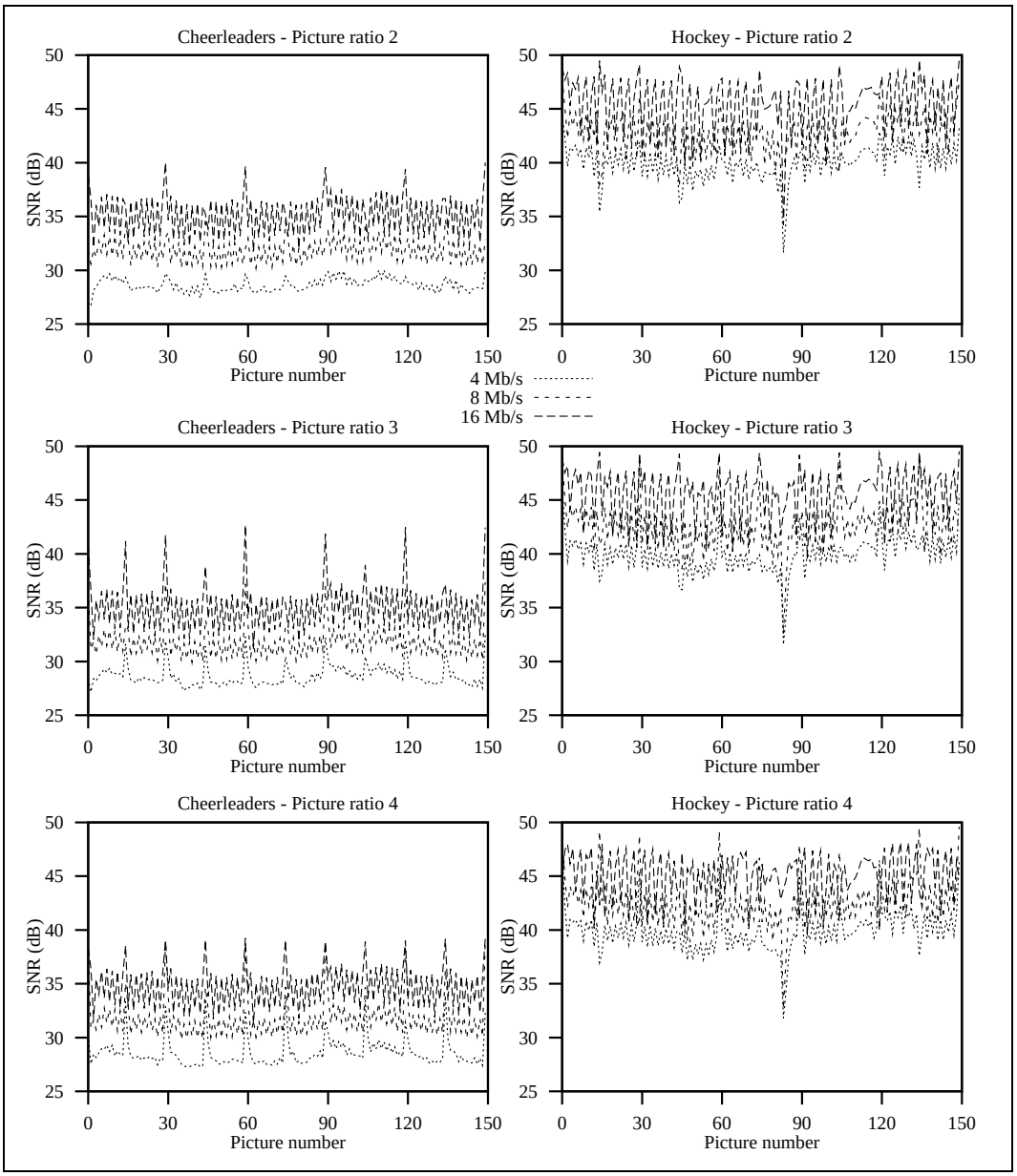

Figure 6.34: SNR in Controlled MPEG Encoding.

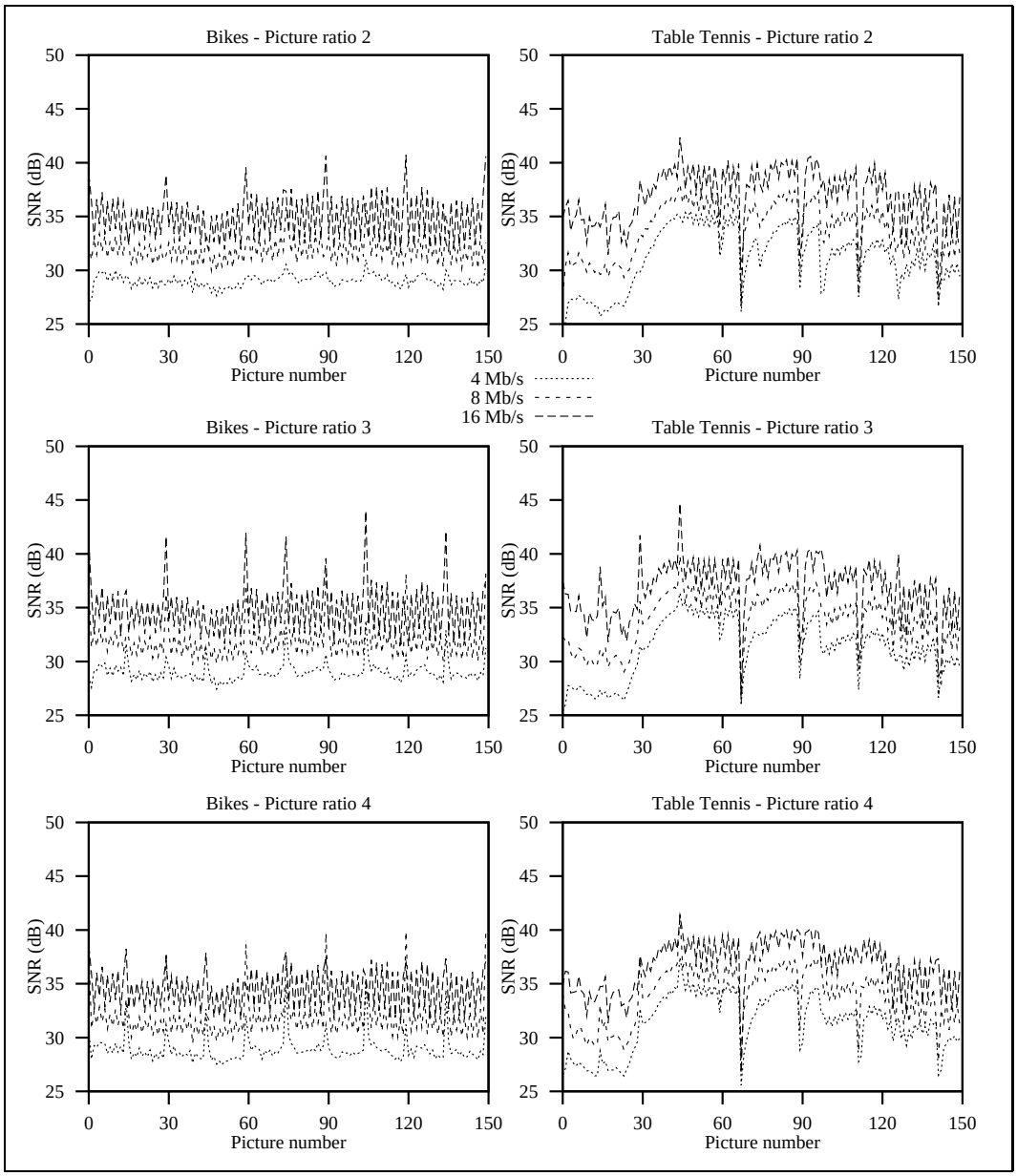

Figure 6.35: SNR in Controlled MPEG Encoding.

assumption, resource reservation is performed as described in Section 6.3.3 for transmission of raw video. The time cycle duration must be an integer multiple M of the video frame period; for the sake of simplicity, we now consider $M = 1$ keeping in mind that the what is described in Section 6.3.4 applies to the general case. In each time cycle, F^I bits are allocated during one TF.

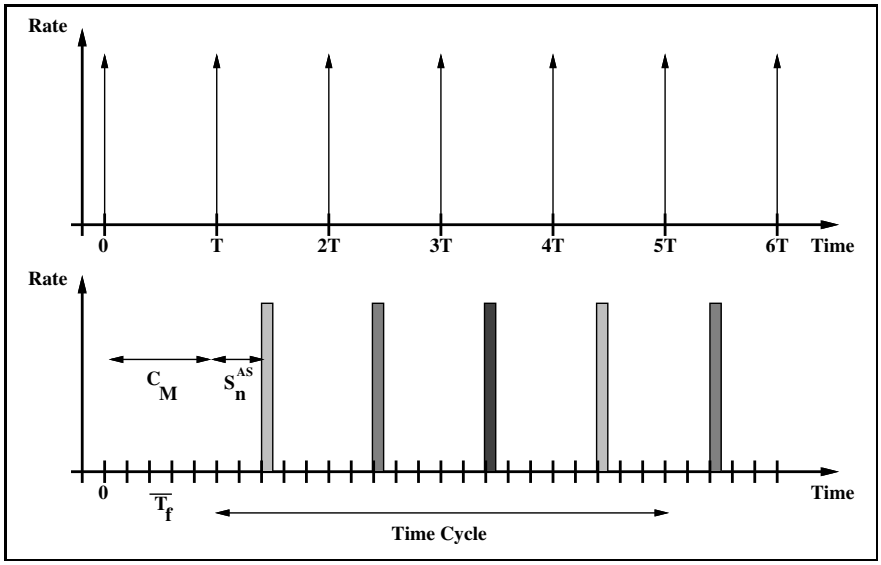

Figure 6.36: I-frame Only Coding and Bandwidth Reservation with Time Driven Priority

The bits encoding a picture are buffered from when they are produced to when the TF reserved for their transmission is scheduled. This buffering can take place, for example, in the packetization function. Temporarily assuming that encoding each picture takes the maximum coding time C_M, each picture spends in the packetization function buffer the application synchronization component of the network shaping delay $S_n^{AS} \in [0,T]$, as shown in Figure 6.36. As explained in Section 6.3.4, if the clock driving the capture card (and hence the pace at which encoded pictures exit the encoder) is not synchronized with the network interface, S_n^{AS} slowly varies in the given interval. If the two capture card and network interface are synchronized, S_n^{AS} is constant and the time in which pictures are captured can be chosen so that $S_n^{AS} = 0$. Otherwise, the end-to-end delay is given by

$$\Delta_{VBR-I}^{TDP} = C_M + S_n^{AS} + L \cdot T_f + D + P_d \tag{6.16}$$

If the encoding time of a picture is less than C_M, it is buffered by the packetization function longer than S_n^{AS}. This does not affect the end-to-end delay because if the picture was sent as soon as encoded, it would be delayed by the decoder to compensate the processing delay variation. Thus, this configuration allows to simplify the receiver that does not need to compensate network jitter and processing delay variation.

Complex Scheduling

When both intra-frame and predictive coding are exploited, two different bounds are imposed for the two different types of frames: different amounts of bits should be reserved in the TFs intended for sending I-frames and those for P-frames. The time cycle must be set to an integer multiple M of the GOP period and $N \cdot M$ TFs must be reserved within the time cycle. Figure 6.37 shows a sample reservation with $M = 1$ and $N = 4$; the upper diagram shows the amount of bits generated to encode each picture assuming that they are instantly available after the maximum encoding delay C_M from the capture[8]. The lower diagram shows the amount of bits reserved in the TFs: F^I in one TF and F^P in the following $N - 1$ TFs. The choice of the TFs to be reserved on each link on the path between sender and receiver is called *complex scheduling*. The choice of the TFs impacts both network performance (in terms of maximum number of real-time connections concurrently supported) and the end-to-end delay of the videoconference call.

Assuming uniform distribution within the time cycle of the reserved TFs, the end-to-end delay is given by (6.16) as in the case in which only intra-frame coding is exploited. Nevertheless, in this case the application synchronization component of the network shaping delay has a different variation interval; for $M = 1$, $S_n^{AS} \in [0, N \cdot T]$. A network shaping delay of the order of the GOP period is definitely not acceptable especially in videoconferencing applications where, due to the slow motion in the video sequence, it is convenient to use a long GOP (large N). Thus, synchronization between video encoder and network interface is essential to keep the end-to-end delay below the 100 ms bound and the buffer of the packetization function small.

Assumption 3 *The capture card (as well as the encoder) is synchronized*

[8]It has been already shown in the case of intra-frame only coding that this assumption does not constitute a limit of the system, but instead allows the decoder to be simplified.

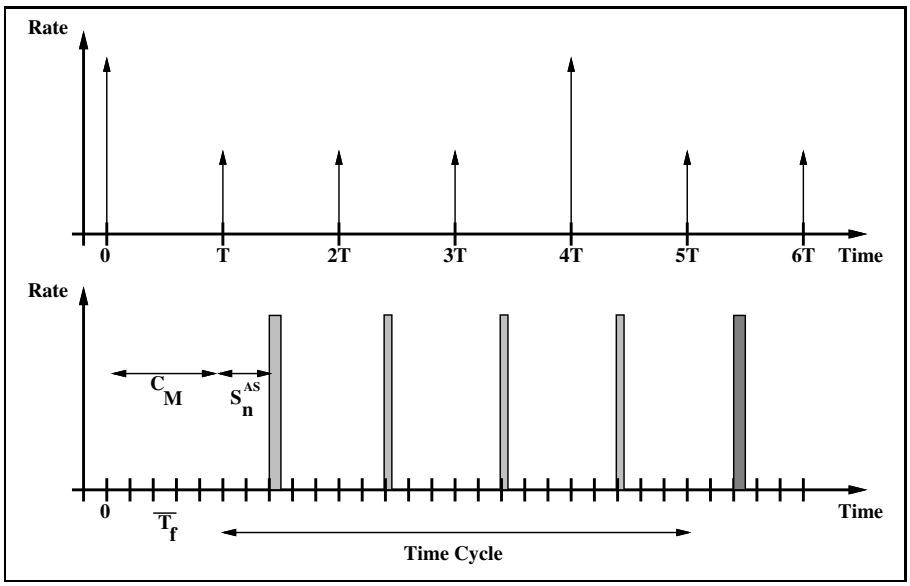

Figure 6.37: Time-Driven Priority and Complex Scheduling.

with the network interface.

If the TFs reserved to a videoconference call are not uniformly distributed within the time cycle the inter-arrival time between video frames is not constant. The receiver compensate to this variability by buffering video frames received before schedule and delaying the whole video stream. Since this delay is due to the adaptation of the transmission time to the characteristics of the network, we categorize this delay as a *scheduling component* of the network shaping delay S_n^{Sched}. If Assumption 3 holds, the end-to-end delay can be written as

$$\Delta_{VBR}^{TDP-CxSc} = C_M + S_n^{Sched} + L \cdot T_f + D + P_d$$

where $S_n^{Sched} \in [0,N \cdot T]$: it is fixed when the videoconference call is placed and is kept small by a wise scheduling in the network.

Reducing Decoding Time

The decoding time D can be reduced if the decoder does not have to wait the whole picture before starting processing it. This can be obtained by inserting encoded video frames in smaller packets that are sent as soon as

the corresponding bits have been produced by the encoder. This does not affect any other component of the end-to-end delay given by (6.16).

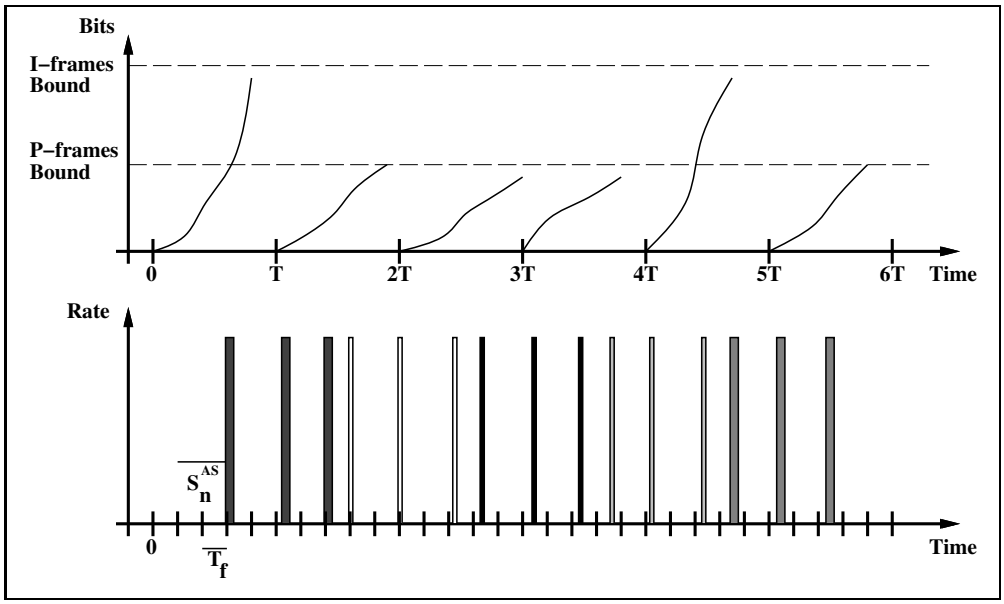

Figure 6.38: Time-Driven Priority and Complex Scheduling.

It is thus better to allocate more than one TF per picture and send a packet in each of them (Figure 6.38), than to allocate a single TF during which a whole encoded picture is sent (Figure 6.37). Choosing this TFs encompasses two non trivial issues: determining their position inside the time cycle and their number so that the end-to-end delay be minimized. In the following these problems are described and the trade-offs between different choices are outlined, but a solution is not proposed because it is out of the scope of this work.

When a reserved TF is scheduled on the sender link, enough bits must have been produced by the encoder, otherwise the amount of transmitted data is smaller than the one reserved. This is called *encoder underflow* and it critical not just because the allocated bandwidth is underutilized (it can be exploited by best effort traffic), but because, if the total amount of bits actually produced to encode the picture is very close to the global amount of bits reserved for the picture over all the TFs, the remaining reserved TFs will not have enough capacity to carry the other bits that has to be sent in the scheduled TF. On the other hand, if the TFs reserved for a picture are chosen later (with respect to the beginning of the video frame period), the benefit

of using more TFs to transmit a picture is reduced, i.e., the end-to-end delay is increased.

The software encoder that has been used in this work shows the timing plotted in Figure 6.39 and Figure 6.40 when encoding frames with bounded dimension. The curves show that the production rate is quite constant, i.e., the position of TFs can be estimated with a simple linear model which introduces some safety margin (by shifting the TFs towards the end of the video frame period). Figures 6.39 and 6.40 show that a safety margin of few ms yields low probability of encoder underflow. Moreover, a videoconference scene has usually more uniform image complexity and motion, than the scenes used in these experiments, thus showing a more regular production rate.

If the instantaneous amount of bits produced by the encoder is below the amount predicted by the model, the bits that cannot be fit in the TFs reserved for the picture can be sent as best effort; if the network is congested, they do not timely get to the receiver and the visual quality of the reconstructed scene temporarily degrades.

The number of the reserved TFs per picture should be kept as high as possible because the smaller the transmission unit, the earlier the decoder can start decoding. Nevertheless, the higher the number of reserved TFs, the higher the probability of encoder underflow (the model averages the real production rate on a shorter interval). Moreover, even though in this work packet header overhead is being neglected, it has to be taken into account when choosing the dimension of the transmission unit.

The amount of bits to be reserved in each TF must be chosen finding a trade-off between the reduction in the decoding delay, on one hand, and the possibility of encoder underflow and the transmission overhead, on the other hand.

6.4.4 Asynchronous Packet Switching

A videoconferencing system can be constructed over an asynchronous packet switched network. As soon as the encoder produces enough bits to assemble a packet of dimension P_s, the packet is sent into the network where it is experience a variable queuing delay. As shown in Figure 6.41, the receiver must exploit a replay buffer to compensate the network delay variation. The end to end delay is thus given by

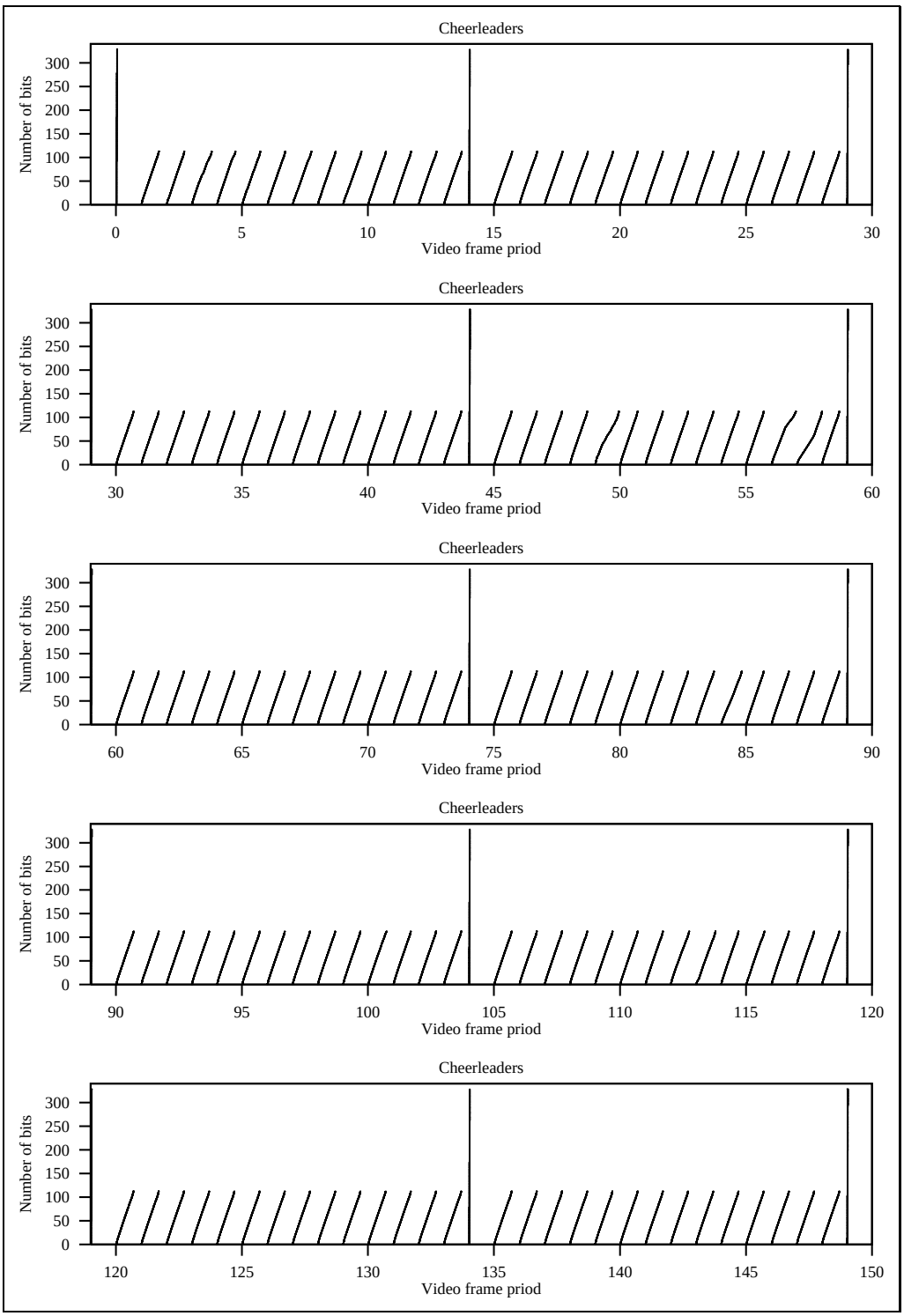

Figure 6.39: Production of Bits when Encoding the "Cheerleaders" Sequence.

