Communication Performance Models

Stefan Böcking[†]

International Computer Science Institute, Berkeley

TR-95-013

March 1995

Abstract

Communication performance models enable distributed real-time and multimedia applications to describe their performance requirements as regards throughput, delay and loss behavior of a particular communication service. The purpose of this paper is to give a basic understanding of communication performance models by presenting four different models: two models designed by the Tenet Group, one model on which ATM channel traffic is characterized by the ATM Forum, and the RFC 1363 Flow Specification of the Internet community. Besides their presentation in a unified terminology, their usability is shown by a video-on-demand example.

Keywords: multimedia communication, real-time communication, quality-of-service (QoS).

[†] Siemens AG, Corporate Research and Development, Otto-Hahn-Ring 6, 81730 Munich, Germany.
e-mail: Stefan.Boecking@zfe.siemens.de.

1 Introduction

Today's widely-used communication systems, best represented by the Internet Protocol Suite, generally provide insufficient service support for distributed real-time and multimedia applications when it comes to performance requirements and guarantees[1]. Only in networks with enough resources, so that resource contention is rare or impossible, protocols without performance guarantees may be used for real-time and multimedia applications. However, when resource contention is more frequent, these protocols are unsatisfactory because of their lack of functions to protect real-time and multimedia communications from being arbitrarily interrupted by other communications.

Distributed real-time applications such as for controlling a power plant or an industrial manufacturing process have stringent timing constraints on operating remote sensors and actuators used to interact with the real world [BuW90]. Because of the awareness that missing deadlines may cause disastrous implications, most real-time systems require a communication system which provides deterministic performance guarantees or bounds with respect to throughput, delay and loss.

Distributed multimedia applications integrate discrete media, such as text and graphics, with continuous media such as audio and video. Continuous media used in an interactive communication[2] has also timing constraints. Nevertheless, the consequences of a missed deadline are less disastrous than in hard real-time applications, because a certain percentage of deadline violations are usually tolerated by the human user. Thus, the effort spent for real-time applications to constitute a deterministic environment is mostly unnecessary for multimedia applications[3]. On the other side, the performance guarantees given by today's communication systems - typically best-effort - are also too less to be satisfying. More suitable performance commitments,

1. Other missing capabilities, such as the support of multipoint communications and the provision of unreliable connection-oriented services, are beyond the paper's scope.
2. Communications in which human beings are involved.
3. Real-time software is typically analyzed *a priori* by schedulability tests in order to determine whether the imposed timing requirements can be met deterministically. These tests require the prediction of the software's execution time, which implies that certain programming features such as recursion, dynamic allocation of memory and dynamic creation of processes are avoided, and that features such as interrupt handling, inter-process and network communication are timely bounded [BuW90].

which have been recently suggested, are examples of efforts to fill the gap between fully deterministic guarantees and best-effort services. [Par92], [FerV90] and [CSZ92] describe techniques to give imperfect, statistical and predicated performance guarantees, respectively.

In this paper we present four different communication performance models which enable distributed real-time and multimedia applications to describe their performance requirements and guarantees as regards throughput, delay and loss behavior. Each model provides a different set of parameters to characterize the user's traffic shape and the provider's guarantee commitment resulting in different communication services.

The rest of the paper is organized into three sections: Section 2 explains the terminology which is used to describe the performance models uniquely. Section 3 then contains the four models which we have selected[1]:

- two models designed by the Tenet Group,
- a model on which ATM channel traffic is characterized by the ATM Forum,
- and the RFC 1363 Flow Specification of the Internet community.

Section 4 demonstrates the usability of these models at a video-on-demand example.

1. Besides the presented models, other models have been proposed recently such as the BERKOM-II FlowSpec [DaG94a], the Simplified QoS Model [DaG94b], and the OSI95 [Dan94]. These approaches are similiar to one of the models presented, i.e. they can be derived easily.

2 Simple Networking Framework

Comparing performance models of different communication services is burdened by the fact that each of them uses its own terminology. Although there is a framework standardized by the ISO committee for networking called the *Open System Interconnection* (OSI) model, mostly two reasons are raised against using it: (1) its incomprehensibility and (2) its lack of coping with new service capabilities such as guaranteed performance qualities, multicast and low service latency. Nevertheless, some of the basic ideas of the OSI model such as the distinction between services and protocols, which is in nature an object-oriented approach, are very useful. Thus, we have developed a *Simple Networking Framework* (SNF), which is based on the OSI model but introduces an object-oriented architecture and a more appropriate communication model which enables us to describe uniquely a wide range of conventional and newly required services[1] [Boe94].

For the purpose of this paper, this section only summarizes parts of the SNF needed for the subsequent description of communication performance models, which are the service and communication model in section 2.1, the continuous traffic model in section 2.2. and the performance parameters in section 2.3.

2.1 Service and Communication Model

A computer network denotes an interconnected collection of autonomous computers called *nodes* of the network. Applying a *communication service* enables its *service users* to reside on different nodes while cooperating. The communication service is offered by the *service provider*. The provider consists of entities which reside on the network's nodes and execute a *protocol* to provide the communication service. The interaction between user and provider takes place at *service access points*. To each access point, a *user address* is bound which unambiguously locates the access point and thus the attached user within the network (see also Figure 1).

1. The existing connection and datagram models are already occupied by a certain meaning. With a connection, properties are already associated such as bidirectional, point-to-point, explicit establishment and reliable data transfer. Datagrams have properties associated such as unidirectional and unreliable data transfer of a single, self-contained data unit.

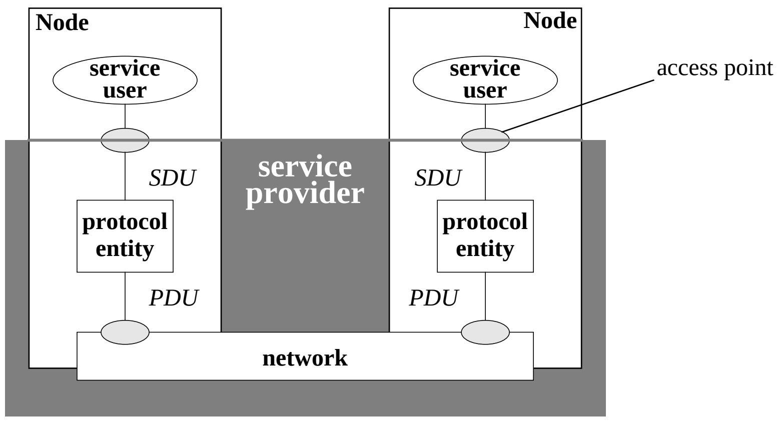

Figure 1: Service Model.

When using a service, user information is exchanged in terms of *service data units* (SDUs). The provider conveys SDUs in one or more protocol data units (PDUs) over the network. Usually, a PDU has three parts: a *header* and *trailer* to exchange protocol information and a *payload* to carry SDUs or parts of them. From the user's point of view, however, SDUs are transferred by the provider transparently.

While using a service, at least two users are logically connected in a *communication association* for a certain time. Only users affiliated to the same association may communicate, i.e. exchange information conveyed in SDUs. Each user has a local *association endpoint* that is used to send or receive SDUs to or from other users participating in the same association. Endpoints are bound to a single access point at a time.

The communication association itself only provides the logical binding between users. Data is exchanged in form of *streams*. A stream is a sequence of SDUs which flows unidirectionally from one *stream source* to one or more *stream sinks*. Sources and sinks are bound to a single endpoint at a time. Because the number of streams and their arrangement within an association is unrestricted, arbitrary communication patterns can be described such as point-to-point or multipoint-to-multipoint structures with unidirectional or bidirectional data flow (see also Figure 2).

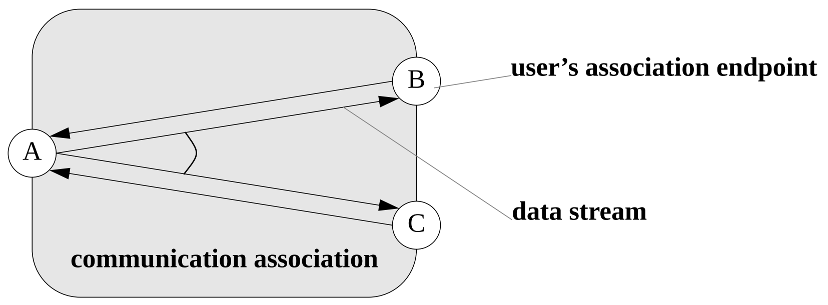

Figure 2: Example of an Association.