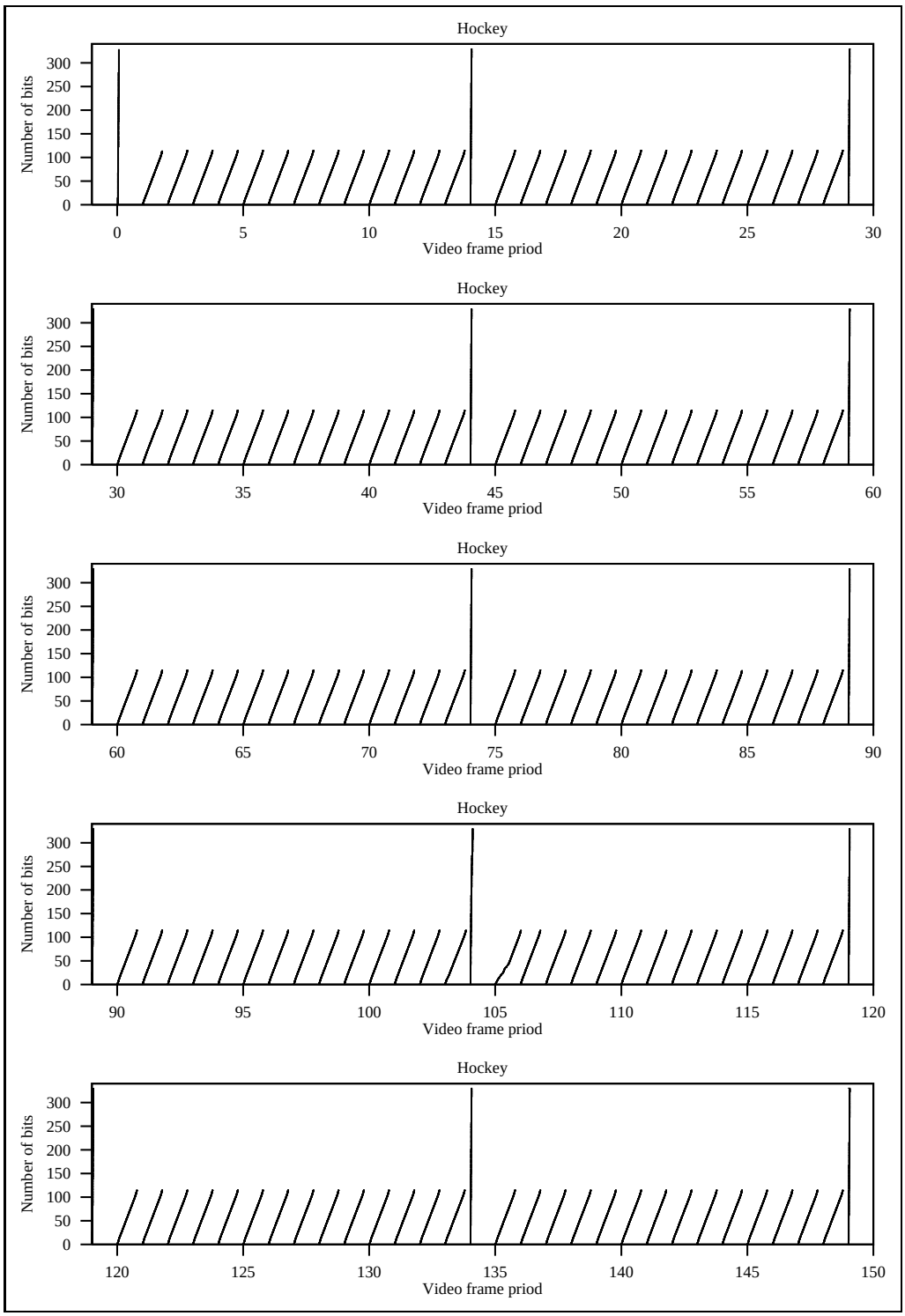

Figure 6.40: Production of Bits when Encoding the "Hockey" Sequence.

$$\Delta_{VBR}^{Async} = C_M + \frac{P_s}{C} + P + Q_M + E_r + D + P_d$$

where C_M is the maximum time required to encode a picture, Q_M is the maximum queuing delay, $E_r \in [0,\Delta Q]$ is the excess resynchronization delay introduced by the replay buffer, and D is the decoding delay. Using small packets and a decoder which starts decoding video frames as soon as data are received, can reduce D. Anyway, the end-to-end delay is dominated by C_M and Q_M.

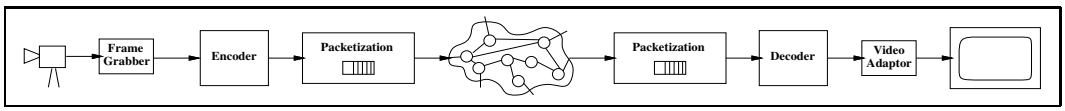

Figure 6.41: Videoconferencing System Configuration with Packet Switched Network between Sender and Receiver.

Traffic Shaping at Network Boundaries

Resources are reserved in the network in order to bound the queuing delay. As discussed in Section 6.3.5, resource reservation is more efficient if traffic shaping is performed at the boundaries of the network, even though it introduces a network shaping delay. Resources are reserved in the network based on the traffic description, given in terms of burstiness and average rate, corresponding to the shaped traffic; the network guarantees the quality on the service (i.e., the bound Q_M on the queuing delay) only if the actual traffic is compliant with the description given at resource reservation time.

The delay globally experienced by a picture due to the traffic shaper depends on the natural bit generation rate of the encoder, the implementation of the traffic shaper, and the characteristics of the shaped traffic. The receiver has to compensate it by means of a replay buffer which introduces a network resynchronization delay. Thus, each packet experiences a network shaping delay S_n^{TS} partly in the traffic shaper and partly in the replay buffer. In Section 6.3.5, we have discussed the constraints on the choice of the leaky bucket parameters and the impact on the end-to-end delay in the case of transmission of raw video. Since in the case of VBR MPEG encoded video the same task is harder and it is not the goal of this work, we do not analyze

S_n^{TS} in more detail. The end-to-end delay of the videoconferencing system is given by

$$\Delta_{VBR}^{Async-TS} = C_M + S_n^{TS} + \frac{P_s}{C} + P + Q_M + E_r + D + P_d$$

Adapting the Encoded Video Stream to the Network

Since the traffic pattern generated by a natural VBR MPEG encoder is not known in advance, it can be incompatible with the shaped traffic description. The non compliant packets can be discarded by either the traffic shaper itself, or a traffic policing function inside the network [50]. For example, if a leaky bucket is exploited to shape the traffic, the token generation rate B and token pool size A determine the burstiness and average rate of the shaped traffic. If the characteristics of the encoded video are not compatible with the values chosen for B and A the excess traffic must be either discarded or sent in the network with "best effort" service. Even though a buffer is inserted before the leaky bucket to adapt the video stream to the traffic description, it can overflow if the two are too different.

The loss of packets is not acceptable in the transmission of MPEG encoded video, especially when the GOP is large. Even though techniques have been proposed to limit the effect of loss [15], it should be better for the videoconferencing system to avoid loss in order to deliver the highest possible quality.

The MPEG encoding process can be controlled to avoid that the traffic shaper discards or sends as best effort traffic packets that cannot be adapted to the traffic description. A rate control function tunes the parameters of the basic MPEG encoder according to the traffic description used to drive the traffic shaper [57, 56]. Due to the unpredictable output of MPEG encoders, this approach does not guarantee against packets not compliant with the traffic description.

Alternatively, the rate control function can tune the MPEG encoder parameters based on feedback information received from the traffic shaper (e.g., the fullness of the buffer preceding a leaky bucket) [48, 12], as shown in Figure 6.42. If this requires to significantly degrade the visual quality of pictures, the resource allocation is renegotiated according to a rate-quantization model that is tuned as the encoding progresses.

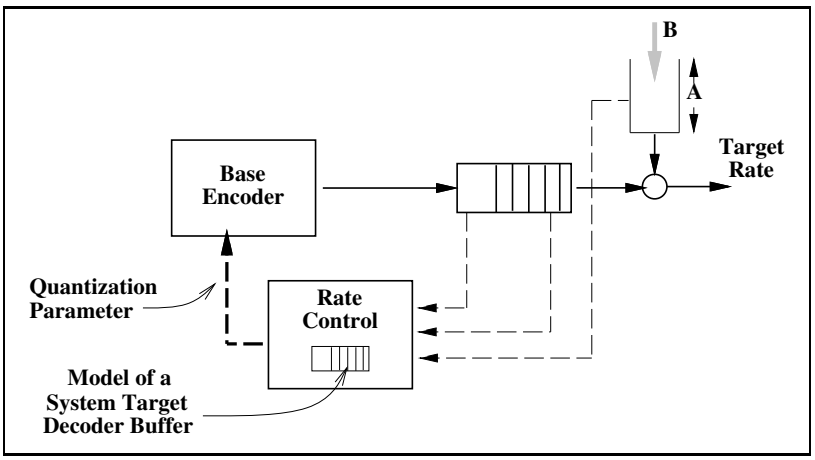

Figure 6.42: MPEG Encoder Controlled using Feedback from a Traffic Shaper.

Hierarchical or *layered encoding* [45] provides another means to adapt the encoded video stream to the service provided by the network. The bits encoding each frame are divided into two separate layers, the high priority layer - which contains the most critical data according to the sensitivity of the human visual system - and the low priority layer. The former must comply to the traffic description because it needs service guarantees, while the latter is sent as best effort. Even though the low priority component gets lost, the video sequence can be reconstructed on the receiver with acceptable quality. The encoded stream can be partitioned dynamically according to feedback information from the traffic shaper (e.g., the fullness of the token bucket) [39]. The fraction of data sent at the high priority layer to better fit the corresponding stream better to the traffic description.

Hierarchical encoding can get the best from a packet switched network by using all the available bandwidth when the network is not overloaded, and getting reasonable visual quality when it is congested. On the contrary, the approach based on controlling the rate of the MPEG encoder varies the quality of images with the objective of fitting the encoded video stream into the traffic description. Thus, visual quality can sometimes be decreased even though the network could possibly support a high rate (i.e., higher quality).

Nevertheless, the hierarchical encoding approach is not efficient because the low priority stream puts a harmful burden on the network when it is overloaded. During congestion, sending low priority packets does not yield any

advantage since they do not get to the destination. Even worse, low priority packets waste resources until they are discarded. The waste is even larger if packets make their way to the destination and there they are discarded due to excessive delay[9].

6.4.5 Circuit Switching

A circuit switched network can be used for the transmission of VBR MPEG encoded video. If the videoconference call is allocated a circuit with bandwidth larger than the instantaneous rate of the encoder and the end-to-end delay is

$$\Delta_{VBR}^{CS} = C_M + Sw + P + D + P_d,$$

where Sw is the switching delay and P the propagation delay in the network. This expression is similar to (6.11) given in Section 6.4.2 for the end-to-end delay of a videoconferencing system that uses VBR MPEG encoding and a direct link between sender and receiver. When a circuit switched network is exploited the switching delay has to be added; moreover, not all the capacity C of links is dedicated to the videoconference call.

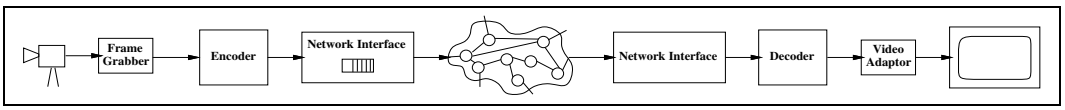

Figure 6.43: Videoconferencing System Transmitting VBR MPEG video over a Circuit Switched Network.

Since the circuit has been dimensioned to support the burstiness on the VBR MPEG encoder, a large part of its bandwidth is wasted. The bandwidth of the circuit can be reduced below the peak rate of the MPEG stream if a buffer is used between encoder and network, as shown in Figure 6.43. It smoothes the rate of the encoder to fit it in the circuit bandwidth at the expenses of a network shaping delay S_n^{CS} added to the end-to-end delay. What is discussed in Section 6.4.4 about the network shaping delay S_n^{TS} introduced by a traffic shaper when using packet switching, applies also to

[9]Take into account that the delay experienced by best effort traffic has a large bound and high variability.

S_n^{CS}. In fact, the shaping performed by the smoothing buffer and the circuit switched network is equivalent to the one due to a traffic shaper with average bandwidth B and maximum burstiness 1 bit (e.g., a leaky bucket with token generation rate B and token pool size $A = 1$). Pictures are buffered in the sender for a time that depends on the rate of the MPEG encoder and the bandwidth of the circuit. A replay buffer on the receiver side compensates the variation of this time thus yielding the fixed contribution S_n^{CS} to the end-to-end delay which is given by

$$\Delta_{VBR}^{CS-TS} = C_M + S_n^{CS} + P + Sw + D + P_d$$

The buffer between encoder and network has the same purpose as the buffer at the ingress of a leaky bucket, which has been described in Section 6.3.5; when it overflows part of the encoded stream is discarded. Dimensioning the buffer and the circuit bandwidth so that discarding of packets is avoided is not possible because the bit rate produced by the MPEG encoder is not known in advance. Thus, the bit generation rate of the encoder must be monitored and regulated by a rate control function which exploits the buffer status as feedback information. The system composed by the basic MPEG encoder, the buffer, and the rate control function is actually a CBR MPEG encoder which is the subject of the next Section.

6.5 Transmission of CBR MPEG Video

The basic MPEG coding process produces a variable bit rate stream. As has been shown in the previous Section, to prevent loss, the natural variability of the bit generation rate must be controlled and adapted to the transmission service. If the transmission service requires data to be sent at Constant Bit Rate (CBR), a CBR MPEG encoder is the choice for building the videoconferencing system.

A CBR MPEG encoder/decoder system introduces a delay that is independent of the transmission scheme adopted and widely varies according to the parameters of the encoder. In the following, the time needed to encode and transmit CBR MPEG video is studied for further steps with the aim of highlighting the different components and the system parameters that influence them. First, the MPEG encoding/decoding system is analyzed from the

point of view of delay when only intra-frame coding is used; the implications on the delay when using also predictive coding are considered next. Lastly, the transmission with different schemes is analyzed in order to devise the end-to-end delay in various configurations.

6.5.1 Intra-frame Coding Only

The case in which only spatial redundancy is eliminated (i.e., the encoded video sequence is composed only of I-frames) is first considered; basically, this is the compression process recommended by Moving JPEG [18] [58] and thus the basic principles discussed in this Section also apply to a videoconferencing system which exploit JPEG to compress video frames.

Coding Shaping Delay

A CBR MPEG encoder is built by smoothing the natural rate of a basic MPEG encoder into a constant *target rate* B. As shown in Figure 6.44, the smoothing is performed by the *encoder buffer*, which is filled at the natural bit rate and emptied at the target bit rate B. The encoder buffer delays each bit depending on the natural bit rate and the target bit rate. Also, bits are buffered in the decoder buffer until a whole picture is received and can be decode. A picture encoded with F bits takes a time F/B to exit the encoder buffer and enter the decoder one. This is the processing delay of the CBR encoder that depends on the picture dimension. Since it not constant, the decoder must compensate it to comply with Requirement 3 which impose that video frames are continuously displayed at a fixed rate. The composition of the processing delay of the encoder/decoder system and the processing resynchronization delay introduced by the decoder is a constant *coding shaping delay* S_c.

Since bits exit the encoder buffer at rate B,

$$S_c \geq \frac{\max_{seq} F}{B}, \tag{6.17}$$

where $\max_{seq} F$ is the maximum number of bits used to encode a picture over the all sequence. (6.17) guarantees that in the time between when a picture is grabbed, to when it is scheduled for displaying, up to $\max_{seq} F$ bits can

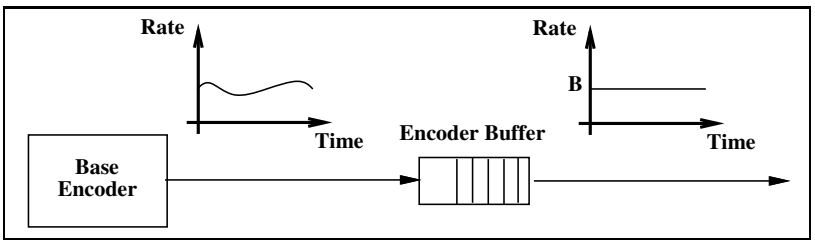

Figure 6.44: Basic MPEG Encoder and Encoder Buffer.

exit the encoder buffer at rate B. In order for the encoded video stream to be continuous (i.e., actually having a constant rate),

$$S_c \geq T, \tag{6.18}$$

otherwise there would be a time interval between two subsequent frames during which no bits are produced. (6.18) shows that the end-to-end delay of a videoconferencing system exploiting a CBR MPEG encoder is larger than the video frame period.

Even though the coding shaping delay is not related to the transmission over a network, it is analogous to the network shaping delay S_n^{TS} introduced by a traffic shaper cascaded to a VBR MPEG encoder (see Section 6.4.4). If a basic MPEG encoder freely fills a finite dimension buffer which is emptied according to a fixed traffic description, the buffer can possibly overflow. In Section 6.4.4 we surveyed some proposals to control the basic MPEG encoding based on feedforward or feedback information, in order to avoid a traffic shaper to discard packets. In the case of a CBR encoder, the rate at which the buffer is emptied undergoes to a very strict traffic description (being it constant). Thus, a rate control function must be introduced to control the basic MPEG encoder in order to avoid the encoder buffer to overflow or underflow.

Rate Control Function

Figure 6.45 shows the components of a CBR MPEG encoder. In principle the rate control function monitors the fullness of the encoder buffer and adjusts the quantization parameter to change the natural bit rate and avoid the buffer to underflow or overflow.

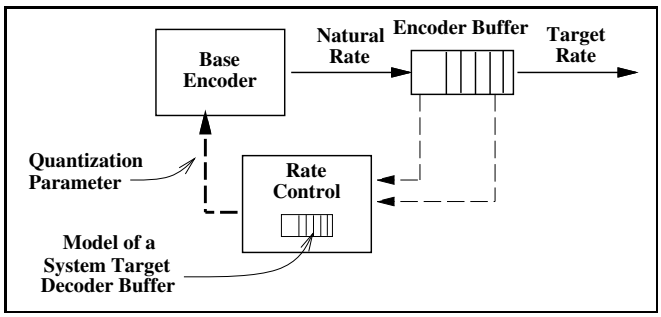

Figure 6.45: Basic Encoder and Rate Control Function.

The compliance of an encoded video stream with the MPEG standard requires that feeding a system target decoder with the stream does not overflow or underflow its buffer, whose dimension is included in the MPEG stream [29]. The MPEG standard specifies that the system target decoder is fed at the constant rate indicated in the stream and it retrieves data from its buffer one *access unit* at a time. The access unit is a whole picture and all the bits encoding a video frame are removed when its decoding is started according to the decoding time specified in the MPEG stream. The buffer of the system target decoder underflows if a whole picture is not present in the buffer when its decoding is scheduled.

Thus, beside monitoring the encoder buffer, the rate control function should keep an updated model of the buffer of a system target decoder, as shown in Figure 6.45. It is filled at the target rate B, and emptied by one picture at the instant in which decoding is scheduled, as shown in Figure 6.46. The quantization parameter (and thus the natural bit rate) is tuned so that the stream compliance is granted.

Actually, if the encoder buffer has the same size of the system target decoder one and the time to encode a picture is assumed to be null, the fullness levels of the two buffers are strictly related. In particular the encoder buffer overflows (underflows) if and only if the system target decoder one underflows (overflows).

In practice, the rate control function can model a *virtual buffer* which is updated according to the dimension of pictures and the target rate B. If the virtual buffer does not underflow or underflows, neither the encoder buffer nor the system target decoder buffer do. The MPEG standard provides guidelines on how to manage the virtual buffer and modify the quantization

parameter [25]; anyway, each CBR MPEG encoder implementation can use its own approach [63, 59, 13, 56].

This virtual buffer constrains the number of bits used to encode each picture. Since according to (6.17) the maximum number of bits used to encode a picture provides a lower bound on the coding shaping delay, we analyze in more detail the virtual buffer implementation of the software encoder `dvdenc` with the objective of devising its impact on the end-to-end delay.

The rate control function models a buffer, called Video Buffer Verifier (VBV), that has the dimension of the system target decoder buffer and is drained at the constant bit rate B. Before encoding a picture, the rate control function sets the maximum and minimum dimensions of the picture. The maximum dimension is set by conservatively assuming that it must not exceed the space available in the buffer[10]. The minimum dimension guarantees that the buffer is not emptied before the end of the video frame period. The VBV is then updated according to the actual number of bits produced. In the following Section, the lower and upper bounds on the dimension of pictures in an CBR MPEG stream are given in terms of the size of the VBV.

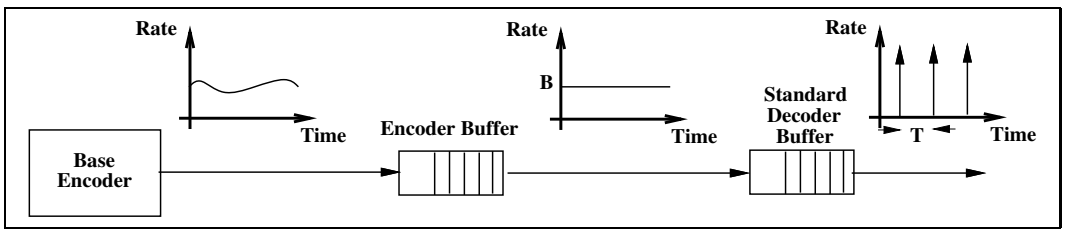

Figure 6.46: Standard Decoder Buffer Model Kept by the Control Function.

Dimension of Encoded Pictures

In the video frame period T, a CBR MPEG encoder working at the target rate B produces $B \cdot T$ bits. The rate control function can be designed in order to use exactly $B \cdot T$ bits to encode each picture. Even though this would guarantee a constant bit rate, the visual quality of images would not be uniform. Instead picture dimension is allowed to vary and is controlled according to the VBV.

[10]This conforms to conservative assumption that even if all the bits were injected in the VBV instantly, it would not overflow.

Claim 1 *The dimension of the VBV is the upper bound on the dimension of pictures in an MPEG compliant stream:*

$$V_s \geq \max_{seq} F$$

where V_s is the dimension of the VBV and F is the dimension of a picture.

Proof The VBV must be larger than the largest encoded picture because it must contain a whole picture before it is started being decoded. Moreover, as the VBV overflow must be avoided to guarantee compliance with the MPEG standard, the rate control function controls the basic encoder so that the number of bits used to encode a picture is not larger than the VBV dimension.

The lower bound on encoded picture dimension is also related to the VBV dimension. To show how this lower bound is determined, we first analyze how picture dimension affects the encoder buffer level. We assume that the encoder buffer and the VBV as the same dimension; this is not a restriction because if the VBV does not underflow, the encoder buffer never contains more bits than the VBV dimension.

In order to have enough bits in the encoder buffer so that it is emptied at the constant rate B during the time T between two pictures, a picture should be encoded with at least $B \cdot T$ bits. If some backlog is present in the buffer at the beginning of the video frame period, the number of bits used for encoding the picture can be smaller. Thus, the lower bound F_m on picture dimension is the minimum allowed picture dimension when the buffer contains the largest backlog.

The backlog is due to bits encoding the previous picture that has not been sent out yet, as exemplified in Figure 6.47. The upper diagram shows that the encoder processes the bits of each raw image during a single video frame period (thin dashed line). The diagram also plots the number of bits generated by the basic encoder versus time for each picture (thick continuous line); the total number of bits per picture varies for different pictures. The CBR encoder emits bits at constant rate B (as shown by the thick dashed line in the lower diagram), thus sending out the same amount of bits $(B \cdot T)$ in each video frame period (last point of the thick dashed line in the upper

diagram). If the dimension of a picture exceeds $B \cdot T$ (e.g., the first picture in Figure 6.47), excess bits are temporarily buffered in the encoder buffer (whose level is depicted by the thin continuous line) until they have been all sent out (after the end of the video frame period).

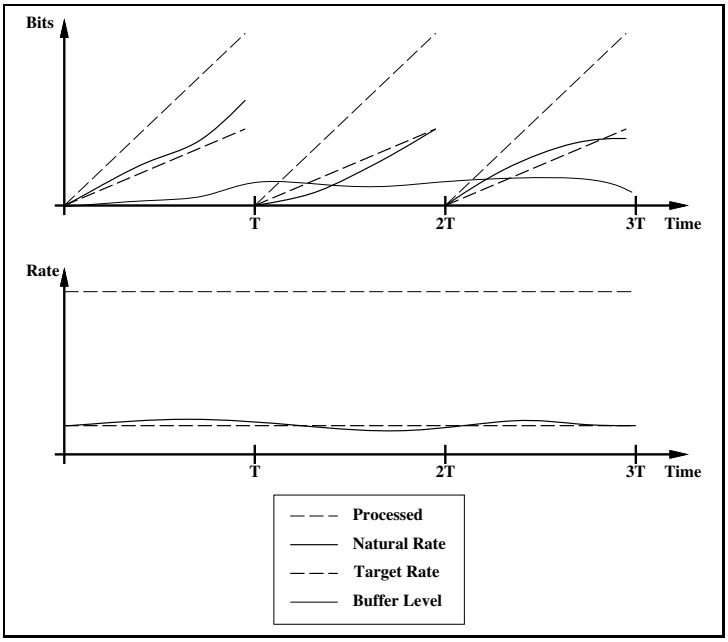

Figure 6.47: Bit Production Rate and Encoder Buffer Fullness.

The largest backlog V_b remains in the buffer after a picture has filled it. Before starting encoding a picture the rate control function checks the VBV and allows the picture to be large enough to fill it. If the maximum allowed amount of bits is produced, the level of the VBV (and the encoder buffer) at the end of the video frame period is

$$V_b = V_s - B \cdot T$$

because during the video frame period $B \cdot T$ bits have been drained from the buffer. The rate control function determines the minimum allowed dimension of the next picture as the number of bits that must be inserted in the VBV so that it contains at least $B \cdot T$ bits, i.e., the amount of bits to be drained during the video frame period:

$$F_m = B \cdot T - V_b = 2 \cdot B \cdot T - V_s$$

This is the lower bound on the picture dimension.

In summary, the dimension of the VBV is the maximum number of bits usable to encode a picture. Moreover, the minimum number of bits usable to encode a picture is bounded by the VBV dimension, the video frame rate, and the target rate of the encoder. Since the natural dimension of MPEG encoded pictures is highly variable (see Section 6.4.1, the more the dimension of pictures is allowed to vary, the more uniform is their visual quality. The maximum possible variation of picture dimension is given by

$$\Delta F_M = V_s - F_m = 2 \cdot (V_s - B \cdot T) \tag{6.19}$$

The larger the VBV dimension, the larger the possible variation of dimension of encoded pictures.

The actual number of bits used to encode each picture is determined by the rate control function. The rate control function must use wisely the allowed variability in picture dimension is fundamental to keep the visual quality uniform throughout pictures and scenes. The task is made more difficult by the fact that, under Assumption 3 of continuous playing made in Section 6.2.2, the amount of bits used to encode a picture is constrained by the amounts used in the preceding ones. This is stated by the following Claim.

Claim 2 *If a picture has its maximum allowed dimension (i.e., it fills the VBV), the following picture must be smaller than* $B \cdot T$.

Proof When a picture is encoded with the maximum allowed number of bits, at the end of the video frame period in which it has been captured, $B \cdot T$ has been drained from the VBV. Being this the number of bits needed to fill the VBV, it provides the maximum allowed dimension for the following picture.

Startup Shaping Delay

The rate control function uses the VBV to guarantee that the MPEG stream complies with the MPEG standard and that the encoder buffer does not overflow or underflow. In this Section we show that, even if the dimension

of a picture is larger than the minimum allowed by the VBV, the encoder buffer can underflow. We also describe how the CBR encoder copes with this situation.

Figure 6.47 shows the timing associated with intra-frame coding. The thin dashed line in the upper diagram shows the amount of bits processed versus the time. This is larger than the amount of bits produced by the basic encoder (continuous line) because of the yielded compression; the punctual slope of the line is the instantaneous natural rate. The objective of the CBR encoder is to produce bits at the constant target rate B, as shown by the thick dashed line; the encoder buffer is used to smooth the variations of the natural bit rate, as exemplified by the lower diagram of Figure 6.47.

When the instantaneous natural bit rate is smaller than the target rate, the encoder buffer level lowers, as shown, for example, in the upper diagram of Figure 6.47 during the second video frame period by the thin continuous line. If the natural rate is smaller than the target rate when the buffer is empty, the encoder buffer underflows. This happens, for example, in the situation depicted in Figure 6.48: at the beginning of the encoding of a scene, the natural rate is lower than the target rate. This can be prevented by waiting a *startup shaping delay* when the encoding of a scene starts, before starting sending bits out of the encoder buffer. The startup shaping delay is obtained by fixing a fullness threshold in the buffer and waiting it to be crossed before the MPEG stream starts flowing out of the CBR encoder.

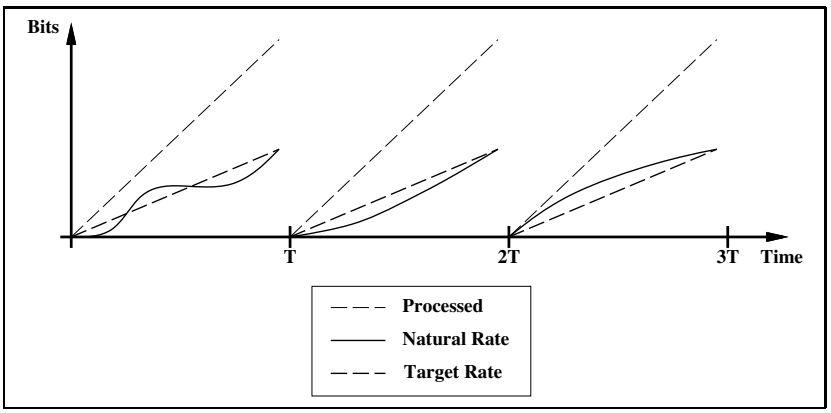

Figure 6.48: I-frames Only Encoding.

The threshold represents the tolerance of the system to a natural bit rate lower than the target rate. Since the natural rate is controlled by the rate

control function, the threshold can be as low as desired, provided that the rate control is aggressive enough to increment the natural rate so that the buffer does not underflows. Nevertheless, the lower the threshold, the smaller the area of the picture over which the natural production rate can be smaller than the target rate, i.e., the area over which the rate control function must intervene. Thus, the visual quality of images is not uniform because the quantization parameter changes suddenly. On the other hand, the higher the threshold, the larger the startup shaping delay.

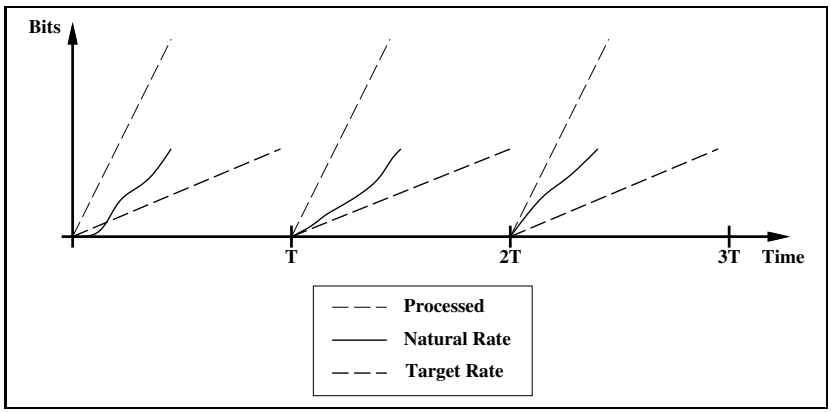

Figure 6.49: Fast I-frame Only Encoding.

The choice of the tradeoff between long startup shaping delay and non-uniform visual quality (and aggressive control function) is left to the specific encoder implementation. The chosen value of the startup shaping delay is not very critical because, if the encoder is powerful enough the target rate is much lower than the average natural rate, as shown in Figure 6.49. This means that the main aim of the encoder buffer is not really smoothing the variation of the natural rate, but lowering it. Thus, it is not necessary to introduce a large startup shaping delay to cope with actual bit production rate variation, i.e., the fraction of buffer to be filled before starting transmitting should not be large because the buffer is emptied much slower than it is filled on the average.

Shaping Delay Implementation

After having described which are the bounds on the dimension of pictures, we analyze their impact on the delay and how it is introduced by the CBR encoder/decoder system.

Pictures experience the startup shaping delay entirely in the encoder buffer. The encoder introduces the startup shaping delay when it starts encoding a scene: no bit exits from its buffer until the buffer level crosses a predefined threshold. In order to keep the startup shaping delay small, the rate control function must increase the natural rate if the threshold is not crossed within a predefined amount of time.

The coding shaping delay is experienced by pictures partly in the encoder buffer and partly in the decoder one. Nevertheless, the value of the coding shaping delay is chosen by the encoder that accordingly sets the decoding and presentation time stamps in the MPEG stream [29]. This determines the amount of time spent by each picture in the decoder buffer so that the overall delay be S_c.

The MPEG standard does not state how the encoder should choose the coding shaping delay, i.e., set the time stamps; nevertheless, this choice is crucial to the visual quality of the resulting stream. In fact, according to Equation (6.17), given the coding shaping delay, picture dimension cannot exceed $S_c \cdot B$. The upper bound on the dimension of pictures encoded by a CBR encoder is determined by the VBV dimension, as stated by Claim 1. As a consequence, once the coding shaping delay has been set, the encoder must set the VBV dimension according to

$$V_s \leq S_c \cdot B \tag{6.20}$$

In Section 6.5.1 it has been shown that VBV dimension affects picture dimension, its variability, and hence visual quality of the encoded stream. Thus, a small coding shaping delay reduces the end-to-end delay of the system, but delivers non constant quality. The implementation of the rate control function must find a tradeoff.

Once the startup and coding shaping delays has been chosen (and the VBV dimension set accordingly), the production of a suitable MPEG stream is guaranteed by the rate control function avoiding the VBV to overflow or underflow. It is always possible to increase the natural bit rate because unencoded pictures are much larger than encoded ones. It is always possible to decrease the bit production rate, at worst by stopping encoding and using a particular code in the MPEG stream that represents a MB in which all the DCT coefficients are null.

Experimental Data

The `dvdenc` software MPEG encoder used to provide the experimental data shown in this work has been implemented for MPEG compression of video sequences to be stored on video disks. Thus, it is not optimally designed for a real-time videoconferencing system. As a consequence, the control function of the CBR encoder is not aggressive enough to properly limit picture dimension when the VBV is small.

The minimum VBV dimension is $B \cdot T$. According to Claim 1, a smaller dimension would deliver pictures smaller than $B \cdot T$; in this case the encoder would not be able to emit bits at the target rate for a whole video frame period. Due to the limited aggressiveness the control function, the dimension of the buffer must be chosen quite larger that this minimum and thus its impact on picture dimension is not clearly visible.

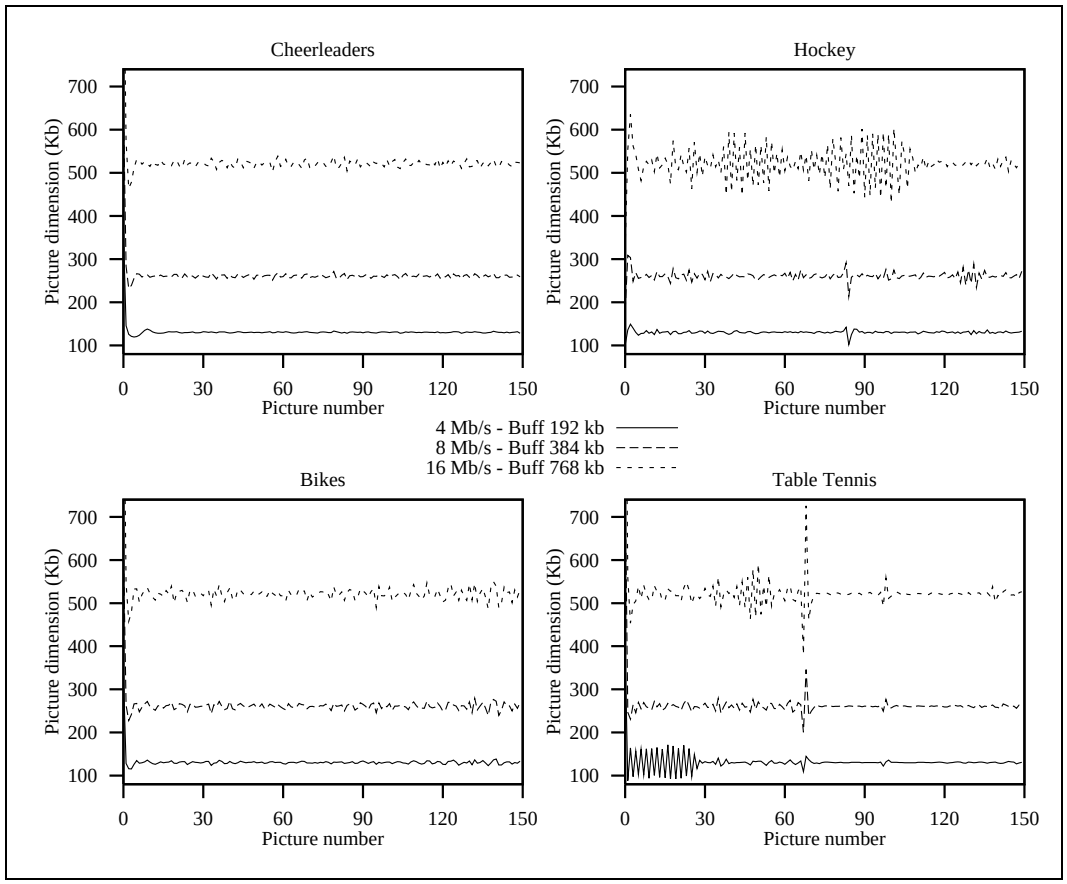

Figure 6.50: Frame Dimension in Intra-frame Only CBR Encoding.

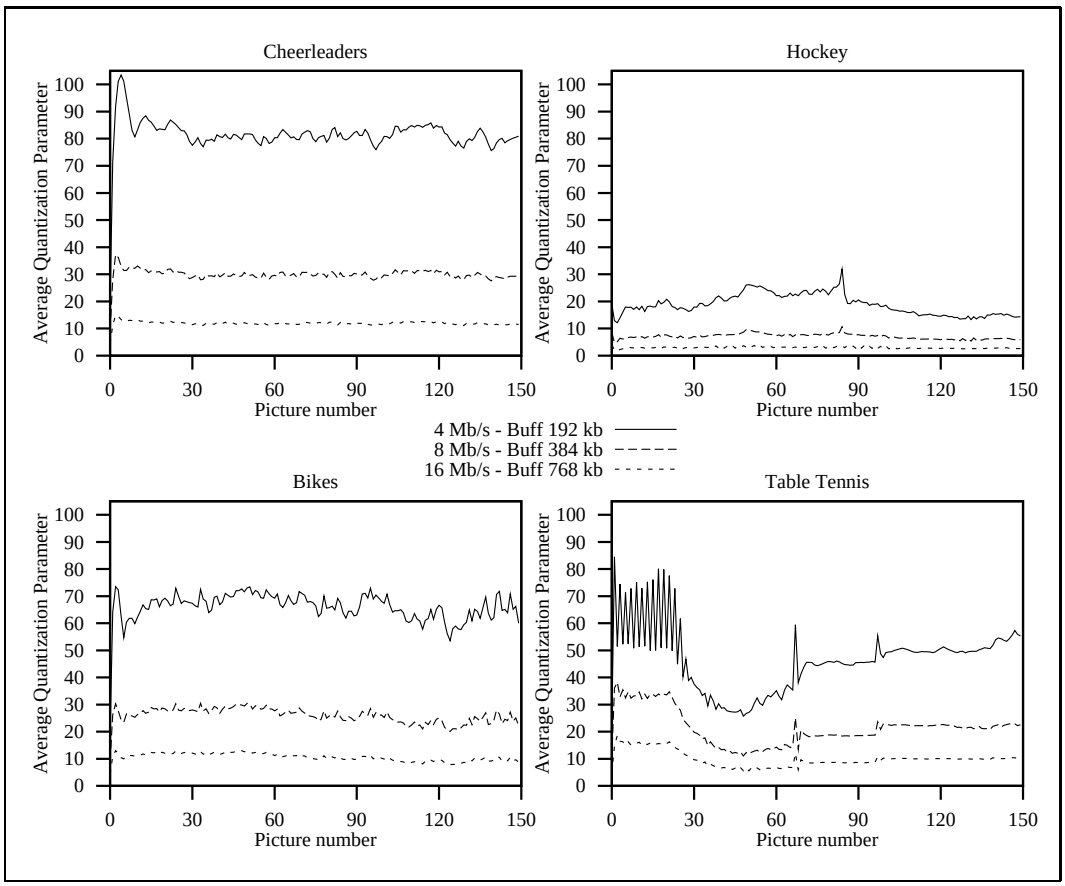

Figure 6.51: Average Quantization Parameter in Intra-frame Only CBR Encoding.

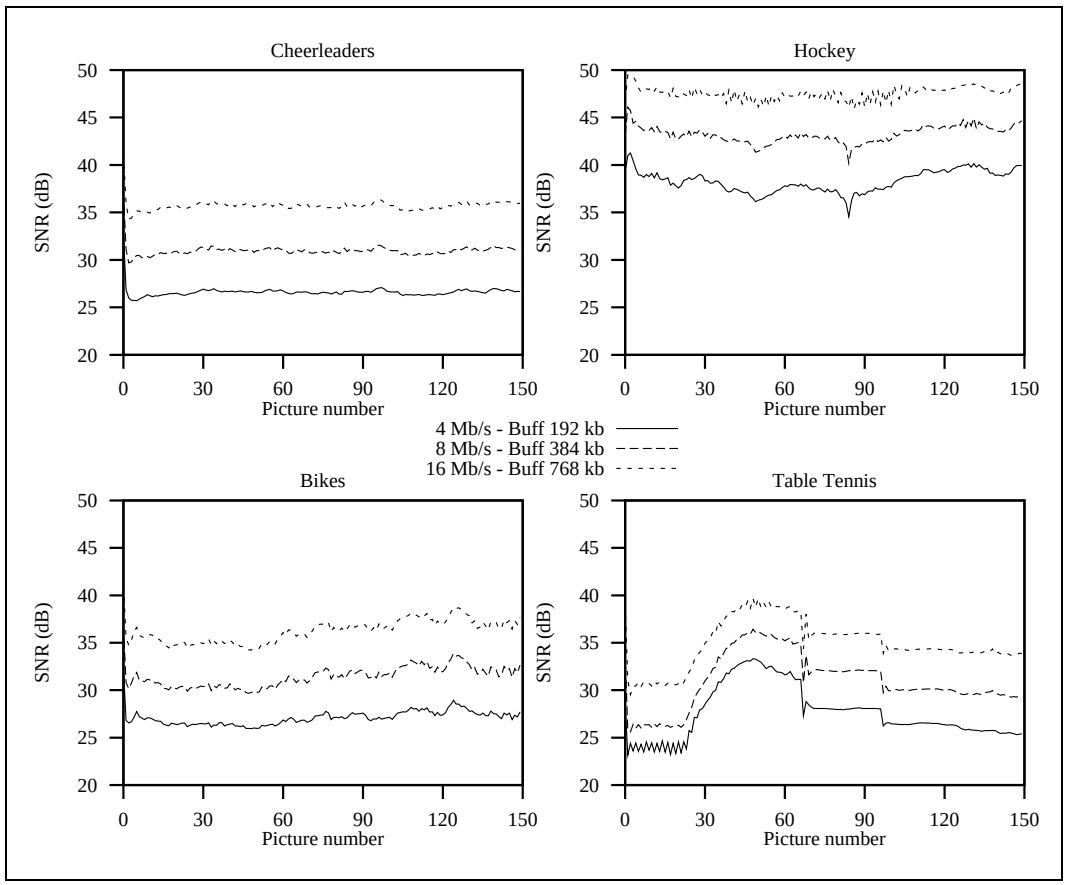

Figure 6.52: SNR in Intra-frame Only CBR Encoding.

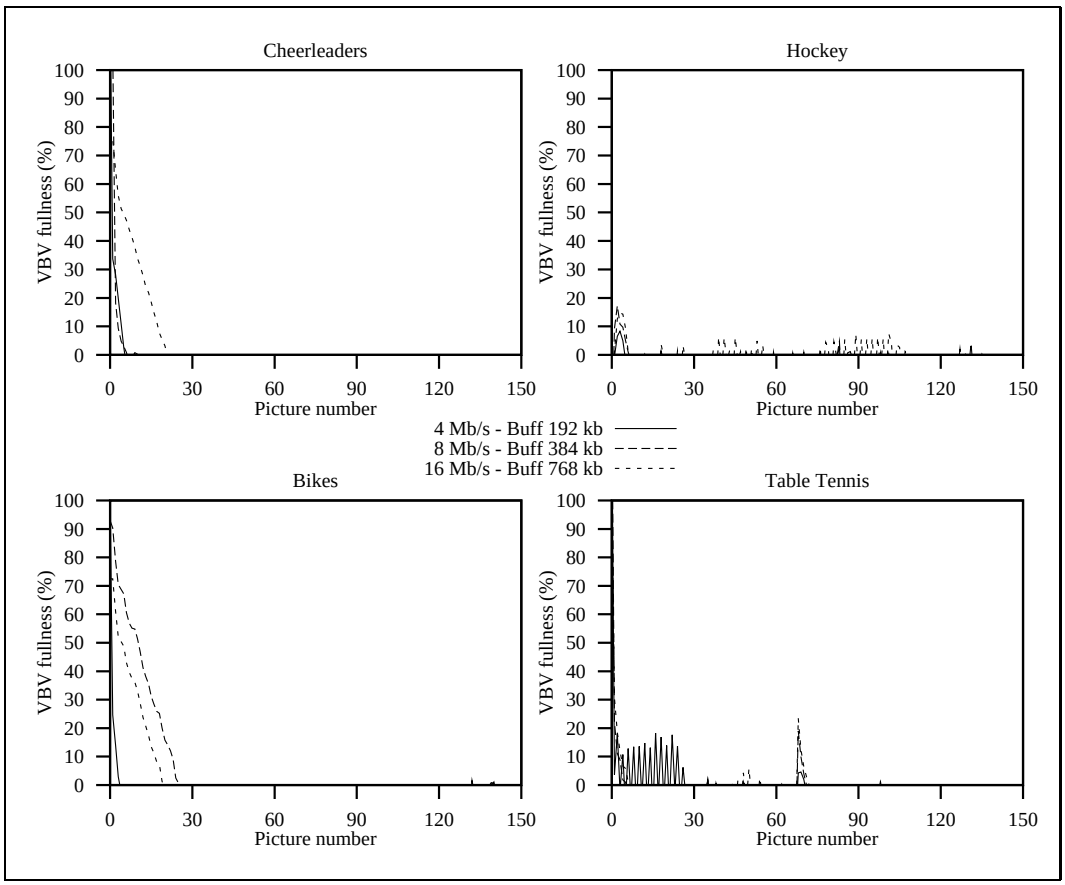

Figure 6.53: VBV Fullness in Intra-frame Only CBR Encoding.

Figure 6.50 shows the number of bits produced by intra-frame coding the four sequences used throughout this work. Three different target rates are used for the encoding of each sequence in order to show how the parameter influences the number of bits per frame produced and its variation. The ratio between VBV dimension and target rate is kept constant so that all the configurations deliver the same coding shaping delay.

As the rate increases the dimension of frames grows larger and the quantization parameter (averaged on the whole frame) decreases, as shown in Figure 6.51. Accordingly, the quality of the encoded sequence becomes higher, as shown by the SNR plotted in Figure 6.52.

Since in the experiments the ratio between VBV dimension and target rate is constant, according to Equation (6.19), more variability is allowed for picture dimension as the target rate increases. This is confirmed by the plots in Figure 6.50.

In the "hockey" scene at 16 Mb/s the dimension of adjacent pictures varies very much due to the small value of the quantization parameter (see Figure 6.51). The global distortion level G is an integer and is varied by 1; due to the low value of G, the relative variation is large and is reflected in a large variation of picture dimension.

6.5.2 Intra-frame and Predictive Coding

Predictive coding yields better compression than intra-frame coding, as qualitatively shown in Figure 6.54 and confirmed by the experimental results shown in Figure 6.19. Thus, if predictive coding is used to encode a scene with a given visual quality, the whole number of bits produced is smaller than if only intra-frame coding is used; the larger the GOP (i.e., the number of P-frames per I-frame), the smaller the amount of bits produced[11]. Moreover, the slower the scene, the smaller the dimension of P-frames; since videoconferencing scenes are likely slow, predictive coding must be used to increase compression.

In the following of this Section, the impact of predictive coding on coding shaping delay is studied. We show that coding shaping delay makes CBR

[11] Actually, as shown in [45], there is a maximum GOP dimension beyond which enlarging the GOP does not decrease the amount of bit generated because the prediction error accumulates and requires many bits to be encoded.

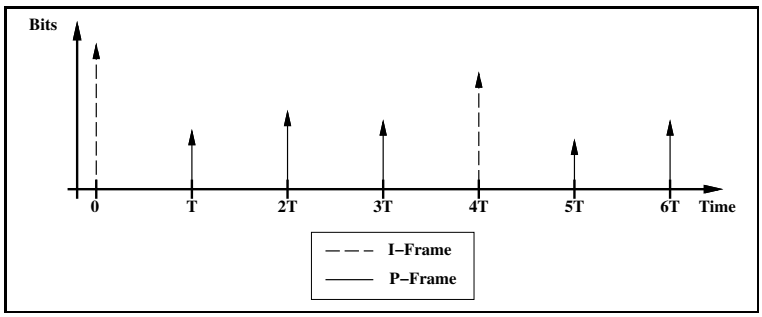

Figure 6.54: Amount of Bits Encoding Each Picture.

MPEG with predictive coding unsuitable for videoconferencing applications. In order to provide a better understanding of coding shaping delay, we first show where pictures experience it. In Section 6.5.2 we show that when a CBR MPEG encoder uses predictive coding, given the target rate, the slower the sequence, the higher the visual quality of the encoded sequence. Nevertheless, the coding shaping delay is larger than when using only intra-frame coding. Then, we show how predictive coding increases the coding shaping delay when encoding at constant quality. Lastly, the impact of predictive coding on the startup shaping delay is considered.