The *quality of service* (QoS) denotes properties as regards reliability, security, performance, synchronization, priority, or cost of the communication. QoS consists of *association qualities*, defining inter-stream properties such as the synchronization behavior of related streams, and *stream qualities*, defining the intra-stream properties such as the reliability or performance of the SDU flow. Different stream qualities may be assigned to streams within the same association.

The sequence of using a communication service usually has three phases: establishment, data transfer and release. In the establishment or release phase of an association, a user is called *initiator* when starting an action and *responder* when waiting to be indicated of an action. In the data transfer phase, a user is called *producer* when issuing SDUs at a stream source and *consumer* when accepting SDUs from a stream sink.

2.2 Continuous Traffic Model

In this section, we give examples of typical audio, video and real-time data streams. By observing the traffic characteristics of these streams, we will conclude by the end of this section that these streams can be uniquely described by a continuous traffic model.

Distributed real-time and multimedia applications require a communication service to transfer time-constrained audio samples, video frames, sensor inputs, or actuator commands over the network. Therefore, the service must provide performance parameters enabling applications to characterize their stream's traffic.

Time constraints are necessary for audio and video streams only if human beings are involved. Human beings are very sensitive to delay and jitter but also able to tolerate transient delay and jitter violations. In general, we can distinguish two types of interactive communications: (1) live communications, where producer and consumer are human beings, and (2) playback communications, where the producer is a playback device at which audio/video data is stored and only the consumer is a human being.

Audio Streams

An *audio stream* is a sequence of SDUs conveying one or more audio samples between users. The sample size and the sampling rate depend on the audio device. For example, devices following the CCITT G.711 recommendation produce or consume fixed-sized samples of 8 bit each at a constant rate of 8000 samples per second, which is telephone quality requiring a bandwidth of 64Kbit/s. Audio of CD quality usually has a sample size of 16 bit at a rate of 44100 samples per second for each stereo channel and therefore requires a total bandwidth of 1.4 Mbit/s for a typical two-channel-stereo stream. When the sample size is small, it is more efficient to convey multiple samples in a single SDU. But accumulating samples in a single SDU introduces additional delay, to which live communications are sensitive. A recommended maximum delay for audio streams between human users is 32 ms [Sch93]. Thus, choosing a delay of 20 ms in our previous example will yield SDU sizes of 160 bytes for telephone quality and 3528 bytes for CD quality, which are still small sizes.

The typical traffic shape of audio streams is a sequence of SDUs whose (small) size and interval are constant. If the audio device is able to detect and suppress silence periods or is a playback device, able to stop and continue the playback arbitrarily, subsequent SDUs may be separated by variable-length silence period no smaller than the constant interval. If multiple samples are carried in a single SDU, short silence periods may cause gaps in SDUs or if the silence period starts within a SDU but continues over the SDU end, a smaller SDU can be sent. In this case, the SDU sizes of an audio stream may vary.

Video Streams

A *video stream* is a sequence of SDUs conveying whole video frames or parts of them between users. The frame size and rate depends on the video device. A video source without compression for NTSC TV quality has a constant frame size of 900 Kbyte (i.e.

an image size of 640x480x24), and generates or displays constantly 30 frames per second. A video source with compression will typically produce a constant flow of frames with varying frame sizes. The size variation between subsequent frames depends on the compression method and the image information. Having a live video source, the image information and therefore its size is usually unpredictable. Only if the source is a video playback, the frame size variation can be estimated *a priori*.

The traffic of video streams can be generally described as a sequence of SDUs of variable size and constant interval[1]. If the source is a video playback device, which allows to stop and continue the playback arbitrarily, subsequent SDUs may be separated by a silence period no shorter than the constant interval.

Real-Time Data Streams

A *real-time data stream* is a sequence of SDUs conveying real-time events, such as sensor inputs received from a temperature transducer or actuator commands sent to a valve, between users. The size of the transferred real-time data is variable but usually very small. The interval between real-time events is either constant or variable depending on whether the events are generated periodically or aperiodically. Usually, the aperiodicity is bound by a minimum interval between two subsequent real-time events. Thus, we can describe aperiodic flows as periodic ones by defining the minimum interval as the constant interval and the time between subsequent SDUs as a stream pause if it is greater than the constant interval.

The traffic shapes of audio, video and real-time data streams can be uniquely described by a so called *continuous traffic model*: a sequence of SDUs with a constant interval and a variable size and pauses between SDUs greater than the constant interval[2].

2.3 Performance Parameters

In this section, we discuss in general terms the performance quality parameters of a stream during its data transfer phase. Therefore, we define parameters to describe throughput, delay, jitter and loss of the stream’s SDU flow.

1. The uncompressed video’s constant frame size is subsumed by the variable-size definition.
2. Note that virtually any traffic shape, not only audio, video and real-time streams, can be described with this continuous traffic model.

To start with, let us assume a unicast stream between one producer and one consumer. Let *n* denote the *n*'th SDU of this stream, where $n = 1,2,3,...$ Furthermore, let $t_{req,n}$ and $t_{ind,n}$ denote the transfer request and indication time of the *n*'th SDU at source and sink of a stream, respectively (see also Figure 3).

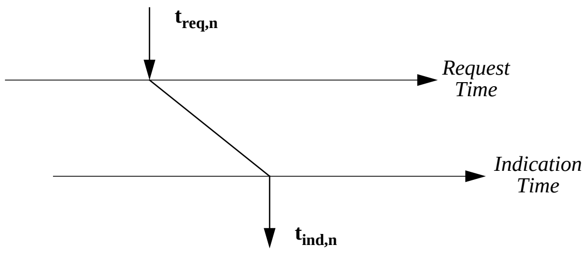

Figure 3: Time Diagram Showing the Time Relation between Transfer Request Invoked at Stream Source and Transfer Indication at Stream Sink of the n'th SDU.

Throughput

The throughput of a stream defines the amount of user data transferred in a certain time unit between source and sink. Throughput can be defined as bandwidth *B* without considering number and size of the SDUs, or by a parameter pair combining either SDU size and interval or SDU size and rate. The SDU size *S* usually denotes the maximum amount of bytes (or octets) conveyed in a single SDU. The SDU interval *I* denotes the time distance between two subsequent transfer requests. The SDU rate *R* which denotes the number of SDUs transferred per second, can be derived from the SDU interval, and vice versa:

$$R = \frac{1}{I}.$$

With SDU size and rate given, the corresponding bandwidth can be estimated by

$$B = R \times S.$$

If only the bandwidth is given, multiple size and rate combinations are possible. This may become a drawback, if the provider's estimation of the transfer delay and required processing time is based on the SDU number and size. In this case, the provider can only assume the worst-case combination as regards the resource requirements.

The upper bound of a stream's throughput B_{max} is given by the maximum SDU size S_{max} and the minimum SDU interval I_{min}:

$$B_{max} = \frac{1}{I_{min}} \times S_{max}.$$

With a maximum throughput parameter, only the worst case traffic of a stream is described. Based on this stream traffic description, the service provider allocates resources and usually determines the charge for a stream. Thus, a more elaborate characterization is desired if the traffic is *bursty* in order to save resources and to reduce costs. The service provider may motivate service users to characterize more precisely their traffic by an appropriate charging policy [CSE93].

Bursty streams have peak times during which SDUs of size S_{max} are requested for transfer every I_{min} time units. When we observe a bursty stream over a time period T_{avg}, the average SDU size will be smaller than S_{avg} but greater than S_{min} and the average SDU interval will be greater than I_{avg}. The burst size N_{burst}, which is the maximum number of subsequent SDUs of size S_{max} and interval I_{min}, is defined as

$$N_{burst} = \left\lfloor \frac{(S_{avg} - S_{min}) \times \left\lfloor \frac{T_{avg}}{I_{avg}} \right\rfloor}{S_{max} - S_{min}} \right\rfloor ,$$

where $S_{max} \geq S_{avg} \geq S_{min}$ and $T \geq I_{avg} \geq I_{min}$.