Buffering Issues Related to the Coding Shaping Delay

In Section 6.5.1 it has been shown that a CBR MPEG encoder/decoder system introduces a coding shaping delay in order to allow pictures to be encoded at constant rate with variable amount of bits. Pictures experience part of it in the encoder buffer and part in the decoder buffer, as explained in detail in this Section.

Figure 6.55 shows how the bits of an encoded video stream are produced and buffered by a CBR encoder/decoder system using both intra-frame and predictive coding. The upper diagram qualitatively shows that I-frames are larger than P-frames. The natural rate averaged on a video frame period while the basic encoder is intra-frame coding, is larger than the target rate; the natural rate while predictive coding is smaller than the target rate.

The lower diagram plots the fullness level of the encoder (continuous line) and the decoder (dashed line) buffer. P-frames find a backlog in the encoder buffer due to the high number of bits produced by the basic encoder for the previous

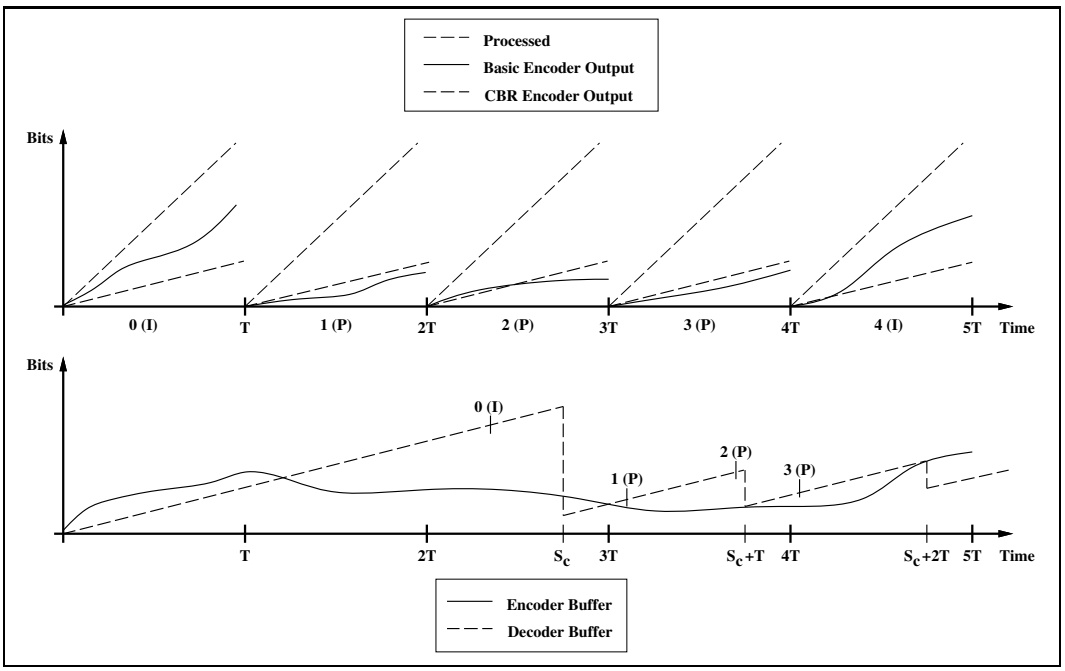

Figure 6.55: Natural and Target Generation of Bits and Encoder and Decoder Buffer Fullness.

pictures and still not sent out by the CBR encoder. The decoder buffer level is incremented linearly and decremented by retrieval units corresponding to a frame. Encoded pictures are not retrieved from the buffer and decoded as soon as they arrive because they experience the processing resynchronization delay unless their dimension is the maximum allowed by the system. For example, picture 0, which is intra-frame coded, fully enters the decoder buffer (exits the encoder) at the time tagged `0(I)`, but it is not decoded until the instant S_c, i.e., until the coding shaping delay is elapsed from its capture.

Coding Shaping Delay at Constant Target Rate

We consider the encoder operating at a fixed target rate and study the effect of the coding shaping delay on the visual quality of the encoded stream. This is a realistic scenario as CBR MPEG encoders usually have their fixed target rate and change the ratio between I-frame and P-frame dimension according to the motion in the scene being encoded. If it is slow, P-frames are small and more bits out of the GOP budget are used in encoding I-frames. If a scene is fast, P-frames require more bits and the dimension of I-frames decreases

accordingly. In order to have a simple (even though not accurate) measure of the visual quality which allows different configuration to be compared, we make the following Assumption.

Assumption 4 *If the same scene is encoded more than once with different encoding parameters, the visual quality of the encoded streams obtained with the different experiments can be compared by comparing the dimension of I-frames: larger I-frames mean better visual quality.*

According to this Assumption and to what stated above, given the target rate, the fast moving is a scene, the lower the quality.

In a videoconferencing scenario, scenes are usually slow and thus, given a target rate, the encoder produces large I-frames. By doing this it introduces a large coding shaping delay which is lower bound by the dimension of the largest picture according to Inequality (6.17).

We now devise an expression that gives the coding shaping delay as function of P-frame dimension and other parameters of a CBR encoder that exploits predictive coding. We assume that the encoder uses a constant number of bits to encode each GOP: it must produce these bits in a GOP period, i.e., each GOP is encoded with $N \cdot T \cdot B$ bits. Actual CBR do not usually comply to this assumption; thus, the expression we devise does not provide the coding shaping delay of any particular encoder. Nevertheless, it gives the flavor that CBR encoders for videoconferencing applications should better not exploit predictive coding in order to reduce the end-to-end delay.

According to the foregoing assumption, the following equation holds for each GOP

$$F^I + \sum_{i=1}^{N-1} F_i^P = N \cdot T \cdot B \tag{6.21}$$

where F^I is the dimension of the I-frame in the GOP, F_i^P is the dimension of the i^th P-frame. By extracting F^I from Equation (6.21), we obtain, for each GOP,

$$F^I = N \cdot T \cdot B - \sum_{i=1}^{N-1} F_i^P \tag{6.22}$$

The smaller the number of bits used to encode P-frames, the larger the dimension of the I-frame in the GOP.

Equation (6.18) in Section 6.5.1 gives the coding shaping delay $S_c = \max_{seq} F/B$, where $\max_{seq} F$ is the dimension of the largest encoded picture in the sequence. When predictive coding is exploited, the largest picture is the I-frame in the GOP with the smallest P-frames. For the sake of simplicity (also in the notation), we assume that each I-frame has the same dimension, i.e., $\max_{seq} F = I$ as given by Equation (6.22); substituting in Equation (6.18):

$$S_c = \frac{N \cdot T \cdot B - \sum_{i=1}^{N-1} F_i^P}{B} = N \cdot T - \frac{\sum_{i=1}^{N-1} F_i^P}{B} \tag{6.23}$$

Picture Dimension The smaller the P-frames, the longer the coding shaping delay that the encoding/decoding system must introduce. If the encoded scene is completely static, the P-frames are encoded with a very small amount of bits; in principle 0. Thus, the I-frame grows using all the bits intended for the encoding of the GOP, i.e., $N \cdot T \cdot B$; this corresponds to a coding shaping delay equal to the GOP period $N \cdot T$.

Thus, a CBR encoder which aims at yielding a short coding shaping delay must bound the ratio between the dimension of I-frames and P-frames, even though the scene to encode is slowly moving and would allow a large ratio. The key element in determining the number of bits used to encode pictures is the VBV dimension V_s; a dimension equal to $N \cdot T \cdot B$ allows the encoder to take full advantage of predictive coding when the scene is particularly slow and to use a large amount of bits to encode I-frames. A larger dimension allows the encoder to deliver even more uniform quality by smoothing sudden increase in the complexity of pictures and in the motion over more than a GOP period. Nevertheless, this requires larger coding shaping delay and buffers in both encoder and decoder.

Beside the VBV, CBR encoders better have other means for controlling the number of bits used to encode pictures. For example, the software encoder `dvdenc` before encoding a picture sets a target dimension; this is chosen according to a predefined amount of bits to be used in the encoding of each GOP and a preferred ratio between dimension of I-frames and P-frames. This is particularly useful when a particularly complex picture (i.e., with low spatial redundancy) is being intra-frame coded; if the encoder used only the

VBV to determine the target for the picture dimension, in order to deliver good quality, it would use the maximum number of bits allowed by the system to encode a picture, i.e., it would fill up the VBV. According to Claim 2, the dimension of following picture(s) would be bound by $B \cdot T$; this bound can be too small, particularly if the scene becomes suddenly fast. In fact, P-frames are encoded with a number of bits that is not sufficient to show the same visual quality of the I-frame corresponding I-frame and thus the quality of the GOP is lowered.

GOP Size Increasing the GOP size (i.e., N) increases the coding shaping delay because the percentage of small pictures (the P-frames) increases and so I-frames are made larger to keep the target rate; even though according to Assumption 4 this increases the visual quality of the encoded scene, it should be avoided to keep the delay smaller[12].

Using a low video frame rate (i.e., large T) increases the coding shaping delay, as shown by Equation (6.23); the longer the video frame period T, the larger the coding shaping delay. Nevertheless, Equation (6.22) shows that increasing the video frame period also improves the quality of the encoded scene because I-frames become larger.

Conclusion 1 *Given the target rate of a CBR MPEG encoder, predictive coding increases the visual quality. Nevertheless, every parameter setting aimed at quality improvement increases the coding shaping delay.*

Coding Shaping Delay at Constant Visual Quality

We now study the impact of predictive coding on the coding shaping delay, given the visual quality the CBR encoder must deliver. We assume to encode the same scene more than once with different amounts of compression due to predictive coding. I.e., for each experiment the parameters of the encoder are changed so that predictive coding is more or less heavily exploited and the visual quality delivered is the same.

The visual quality of the encoded streams is compared by comparing the dimension of I-frames, according to Assumption 4; we need a way to compare

[12]Increasing the GOP size also renders the encoded stream more sensitive to errors and losses.

the effectiveness of predictive coding. The exploitation of predictive coding is as more effective as much P-frames are smaller than I-frames and as much GOP size N is larger.

According to Assumption 4, if the same scene is encoded with same visual quality, but using a different number of P-frames per GOP, the I-frames produced in each experiment have the same dimension F^I. Assumption 4 is a rough approximation, but it is close to reality when dealing with simple slow moving scenes (such as a typical videoconference); moreover, the aim of this Section is only to give the flavor of the impact of predictive coding on the coding shaping delay, not to provide a general analytical relationship between the two. In Section 6.4.1 we stated that usually the dimension of I-frames is from 2 to 4 times the dimension of P-frames; the slower the encoded scene, the larger the ratio between I-frame and P-frame dimension. In the context of the analysis carried on in this Section, we assume that P-frames are encoded with constant amount of bits P and exploit as measure of yielded compression the ratio

$$\alpha = \frac{F^I}{F^P}$$

The rate of the encoded stream can be roughly expressed as

$$B^N = \frac{F^I + (N-1) \cdot F^P}{N \cdot T} \tag{6.24}$$

where N is the number of pictures per GOP and F^P is P-frame dimension, which we assume to be constant for all the P-frames. Equation (6.24) can be rewritten as

$$B^N = \frac{N + \alpha - 1}{\alpha \cdot N \cdot T} \cdot F^I \tag{6.25}$$

To better highlight the effect of predictive coding on the rate of the encoder, we also provide the encoder rate when only intra-frame coding is used:

$$B^I = \frac{F^I}{T} \tag{6.26}$$

By merging this equation with (6.25), the latter can be written as

$$B^N = B^I \cdot \frac{N + \alpha - 1}{\alpha \cdot N} \tag{6.27}$$

Due to the approximations made to devise this Equation, it does not exactly reflect the real behavior, but it gives the flavor that the more effective predictive coding (i.e., the ratio α), the smaller the target rate needed to deliver the same visual quality. Also, the larger the number of P-frames per GOP (N), the smaller the target rate.

We now analyze the effect of predictive coding at constant quality on coding shaping delay. Equation (6.18) in Section 6.5.1 gives the coding shaping delay as $S_c = \max_{seq} F/B$, where $\max_{seq} F$ is the maximum dimension of frames and B is the target rate of the CBR encoder. According to Assumption 4, $\max_{seq} F = F^I$ and the coding shaping delay is expressed as

$$S_c = \frac{F^I}{B}$$

Being F^I constant when the quality is constant, the target rate of the encoder B is the only factor which determines the coding shaping delay in different configurations. By substituting the value of B with B^N given by Equation (6.25)

$$S_c = \frac{\alpha \cdot N \cdot T}{N + \alpha - 1}$$

The more predictive coding is effective (i.e., α), the larger the coding shaping delay. Typical values of α are between 2 and 4; the slower the motion in the scene, the larger the value of α. In a videoconferencing application, it is likely to have $\alpha = 4$; with $N = 15$ at 15 fps, $S_c = 222$ ms, i.e., the CBR encoder alone introduces in the videoconferencing system a delay larger than the 100 ms bound essential for interaction. Similar considerations apply also to the increase of N.

Conclusion 2 *Using a CBR MPEG encoder with predictive coding to build a videoconferencing system, reduces the bandwidth requirement, but it is not acceptable from the point of view of the end-to-end delay.*

Startup Shaping Delay

As shown in Figure 6.56, a typical MPEG encoder encodes I-frames with a large amount of bits in a small time, with respect to P-frames. As a consequence, it is unlikely that the natural rate be lower than the target rate when the encoding of a GOP begins. After, the buffer has some backlog due to the large amount of bits encoding the I-frame.

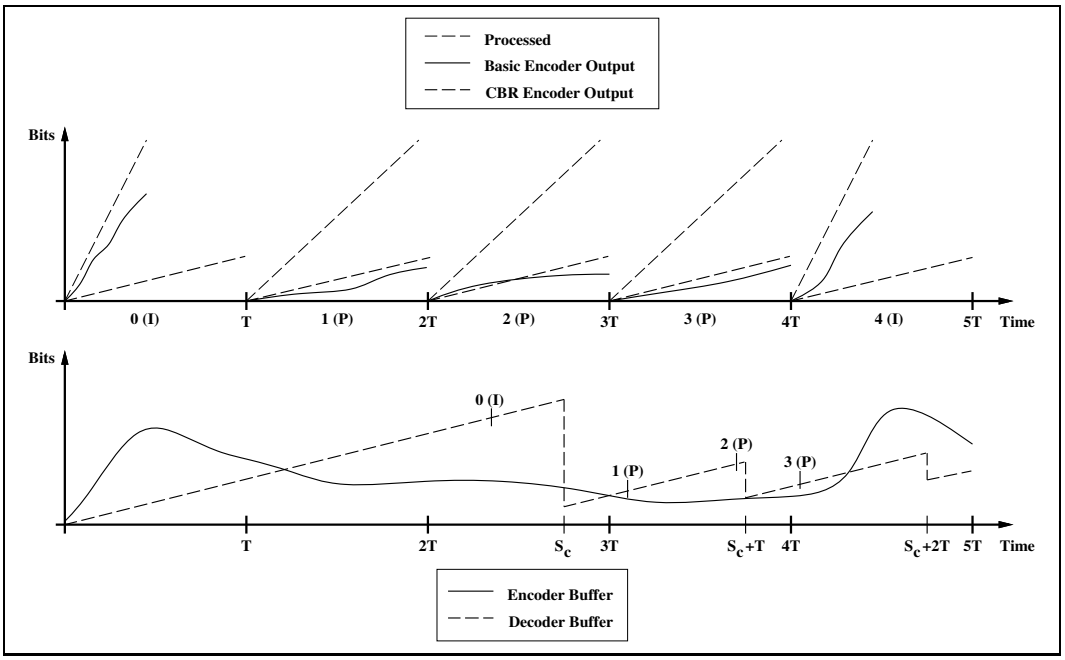

Figure 6.56: Realistic Bit Generation.

The encoder buffer can happen to underflow only when P-frames are particularly small; the highest underflow probability is after the when the buffer gets empty at the end of a GOP with small P-frames. During the rest of the time a backlog is likely to be present in the encoder buffer, as can be seen in Figure 6.55. Thus, even though the startup shaping delay is underdimensioned, the effects (non uniform quality due to a sudden decrease of the quantization stepsize in order to increase the natural rate) are visible on a small percentage of the pictures.

6.5.3 Experimental Data

Figures 6.57-6.60 show the experimental data obtained by encoding the four sample sequences at three different rates. The ratio between the VBV dimension and the rate is kept constant to a value that guarantees the VBV not to be overflowed, notwithstanding the low aggressiveness of the rate control function. The GOP dimension is set to $N = 15$, i.e., each I-frame is followed by 14 P-frames.

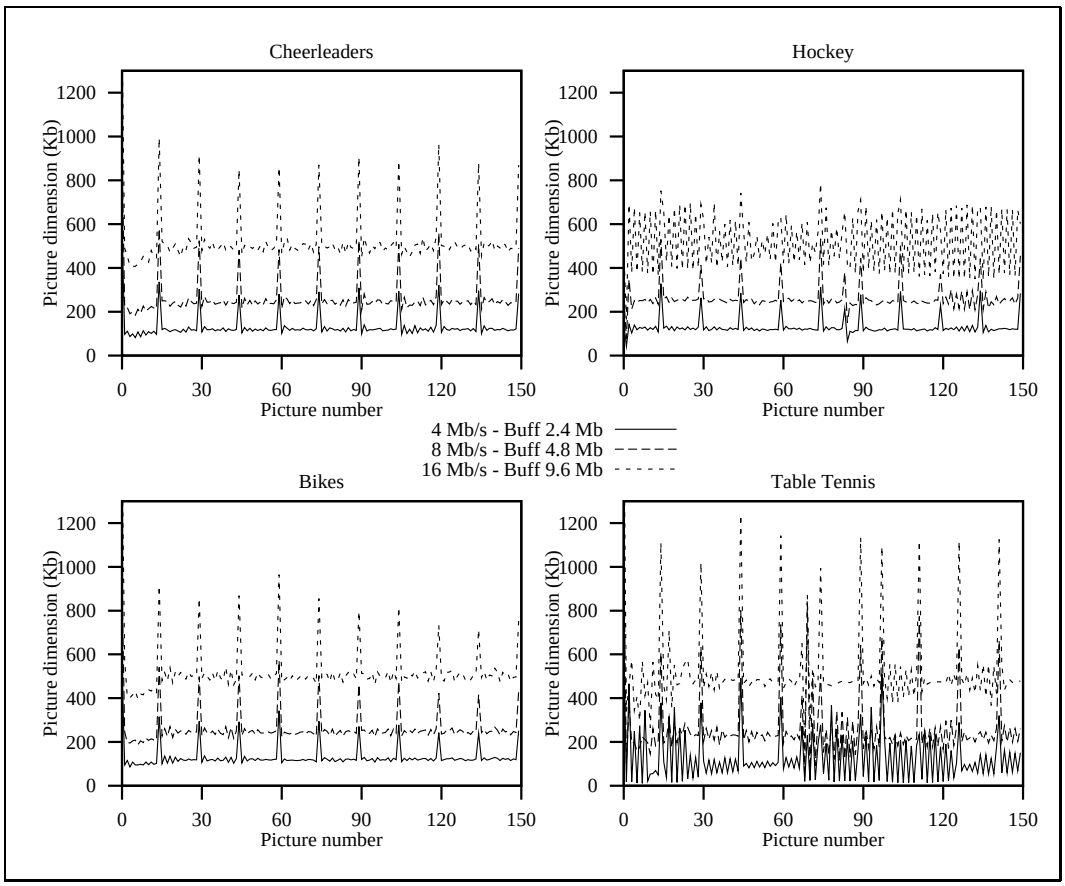

Figure 6.57: Dimension of Pictures in CBR MPEG Encoding with $N = 15$.

Figure 6.57 shows the dimension of each picture. I-frames are typically larger than P-frames and the dimension of both I-frames and P-frames is variable. Especially at low bit rates the dimension of I-frames is double or more than double the dimension of P-frames: in order for this to be possible the coding shaping delay must be large.

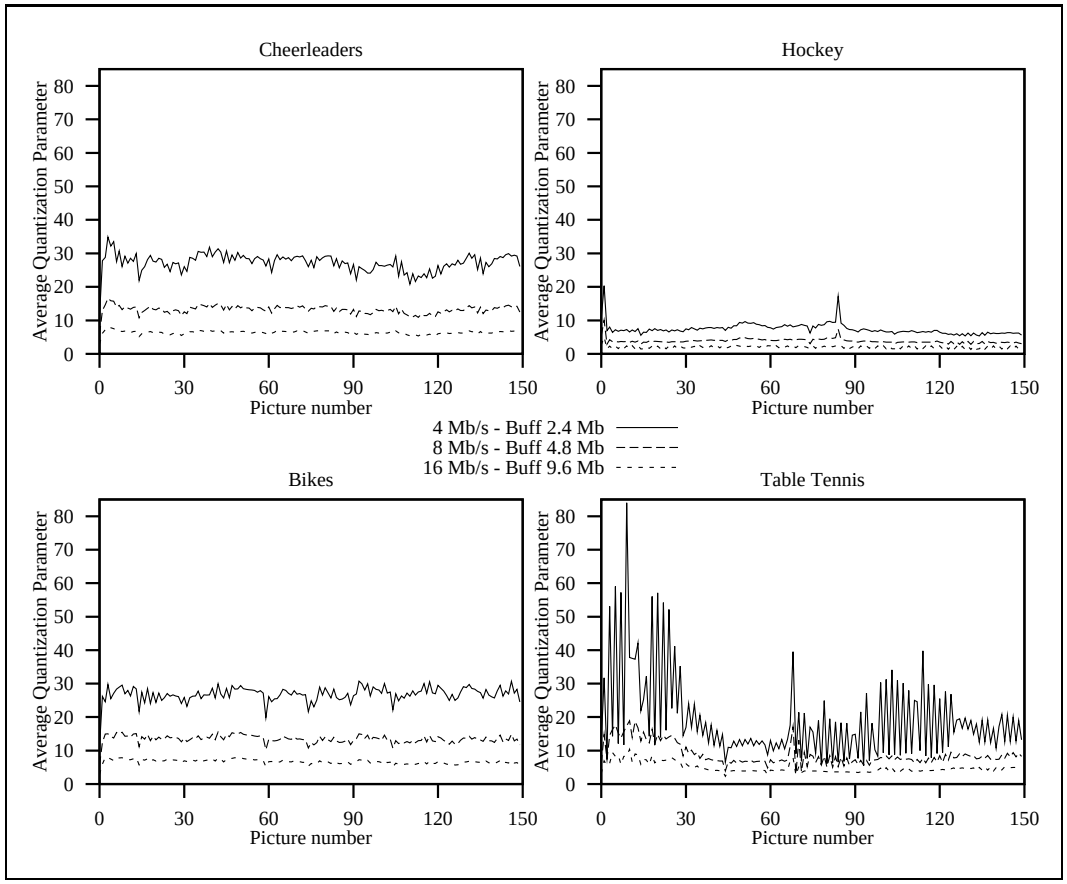

Figure 6.58: Average Quantization Parameter in CBR MPEG Encoding with $N = 15$.

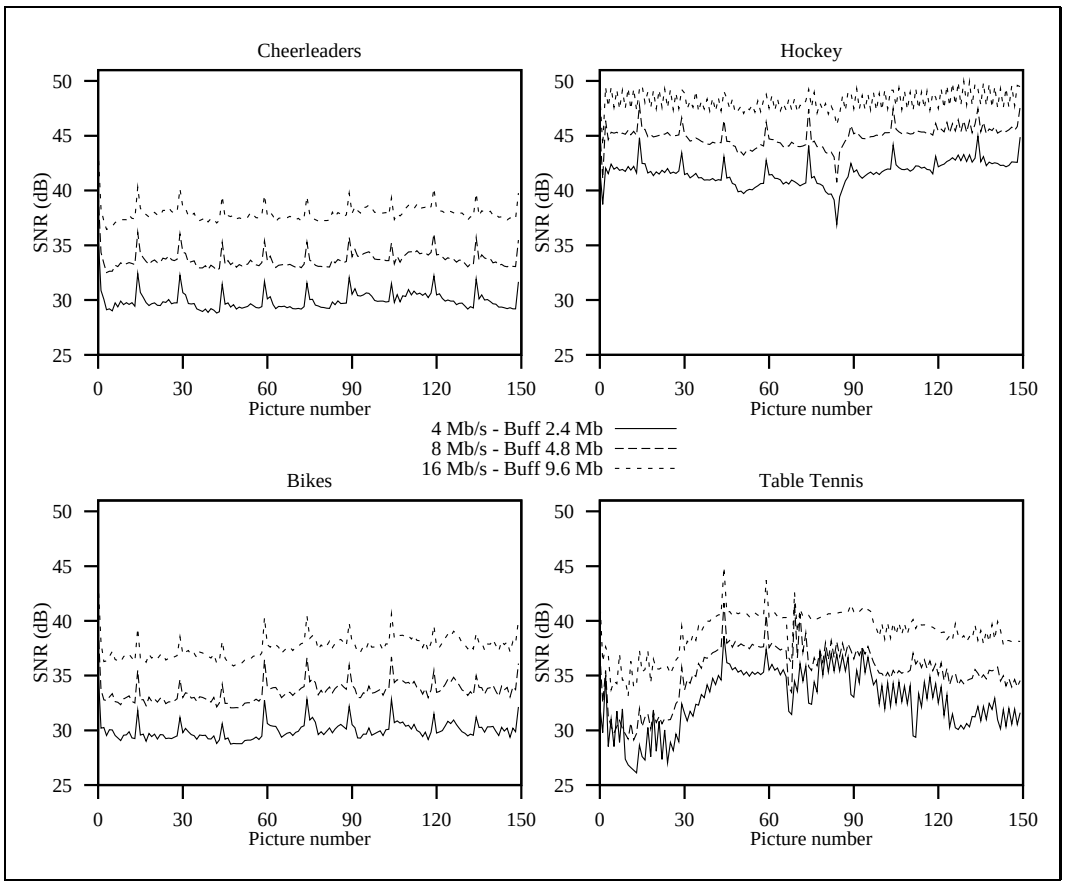

Figure 6.59: SNR in CBR MPEG Encoding with $N = 15$.

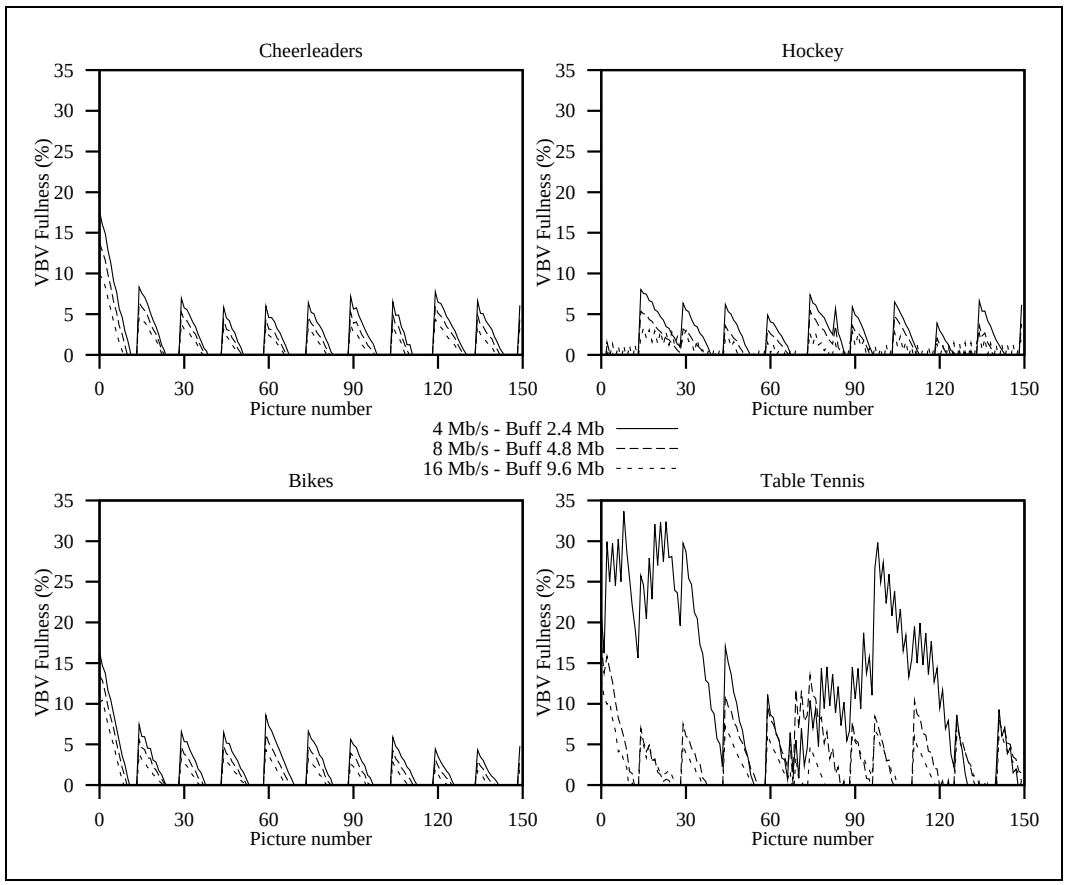

Figure 6.60: VBV Fullness in CBR MPEG Encoding with $N = 15$.

In the "hockey" sequence encoded at 16 Mb/s there is almost no difference between the dimension of I-frames and P-frames. This is due to the fact that the camera is panning over the hockey court to follow the game. Pictures are simple (most of the screen is the white ice of the court) so that I-frames can be encoded with few bits; the P-frames are quantized with a small quantization parameter (Figure 6.58). Since the camera is panning, motion concerns the whole images and P-frames dimension is large: the first P-frame in the GOP is not much smaller than the I-frame. In order to reduce P-frame dimension the quantization parameter is decreased; being it small, decreasing it by one unit has a significant effect on the picture dimension: the second P-frame is much smaller than the first one. This causes the VBV level to decrease (Figure 6.60) and the quantization parameter to be reduced again to avoid underflow. This behavior is kept for the whole GOP thus leading to the oscillation in P-frames dimension and picture quality (see SNR in Figure 6.59). Figure 6.58 shows the average quantization parameter value used on each frame.

Figures 6.61-6.64 show the experimental data obtained using a GOP of 30 pictures. The results are quite similar to the one obtained with $N = 15$ except for the "Table tennis" sequence in which I-frame dimension increases (bottom right plot in Figure 6.61) and the VBV gets more full and empties more slowly (bottom right plot in Figure 6.64).

The scenes used in these experiments are not videoconferencing sequences; a videoconference is usually characterized by low motion and pictures having low complexity. Thus, when compressing them with a CBR MPEG encoder a fairly small amount of bits can be sufficient to encode I-frames with good quality; when encoding a P-frame the error between the MB that is being encoded and the reference MB is small, and thus the amount of bits required is low.

In order to emphasize this behavior of a CBR MPEG encoder, we exploited to encode a static scene. This is an extreme case of slow moving scene but is likely for a videoconferencing application if, for example, the camera is pointed at a whiteboard while somebody is explaining its content. We obtained two static scenes by repeating 120 times the same image of the "cheerleaders" and "hockey" scenes, respectively.

Figure 6.65 shows the dimension of frames for various target rates and VBV dimensions. In any plot the ratio between I-frame and P-frame dimension

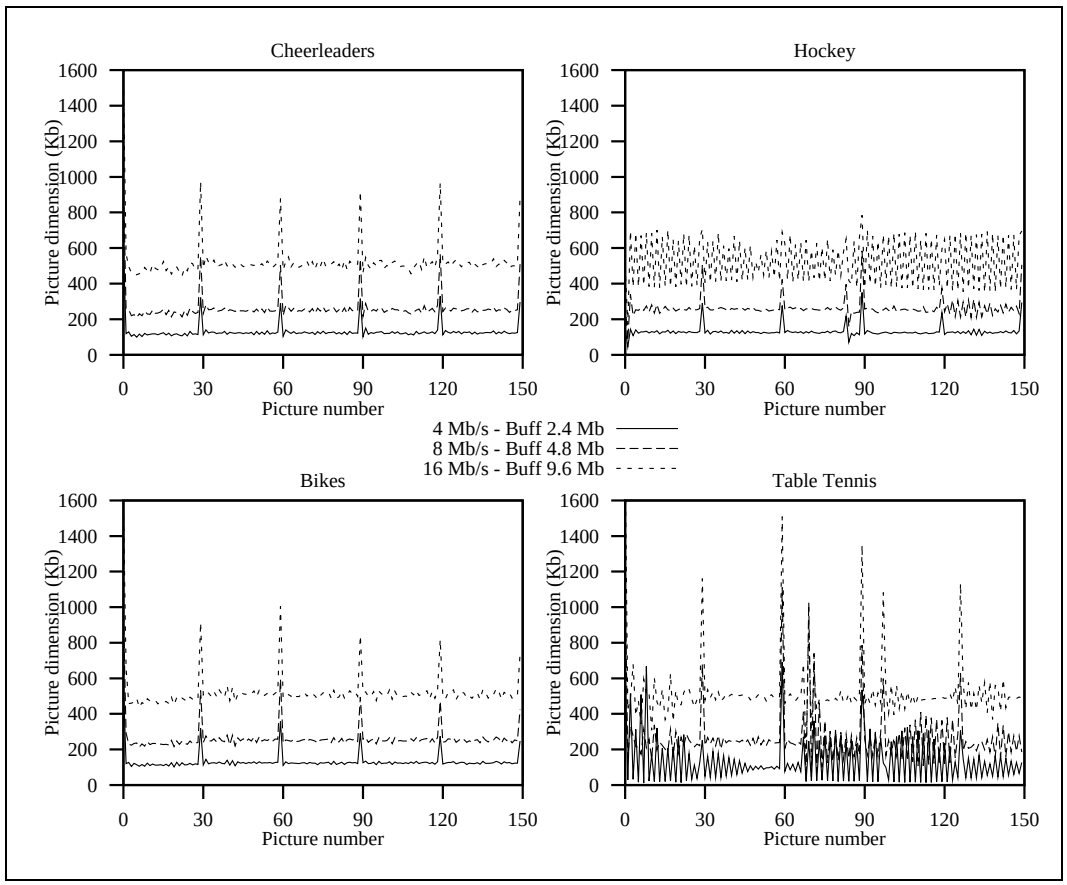

Figure 6.61: Dimension of Pictures in CBR MPEG Encoding with $N = 30$.

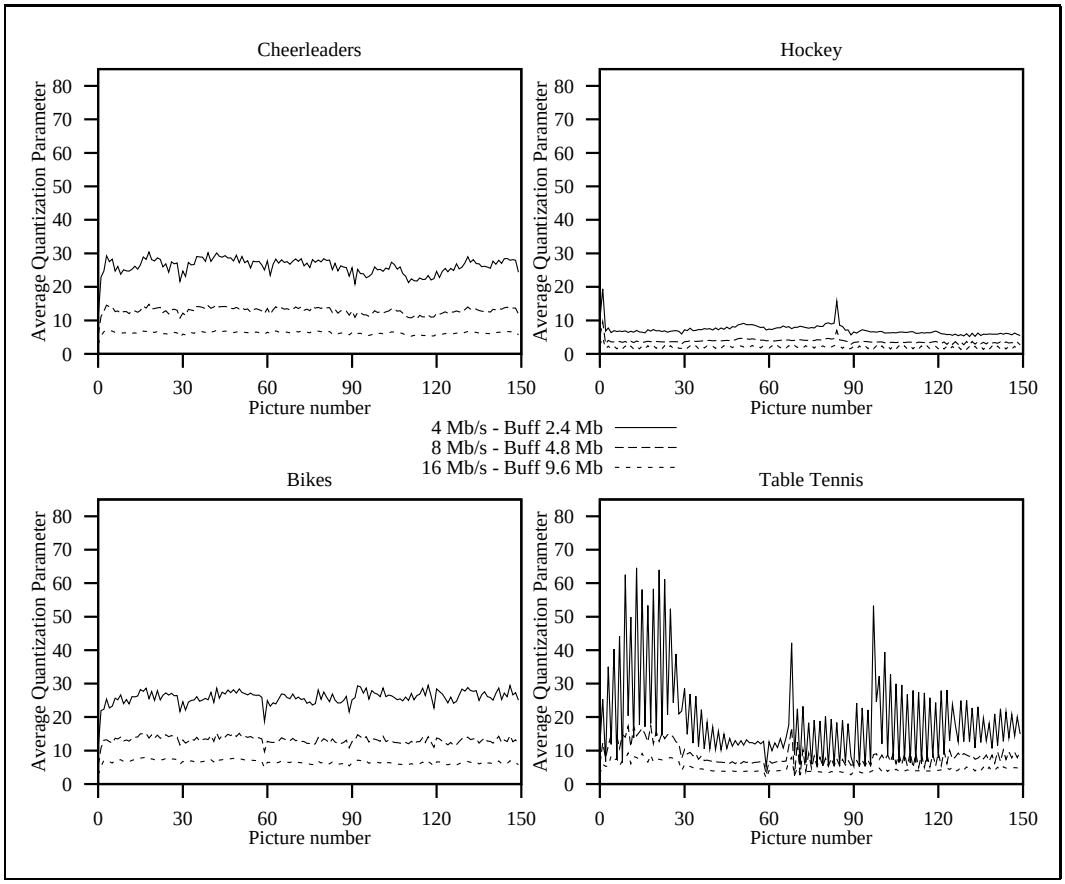

Figure 6.62: Average Quantization Parameter in CBR MPEG Encoding with $N = 30$.

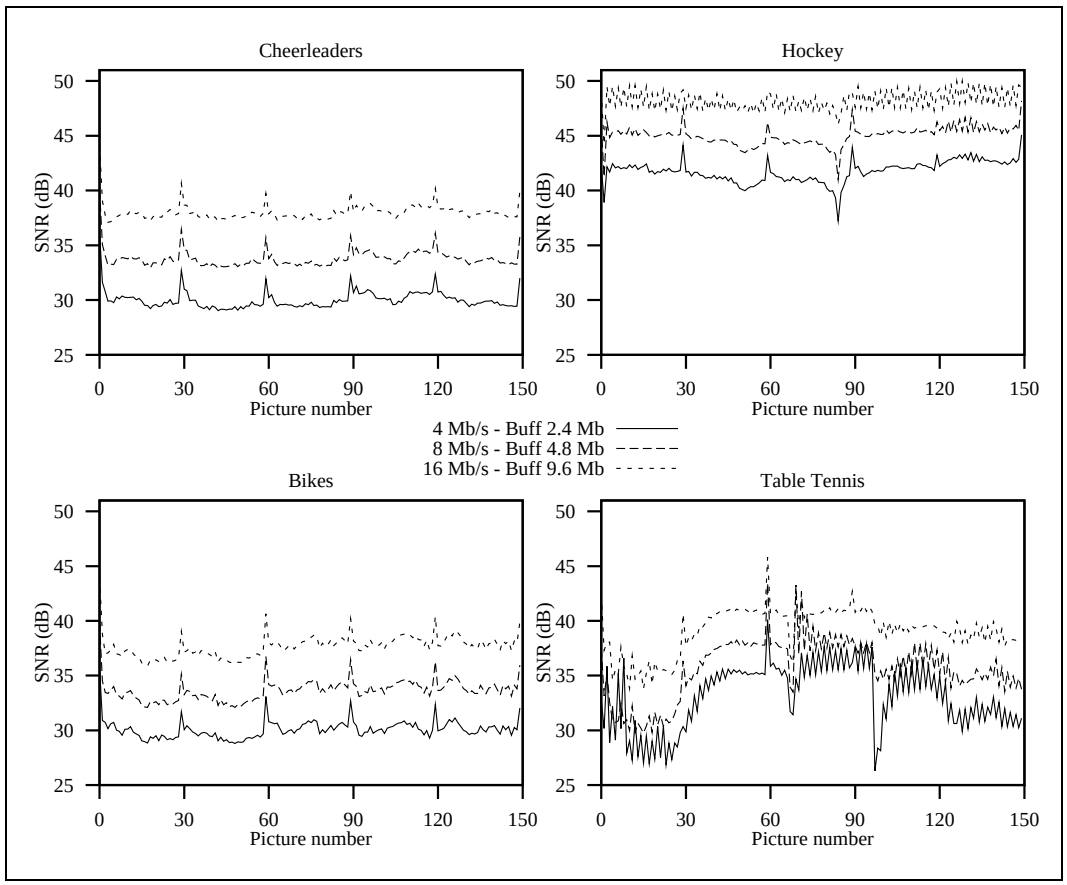

Figure 6.63: SNR in CBR MPEG Encoding with $N = 30$.

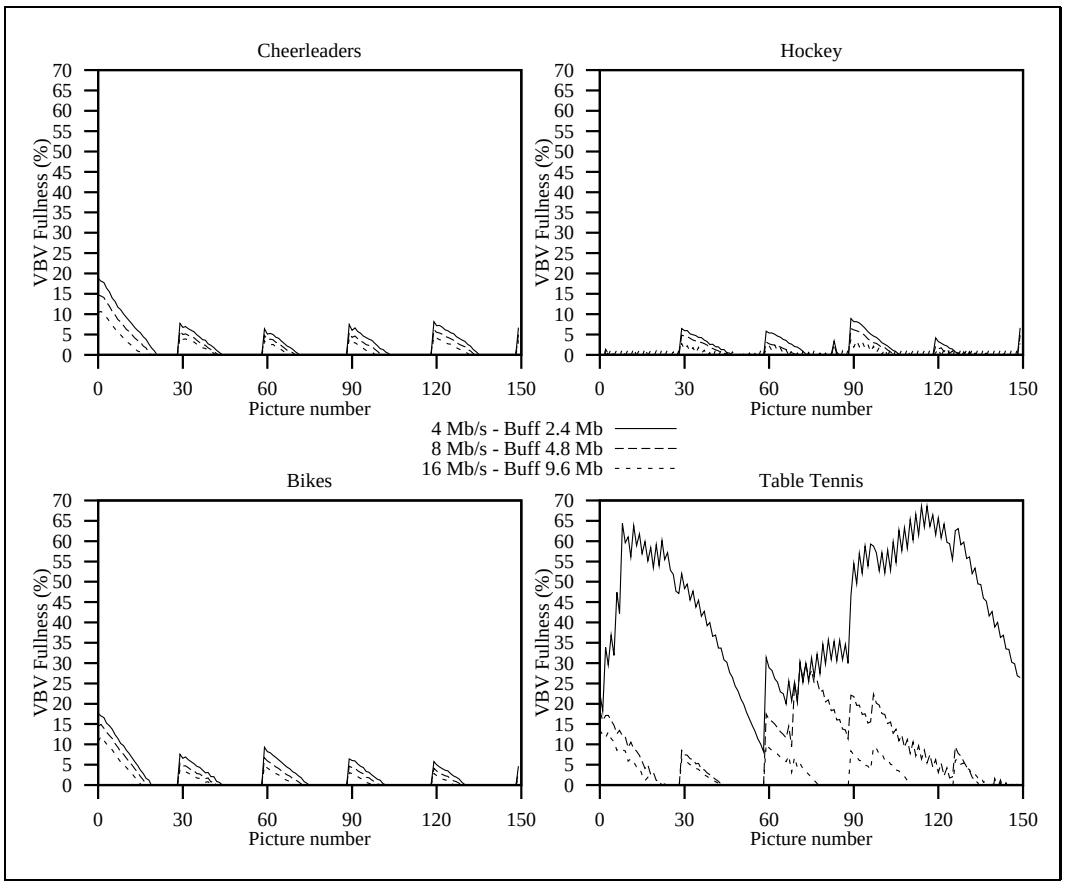

Figure 6.64: VBV Fullness in CBR MPEG Encoding with $N = 30$.

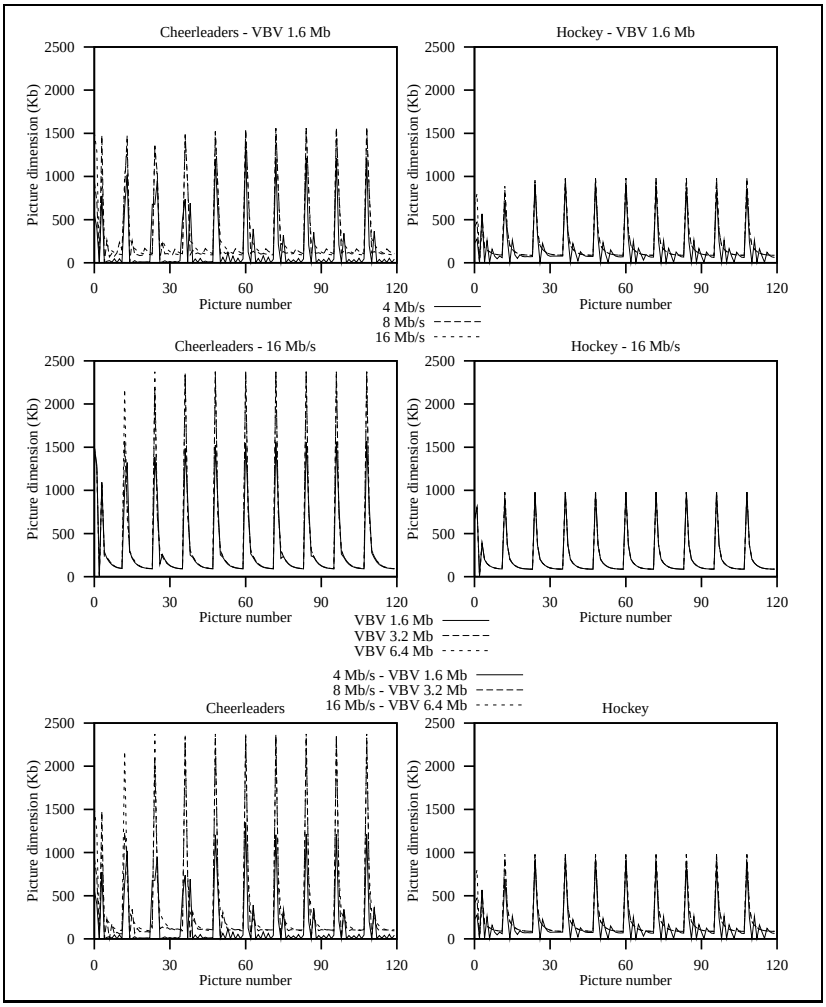

Figure 6.65: Dimension of Pictures in CBR MPEG Encoding of a Static Scene

is larger than when the corresponding non static scene is encoded at the same target rate. P-frames are small because they must only encode the difference between their reference picture and the raw picture that is due to the quantization. Thus, the SNR delivered by P-frames is high, as shown in Figure 6.67.

The plot obtained by encoding the "cheerleaders" scene at 16 Mb/s with various VBV dimensions (second row, first column in Figure 6.65) shows that the larger is the VBV dimension, the largest are the I-frames. This does not hold for the "hockey" scene (second row, second column) because, being the pictures less detailed, even though the rate control function would allow a larger amount of bits to be produced, I-frames do not grow because the quantization parameter has the minimum allowed value and cannot be further reduced (see Figure 6.66).

Figure 6.68 shows that in many cases the VBV is (almost) completely filled, while in others the fullness level is of the order of the 20 %. Nevertheless, when dealing with real-time encoding (such as in videoconferencing applications), the scene to be encoded is not known in advance. If the videoconferencing system must be able to deliver a good quality, especially when the camera is statically set on some image, the VBV must be large enough to allow I-frames to grow and capture the details of the image[13] (e.g., second row, first column in Figure 6.68). When pictures are not detailed (e.g., second row, second column in Figure 6.68) or the motion is higher (Figure 6.60) only a fraction of the VBV is filled. In any case, the coding shaping delay is large because it must be set according to the VBV dimension in order to guarantee continuous playing.

6.5.4 Dedicated Link between Sender and Receiver

For the sake of completeness of this study, the system configuration in which sender and receiver are connected through a dedicated link (Figure 6.69) is taken into account. Having the whole link capacity at disposal for sending the CBR MPEG stream does not provide particular advantages. Only a fraction of the link capacity, corresponding to the target rate B of the encoder is actually exploited and the coding shaping delay provides a significant contribution to the end-to-end delay.

[13]Note that as the scene is not moving, the human eye is more sensitive to errors.