The sustainable or average bandwidth available is given by

$$B_{avg} = \frac{1}{I_{avg}} \times S_{avg}.$$

Delay

Transferring SDUs between stream source and sink consumes time spent for processing and queuing in the nodes and for transmission in the network. Thus, each SDU will experience a certain delay between its transfer request and indication. An upper bound on the delay, D_{max}, denotes the maximum time any SDU of a stream will need to be transferred (see also Figure 4).

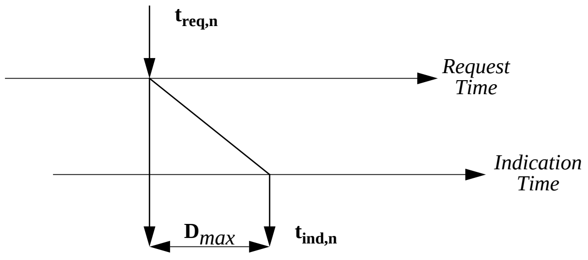

Figure 4: Maximum Delay any SDU of a Stream can Experience.

Jitter

SDUs of the same stream may experience different transfer delays. The delay variation is usually called *jitter*. Jitter can be viewed as the difference between the maximum delay D_{max} and the minimum delay D_{min} any SDU of a stream can experience between its transfer request and indication. Alternatively, jitter can be viewed as the variation of a mean delay given by D. In this case, the upper bound of the transfer delay can be deducted by $D + J$, and the lower bound by$D - J$. Subsequently, we assume that jitter J is defined as the difference between D_{max} and D_{min} (see also Figure 5).

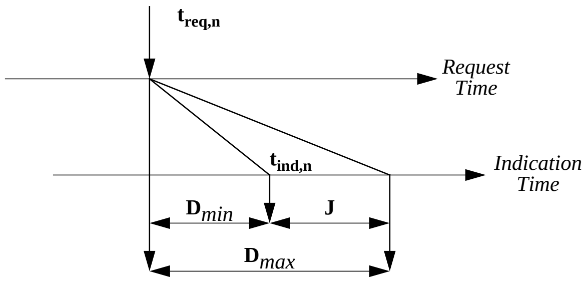

Figure 5: SDU Jitter.

Loss

SDUs may be lost due to damage, congestion or missed deadline. SDUs are transferred by the service provider in one or more protocol data units (PDUs) over the network. A SDU is only lost, if all its PDUs are lost. When only some PDUs are lost, the SDU is called *incomplete*.

PDUs may be damaged due to transmission interferences in the network or hardware errors in the node. Whether a PDU is dropped because of damage depends on the PDU part which is affected. Besides SDU information, each PDU may convey protocol information such as sequence numbers and stream identifiers. An irreparable damage of the protocol information prevents the proper assignment of the PDU to the relevant protocol context. Consequently, the PDU is dropped by the service provider. On the other hand, if only the payload of the PDU is damaged, the PDU can be properly processed, at least in the protocol context. If all PDUs of a SDU arrive but only portions of the conveyed SDU information are damaged, the SDU is called *damaged* SDU. Making a distinction between incomplete and damaged SDUs is mostly uninteresting to the service user, therefore we call a SDU as *corrupted* when it is either damaged or incomplete or both.

PDUs may also be dropped because of node congestion, which occurs when the buffer resources are exceeded at the time the new PDU arrives. During congestion, the node may either reject to receive the new PDU, or drop the oldest PDU in the buffer, or discard the PDU with the lowest priority, for example.

Furthermore, SDUs arriving after their deadline which is imposed by the maximum delay bound D_{max} may also be dropped. If a SDU is conveyed in multiple PDUs, it may occur that only a subset of the PDUs arrive on time meeting the deadline. In this case, the SDU may be indicated to the service user as incomplete.

We can summarize that a SDU is lost because all PDUs in which the SDU is conveyed are either damaged or dropped due to buffer overflow, or have experienced a transfer delay exceeding the defined upper bound.

The loss behavior of a stream can be described by a single loss parameter which denotes the upper bound of the tolerated SDU losses hiding the cause why SDUs are lost. Alternatively, separate upper bounds can be given in order to differentiate the tolerance on whether loss is caused by damage, congestion or missed deadline. In general, upper bounds may be defined either as *rate* or *ratio*. A rate defines the maximum number of lost SDUs, N_{loss}, which may occur in a given time interval I_{err}, assuming maximum throughput utilization. A ratio gives the proportion between N_{loss} and the number of totally transferred SDUs, N_{total}. N_{loss} can be calculated by

$$N_{loss} = N_{damage} + N_{congestion} + N_{delay},$$

where N_{damage}, $N_{congestion}$ and N_{delay} denote the number of lost SDUs because of damage, congestion or missed deadline respectively.

Furthermore, a ratio can be interpreted as *probability*. A loss probability P_{loss}, for example, can simply be deducted by

$$0 \le P_{loss} = \frac{N_{loss}}{N_{total}} \le 1,$$

where $N_{total} \ge N_{loss}$. Consequently, the probability that a SDU is delivered is

$$1 - P_{loss}$$

or

$$(1 - P_{damage})\ (1 - P_{congestion})\ (1 - P_{delay})\ ,$$

if P_{loss} is defined as separate damage, congestion and delay probabilities.

Another aspect of SDU losses is their burstiness. For example, during a longer node congestion period, more than one SDU of a stream may be dropped resulting in a burst of lost SDUs in sequence. The burst loss behavior can be described by a N_{err_burst} parameter which denotes the maximum number of SDUs which are tolerated to be lost in sequence.

2.4 Performance Guarantees

With the performance parameters previously described, the service user defines bounds on the throughput, delay and loss of the stream to be established. The provider may commit to guarantee these bounds with certain restrictions such as shown in the following table:

Guarantee	**Description**
Best effort	Bounds will be kept best as possible. There is no guarantee that the required performance is ever provided.
Imperfect	Bounds will be theoretically kept but all or some components of the provider are not able to guarantee the bounds [Par92].
Predicted	Bounds will be kept under the assumption that the future load of the provider corresponds to the load currently observed [CSZ92].
Statistical	Bounds will be kept with a given probability. The probability is guaranteed as long as the provider's hardware functions properly [Fer90].
Deterministic	Bounds will be kept as long as the provider's hardware functions properly.

Besides viewing the performance parameters as upper or lower bounds, we can treat them also as mean values obeying to a certain stochastic distribution [WSW93]. Furthermore, the performance parameters may be defined as ranges in which the mean value lies with a given probability[1].

1. For example, the mean transit delay lies between 18 and 22 ms with a confidence of 95%.

3 Communication Performance Models

In this section, we present four different communication performance models and their parameters by using the terminology of the Simple Networking Framework as described in the previous section. However, to ease the referencing to the original papers, where the models are specified, we use the original parameter names.

The parameters have been categorized according to whether they describe attributes of throughput, delay, jitter or loss. Parameters not falling in any of these categories are listed under Miscellaneous.

3.1 The Tenet Protocol Suite

The Tenet protocol suite has been designed by the Tenet Group at University of California at Berkeley and the International Computer Science Institute. The Tenet suite comprises transport and network functionality in order to provide real-time communication associations consisting of a single real-time stream. This stream is a sequence of SDUs which flow unidirectionally and unreliable (i.e. no retransmission) but with performance guarantees between two transport users[1]. The Tenet suite consists of five protocols: two of them are concerned with the control and management of real-time associations whereas the other three coping with the streams' data transfer phase. The protocols are as follows:

- RTIP (Real-Time Internet Protocol) is a data transfer protocol at the network layer.
- RMTP (Real-time Message Transport Protocol) is a data transfer protocol at the transport layer providing an asynchronous real-time stream.
- CMTP (Continuous Media Transport Protocol) is a data transfer protocol at the transport layer providing a continuous real-time stream.
- RCAP (Real-time Channel Administration Protocol) is a control and management protocol responsible for establishment, release and modification of real-time associations.
- RTCMP (Real-Time Control Message Protocol) is a control and management protocol monitoring the data transfer protocols.

1. A point-to-multipoint version is in preparation [FBZ92].

Two performance models are offered by the Tenet suite in order to characterize the SDU stream of a real-time transport association. Because the models are related to a specific Tenet transport protocol, we will subsequently refer to them as RMTP and CMTP performance model.