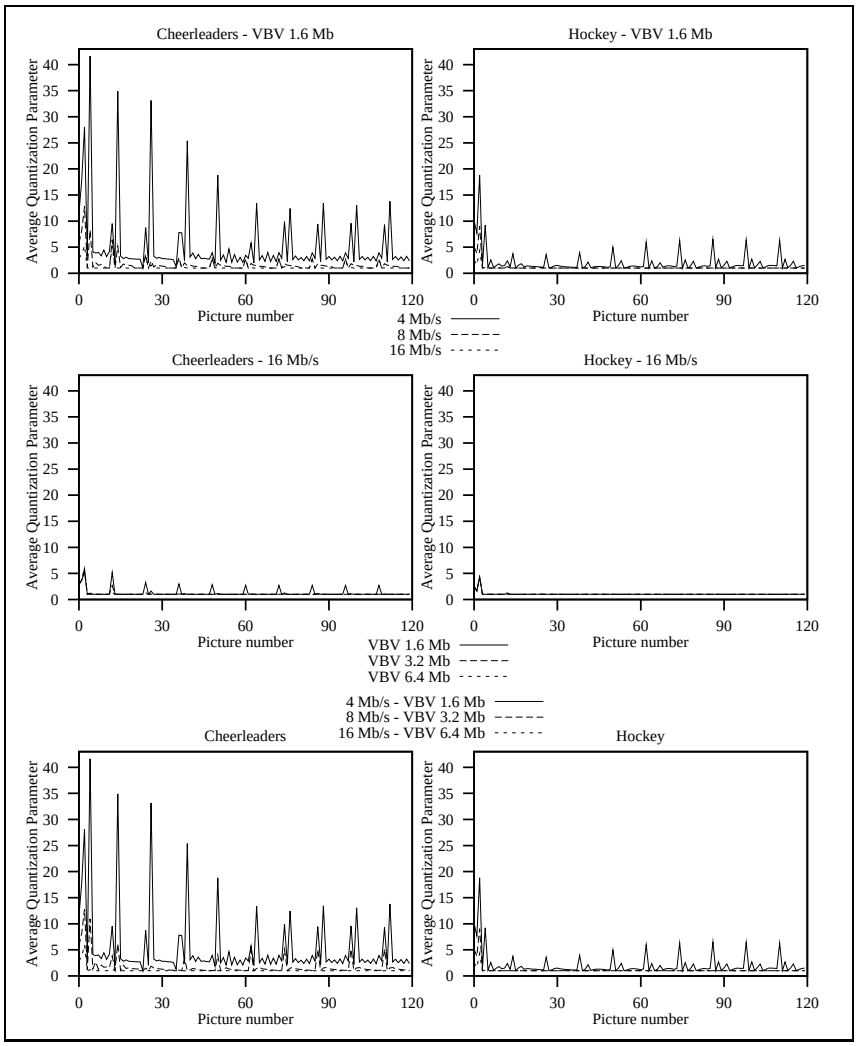

Figure 6.66: Average Quantization Parameter in CBR MPEG Encoding of a Static Scene

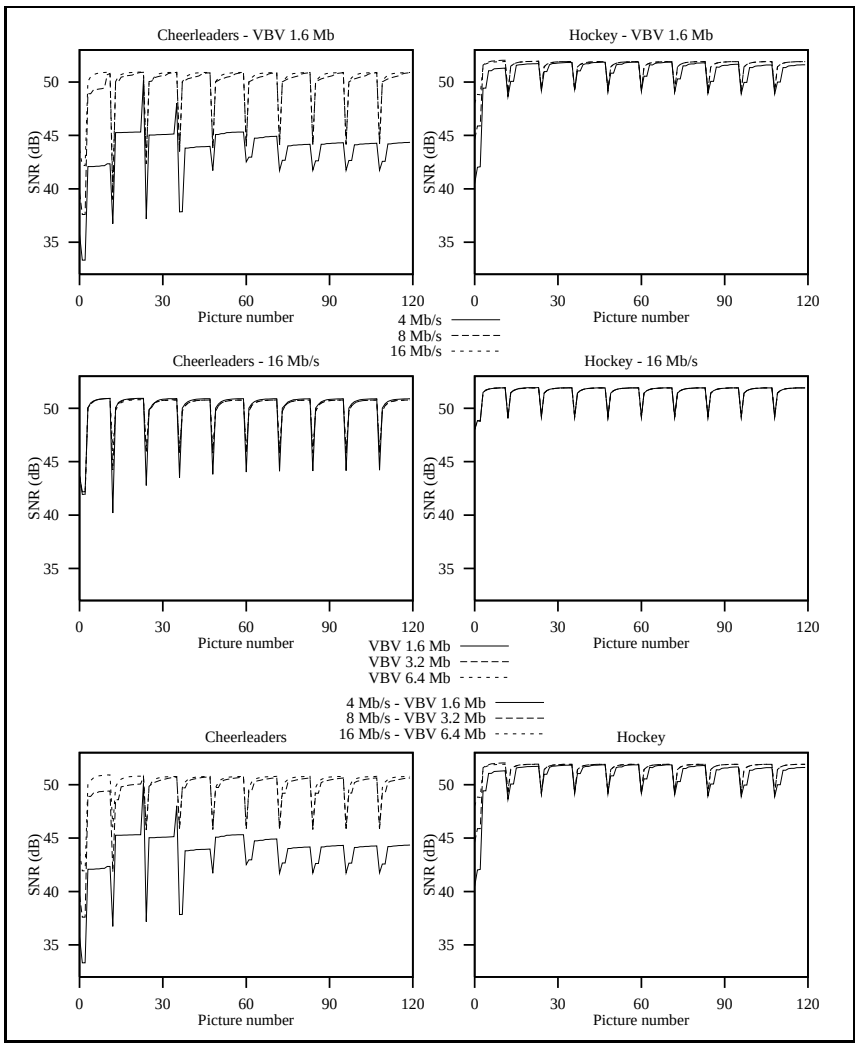

Figure 6.67: SNR in CBR MPEG Encoding of a Static Scene

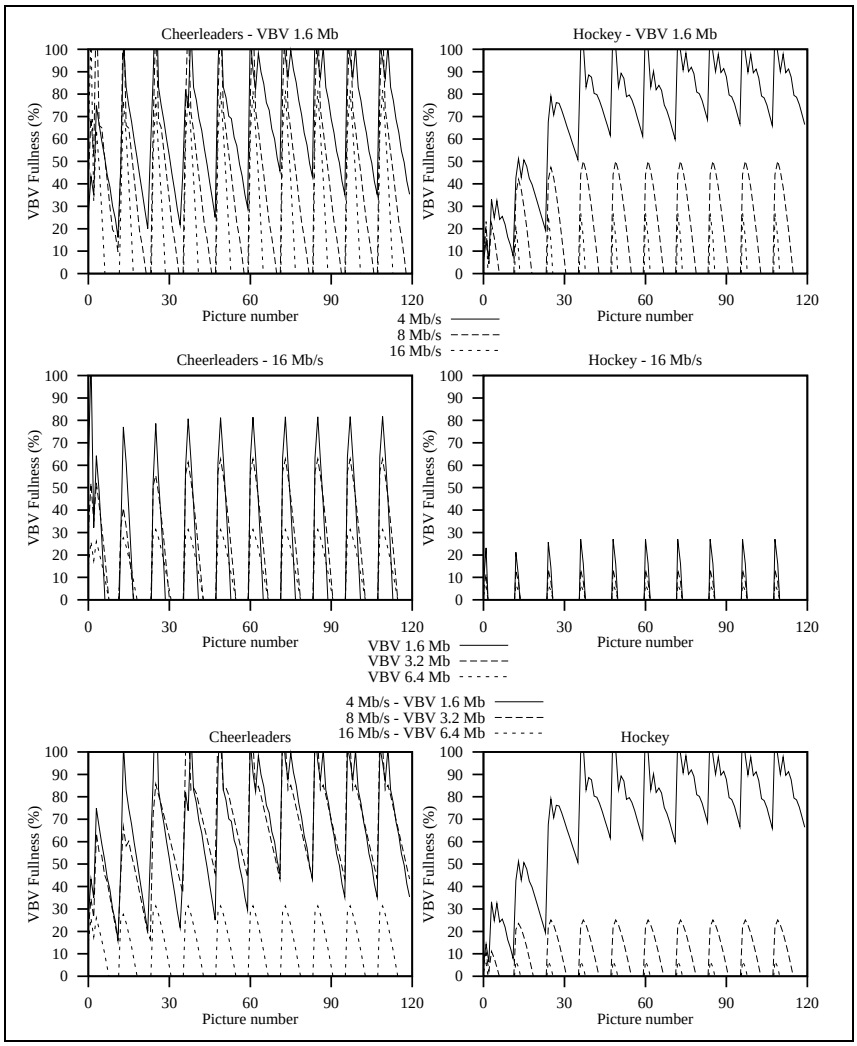

Figure 6.68: VBV Fullness in CBR MPEG Encoding of a Static Scene

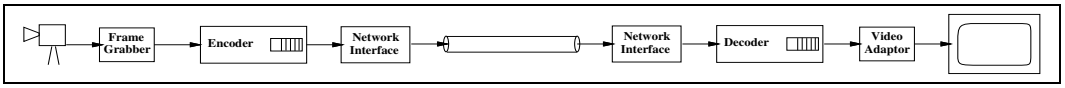

Figure 6.69: Architecture of a Videoconferencing System Exploiting a CBR MPEG Encoder and a Dedicated Link between Sender and Receiver.

Given the propagation delay P, which depends on the distance between sender and receiver, the end-to-end delay of the system is

$$\Delta_{CBR}^{Ded} = S_c + S_s + P + D + P_d \tag{6.28}$$

where D is the decoding delay, S_c is the coding shaping delay, and S_s is the startup shaping delay. This represent a lower bound in the end-to-end delay obtainable from a videoconferencing system exploiting CBR MPEG encoding.

6.5.5 Circuit Switching

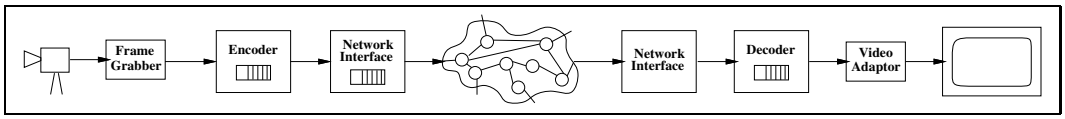

Figure 6.70: Architecture of a Videoconferencing System Exploiting a CBR MPEG Encoder and a Circuit Switched Network.

CBR MPEG encoding is suited for transmission over circuit switched networks: when a CBR MPEG encoder is used to build a videoconferencing system (Figure6.70), the most effective way to connect sender and receiver is through a circuit that has a bandwidth equal to the target rate of the encoder. The contribution of the network to the end-to-end delay is the switching delay Sw and the propagation delay P which depends on the distance between sender and receiver. The end-to-end delay of the system is

$$\Delta_{CBR}^{CS} = S_c + S_s + Sw + P + D + P_d \tag{6.29}$$

where D is the decoding delay, S_c is the coding shaping delay, and S_s is the startup shaping delay. This delay almost equals the one obtained with a dedicated link between sender and receiver: the only further contribution is the switching delay. Thus, it can be said that it represents as well a lower bound in the end-to-end delay obtainable from a videoconferencing system exploiting CBR MPEG encoding.

6.5.6 Packet Switching with Time-Driven Priority

CBR MPEG encoding is best suited for transmission over circuit switched networks. Nevertheless, in the following two Sections we consider the end-to-end delay obtained when CBR encoded video is sent through a packet switched network. In this Section, time-driven priority is considered to be exploited in the network. The network shaping delay introduced in the sender and receiver is analyzed first; then, the end-to-end delay is considered.

Network Shaping Delay

When time-driven priority is used in the network, the time cycle is set to T_c, the video frame period, and $B \cdot T_c$ bits must be reserved in each time cycle. The bits exiting the CBR MPEG encoder are buffered in the buffer of the packetization function depicted in Figure 6.71, until they can be included in a packet. Each bit experiences a different delay, but due to Requirement 3 of continuous playing, a network resynchronization delay must be introduced on the receiver side so that each bit experiences the same whole delay S_n^{Pack} which is called the *packetization* component of the network shaping delay.

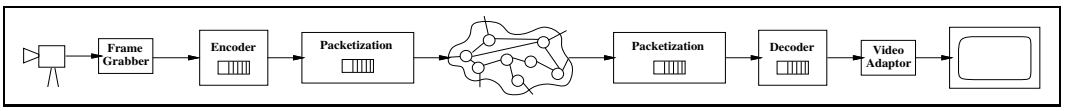

Figure 6.71: Architecture of a Videoconferencing System Exploiting a CBR MPEG Encoder and a Packet Switched Network with Time-Driven Priority.

The packetization function on the receiver side introduces the network resynchronization delay; the packetization function knows the delay experienced in the buffer of the sender packetization function by the bits of each received packet and can compensate it. The compensation is done by buffering the bits and providing to the decoder a bit flow at the constant rate B. If the TFs reserved to the videoconference call are evenly distributed in the time cycle, no network resynchronization is required.

The dimension of the buffer of both packetization functions must be at least $B \cdot S_n^{Pack}$. P_n^{Pack} is the maximum time a bit spends in the sender buffer before being retrieved and inserted in a packet; the buffer, which is filled at constant rate B, does not overflow if it can store the $B \cdot S_n^{Pack}$ bits generated by the

decoder in the meantime. S_n^{Pack} is as well the maximum time spent by a bit in the receiver buffer; the buffer cannot contain more than $B \cdot S_n^{Pack}$ bits otherwise the last one would take more than S_n^{Pack} to exit at the constant rate B.

The actual value of S_n^{Pack} depends on how the TFs have been reserved within the time cycle for the videoconference; thus, it is fixed at resource reservation time and is constant over the whole videoconference call (or until the bandwidth is reallocated). If, for example, a single TF is reserved, $S_n^{Pack} = T_c$ because the first bit exiting the CBR encoder after a packet has been assembled and sent has to wait a whole time cycle before being inserted in a packet. Intuitively, the smaller and the more uniformly distributed inside the time cycle are the packets, the shorter the network shaping delay S_n^{Pack}.

Exploitation of small and uniformly distributed packets is encouraged also by other arguments. In Section 6.4.3 it has been shown that, if the decoder starts decoding as soon as some of the bits encoding a picture enter its buffer, the decoding time is reduced by using small packets. In [35] is shown that reserving a small amount of bits in many TFs is convenient also from the point of view of the network because balancing the load along the time cycle provides lower blocking probability. Using small packets has a further advantage when time-driven priority is exploited in the transmission of CBR MPEG encoded video.

To identify the schedule that minimizes the network shaping delay, the structure of the MPEG stream must be taken into account. The MPEG standard specifies that the bit stream produced by a CBR MPEG encoder be structured as a *transport stream* (see Section A.5) in order to make more robust its transmission over packet switched networks. The transport stream is composed of packets having fixed dimension of 188 bytes; each packet contains bits encoding a single picture. The network shaping delay can be minimized if the TFs are reserved so that as soon as an MPEG packet is produced, a reserved TF is scheduled and the packet is sent[14]. This requires to reserve 188 bytes in TFs equally spaced by $(188 \cdot 8)/B$ and have the encoder synchronized with the network interface so that the last bit of each MPEG packet is

[14]Breaking an MPEG packet and sending it during two TFs decreases robustness to losses and errors.

emitted just before a reserved TF is scheduled[15]. If the encoder is not synchronized with the network interface, there is an application synchronization component of the network shaping delay $S_n^{AS} \in [0,(188 \cdot 8)/B]$.

In the general case, the network shaping delay is given by the composition of the two components, i.e.,

$$S_n = S_n^{AS} + S_n^{Pack}$$

and it is experienced partly in the sender and partly in the network resynchronization function of the receiver.

Excess Resynchronization Delay

In what has been discussed so far, the network resynchronization delay is assumed to be introduced by an ad hoc function which can be implemented, for example, in the software layers below the application. Nevertheless, it can also be introduced in the decoder buffer; buffering is not required in the receiver software below the decoder[16] and the decoder buffer must be enlarged. If the decoder is synchronized with the network interface, the buffer must be enlarged by, at least, $B \cdot S_n^{Pack}$. This stems from the fact that when time-driven priority is exploited in the network, the packetization component of the network shaping delay is the only variable component of the network delay.

If the decoder is not synchronized with the network interface, when the first packet is received the decoder does not know how much time the bits have spent in the buffer of the sender packetization function. Thus, the decoder uses the conservative assumption that the bits have experienced the minimum delay and the decoding of pictures must be delayed accordingly[17]. If the

[15]Actually, even in this situation, the bits of each packet has to be buffered in the packetization function until all the 188 bytes have been sent out of the encoder. However, this time does not contribute to the end-to-end delay because it is already taken into account by the coding shaping delay: the bits would be anyway kept in the decoder buffer to introduce the processing resynchronization delay. The only impact on the end-to-end delay is a possible increase of the decoding time if the decoder has an access unit smaller than the MPEG packet.

[16]Actually, a buffer is needed only to buffer each incoming packet while the header is processed, but this is not relevant to this work.

[17]As explained in Section 6.3.5, the replay buffer shows the same behavior when introducing the network resynchronization delay to compensate the variation of the queuing

assumption is true, the network shaping delay experienced by each bit partly in the sender packetization function, partly in the decoder buffer is actually S_n^{Pack} and the decoder buffer must be enlarged by, at least, $B \cdot S_n^{Pack}$. If the assumption is not true, a excess resynchronization delay E_r is introduced, i.e., bits are buffered longer than necessary; the decoder buffer must be dimensioned accordingly so that it doesn't overflow due to the excess buffering time E_r. In the following, the maximum value of E_r and the minimum dimension of the buffer are calculated.

The largest excess resynchronization delay is introduced when the first bit in the first received packet has already experienced a delay S_n^{Pack} in the buffer of the sender packetization function. The decoder should not introduce any network resynchronization delay; instead the decoding is delayed because it is assumed that the bits experienced minimum delay in the packetization function. In order to calculate the time spent by the first packet in the decoder buffer, it is necessary to compute the minimum delay experienced in the sender packetization function.

Bits experience minimum delay due to packetization, if a packet of dimension P_f is sent as soon as the CBR encoder emits the bit number P_f. In this case, the first bit in the packet had been in the buffer of the sender packetization function for the time the CBR encoder takes to produce P_f bits, i.e., P_f/B. Since decoding of the first packet must start after S_n^{Pack} from when the first bit was produced by the CBR encoder, the network resynchronization delay experienced in the decoder buffer (which is also the upper bound on the excess resynchronization delay E_r) is

$$S_n^{Pack} - \frac{P_f}{B} \tag{6.30}$$

The overall time spent by bits in the decoder buffer due to network resynchronization delay and excess resynchronization delay is thus given by

$$S_n^{Pack} + E_r$$

where

$$E_r \in \left[0, S_n^{Pack} - \frac{P_f}{B}\right]$$

delay in asynchronous packet switched networks.

The decoder buffer must be oversized according to the worst case, i.e., in order to introduce the network resynchronization delay and be robust against the excess resynchronization delay, it must be enlarged by

$$\left(2 \cdot S_n^{Pack} - \frac{P_f}{B}\right) \cdot B$$

The actual value of $S_n^{Pack} \in [0,T_c]$ depends on the TFs reserved for the videoconference call. Thus, S_n^{Pack} is known at reservation time by the network interface; this can be provided with a mechanism to communicate it to the decoder. If such a mechanism is not available, the decoder does not know the actual value of S_n^{Pack} and delays decoding according to the worst case $S_n^{Pack} = T_c$; i.e., decoding is started after

$$T_c - \frac{P_f}{B} \tag{6.31}$$

from the reception of the first packet. If the actual packetization component of the network shaping delay is null (because TFs had been allocated optimally) and the first packet has experienced S_n^{Pack} in the sender, it is useless to delay decoding by (6.31), i.e., this value is the excess resynchronization delay. In particular (6.31) is the maximum E_r value; the minimum excess delay is introduced when the bits in the first packet has experienced minimum delay and decoding should be delayed by (6.30):

$$E_r \in \left[T_c - S_n^{Pack}, T_c - \frac{P_f}{B}\right]$$

The decoder buffer must thus be enlarged by

$$\left(S_n^{Pack} + T_c - \frac{P_f}{B}\right) \cdot B$$

This shows that network resynchronization can be performed inside the decoder; this eliminates the need for an ad hoc buffer. Nevertheless, if the decoder is not tightly connected to the network interface (i.e., it does not have information about the schedule of the reserved TFs and the TF in which each packet is received), the excess resynchronization delay is larger.

End-to-end Delay

The general expression for the end-to-end delay is

$$\Delta_{CBR}^{TDP-SH} = S_c + S_s + L \cdot T_f + S_n + E_r + D + P_d \tag{6.32}$$

where

- S_c is the coding shaping delay introduced by the encoder/decoder system to produce the encoded stream at the constant target rate;
- S_s is the startup shaping delay introduced by the encoder to avoid the encoder buffer to overrun when the natural rate is smaller than the target rate;
- $L \cdot T_f$ is the time taken by a packet to travel through the network from source from destination,;
- $S_n = S_n^{Pack} + S_n^{AS}$:
  - $S_n^{Pack} \in [0,T_c]$ depends on the number, dimension, and position of the TFs reserved to the videoconference call and is constant over the duration of the call,
  - $S_n^{AS} \in [0,T_c]$ is due to the lack of synchronization between encoder and network interface; because of the relative drift of the two timing references, this delay can vary inside the given interval during the videoconference call;
- $E_r \in [0,T_c - P_f/B]$ is not null when the network resynchronization delay is introduced by the decoder and it is not synchronized with the network interface.

If resource reservation is optimal ($S_n^{Pack} = P_f/B$), the encoder is synchronized with the network interface ($S_n^{AS} = 0$), and network resynchronization is performed with information from the network interface ($E_r = 0$), (6.32) is very similar to (6.29). The terms $Sw + P$ appearing in the latter are substituted by the term $L \cdot T_f$ in the former; all of these terms depend on the propagation delay and the number of nodes on the path between sender and receiver. When the sender is topologically close to the receiver, the coding

shaping delay S_c is the main contribution in both Δ_{CBR}^{CS} and Δ_{CBR}^{TDP}. For example, if S_c is around $3 \cdot T$ at 30 frames per second, and the propagation delay P and switching delay Sw are negligible, $\Delta_{CBR}^{CS} \simeq 100\ ms$. If a packet switched network with time-driven priority is exploited in the same scenario, it is likely that packets are forwarded by one router, i.e., $L = 3$. Moreover, if small packets are used ($P_f = 188$ bytes) and the rate of the CBR encoder is 1.5 Mb/s,

$$\Delta_{CBR}^{TDP} = \Delta_{CBR}^{CS} + S_n^{Pack} + 3 \cdot T_f \simeq 100 + 1 + 0.375 \simeq \Delta_{CBR}^{CS}$$

Thus, the short distance transmission of CBR MPEG video over a packet switched network with time-driven priority yields almost the same end-to-end delay as the transmission over a circuit switched network.

If sender and receiver are not close (i.e., the propagation delay is not negligible), the difference between Δ_{CBR}^{TDP} and Δ_{CBR}^{CS} depends on the speed and the number of intermediate nodes on the path. The faster the packet switches, the smaller the difference.

A suboptimal schedule of the TFs reserved for the videoconference call increases the end-to-end delay; nevertheless, suboptimal schedules can be accepted in order to reduce the probability that calls be blocked.

6.5.7 Asynchronous Packet Switching

Transmission of a CBR MPEG stream over a packet switched network requires the stream to be segmented in packets and then reconstructed in the receiver. In order to better focus on the issues related to packetization, in this Section a configuration in which sender and receiver are connected by a direct link is first considered. Then, a multi-hop packet switched network between sender and receiver is taken into account.

Single-Hop Packet Network

With the system configuration depicted in Figure 6.72, sender and receiver are connected through a direct dedicated link on which they send data in packets, as shown by the packetization function in Figure 6.72. This configuration is not necessarily used in real applications, but allows us to focus on the delay due to packetization.

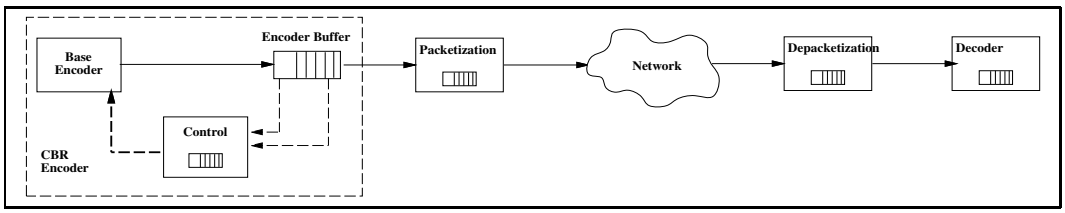

Figure 6.72: MPEG Encoder and Packetization Function.

Assumption 5 *Transmission of a packet cannot start until all the bits that are to be put in its payload has been produced by the encoder.*

This Assumption conforms with reality since many existing protocols need to insert in the header of the packet a check sequence that can be calculated only from the whole packet payload. Nevertheless, due to this Assumption the packetization function introduces a delay in the system. The bits produced by the encoder are stored in the buffer of the sender packetization function. Given a packet size P_s, when the buffer contains P_s bits it is emptied (as shown in Figure 6.73) and a packet is sent over the network.

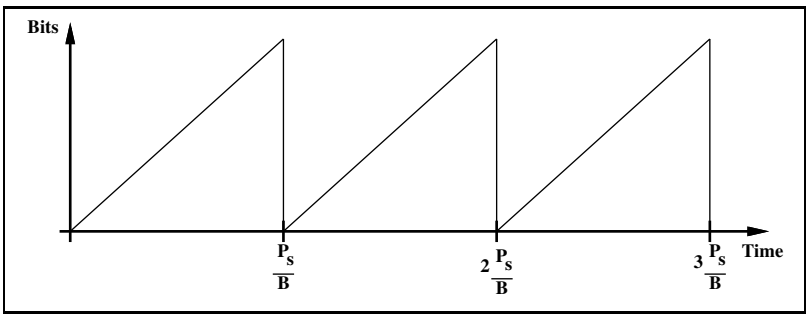

Figure 6.73: Fullness of the Buffer of the Packetization Function.