3.1.1 RMTP Performance Model

The Real-time Message Transport Protocol, RMTP [VeZ91], is part of the Tenet protocol suite. RMTP provides unidirectional real-time data transfer between two service users. RMTP assumes that a real-time communication association exists, established by RCAP, the real-time channel administration protocol. The offered parameters are as follows[1]:

Parameter		Unit	Description	SNF Notation
throughput	x_min	2^{-16} sec	minimum SDU inter-arrival time	I_{min}
	x_avg	2^{-16} sec	average SDU inter-arrival time	I_{avg}
	I	2^{-16} sec	averaging interval	T_{avg}
	s_max	byte	maximum SDU size	S_{max}
delay	D	2^{-16} sec	SDU transfer delay bound	D_{max}
	Z	10^{-4}	delay violation probability bound	$1\text{-}P_{delay}$
jitter	J	2^{-16} sec	delay jitter bound	J
loss	W	10^{-4}	buffer overflow probability bound	$1\text{-}P_{congestion}$

Throughput

The *x_min* parameter denotes the minimum interval between any two subsequent SDU transfer requests at a stream source. The parameter *x_avg* gives the average interval between subsequent SDU transfer requests over any averaging interval of duration *I*. With these three parameters, *x_min*, *x_avg* and *I*, the burstiness of the stream traffic can be characterized. A burst of SDU requests can be given as the maximum number of subsequent requests with an interval of *x_min*:

$$N_{burst} = 1 + \left\lfloor \frac{I}{x\_avg} \right\rfloor ,$$

1. Assuming a 32 bit integer to store the time value, this format (2^{-16} s) permits the representation of time values up to 18 hours of a granularity of 15 micro seconds.

assuming $I > x\_avg \geq x\_min$. The maximum bandwidth, which is required during a SDU burst, is

$$B_{max} = \frac{1}{x\_min} \times s\_max,$$

whereas the sustainable or average bandwidth is

$$B_{avg} = \frac{1}{x\_avg} \times s\_max.$$

Delay

The delay parameter D denotes the maximum time between transfer request and indication for any SDUs of the stream. The parameter Z gives the lower bound of the probability that the experienced delay of a SDU, D_n, within the requested delay D, i.e.

$$P(D_n \leq D) \geq Z$$

If Z is one, all successful transferred SDUs are delivered on time. Otherwise, at least $100Z\%$ of all SDUs are timely delivered.

Jitter

The jitter parameter J denotes the maximum delay variation that any SDU may experience. Substracting the jitter from the maximum delay D gives the lower delay bound for every SDU in the stream, i.e. the minimum amount of time SDUs need to be transferred. The jitter parameter is optional.

Loss

The buffer overflow probability parameter W denotes the lower bound of the probability that a SDU is not dropped because of buffer overflow (congestion). Thus, $1 - W$ denotes the probabilistic upper bound on SDU losses due to congestion. If W is 1, no SDU loss is accepted. Otherwise, at least $100W\%$ of all SDUs are successfully delivered. The parameter W considers congestion as the only reason for SDU loss. Whereby, loss due to delay violation is covered by the delay violation probability Z. Loss because of SDU damage is neglected. The number of SDUs delivered correctly and on time is at least $100WZ\%$ of the total number of transferred SDUs.

The delay violation probability bound *Z* and the buffer overflow bound *W* are used to classify the performance guarantee. A deterministic guarantee (Z=1, W=1), gives an absolute delay bound, i.e. all SDUs are delivered in time without loss due to congestion. A statistical guarantee gives a probability delay and/or a buffer overflow bound, therefore a defined number of SDUs may miss their playback point or may be lost due to congestion, and a best-effort guarantee (Z=0, W=0) gives no commitment whether SDUs will meet the issued delay bound or how many will be lost due to congestion.

3.1.2 CMTP Performance Model

The Continuous Media Transport Protocol, CMTP [WoM91, WoM92, FGM92], is part of the Tenet protocol suite. CMTP provides unidirectional real-time data transfer between two service users optimized for continuous media. CMTP assumes that a real-time communication association exists, established by RCAP, the real-time channel administration protocol of the Tenet protocol suite.

Continuous media traffic is characterized by a stream of SDUs sent periodically at a constant interval with variable SDU sizes. A stream can arbitrarily be stopped and continued by the producer. Stopping a stream may be performed either explicitly by invoking a stop primitive or implicitly by discontinuing the invocation of transfer requests. During a stream pause, the producer may weaken some of the performance parameter (T, S_{min}, S_{max}, S_{avg}, D_{stream}, N_{max}, W_{err}) of the stream, which will become valid when the transmission resumes.

The following table shows the offered parameters:

Parameter		Unit	Description	SNF Notation
Throughput	T	µsec	SDU interval	I_{min}
	S_{max}	byte	Maximum SDU size	S_{max}
	S_{avg}	byte	Average SDU size	S_{avg}
	N_{avg}[a]	#T	Averaging time period	T_{avg}/I_{min}
	S_{min}	byte	Minimum SDU size	S_{min}
Delay	D_{stream}	2^{-16} sec	SDU delay	D_{min}
Jitter				

Parameter		**Unit**	**Description**	**SNF Notation**
Loss	S_{err}	byte	Granularity of data errors	
	W_{err}	10^{-4}	PDU loss probability	$(1-P_{congestion})(1-P_{delay})$
	REPLACE	boolean	PDU replace indicator	
	DUMMY		Replace PDU	
Misc.	$STDU_{max}$	byte	Maximum STDU size	
	CONST_SIZE	Boolean	STDU constant size indicator	
	CONST_NUM	Boolean	STDU constant number indicator	
	N_{max}	#STDU	Maximum STDU number	
	Buffer		Interface buffer	
	S_{Sslack}	byte	Producer workahead	
	S_{Rslack}	byte	Consumer delay	

a. To avoid ambiguities in this paper, the original name I_{avg} has been changed to N_{avg}.

Throughput

The interval parameter T denotes the constant time distance between the start of any two subsequent SDUs of the stream. The SDU size has a lower and upper bound, S_{min} and S_{max}. The producer must conform to the negotiated maximum size, whereas the minimum size is only a hint for the service provider to optimize resource allocation which can be violated by the producer, i.e. sending SDUs with a smaller size than S_{min}. Besides the maximum throughput, the user may also define an average throughput. The service provider may use this information for a more efficient resource allocation. The average throughput is characterized by an averaging interval I_{avg}, in which the average SDU size is smaller than S_{avg}. The averaging interval can be calculated by the given N_{avg} and T parameters:

$$I_{avg} = N_{avg} \times T.$$

Delay

The delay parameter D_{stream} denotes the maximum transfer time any SDU of the stream may experience including the maximum delay jitter. Because the interval between subsequent transfer indications is constant, all SDUs will experience, from the consumer's point of view, the same delay as the first SDU, D_1:

$$D_{stream} \geq (D_n = D_{n-1}) \text{, for } n > 1.$$

Jitter

CMTP transparently compensates any delay jitter which may occur. Thus, no jitter parameter is provided.

Loss

The S_{err} parameter denotes the granularity of data errors by the maximum amount of SDU information which may be lost or corrupted by a single error. SDU information of size S_{err}, are subsequently called *error data units* (EDUs). EDUs are conveyed in one or more PDUs, or a PDU may contain multiple EDUs.

The loss parameter W_{err} denotes the lower probability bound that an EDU is delivered. In other words, W_{err} is the ratio between number of arrived and totally sent EDUs. $1 - W_{err}$ gives the upper bound on the probability that an EDU is lost due to congestion or missed deadline. Loss caused by EDU damage is not incorporated in W_{err}. However, if damaged SDU information in an EDU is detected, it is either delivered to the consumer or replaced by a pre-defined bit pattern referenced by the *DUMMY* parameter. A boolean parameter called *REPLACE* determines whether the substitution should be performed.

Miscellaneous

SDUs are structured into several *stream data units* (STDUs). The boundaries of a STDU is preserved during its transfer. The size of STDUs and the number of STDUs per SDU is controlled by the two boolean parameters *CONST_SIZE* and *CONST_NUM*. If CONST_SIZE is true, all STDUs have the same size given by $STDU_{max}$ and the number of STDUs per SDU is either constant or variable depending whether CONST_NUM is true or false. If CONST_SIZE is false, $STDU_{max}$ gives the upper bound of STDU size and N_{max} the maximum number of STDU per SDU. All in all, three different SDU patterns can be defined consisting of either fixed number of fixed-size STDUs, fixed number of variable-size STDUs, or variable number of fixed-size STDUs.