Since bits are produced by the CBR MPEG encoder at constant rate B, packets leave the sender at regular intervals P_s/B, as shown in Figure 6.73. The generated traffic has the same characteristics of data exiting a leaky bucket with token generation rate B equal to the target rate of the encoder and token bucket size $A = P_s$. Packet size must be chosen according to the structure of the MPEG transport stream so that a single MPEG packet is not sent into two packets. Moreover, we assume that a packet is sent each time the last bit encoding a picture exits the encoder, independently from its dimension:

Assumption 6 *A packet contains bits encoding only one picture.*

Thus, the real behavior of the packetization buffer can be different from the one depicted in Figure 6.73, but the basic principles discussed in this Section still hold.

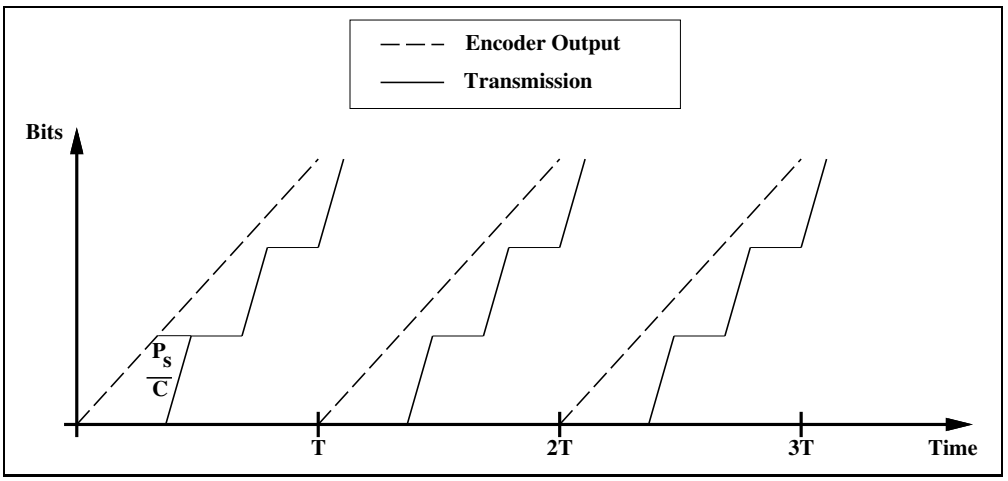

Figure 6.74: Generation and Transmission of Bits; Fullness Level of the Decoder Buffer.

As shown by Figure 6.74, packets are sent at the full speed C of the link connected to the sender. Each bit of a packet experiences a different delay from when it is generated by the CBR encoder to when they it is put on the link.

No Buffering in the Depacketization Function We first assume that packets get directly to the decoder buffer upon arrival at the receiver, as shown by Figure 6.75; the depacketization function has the only task of stripping off the packet header[18].

Each bit of a packet experiences a different delay due to packetization; nevertheless we are interested in the delay experienced by the whole picture. The system target decoder specified by the MPEG standard, the bits encoding a picture are removed from the decoder buffer all together when decoding

[18]This assumption is not realistic as the packetization function usually needs to buffer the whole packet in order to possibly check the error control code. This is not relevant to what is explained in the following: it means that Figure 6.75 should be modified by having each packet enter the decoder buffer all at once.

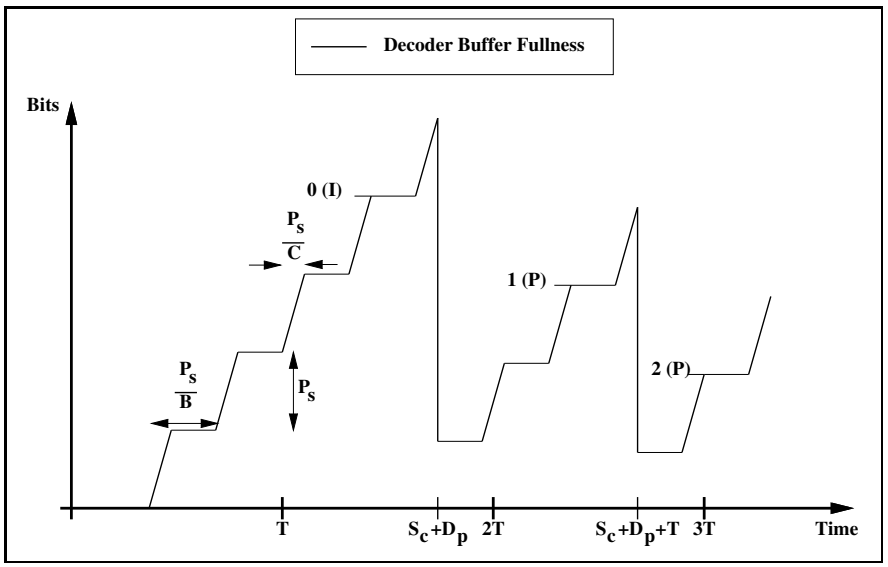

Figure 6.75: Fullness Level of the Decoder Buffer.

starts. According to Assumption 6, the last bit of a picture is the last bit in a packet; in the worst case a time P_s/C elapses from when this last bit exits the CBR MPEG encoder to when it enters the decoder buffer. Thus, decoding is delayed by the *packetization delay* D_p that is the transmission delay of a packet, i.e.:

$$D_p = \frac{P_s}{C} \tag{6.33}$$

Maximum dimension packets experience the packetization delay entirely in the sender packetization function; packets of dimension $P < P_s$ spend $(P_s - P)/C$ in the decoder buffer. Since no further data arrive during this time and after it the whole picture is retrieved, the decoder buffer must not be oversized. Considering also the delay introduced by the encoder/decoder system (S_c, S_s, and D) and the propagation delay, the end-to-end delay is given by

$$\Delta_{CBR}^{Async} = S_c + S_s + D_p + P + D + P_d \tag{6.34}$$

The packet size shall be chosen as small as possible (possibly equal to the dimension of MPEG packets) in order to decrease the packetization delay

and possibly the decoding time if the decoder processes bits as soon as they enter its buffer.

Buffering in the Depacketization Function We assume now that the depacketization function in the receiver reconstructs the CBR MPEG stream, as shown in Figure 6.76, by buffering incoming packets and feeding the decoder buffer at the constant rate B. Received bits are feeded to the decoder only after the whole packet has been received. Thus, bits exiting the CBR MPEG encoder are buffered in the buffer of the packetization (sender) and depacketization (receiver) functions for an overall time

$$D_p = \frac{P_s}{B} + \frac{P_s}{C} \tag{6.35}$$

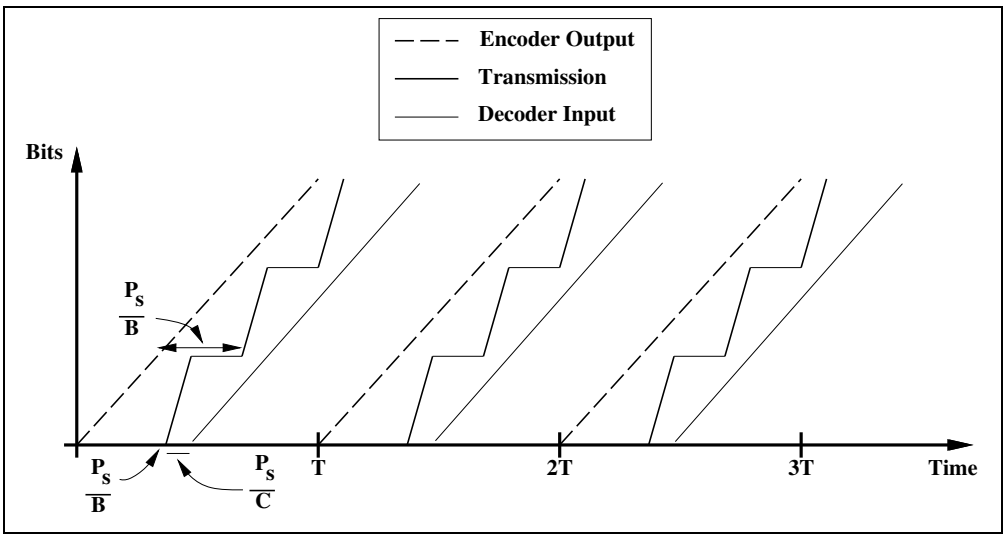

Figure 6.76: Generation and Transmission of Bits and Output of the Receiver Packetization Function.

The packetization delay introduced in this system configuration is P_s/B larger than the one given by Equation (6.33). Thus, if the depacketization function reconstructs a CBR MPEG stream, the end-to-end delay is increased. The P_s/B increment is necessary if Assumption 6 is dropped and the last bit of a picture must not be the last of the packet in which it is sent. In fact, the sender packetization function delays the first bit of each packet by P_s/B, as shown in Figure 6.76; if decoding is not delayed by the same

amount of time, the decoder buffer underflows when, for example, the last bit of a picture happens to be the first in a packet.

Multi-Hop Configuration

Figure 6.77 shows the architecture of a videoconferencing system that exploits a CBR MPEG encoder and an asynchronous packet switched network. This configuration adds the queuing delay, the network resynchronization delay, and the excess resynchronization delay E_r to the end-to-end delay given by Equation (6.34):

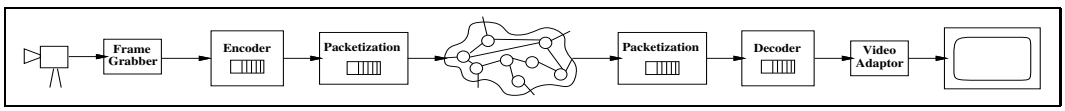

Figure 6.77: Architecture of a Videoconferencing System Exploiting a CBR MPEG Encoder and an Asynchronous Packet Switched Network.

$$\Delta_{CBR}^{Async} = S_c + S_s + D_p + P + Q_M + E_r + D + P_d \tag{6.36}$$

where P is the propagation delay on the links on the path from source to destination, Q_M is the maximum queuing delay experienced by packets in the network, $E_r \in [0,\Delta Q]$ is the excess resynchronization delay introduced by the replay buffer (being it in the packetization function or in the decoder) due to lack of synchronization between sender and receiver network interfaces (see Section 6.3.5), ΔQ is the variation of the queuing delay. The packetization delay D_p is given by Equation (6.33) or Equation (6.35) depending on the behavior of the depacketization function (see Section 6.5.7).

Since bits are produced at the constant rate B and packets are sent at constant pace, the dimension of the replay buffer must be, at least,

$$2 \cdot \Delta Q \cdot B$$

This expression provides also the increment required in the dimension of the decoder buffer in case it is used to compensate the network delay variation.

Packetization and Startup Shaping Delay

The packetization function can impact the startup shaping delay. In many practical cases the mean bit production rate is higher than the target rate; thus the startup shaping delay is essential only while encoding the first MBs of a picture in order to compensate a possible instantaneous low bit production rate. If the packetization function is included in the closed control loop of the encoder, the startup shaping delay can be reduced or even eliminated. The control function takes into account that bits are not removed from the encoder buffer for a time $\frac{P_s}{B}$. Thus, even if the initial bit production rate is low, the average over this time interval can be sufficient to produce a packet worth of bits.

6.6 Summary

In this work we analyzed the end-to-end delay of videoconferencing over packet switched networks. Our key findings are summarized in Table 6.2. The main design objective is to keep the end-to-end delay below 100 ms. This requirement comes from the human hearing sensitivity for delays larger than 100 ms and the requirement for lip-synchronization, i.e., the need for the audio and video to be synchronized.

Controlling the delay also requires to control the amount of buffers used at the (i) sender, (ii) network and (iii) receiver. The control of the buffer sizes can have adverse consequences:

1. Decreasing the buffers inside the network can increase the packet loss inside the network and degrade the quality of the received video.

2. Limiting the MPEG encoder buffer size will limit the maximum I-frame size and degrade the compressed MPEG video quality.

Thus, what we found and formulated, in this study, are some tradeoff between the perception quality due to delay and the received picture quality due to loss and compression.

We found interesting results some of which are counter intuitive. This in turn illustrates the importance of this sort of study.

	Dedicated Link	Circuit Switching	Time-Driven Priority	Asynchronous Packet Switching
Raw Video	$\Delta_{Raw}^{Ded} = \frac{F_r}{C} + P + P_d$	$\Delta_{Raw}^{CS} = S_n + P + Sw + P_d$	$\Delta_{Raw}^{TDP} = S_n + L \cdot T_f + P_d$ $S_n = S_n^{AS} + (N_r - 1) \cdot T_f$ $S_n^{AS} \in [0, T]$	$\Delta_{Raw}^{Async} = \frac{F_r}{C} + P + Q_M + E_r + P_d$ $E_r \in [0, \Delta Q]$ $\Delta_{Raw}^{Async-Sh} = S_n + \frac{P_s}{C} + P + Q_M + E_r + P_d$
VBR MPEG	$\Delta_{VBR}^{Ded} = C_M + P + D + P_d$	$\Delta_{VBR}^{CS} = C_M + Sw + P + D + P_d$ $\Delta_{VBR}^{CS-TS} = C_M + S_n^{CS} + P + Sw + D + P_d$	$\Delta_{VBR-I}^{TDP} = C_M + S_n^{AS} + L \cdot T_f + D + P_d$ $S_n^{AS} \in [0, T]$ $\Delta_{VBR}^{TDP-CxSc} = C_M + S_n^{Sched} + L \cdot T_f + D + P_d$ $S_n^{Sched} \in [0, N \cdot T]$	$\Delta_{VBR}^{Async} = C_M + \frac{P_s}{C} + P + Q_M + E_r + D + P_d$ $\Delta_{VBR}^{Async-TS} = C_M + S_n^{TS} + \frac{P_s}{C} + P + Q_M + E_r + D + P_d$
CBR MPEG	$\Delta_{CBR}^{Ded} = S_c + S_s + P + D + P_d$	$\Delta_{CBR}^{CS} = S_c + S_s + Sw + P + D + P_d$	$\Delta_{CBR}^{TDP} = S_c + S_s + L \cdot T_f + S_n + E_r + D + P_d$ $S_n = S_n^{Pack} + S_n^{AS}$ $S_n^{Pack} \in [0, T_c]$ $S_n^{AS} \in [0, T_c]$ $E_r \in [0, T_c - P_f/B]$	$\Delta_{CBR}^{Async} = S_c + S_s + D_p + P + Q_M + E_r + D + P_d$ $D_p = \frac{P_s}{B} + \frac{P_s}{C}$

Table 6.2: Summary of the Configurations Considered in this Work.

1. Transmission of raw video does not necessarily provide the shorter delay. This is because of the transmission time needed for large number of bits of high definition pictures.

2. MPEG CBR encoding of a fixed scene introduces long delay. This is because the inter-frame coding of P-frames requires few bits, and therefore, a single I-frame can use all the CBR capacity allocated to the group of picture (GOP). As a result, the I-frame transmission will last the entire GOP period.

3. Using asynchronous packet switching with statistical multiplexing is challenging. This is because the distribution of the delay variation or jitter inside the network can be large and with heavy tail. Thus, conservative design with large replay buffer would result in large delay, and small replay buffer would result in occasional underflow/overflow of the replay buffer and distortion of the video viewed by the user at the receiver.

4. If the capture card and the display buffer are using the same reference clock the delay can be decreased by one video frame period. For 10 frames per second this means delay reduction of 100 ms and for 30 frames per second the delay reduction is 33 ms.

5. MPEG VBR video over asynchronous packet switched network requires high equivalent capacity or effective bandwidth, which can be too much to be wasted over wide area links. Therefore, in some case it may be better to use only I-frames, i.e., to use for example Motion-JPEG.

In conclusion, MPEG-based videoconferencing is possible, however, in order to keep the end-to-end delay below human perception, there are four requirements:

1. The capture (frame grabber) card and display (video) adapter should have a common time reference clock.

2. The pictures should be sent right after the compression is completed as a variable bit rate (VBR) stream.

3. The network jitter should be controlled with a well defined bound.

We also showed that with time-driven priority and complex scheduling it is possible to have the following properties for VBR MPEG:

1. Bound of 250 μseconds on the jitter.
2. No loss even if the link is fully utilized.
3. The end-to-end delay is dominated by the propagation delay plus $L \cdot 125\mu$seconds (L depends on the number of hops). This delay is shorter than the delay that can be obtained over a circuit switched network. On a circuit switched connection a CBR encoder must be deployed which introduces a coding shaping delay larger than the video frame period. If the scene is slow, this delay can be as large as the duration of a group of pictures.

Chapter 7

Conclusions

The advances in computer and communications technology have opened the way to a set of demanding applications like videoconference. Many of these and legacy (e.g., telephony) applications require a real-time service which has traditionally been provided over circuit switched networks. However, the increasing diffusion of packet switching networks, the possibility of integrating on the same network different services, and the promise that statistical multiplexing can keep costs low are raising growing interest for supporting real-time services over packet switched networks.

Providing a real-time service basically requires the network to keep a time relationship between the sender and the receiver, by limiting the variation of the delay experienced by packets into the network. If not properly dealt with, queuing gives the most variable contribution to the end-to-end delay; it can be controlled by properly (1) limiting the number of data flows sharing network resources and (2) scheduling transmission of packets on links. The first task is accomplished by call admission control which has not been in the scope of this work. The second issue is responsibility of packet scheduling algorithms which can be roughly classified depending on whether they use local or global timing.

The first category is used in *asynchronous* packet switching networks. In this work we studied the performance of two such schemes for the provision of a toll quality telephone service: static priority has been taken into consideration in Chapter 3, and packet by packet generalized processor sharing has been investigated in Chapter 4. Both studies show that *deterministic* quality of service guarantees can be obtained from the network at the expenses of a

low utilization of network resources.

A packet scheduling technique based on global timing (namely time-driven priority) has then been considered. In Chapter 5 it is shown how this technology combined with embedding of virtual rings over general topology networks supports *adaptive* real-time many-to-many group communication. The presented solution is crucial for providing deterministic quality of service guarantees with acceptable resource allocation to large multicast groups like, for example, those possibly involved in world-scale videoconferences.

Chapter 6 compares the quality (in terms of end-to-end delay) perceived by the user of a videoconference session on asynchronous versus synchronous packet switching networks. The work shows that properly using time-driven priority can lead to the lowest end-to-end delay, even compared to circuit switching.

To summarize, real-time services with deterministically guaranteed quality can be provided over both asynchronous and synchronous packet switched networks. The results presented in this work show that the cost (in terms of resource utilization) vary significantly according to the considered scenario. Further study is needed to compare the cost of the two packet switching technologies in various configurations in order to understand if a rationale exists for preferring one over the other. In particular, further work is required on the provision of real-time services to variable rate bit flows.

Bibliography

[1] M. Baldi, Y. Ofek, and B. Yener. Adaptive real time group multicast. In *IEEE INFOCOM '97*, April 1997.

[2] J. C. R. Bennet and H. Zhang. Hierarchical packet fair queueing algorithms. In *ACM SIGCOMM96*, 1996.

[3] J. C. R Bennet and H. Zhang. WF^2Q: Worst-case Fair weighted Fair Queueing. In *IEEE INFOCOM'96*, 1996.

[4] V. Bolotin. Modeling call holding time distributions for CCS network design and performance analysis. *IEEE Journal on Selected Areas in Communications*, 12(3), April 1994.

[5] R. Braden, D. Clark, and S. Shenker. Integrated service in the internet architecture: an overview. Informational RFC 1633, Internet Engineering Task Force, July 1994.

[6] M. Butto, E. Cavallero, and A. Tonietti. Effectiveness of the 'leaky bucket' policing mechanism in ATM networks. *IEEE Journal on Selected Areas in Communications*, 9(3):355 – 342, April 1991.

[7] R. Callon. Integrated PNNI for multi-protocol routing. Technical Report 94-0789, ATM Forum, September 1994.

[8] S. Casner and V. Jacobson. Compressing IP/UDP/RTP headers for low-speed serial links. Internet draft, November 1996.

[9] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanisms. In *ACM SIGCOMM'92*, pages 14–26, August 1992.

[10] R. Cruz. Service burstiness and dynamic burstiness measures: A framework. *Journal of High Speed Networks*, 1(2):105 – 127, 1992.

[11] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. *ACM Computer Communication Review (SIGCOMM'89)*, pages 3–12, 1989.

[12] W. Ding and B. Liu. Joint encoder and channel rate control of VBR video over ATM networks. In *Visual Communications and Image Processing*, volume 2668, pages 392 – 407, 1996.

[13] W. Ding and B. Liu. Rate control of MPEG video coding and recording by rate-quantization modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1):12 – 20, February 1996.

[14] D. Dykeman and M. Goguen. *Private Nework-Network Interface Specification - Version 1.0*. The ATM Forum, March 1996.

[15] J. Feng, K.-T. Lo, H. Mehrpour, and A. E. Karbowiak. Cell loss concealment method for MPEG video in ATM networks. In *GLOBECOM '95*, pages 1920–1924, 1995.

[16] D. Ferrrari, A. Banerjea, and H. Zhang. Network support for multimedia. a discussion of the tenet approach. *Computer Networks and ISDN Systems*, 26:1267 – 1280, 1994.

[17] The ATM Forum. *ATM User-Network Interface Specification - Version 3.1*. The ATM Forum, September 1994.

[18] B. Fufht. A survey of multimedia compression techniques and standards. Part I: JPEG standard. *Real Time Imaging*, (1):49 – 67, 1995.

[19] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):47 – 58, April 1991.

[20] R. Gareiss. Voice over the Internet. *Data Communications*, pages 93–100, September 1996.

[21] Golestani. A Self-Clocked fair queueing scheme for boradband applications. In *IEEE INFOCOM'94*, pages 636–646, June 1994.

[22] S. J. Golestani. Congestion-free transmission of real-time traffic in packet networks. In *IEEE INFOCOM'90*, pages 527–536, San Francisco, CA, June 1990.

[23] S. J. Golestani. A stop-and-go queuing framework for congestion management. In *ACM SIGCOMM'90*, pages 8–18, Philadelphia, Pennsylvania, September 1990.

[24] S. J. Golestani. Duration-limited statistical multiplexing of delay-sensitive traffic in packet networks. In *ACM SIGCOMM'91*, pages 323–332, April 1991.

[25] Moving Pictures Experts Group. *MPEG-2 test model 5.* ISO/IEC JTC1/SC29/WG11/N0400, April 1993.

[26] A. Gupta, W. Heffner, M. Moran, and C. Szyperski. Network support for realtime multi-party applications. In *Fourth International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'93)*, Lancaster, England, November 1993.

[27] A. Gupta, W. Howe, M. Moran, and Q. Nguyen. Resource sharing for multi-party real-time communication. In *IEEE INFOCOM'95*, Boston, MA, April 1995.

[28] C. Horne and A. Puri. Video coding with adaptive quantization and rate control. In *Visual Communications and Image Processing*, volume 1818, pages 798 – 806, 1992.

[29] ISO/IEC. *Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s.* International Organization for Standardization, 1993.

[30] ITU-T. *Recommendation I.327.* B-ISDN Functional Architecture, March 1993.

[31] ITU-T. *Recommendation I.363.* B-ISDN ATM Adaptation Layer (AAL) Specification, March 1993.

[32] S. Jung and J. S. Meditch. Adaptive prediction and smoothing of MPEG video in ATM networks. In *IEEE International Conference on Communications*, pages 832 – 836, 1995.

[33] G. Karlsson. Asynchronous transfer of video. *IEEE Communications Magazine*, pages 118 – 126, August 1996.

[34] Y. Katsube, K. Nagami, and H. Esaki. Router architecture extensions for ATM: overview. Internet draft, October 1996.

[35] C-S. Li, Y. Ofek, A. Segall, and K. Sohraby. Pseudo-isochronous cell switching in ATM networks. In *IEEE INFOCOM'94*, pages 428–437, June 1994.

[36] C-S Li, Y. Ofek, and M. Yung. "Time-Driven Priority" flow control for real-time heterogeneous internetworking. In *IEEE INFOCOM'96*, 1996.

[37] L.-J. Lin, A. Ortega, and C.-C. J. Kuo. A gradient based rate control algorithm with applications to MPEG video. pages 392 – 395, 1995.

[38] E. Linzer. A robust MPEG-2 rate control algorithm. Unpublished technical report, IBM – T. J. Watson Research Center, department 924A.

[39] W. Luo and M. El Zarki. Adaptive data partitioning for MPEG-2 video transmission over ATM based networks. In *International Conference on Image Processing*, volume 1, pages 17 – 20, October 1995.

[40] J. Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, 35(4):435–438, April 1987.