CMTP uses a shared circular buffer as service interface model at the stream source and sink. Both producer and consumer denote the memory location with the buffer parameter. The producer writes SDUs to the stream source buffer, the provider reads with a periodicity of the given SDU interval the source buffer in order to transfer them to the sink buffer, where the SDUs are periodically read by the consumer. No restriction is given at what granularity to access the buffer. In the contrary, producer and consumer may access their interface buffers in a application-dependent way, e.g. as sequences of bytes, STDUs, or SDUs.

During each SDU interval, the producer may write at most S_{max} bytes to the source buffer, which corresponds to a single maximum-sized SDU. The maximum number of bytes the producer may work ahead for one of the next SDU intervals, is given by the parameter S_{Sslack}. Thus, the source buffer is at least of size

$$S_{max} + S_{Sslack}.$$

At stream sink site, the provider assumes that the consumer reads a complete SDU every SDU interval. The parameter S_{Rslack} defines the maximum number of bytes the consumer may postpone reading to one of the next SDU intervals. Thus, the sink buffer is at least of size

$$S_{max} + S_{Rslack}.$$

3.2 ATM Performance Model

Asynchronous Transfer Mode (ATM) is a concept for cell networking being developed by the ITU-T (former CCITT) and by an interest group of users and vendors called *ATM Forum* [ATM93]. An ATM network provides communication associations at its user-network interface which may consist of several streams:

ATM Association	**Description**
unidirectional point-to-point	Consists of a single stream with a single source and sink.
bidirectional point-to-point	Consists of two unidirectional point-to-point streams in opposed direction and possibly with different stream qualities.
unidirectional point-to-multipoint	Consists of a single stream with multiple stream sinks.

The communication association control, i.e. establishing and releasing associations, is separated from the data transfer part and is performed by a signalling protocol called *Q.2931*. The latter is a modified version of the ISDN signalling protocol, *Q.931*.

The performance parameters offered at the ATM signalling interface are slightly different in comparison to the parameters used to characterize the data stream's traffic at the ATM data interface. The signalling interface parameters can be viewed as a simplified version of the data interface parameters. Thus, we describe first the data interface parameters, and later the performance model used during ATM signalling.

The following table shows the offered parameters to describe a data stream at the ATM data interface:

Parameter		**Unit**	**SNF Notation**
Throughput	Peak Cell Rate (CLP=0)	cell/sec	
	Peak Cell Rate (CLP=0+1)	cell/sec	$1/I_{min}$
	Cell Delay Variation Tolerance	time unit	
	Sustainable Cell Rate (CLP=0)	cell/sec	
	Sustainable Cell Rate (CLP=0+1)	cell/sec	$1/I_{avg}$
	Burst Tolerance	time unit	$(N_{burst}-1)(I_{avg}-I_{min})$
Delay	Cell Transfer Delay		D_n-D_1, $n > 1$
	Mean Cell Transfer Delay		
Jitter	Cell Delay Variation		
Loss	Cell Error Ratio		
	Severely-Errored Cell Block Ratio		
	Cell Loss Ratio		N_{loss}/N_{total}
	Cell Misinsertion Rate		
Misc.	Tagging	boolean	

Throughput

At the ATM layer, PDUs are called *cells*. Each cell has a constant size of 53 bytes. Because 5 bytes are used for protocol information, only 48 bytes are available for SDU information in the payload field.

The *Peak Cell Rate* (PCR) parameter defines the maximum number of SDU transfer requests per second the producer is allowed to issue at the stream source. The minimum SDU interval I_{min} is given by the inverse of PCR[1]. The maximum throughput is

$$B_{max} = \frac{1}{I_{min}} \times 48 bytes.$$

Given a constant SDU flow at peak cell rate, the *Cell Delay Variation* (CDV) *Tolerance* parameter τ denotes the maximum time a SDU transfer request may be issued earlier than scheduled. If the CDV Tolerance is greater than the SDU interval, the producer may issue several SDU requests in a single burst whose maximum number is

$$N_{burst} = 1 + \left\lfloor \frac{\tau}{I_{min} - \delta} \right\rfloor,$$

where δ is the time required to send 53 bytes (one PDU) at the ATM layer data rate (e.g. 150 Mbit/s).

The optional *Sustainable Cell Rate* (SCR) parameter denotes the average rate of SDU requests throughout the whole stream. SCR is defined by the maximum number of requests per second which must be lower than PCR.

Given a SDU flow at Sustainable Cell Rate, the *Burst Tolerance* parameter τ_s denotes the maximum time each SDU transfer request may be issued earlier, i.e. before its scheduled request time. The Burst Tolerance parameter is only significant, when SCR is defined. Thus, either both or none of them has to be given. With SCR and Burst Tolerance defined, the maximum number of subsequent SDUs sent as a single burst at Peak Cell Rate is

$$N_{burst} = 1 + \left\lfloor \frac{\tau_s}{I_{avg} - I_{min}} \right\rfloor,$$

where I_{avg} is the average SDU interval given by the inverse of SCR. Subsequent bursts of size N_{burst} can be sent with a minimum distance of $(I_{avg} - I_{min}) \times N_{burst}$.

1. Strictly speaking, the inverse of PCR has to be multiplied by 1 cell, so that the resulting interval unit is second.

Delay

The *Cell Transfer Delay* parameter denotes the (maximum) time between transfer request and corresponding indication for any SDU of the stream. The *Mean Cell Transfer Delay* parameter gives the arithmetic average of a specified number of cell transfer delays.

Jitter

The *Cell Delay Variation* (CDV) is described by two parameters: 1-point and 2-point CDV. The *1-point CDV* parameter denotes the maximum difference between reference (or scheduled) and actual SDU indication time, t_{ref} and t_{ind}. Assuming a minimum SDU interval of I_{min}, the reference indication time is defined as follows:

$$t_{ref, n+1} = \begin{cases} t_{ref, n} + I_{min} \text{ if } t_{ref, n} \geq t_{ind, n} \\ t_{ind, n} + I_{min} \text{ otherwise} \end{cases},$$

where $t_{ref,0} = t_{ind,0} = 0$ and $n > 0$. Because the difference may be positive or negative, indicating either an early or late SDU arrival, two values may be used to bound the 1-point CDV.

The *2-point CDV* parameter denotes the maximum difference between reference and actual SDU transfer delay, d_{ref} and d_n. As reference delay, the actual delay experienced by a reference SDU is used. Because the 2-point CDV difference may also be positive or negative, two values may be used to bound the upper and lower transfer delay.

Loss

Cells in ATM, and therefore SDUs, are either successfully transferred, errored, lost or misinserted: Successfully transferred SDUs arrive at the stream sink without any damage, errored SDUs arrive at the stream sink with damage due to physical interferences, lost SDUs do not arrive due to congestion or SDU header damage or missed deadline, and misinserted SDU arrives at the wrong stream sink due to undetected SDU header damage. Additionally, the loss behavior is also described in terms of *severely-errored SDU blocks*. A SDU block consists of *N* consecutive SDUs. If at least *M* of the SDUs are either errored, lost or misinserted, the SDU block is considered as severely-errored.

The *Cell Error Ratio* parameter denotes the proportion between total errored SDUs to total successfully transferred SDUs plus total errored SDUs. The *Cell Misinsertion Rate* gives the total number of misinserted SDUs observed during a specified time interval divided by the time interval duration. The *Cell Loss Ratio* parameter denotes the proportion of total lost SDUs to total transmitted SDUs.

The *Severely-Errored Cell Block Ratio* parameter denotes the proportion between the maximum number of severely-errored SDU blocks and the number of SDU blocks sent throughout the stream's lifetime (which are the total sent SDU blocks). All SDUs of a severely-errored SDU block are not considered in the calculation of Cell Error Ratio, Cell Loss Ratio and the Cell Misinsertion Rate.

Miscellaneous

Both the producer and the service provider may mark a SDU to be of low or high priority. Therefore, each cell header contains a single bit called the *Cell Loss Priority* (CLP) bit. If congestion in the node or the network occurs, the provider will first drop SDUs with low priority (i.e. CLP set) before it starts discarding SDUs with high priority (i.e. CLP cleared). Also the provider may set the CLP bit in any SDU, when it detects that the corresponding producer sends faster than negotiated. With the *Tagging* parameter, the service user may select whether a non-conforming high-priority SDU should either be discarded or tagged by the provider.

Using the CLP bit, the service user may produce a SDU flow of low and high priority. Therefore, the service user may additionally define a peak and sustainable cell rate for SDUs transferred with high priority, *PCR(CLP=0)* and *SCR(CLP=0)*. The following combination of PCRs, SCRs, and Tagging parameters are allowed:[1]

PCR (CLP=0+1)	**PCR (CLP=0)**	**SCR (CLP=0+1)**	**SCR (CLP=0)**	**Tagging**
×	×	–	–	yes/no
×	–	–	×	yes/no
×	–	×	–	no
×	–	–	–	no

1. The × sign denotes parameter is specified by the service user.

Example: PCR(CLP=0+1), PCR(CLP=0). Let PCR(CLP=0) be 5 SDUs per second and PCR(CLP=0+1) be 8 SDUs per second, the producer may issue 5 high and 3 low priority SDUs in a single burst. If the Tagging option is enabled and the producer issues 8 high priority SDUs, 3 of them will be downgraded to a low priority SDU. If both PCR values are identical, all SDUs can be sent with high priority at PCR(CLP=0+1). If PCR(CLP=0) is set to zero, all SDUs can only be sent with low priority at PCR(CLP=0+1).

Example: PCR(CLP=0+1), SCR(CLP=0), N_{burst}. Let PCR be 5 SDUs per second and SCR be 2 SDUs per second, the producer may issue 2 high priority SDUs and 3 low priority SDUs in a single burst. If the Tagging option is enabled and the producer issues 5 high priority SDUs, 3 of them will be downgraded. Thus, the producer is allowed to send bursts of low priority SDUs at the rate of PCR.

Example: PCR(CLP=0+1), SCR(CLP=0+1), N_{burst}. Let PCR be 5 SDUs and SCR be 2 SDUs per second, the producer may issue 5 high or low priority SDUs in a single burst. Because no Tagging is allowed, SDUs may be dropped due to congestion (but still low priority SDUs before high priority ones).

Using the ATM signalling to establish a communication association, only some of the above shown parameters can be defined by the service user in order to describe the required data stream performance; other presented parameters are pre-defined by the ATM provider. When establishing an association, the user may define the PCR(CLP=0), PCR(CLP=0+1), SCR(CLP=0), and SCR(CLP=0+1) for every stream of the association. Instead of a Burst Tolerance parameter, the user may directly define the Maximum Burst Size N_{burst}. The ATM provider re-calculates the Burst Tolerance τ_s from N_{burst} by

$$\tau_s = (N_{burst} - 1) \times (I_{avg} - I_{min})$$ [1].

CDV Tolerance and the cell payload size are pre-defined by the ATM provider, thus not available in the ATM signalling parameter set. Severely-Errored Cell Block, Cell Transfer Delay, Mean Cell Transfer Delay and Cell Delay Variation are also pre-defined but the service user

1. Because τ_S can not unambiguously be re-calculated by the N_{burst} formula, the ATM Forum has determined that this one has to be used.

may select between different settings. A pre-defined setting of these performance parameters is called *QoS class*. The ATM Forum specification recommends that an ATM provider should initially offer at least one QoS class for each of the following service classes:

Service Class	**Intended Application Class**
A	circuit emulation, constant bit rate video
B	variable bit rate audio and video
C	connection-oriented data transfer
D	connectionless data transfer

The subsequent table summarizes the performance parameters which the user can define individually per stream during association establishment:

Parameter		**Unit**	**SNF Notation**
Throughput	Peak Cell Rate (CLP=0)	cell/s	
	Peak Cell Rate (CLP=0+1)	cell/s	$1/I_{min}$
	Sustainable Cell Rate (CLP=0)	cell/s	
	Sustainable Cell Rate (CLP=0+1)	cell/s	$1/I_{avg}$
	Maximum Burst Size		N_{burst}
Delay	QoS class		
Jitter			
Loss			
Misc.	Tagging	boolean	

3.3 RFC 1363 Performance Model

This model is a product of the Internet comunity published as *Request for Comments* (RFC) 1363 by Partridge [Par92]. The RFC specifies a *flow specification* whose parameters characterize the traffic of a SDU stream between two users. The following table shows the offered parameters:[1]

Parameter		**Unit**	**SNF Notation**
Throughput	Maximum Transmission Unit	byte	S_{max}
	Token Bucket Rate	$V \times 2^E$ bytes/sec	$1/I_{avg}$
	Token Bucket Size	$V \times 2^E$ byte	T_{avg}/I_{avg}
	Maximum Transmission Rate	$V \times 2^E$ bytes/sec	$1/I_{min}$
Delay	Minimum Delay Noticed	$V \times 2^E$ µsec or enum	D_{max}
Jitter	Maximum Delay Variation	$V \times 2^E$ µsec	J
Loss	Loss Sensitivity	$V \times 2^E$ SDU or enum	N_{loss}
	Burst Loss Sensitivity	$V \times 2^E$ SDU	N_{err_burst}
	Loss Interval	$V \times 2^E$ SDU	I_{err}
Misc.	Quality of Guarantee	enum	

Throughput

PDUs in the RFC 1363 model are called *packets*. The maximum size of SDU information conveyed in a packet is bounded by the *Maximum Transmission Unit* (MTU) parameter. Assuming no segmentation, MTU also determines the maximum SDU size.

The traffic is described by the required maximum and average bandwidth. The *Maximum Transmission Rate* parameter denotes the maximum bandwidth in bytes per second, whereas the *Token Bucket Rate* parameter ρ defines the average stream bandwidth observable in every time period T_{avg}. The *Token Bucket Size* parameter β denotes the number of bytes over which the average bandwidth is measured. Thus, T_{avg} can be estimated by

$$T_{avg} = \frac{\beta}{\rho}.$$

1. Most of the parameters are defined in the format: $V \times 2^E$. V is a 8-bit value and E is a 7-bit value. Enum means enumeration type.

The given maximum bandwidth must be greater than or equal to the specified average bandwidth, and the size of the averaging interval β must be greater than or equal to MTU. The producer may issue every second a maximum number of SDU transfer requests which is bounded by

$$\left[\left\lceil \frac{\beta}{MTU} \right\rceil, \beta\right].$$

Note, that the second term gives the maximum number of minimum-sized SDUs conveying only a single byte of user information. The maximum number of maximum-sized SDUs issued in sequence is

$$N_{burst} = \left\lfloor \frac{\beta}{MTU} \right\rfloor.$$

Delay

The *Minimum Delay Noticed* parameter denotes either a value or a class distinguished by the first bit in the parameter field. A value denotes the transfer delay, *below* which the service user is insensitive to improvements. The provider is encouraged to establish a stream with a delay close to the specified value but not better! Instead of a value, a delay class can be selected: Class 0, denotes that the application is insensitive to delay, i.e. accepts any delay; Class 1, denotes a user which is sensitive to delay, i.e. the provider should avoid satellite links, for example.

Jitter

The *Maximum Delay Variation* parameter denotes the maximum time difference between maximum and minimum transfer delay of any SDU in the stream.

Loss

The *Loss Sensitivity* parameter denotes either a value or a loss class. A value gives the maximum number of lost MTU-sized SDUs in any interval whose length is defined by the *Loss Interval* parameter. Instead of a value, a loss class can be selected: Class 0, denotes that the service user is insensitive to loss; Class 1, denotes that the user desires a loss rate lowest as possible.

The *Burst Loss Sensitivity* parameter bounds the maximum number of MTU-sized SDU which are lost in sequence. A value of 0 indicates that the application is insensitive to burst losses.

Miscellaneous

The *Quality of Guarantee* parameter denotes a certain guarantee class which are as follows:

Class	**Description**
0	Best effort guarantee.
100 (hex)	Imperfect guarantee.
200 (hex)	Predicted guarantee.
201 (hex)	Predicted guarantee or downgrade to predicted.
300 (hex)	Deterministic guarantee.
301 (hex)	Deterministic guarantee or downgrade to imperfect.