[41] P. Newman, G. Minshall, T. Lyon, and L. Huston. IP Switching and gigabit routers. *IEEE Communications Magazine*, January 1997.

[42] Y. Ofek. Integration of voice communication on a synchronous optical hypergraph. *INFOCOM'88*, 1988.

[43] Y. Ofek. Generating a fault tolerant global clock using high-speed control signals for the MetaNet architecture. *IEEE T. on Communications*, pages 2179–2188, May 1994.

[44] Y. Ofek and B. Yener. Reliable concurrent multicast from bursty sources. In *IEEE INFOCOM'96*, 1996.

[45] P. Pancha and M. El Zarki. MPEG coding for variable bit rate video transmission. *IEEE Communications Magazine*, pages 54 – 66, May 1994.

[46] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The sigle-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[47] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[48] D. Reininger, G. Ramamurthy, and D. Raychaudhuri. VBR MPEG video coding with dynamic bandwidth renegotiation. In *IEEE International Conference on Communications*, pages 1773 – 1777, June 1995.

[49] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow. Tag switching architecture overview. Internet draft, January 1997.

[50] I. E. G. Richardson and M. J. Riley. Usage parameter control cell loss effects MPEG video. In *IEEE International Conference on Communications*, pages 970 – 974, June 1995.

[51] T. Robinson. Speech analysis. http://squid.eng.cam.ac.uk:80/~ajr/SA95/SA95.html.

[52] R. M. Rodriguez-Dagnino, M. R. K. Khansari, and A. Leno-Garcia. Prediction of bit rate sequances of encoded video signals. *IEEE Journal on Selected Areas in Communications*, 9(3):305 – 313, April 1991.

[53] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. Standard Track RFC 1889, IETF's Audio-Video Transport Working Group, January 1996.

[54] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Standard Track RFC 2212, Internet Engineering Task Force, September 1997.

[55] S. Shenker and J. Wroclawski. General characterization parameters for integrated service network elements. Standard Track RFC 2215, Internet Engineering Task Force, September 1997.

[56] M. Simon, P. Villegas, J. Caballero, and M. Roser. A general approach to output rate control in video coding. In *Visual Communications and Image Processing*, volume 1903, pages 246 – 254, 1993.

[57] S. Singh and S-S. Chan. A multi-level approach to the transport of MPEG-coded video over ATM and some experiments. In *IEEE GLOBECOM '95*, pages 1920–1924, 1995.

[58] R. Steinmetz and K. Nahrstedt. *Multimedia: computing, communications & applications*. Prentice Hall, Upper Saddle River, NJ 07458, 1995.

[59] A. Sultan and H. A. Latchman. Adaptive quantization scheme for MPEG video coders based on HVS (Human Visual System. In *Visual Communications and Image Processing*, volume 2668, pages 181 – 188, 1996.

[60] T. Turletti and C. Huitema. Videoconferencing on the Internet. *IEEE/ACM Transactions on Networking*, 4(3):340 – 351, June 1996.

[61] D. Verma, H. Zhang, and D. Ferrari. Delay jitter control for real-time communication in a packet switching network. In *TriComm '91*, Chapel Hill, NC, April 1991.

[62] G. K. Wallace. The JPEG still pictures compression standard. *Communications of the ACM*, 34(4):30 – 44, April 1991.

[63] L. Wang. Rate control for MPEG video coding. In *Visual Communications and Image Processing*, volume 2501, pages 53 – 63, 1995.

[64] J. Woodard. Speech coding. http://www-mobile.ecs.soton.ac.uk/speech_codecs/.

[65] J. Wroclawski. Specification of the controlled-load network element service. Standard Track RFC 2211, Internet Engineering Task Force, September 1997.

[66] J. Wroclawski. The use of RSVP with IETF integrated services. Standard Track RFC 2210, Internet Engineering Task Force, September 1997.

[67] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *ACM SIGCOMM'90*, pages 19–29, Philadelphia, Pennsylvania, September 1990.

Appendix A

The MPEG Encoding Standard

A video sequence is digitalized by converting into digital form pictures (or video frames) at a fixed pace. Digital video is replayed by displaying pictures at the same fixed pace at which they had been captured. For example, good quality video is digitalized and displayed at 30 frames per second in the US and 25 frames per second in Europe.

Each picture is divided in pixels; the number of pixels in each picture and the number of bits used to encode each pixel determines the picture resolution. Pictures are described by three components which are either the intensity of the red, green, and blue (RGB format) or the intensity of luminance and chrominance (YUV format) of the pixels.

This coding format requires a huge amount of bits to encode sequences. Many compression schemes have been proposed which exploit the redundancy inside (*spatial redundancy*) and among (*temporal redundancy*) pictures in order to reduce the number of bits needed to encode a video sequence. The Moving Pictures Expert Group coding standard [29, 58, 19] specifies how to digitally encode and compress video and related audio mainly for two application fields: (1) storage and subsequent retrieval and display (e.g., on digital video disks), and (2) one way broadcasting (e.g., TV broadcasting and video on demand).

Nevertheless, the MPEG standard has been proposed for being applied also for videoconferencing purposes, even though the encoding algorithm is computationally intensive and it was not designed for real-time operation. The MPEG standard supports various resolutions and output bit rates through

the same coding principles.

This Appendix describes the basic principles of MPEG standard with the aim of highlighting and explaining the characteristics which are particularly relevant to this work.

A.1 General Principles

Compression schemes are based on reduction of temporal and spatial redundancy and quantization of values used to code images. While redundancy reduction is a lossless process, quantization degrades image quality.

Temporal redundancy is eliminated by coding the difference between the picture and a reference one. This introduces strong dependency among pictures and pictures coded by difference cannot be decoded and displayed prior to decoding the reference image. This turns out to be a problem when transmitting a coded sequence because the loss of (part of) a reference picture causes all the referencing images to be corrupted.

The MPEG standard defines three categories of pictures according to whether spatial or temporal redundancy is eliminated:

- I-frames (*Intra-coded* frames) are coded by eliminating spatial redundancy only. Hence they are decoded by themselves, without reference to any other picture.
- P-frames (*Predictive-coded* frames) are encoded with reference to the previous image.
- B-frames (*Bidirectionally predictive-coded* frames) are coded with reference to the previous and following I-frame or P-frame. This allows similarities with a future image (possibly not present within a previous picture) to be exploited to reduce temporal redundancy. This coding yields very good compression, but introduces a large delay because a B-frame cannot be coded until the future reference picture has been captured; the lower the frame rate, the larger the delay introduced. Since in this work we are considering interactive applications which require low delays, bidirectional predictive coding is not exploited and further explained.

When a video sequence is encoded, images are gathered in *Groups of Pictures* (GOPs): the first image is intra-coded and the following pictures in the same GOP are predictive-coded[1].

A.2 Intra-coded Pictures

Intra-frame coding yields compression by eliminating spatial redundancy. In this Section the basic steps of intra-coding are briefly described.

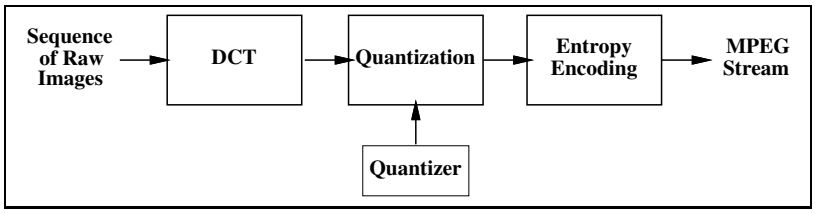

Figure A.1: Block Diagram of the MPEG Encoding Algorithm for I-Frames.

A.2.1 Discrete Cosine Transformation

Raw pictures which are to be encoded with the MPEG standard are digitalized in format similar to YUV. Each image consists of a luminance component and two chrominance components. The former has twice as many pixels as the latter two components.

On each of the three components of a picture 8x8 blocks of pixels (named simply *blocks* in the following) are identified. The Discrete Cosine Transformation (DCT) is performed independently on each block and the representation of the block in the frequency domain is obtained. The DCT takes as input 64 8-bit integer values $s_{x,y}$ representing the value of each pixel in the block and produces 64 12-bit integers $c_{i,j}$ which are the representation of the block in the frequency domain [19]:

$$c_{i,j} = DCT(s_{x,y}) \qquad i,j,x,y \in [0,7]$$

[1]In general, the GOP contains a pattern of B-frames interleaved by P-frames, e.g., IBBPBBPBBPBB.

The $c_{0,0}$ coefficient is called DC-coefficient and represents the fundamental color in the block. The other coefficients, AC-coefficients, represent the separation of lines in either or both directions. E.g., $c_{7,0}$ is the highest frequency appearing in the vertical direction and represents the closest separation of vertical lines in the block. $c_{i,i}$ is proportional to the amount of squares in the block because they are characterized by the same spatial frequency in both directions. E.g., the larger the number of isolated pixels, the highest the value of $c_{7,7}$.

The DCT provides *energy compaction* because the information contained in a block is grouped according to the level of detail in the block. Low spatial frequency coefficients contain the information on the overall aspect of the block; high frequency coefficients contain the information concerning the details of the block. Low frequency coefficients usually have larger values than high frequency ones. The encoding step (see Section A.2.3) provides better compression as the values to be encoded are not uniformly distributed.

Moreover, frequency domain coefficients can be coded with different precision according to how sensitive the human eye is to each of them. Low frequency ones are more important in determining the perceived quality of an image, i.e., they must be encoded with minimal information loss (fine quantization). High frequency components are less critical for the image quality, and thus they can be encoded with less accuracy and higher compression ratio (coarser quantization).

A.2.2 Quantization

Each DCT coefficient is quantized with a different quantizer because each one of them weighs differently on the image quality. The quantization is driven by an 8x8 matrix of 8-bit integers called *quantization matrix*. Each element of the matrix indicates the aggressiveness of quantization on the corresponding DCT coefficient: quantization is performed by dividing the elements of the block by the corresponding element of the quantization matrix. Thus, the higher is the value of quantization elements, the coarser is the quantization, i.e., the greater is the yielded compression. The result of this step is a 10-bit integer value [19].

The quantization matrix is chosen according to the nature of the scene to be encoded and is included at the beginning of the MPEG video stream.

A.2.3 Entropy Encoding

This step produces a compressed stream of bits through two sub steps [18, 62]:

1. *Run-length.* The coefficients obtained by the quantization are run-length encoded with a sequence of couples of symbols:

   - the first symbol, is divided in two parts:
     (a) `run-length` (4 bits) represents the number of coefficients with value zero,
     (b) `size` (4 bits) is the number of bits (between 0 and 10) used to encode the value of the next non zero coefficient;
   - the second symbol is the coefficient value expressed on `size` bits; the value zero does not have to be coded as it is run-length encoded.

   As a consequence, for each non-zero coefficient a number of bits between 8 to 18 is used.

   The DC-coefficient $c_{0,0}$ is encoded using a predictive coding technique, i.e., the difference between the value of the DC-coefficient of the current block and the one of the previous one is encoded. As the two quantized coefficients are both represented on 10 bits, the result is encoded on a maximum of 11 bits.

   The AC-coefficients are encoded according to the zig-zag order shown in Figure 1, thus being considered in increasing frequency order. Low frequency coefficients typically have similar big values, while high frequency ones have smaller values; the probability of zero valued coefficients is higher among the high frequency ones and having them grouped together allows run-length encoding to deliver better compression.

2. *Huffman encoding.* The intermediate symbol sequence produced by the run-length encoder is transformed by replacing the first symbol in each couple of symbols with a variable length code obtained from a Huffman table that has been specified for each component of the picture.

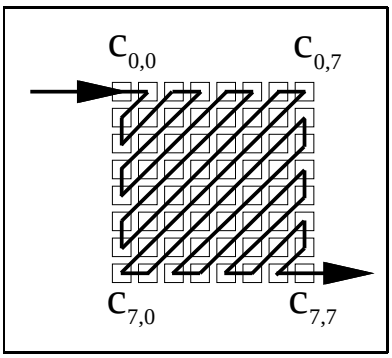

Figure A.2: Encoding Order for Quantized DCT Coefficients.

A.2.4 Controlling I-frame Dimension

As shown in Section 6.4.1 (see Figure 6.19) dimension of encoded pictures is highly variable. Application of the MPEG encoding scheme often requires picture dimension to be controlled: e.g., CBR MPEG encoding (Section 6.5) and transmission of VBR MPEG encoding through packet switched networks with time-driven priority Section 6.4.3). The number of bits obtained when intra-coding a picture (I-frame dimension) depends on the following factors:

- *Resolution* of the digitalized image: it is defined for each conference and it is not usually changed on the fly during the session.

- *Content* of the digitalized image: if the image has few details, many high frequency DCT coefficients have small values. Quantization reduces small values to zero and run-length encoding delivers high compression.

- *Quantizer*: the higher the quantization stepsize, the smaller the values of the quantized coefficients. Smaller quantized values are Huffman encoded with fewer bits; sequences of zero values yield even higher compression thanks to run-length encoding.

- *Variance of the first symbols* produced by run-length encoding: Huffman encoding is as more effective as the number of different first symbols in the sequence is low and each symbol has a high occurrence probability.

Among the foregoing factors, only the quantizer can be changed on purpose and dynamically to control the compression ratio and hence I-frame dimension. The quantization stepsize can be possibly modified on a block by block basis. Since a quantization matrix is exploited, quantization is regulated separately for each coefficient in a block, thus allowing a fine tuning over a wide range of compression ratios (and image qualities).

One of the most debated topics in controlling I-frames dimension is the choice of a quantizer that allows the intended number of bits to be produced, while keeping image quality uniform. The problem is sometimes addressed with reiterative approaches that quantize a picture more than once with different quantizers in order to identify the most suitable quantization stepsize. At each iteration the picture is encoded; if the yielded amount of bits is too big (small), all the coefficients are re-quantized with a coarser (finer) step and re-encoded. After a number of iterations the intended dimension is obtained for the I-frame.

This approach is computationally hard and it does not bound the encoding time, thus not being suited to real-time encoding. Some heuristics can be used to predict the amount of bits that are going to be produced by compressing an I-frame without having to perform the coding process up to the end.

1. The DCT is performed on the whole picture before starting quantizing and encoding the coefficients. By a swift analysis of the yielded coefficients the compression factor provided by Huffman encoding can be roughly estimated and a sensible quantization stepsize can be chosen.

2. During quantization some very simple and computationally light algorithm is run in order to get an estimate of the amount of bits expected as outcome of the encoding step. If the I-frame is expected to grow too large, the coefficients are quantized again with a larger quantization stepsize before encoding the outcome of quantization. If the system has more than one processor, more than one quantization processes, each with different stepsize, can be run in parallel.

3. Blocks are grouped in sets and an equal share of the target I-frame dimension is assigned to each set. After all the blocks in a set have been processed, the yielded number of bits is compared against the

assigned share. If it is significantly grater (smaller) than the target, the coefficients are re-quantized with a larger (smaller) quantizer and re-encoded.

Block grouping must be pseudo-random because an image can have very simple parts whose blocks are encoded with a small amount of bits; more complex parts are characterized by blocks requiring a huge amount of bits. Due to the pseudo-random order with which blocks are encoded, the image sending cannot start before the whole picture has been encoded.

The proposed heuristics can be used singularly or in composition to get better results. Constraints must be imposed on the variability of the quantization stepsize within each image in order to prevent non-uniform quality.

A.3 Predictive-coded Pictures

Predictive-coding reduces both spatial and temporal redundancy. Pictures are encoded by reference to the previous one (having it been intra- or predictive-coded) which is called the *reference picture* in the following. This Section describes the basic steps of predictive-coding; the first two allow temporal redundancy to be reduced before going through the very steps of intra-coding.

A.3.1 Motion Estimation

On the luminance component of the digital image, 16x16-pixel squares are identified, each composed of 4 blocks; one block on each of the chrominance components corresponds to the 16x16-pixel square on the luminance component. Motion estimation operates on *macroblocks* (MBs) which consist of 6 corresponding blocks (4 on the luminance component and one on the two chrominance components). The reference picture is searched for a MB "similar" to the one being encoded; the possibly found MB is called a *predictor*. The difference between the given MB and the predictor is encoded as a representation of the former.

An algorithm, not specified by the MPEG standard, is run in order to identify a predictor in the area of the reference picture around the location corresponding to the MB being encoded. The algorithm behavior is controlled

by two parameters which determine its performance in terms of running time and contribution to the overall picture compression.

1. The *search range* identifies the area around the current MB position in which the reference picture is searched for the predictor. The larger the search area, the higher the probability of finding a very well matching MB, thus yielding good compression. On the other hand, the larger is the search area, the longer it takes to the algorithm to complete the search. The search range can be changed on a picture by picture basis.

2. The *similarity criteria* aims at choosing as predictor the MB which is going to provide the highest compression when the difference from the actual MB is encoded. The more complex the similarity criteria, the better the compression obtained, but the harder the computation.

Motion estimation is the most computationally intensive and time consuming step of the whole encoding process.

A.3.2 Motion Compensation

When a predictor is found for the MB, the difference between each pixel and the corresponding one in the predictor is computed. Since two 8-bit integers are subtracted, the result is a 9-bit integer. If a MB similar enough is not found, each block of the MB is encoded as an I-frame block, i.e., the motion compensation step is skipped.

The complete encoding of a motion compensated MB encompasses also a motion representation which allows the predictor to be identified. Motion is represented through a *motion vector*, i.e., the bidimensional offset of the predictor from the position in the picture of the MB being encoded. The motion vector is encoded as difference from the motion vector of the previous MB.

A.3.3 DCT

The DCT is performed on the output of the previous step. If a predictor has been found, the DCT is performed on the differences between the pixels of

the given MB and those of the predictor. If no predictor has been found, the DCT is applied on the pixels of the MB itself.

A.3.4 Quantization

The result of the DCT is quantized using different quantizers for motion compensated MBs and intra-coded MBs. The motion compensated MBs likely have small coefficients and thus the quantizer must have no deadzone, i.e., the range of values that are quantized to zero must be smaller than the quantization stepsize.

If the coefficients of a motion compensated MB are all zero (i.e., a predictor identical to the MB has been found) the MB is no further processed and it is represented in the MPEG stream by a special 6-bit code.

A.3.5 Entropy Encoding

All the coefficients of motion compensated MBs are encoded using run-length encoding and Huffman encoding (i.e., the DC-coefficient is not treated differently). Blocks of non motion compensated MBs are encoded according to the intra-coding process.

A.3.6 Controlling P-frame Dimension

The number of bits obtained when predictive-coding a picture (P-frame dimension) depends on the following factors:

- *Dynamics of scenes*: if scenes are static, subsequent pictures do not change too much, motion compensation is highly effective, and P-frames are small.
- *Resolution of images*: see Section A.2.4.
- *Search range* and *similarity criteria* used to find the predictor. Both parameters could be used to control the compression ratio, but they are employed too early in the compression process. Thus, on one side, reiterations with different values would be impracticable due to the

large amount of computation required by motion estimation. On the other side, the relationship between changing these parameters and the number of generated bits is not trivial.

- *Quantizer*: see Section A.2.4.
- *Variance of the first symbols* produced by run-length encoding: see section A.2.4.

The same actions proposed in Section A.2.4 for controlling the size of I-frames, can be taken for P-frames. In addition, before encoding a P-frame, the percentage of MBs which can be motion compensated can be roughly estimated by estimating the overall similarity of the two images. Having some knowledge or statistics about the number of bits needed to encode motion compensated MBs, it is possible to roughly estimate the amount of bits that are going to be produced and consequently choose the search range and the quantizers to be used.

A.4 Signal to Noise Ratio

The loss due to the encoding process can be quantitatively measured by the *peak-to-peak Signal to Noise Ratio* (SNR) which is calculated as

$$SNR = 10 \cdot \log_{10} \frac{255^2}{\sum_i (s_i - \hat{s}_i)^2} \tag{A.1}$$

where s_i is the value of the i^{th} pixel before encoding, while $\hat{s}_i$ is the value of the same pixel reconstructed from the MPEG stream. The signal to noise ratio is scaled according to the maximum value of the pixel (peak-to-peak) which is 255.

A.5 Packetization and Streaming

The MPEG standard [29] defines an *elementary stream* for carrying each media: video, audio, or data. A *system stream* is designed to combine a number of elementary streams.

Each elementary stream is packetized; each *packet* has a header identifying the elementary stream to which the packet belongs. Moreover, the header can contain a *presentation time stamp* and a *decoding time stamp* which indicate the time at which the first access unit[2] in the packet is intended to be presented and decoded, respectively. The compliance of an elementary stream with the MPEG standard is checked by feeding a *System Target Decoder* (STD) with the stream and verifying that its buffer does not overflow or underflow.

The STD buffer is filled by the packets of the corresponding elementary stream as soon as they get to the decoder. The buffer is emptied by retrieving a whole access unit at the time specified by the decoding time stamp contained in the packet header. The STD buffer underflows if the complete access unit is not yet in the buffer before its retrieval is scheduled. The buffer overflows if there is not enough space to store a whole packet when it is extracted from the system stream.

Packets, possibly belonging to different streams, are grouped in *packs*, each preceded by a pack header which contains information about the rate of the stream. Moreover, the pack header contains a time stamp, called the *system clock reference*, which indicates the intended time of arrival of the last byte of the time stamp itself. It is used to recover the encoder clock at the decoder.

The first pack of the system stream contains a system header that can be optionally repeated in some other packs. The system header carries information that is relevant to the whole stream and contains the maximum size of the STD buffer to be used with each elementary stream. The current size of the STD buffer optionally included in packet headers must be smaller than the maximum value present in the system header.

The MPEG-2 standard provides two kinds of system stream:

1. The *Program Stream* is designed for error-free environments, e.g., storage and retrieval of video scenes from digital video disks. It uses variable length packets.

2. The *Transport Stream* is intended for exploitation in lossy environments (e.g., transmission over computer networks) and packets have fixed length (188 bytes).

[2]For example, the the access unit in a video stream is a picture.

Appendix B

List of Acronyms and Symbols

A dimension of the token bucket in a leaky bucket shaper.

b_{mb} macroblock level rate control factor of macroblock mb.

B target rate for MPEG CBR encoder or rate of circuit switched connections used to send raw and encoded video.

C capacity of physical links.

C_M maximum coding delay.

CBR Constant Bit Rate.

$c_{i,j}^{mb}$ DCT coefficient (i,j) of MB mb.

$\hat{c}_{i,j}^{mb}$ quantized DCT coefficient (i,j) of MB mb.

D decoding time or decoder delay.

DCT Discrete Cosine Transform.

D_i^T delay experienced in the network by frame i

D_M^T maximum delay experienced by a frame in the network.

D_m^T minimum delay experienced by a frame in the network.

D_i^R delay experienced in the replay buffer or transmission synchronization delay experienced by video frame i.

D_M^R maximum delay experienced in the replay buffer or maximum transmission synchronization delay.

E_r Excess resynchronization delay

F number of bits encoding a frame;

- F_m lower bound on picture dimension,
- F_r number of bits encoding a raw picture,
- F^I dimension of an I-frame,
- F^P dimension of a P-frame.

g_i minimum service rate of a (P)GPS scheduler for flow i.

GPS Generalized Processor Sharing.

G global distortion level that is chosen on a picture by picture basis.

GOP Group Of Pictures.

h_i number of hops traversed by flow i.

J_n network delay jitter.

L number of time frames needed to transfer a packet from sender to receiver with time-driven priority.

M number of video frame periods in the time cycle.

MB MacroBlock.

N number of video frames in a group of pictures.

N_s number of time frames in which bits have been allocated to send a raw or encoded video frame with time-driven priority.

N_r number of time frames between the first and the last TF reserved for the transmission of a picture.

P propagation delay.

PGPS Packet by packet Generalized Processor Sharing.

p_{mb} MB local activity factor of MB mb.

P_s size of a packet.

P_M maximum packet size on the network.

Q queuing delay in the network;

- Q_m minimum queuing delay,
- Q_M maximum queuing delay.

Q_{mb} quantization parameter of macroblock mb.

QoS Quality Of Service.

$q_{i,j}^{\{I|P\}}$ element (i,j) of quantization matrix for I-frames or P-frames.

r_m overall service rate of the scheduler on node m; usually r_m equals the output link capacity.

S_c coding shaping delay.

S_n network shaping delay.

S_n^{CS} network shaping delay due to lack of synchronization between application and network interface.

S_n^{CS} network shaping delay due to circuit bandwidth.

S_n^{TS} network shaping delay due to traffic shaping at the boundaries of asynchronous packet switched networks.

S_s startup shaping delay;

- S_s^m minimum startup shaping delay.

Sw switching delay.

T video frame period.

T_c time cycle duration.

TF Time Frame.

T_f duration of a time frame.

VBV Video Buffer Verifier.

V_b maximum backlog in the encoder buffer.

V_s size of the VBV.

WFQ Weighted Fair Queuing.