4 A Video-on-Demand Example

In this section, we describe how the presented performance models can be applied on a video-on-demand example. The video sequence to be transferred over a network is stored on a digital storage device in a compressed format. Thus, in contrast to a live communication, the resource requirements can be estimated more precisely because the traffic shape is well known.

The exemplary video sequence is encoded in MPEG format. The *Moving Picture Experts Group* (MPEG), a working group within the *International Standards Organization* (ISO), has developed a standard for the coded representation of compressed video (moving pictures) and associated audio [ISO/IEC11172, LeG91]. The encoding scheme was originally designed to store audio and video information on digital storage media such as CDs and DATs. Nevertheless, the MPEG scheme is also applied for transferring audio and video over networks.

A MPEG encoder compresses and multiplexes multiple incoming audio and video signals into a single outgoing MPEG stream producing a variable bit rate whose upper bound is 1.856 Mbit/s[1]. Besides the coded audio and video streams, a MPEG stream contains time-stamps used by the MPEG decoder to re-synchronize the decoded audio and video signals at playback time. Between encoding and decoding, MPEG streams may be stored on a digital storage media or transferred over a network.

The standard does not restrict the number of audio and video sources coded in one MPEG stream. For our example, we have chosen a simple MPEG stream comprising only a single coded video source. MPEG-coded video images are called *pictures* or *frames*. The MPEG encoding scheme for video produces three types of frames[2]: intra, predicted and bidirectional frames.

Intra frames, or *I-frames*, are video images coded without any reference to other frames. They are produced periodically to compensate losses of predicted and bidirectional frames, which only contain image updates. Predicted frames, or *P-frames*, are image updates referring to a

1. This matches well with the typical data rates of digital storage media devices such as CD and DAT players, which is about 1.5 Mb/s. With MPEG phase II, higher bandwidth for MPEG streams will be defined.
2. A fourth type also exists defining *discrete-cosine-coded frames*. But they are only used for a fast forward search mode on digital storage media devices, thus omitted for our example.

past intra or predicted frame. They are usually smaller in size than intra frames. Bidirectional frames, or *B-frames*, provide the highest amount of compression but require as reference the last and next intra or predicted frame.

The number and pattern of predicted and bidirectional frames between intra frames is configurable. Including the first intra frame, a sequence of predicted and bidirectional frames are called *group of pictures* (GOPs).

In our example, a MPEG video stream is used consisting of 1210 encoded images of the movie *Red's Nightmare* [KKS93]. Each image is encoded into a single MPEG frame. The GOP encompasses 30 frames and its pattern of P- and B-frames is shown in Figure 6. The frame rate is 30 frames per second, thus a whole GOP is produced in every second. The frame rate corresponds to a SDU interval I_{min} of 33.3 ms.

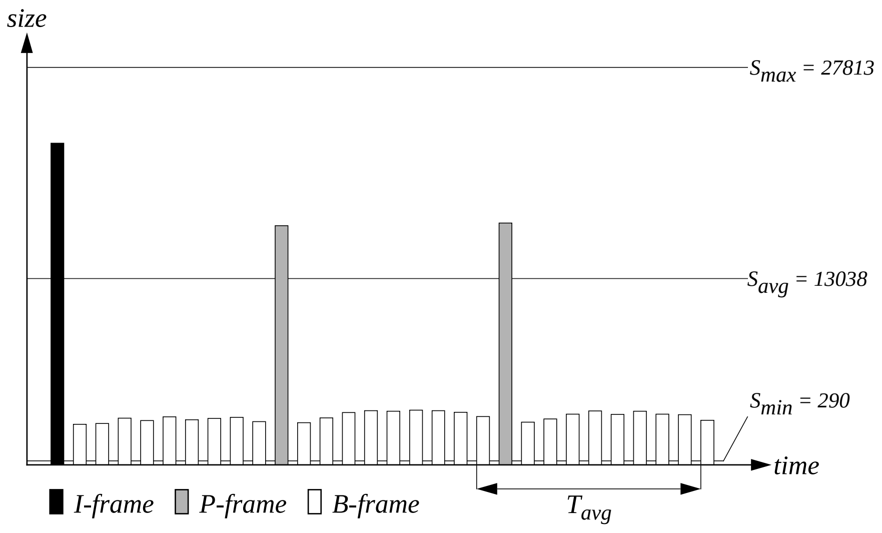

Figure 6: GOP Pattern.

The maximum frame size S_{max} is 27813 bytes required by a P-frame. A B-frame of 290 bytes was identified as the smallest frame, thus defining S_{min}. In an averaging interval T_{avg} of 10 subsequent frames, the maximum average frame size S_{avg} is 13038 bytes.

We assume that the video-display device at the stream sink tolerates a delay jitter *J* of 33.3 ms, i.e. that either the maximum interval between frames may increase to 66.6 ms (i.e. when one frame experiences a minimum transfer delay and the next frame a maximum delay) or two frames may arrive back-to-back (i.e. when one frame experiences a maximum transfer delay and the next frame a minimum delay).

Furthermore, we assume that the human user observing the video stream tolerates a maximum transfer delay D_{max} of 5 seconds (perceived as the video's start delay) and a maximum number of lost SDUs of 90 frames totally and at most one GOP in sequence (which corresponds to N_{loss}=90, N_{total}=1210, and N_{err_burst}=30)[1].

The communication performance models presented in this paper provide to applications different sets of parameters to characterize throughput, delay, jitter and loss behavior of the required communication streams[2]. Subsequently, we discuss the models' ability to describe a communication stream for our video-on-demand example.

RMTP Model

The throughput parameters of the RMTP model have been designed to characterize burstiness in terms of varying SDU intervals. Because our video stream has a constant frame rate, its burstiness (which is caused by varying frame sizes) cannot be described directly. Only if the frames are fragmented by the application into smaller SDUs (e.g. to ATM-like cells), the bursty nature of the video stream can be defined. The following table shows a possible parameter setting for a direct RMTP model usage:

Parameter		**Value**
throughput	x_min	2184×2^{-16} s
	x_avg	2184×2^{-16} s
	I	2184×2^{-16} s
	s_max	27813 byte
delay	D	327680×2^{-16} s
	Z	10000×10^{-4}
jitter	J	2185×2^{-16} s
loss	W	9300×10^{-4}

1. These values are speculative and depending strongly on the human user's tolerance.
2. Miscellaneous parameters have been omitted for simplicity.

The parameter *Z* is set to 1, requesting that all successful transferred SDUs are delivered on time. The buffer overflow probability parameter *W* is given by the ratio of correctly delivered SDUs to total sent SDUs (1210 frames). Tolerating 90 lost frames totally, the minimum number of correctly delivered frames is 1120.

CMTP Model

The CMTP model has been especially designed for continuous data streams. Thus, the parameter setting is straight-forward shown in the subsequent table:

Parameter		**Value**
throughput	T	33333 µsec
	S_{max}	27813 byte
	S_{avg}	13038 byte
	N_{avg}	10 SDU
	S_{min}	290 byte
Delay	D_{stream}	327680×2^{-16} sec
Jitter		
Loss	S_{err}	27813 byte
	W_{err}	9300×10^{-4}
	REPLACE	NO
	DUMMY	NULL

CMTP expects that the consumer reads a complete SDU (frame) every SDU interval *T*. The loss parameter W_{err} denotes the lower probability bound that SDU information of size S_{err} is delivered. Assuming that the error granularity (given by S_{err}) is equal to the maximum SDU size (i.e. a single error results in a lost or corrupted SDU), W_{err} can be given by the ratio of correctly delivered SDUs to total sent SDUs (1210 frames).

ATM Model

SDUs (cells) of ATM streams have a fixed size of 48 bytes. Thus, all frames of the MPEG video stream have to be fragmented before the ATM model can be applied. The Peak Cell Rate is 17400 cells/s, assuming that every frame contains 27813 bytes. The Sustainable Cell Rate is 8149 cells/s which is based on the assumption that all frames are filled up to 13038 bytes. This frame size is the maximum average frame size measured over the averaging interval T_{avg}, which is $10 \times 33.3\,ms$. The Maximum Burst

Size parameter denotes the maximum number of cells that can be sent continuously at Peak Cell Rate. According to the GOP pattern, a sequence of a maximum-sized P-frame (27813 bytes) and B-frame (17875 bytes) denotes the maximum cell burst which consists of 952 cells. A possible parameter setting can be as follows:

Parameter		**Value**
Throughput	Peak Cell Rate (CLP=0)	-
	Peak Cell Rate (CLP=0+1)	17400 cell/s
	Sustainable Cell Rate (CLP=0)	-
	Sustainable Cell Rate (CLP=0+1)	8149 cell/s
	Maximum Burst Size	952 cells
Delay	QoS class	"Service Class B for variable bit rate audio and video"
Jitter		
Loss		

Settings for the delay, jitter and loss parameters are pre-defined in QoS classes. Instead of specifying precisely the required delay, jitter and loss behavior, the application can only select one of the offered QoS classes that seems to be most appropriate. QoS classes are related to a specific application or service class, such as service class B for variable bit rate audio and video applications, which we have selected for our video-on-demand example.

I-frames are more important than P- or B-frames. A lost I-frame makes all related P- and B-frames obsolete and a whole GOP will be lost consequently. Thus, to indicate a higher priority of I-frames their CLP bit should be cleared.

RFC 1363 Model

The throughput parameters of the RFC 1363 model enable applications to characterize stream traffic only as a byte flow. Thus, information about frame rate and size of our video example are lost, although it could be useful to estimate the provider's resource requirements. The following table shows a possible parameter setting for a RFC 1363 model usage:

Parameter		**Unit**
Throughput	Maximum Transmission Unit	27813 byte
	Token Bucket Rate	191×2^{11} bytes/s
	Token Bucket Size	255×2^{9} bytes
	Maximum Transmission Rate	204×2^{12} bytes/s
Delay	Minimum Delay Noticed	153×2^{15} µs
Jitter	Maximum Delay Variation	131×2^{8} µs
Loss	Loss Sensitivity	90×2^{0} SDU
	Burst Loss Sensitivity	30×2^{0} SDU
	Loss Interval	152×2^{3} SDU

The Maximum Transmission Unit parameter is set to the maximum frame size of 27813 bytes, so that each PDU sent by the provider may convey a complete frame. The Maximum Transmission Rate is 834390 byte/s assuming only frames of maximum size. The Token Bucket Rate is set to 391140 byte/s, which is the average bandwidth presuming an average frame size of 13038 bytes. The Token Bucket Size denotes the averaging interval in bytes which is 130380 bytes.

The delay parameter denotes that the video-on-demand application is insensitive to transfer delay improvements which are above 5 seconds. The jitter parameter gives the time difference between maximum and minimum transfer delay, which is 33.3 ms.

The Loss Sensitivity parameter is set to 90 frames (SDUs) which are tolerated to be lost over a sequence of 1210 frames specified in the Loss Interval parameter. The Burst Loss Sensitivity parameter is set to 30, indicating that at most a GOP-sized burst of frame losses is tolerated.

Most of the parameters have to be defined in $V \times 2^E$ format which is cumbersome when precise values are given and the application programmer has, nevertheless, to find the best-fitting combination of value *V* and exponent *E*.

5 Conclusion and Acknowledgments

The presented models have been designed to enable applications to characterize their performance requirements as regards throughput, delay, jitter and loss behavior. All models have their strengths and weaknesses. Subsequently, we emphasize some of their concepts which we found interesting and useful from the application's point of view.

The CMTP model's concept to describe throughput requirements suits best presuming that in general real-time and multimedia traffic can mostly be interpreted as continuous traffic. On the other hand, the delay, jitter and loss parameter definitions of the RFC 1363 model is advantageous because it enables applications either to quantify or to classify their requirements. Also, the ATM model's concept of pre-defined QoS classes which are related to specific application classes requirements is interesting because it simplifies the specification effort and it avoids the QoS negotiation between user and provider because the available provider QoS are well known.

I would like to thank Riccardo Bettati, Mark Moran, and Bernd Petri, who read the paper very thoroughly and gave valuable comments.

6 References

ATM93	ATM Forum: **ATM User-Network Interface Specification**, Version 3.0, Oct 1993.
Boe94	S. Boecking: **A Simple Networking Framework**, in prepartion.
BuW90	A. Burns, A. Wellings: **Real-Time Systems and their Programming Languages**, Addison-Wesley International Computer Science Series, 1990.
CSD93	R. Cocchi, S. Shenker, D. Estrin, L. Zhang: **Pricing in Computer Networks: Motivation, Formulation, and Example**, IEEE/ACM Transactions on Networking, vol 1, no 6, Dec 1993.
CSZ92	D. Clark, S. Shenker, L. Zhang: **Supporting real-time applications in an integrated services packet network: architecture and mechanism**, Proc. of the ACM SIGCOMM ‘92, pp 106-114, Aug 1992.
DaG94a	S. Damaskos, A. Gavras: **The BERKOM-II FlowSpec and its Mapping on ATM**, SUPERCOMM/ICC 94, May, 1994.
DaG94b	S. Damaskos, A. Gavras: **A Simplified QoS Model for Multimedia Protocols over ATM**, High Performance Networking Conference, Grenoble, June/July, 1994.
Dan94	A. Danthine (Ed.): **The OSI95 Transport Service with Multimedia Support**, Research Reports ESPRIT, Project 5341, OSI95, Vol 1, Springer, 1994.
FBZ92	D. Ferrari, A. Banerjea, H. Zhang: **Network Support For Multimedia, A Discussion of the Tenet Approach**, ICSI TR-92-072, Nov 1992.
FeV90a	D. Ferrari, Dinesh C. Verma: **A scheme for real-time channel establishment in wide-area networks**, IEEE Journal on Selected Areas in Communications, vol 8, no 3, pp 368-379, Apr 1990.
FeV90b	D. Ferrari, Dinesh C. Verma: **Buffer Space Allocation for Real-Time Channels in a Packet-Switched Network**, TR-90-022, ICSI, 1990.
Fer90	D. Ferrari: **Client Requirements for Real-Time Communication Services**, IEEE Communications Magazine, Vol 28(11), pp. 65-72, Nov 1990.
Fer91	D. Ferrari: **Design and Applications of a Delay Jitter Control Scheme for Packet-Switching Internetworks**, Proc. 2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Nov, 1991.
FGM92	D. Ferrari, A. Gupta, M. Moran, B. Wolfinger: **A Continuous Media Communication Service and its Implementation**, GLOBECOM’92, Dec 1992.
ISO/IEC11172	Committee Draft ISO/IEC CD 11172: **Coded Representation of Picture, Audio and Multimedia/Hypermedia Information**, Dec 1991.
KKS93	A. Klingler, R. Koenig, M. Schroeder, J. Weigert: **Red’s nightmare**, MPEG movie sequence, IMMD IV, Friedrich-Alexander University of Erlangen-Nuernberg, 1993.
LeG91	D. LeGall: **MPEG: A Video Compression Standard for Multimedia Applications**, Communications of the ACM, Vol 34, No 4, Apr 1991.
Par92	C. Partridge: **A Proposed Flow Specification**, Network Working Group, RFC 1363, Sep 1992.

Par94	C. Partridge: **Gigabit Networking**, Addison-Wesley Professional Computing Series, 1994.
Sch93	H. Schulzrinne: **Issues in Designing a Transport Protocol for Audio and Video Conferences and other Multiparticipant Real-Time Applications**, IETF draft, Oct 1993.
VeZ91	D. Verma, H. Zhang: **Design document for RTIP/RMTP**, ICSI internal document, May 1991.
WoM91	B. Wolfinger, M. Moran: **A Continuous Media Data Transport Service and Protocol for Real-Time Communication in High Speed Networks**, Proc. 2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Nov, 1991.
WoM92	B. Wolfinger, M. Moran: **Design of a Continuous Media Data Transport Service and Protocol**, TR-92-019, ICSI, Apr 1992.
WSW93	A. Wolisz, I. Schieferdecker, M. Walch: **An Integrated Approach to the Design of Communication Protocols**, Proc. of the 4th Workshop on Future Trends of Distributed Computing Systems, Lisboa, Sep 1993.