Enhanced transport protocols

Nicolas van Wambeke, Ernesto Expósito, Guillaume Jourjon, Emmanuel Lochin

5

Enhanced Transport Protocols

Nicolas Van Wambeke, Ernesto Exposito, Guillaume Jourjon, and Emmanuel Lochin

Summary. The deployment of QoS network services and the entitled set of services offered by existing transport protocols have motivated the design of new transport protocols. In this chapter, we present a set of standardised transport protocols and advanced transport protocol mechanisms to support Quality of Service and satisfy new application requirements.

5.1 Introduction

At the network layer, despite several studies on the end-to-end QoS guarantees provisioning, the high complexity of these mechanisms results in the domination of Best-Effort service as the only Internet network service available. The aim of the enhanced transport protocols is to provide an end-to-end QoS-oriented service and QoS optimisations or guarantees for current and future multimedia applications by using available resources and network services (i.e., new network services).

In Sect. 5.2, we introduce the basic transport mechanisms provided by standard transport protocols (i.e., TCP, UDP, SCTP and DCCP) and, in particular, congestion and error-control mechanisms. Then, Sect. 5.3 details transport mechanisms used to implement common transport services. We will see that traditional and more recent transport protocols are not designed to offer QoS-oriented services. Indeed, these protocols have mainly focused on the implementation of congestion-control mechanisms to protect and save network resources (i.e., TCP, SCTP and DCCP) while satisfying application requirements only partially. Finally, Sect. 5.4 presents enhanced transport protocol mechanisms based on QoS-aware error- and congestion-control mechanisms. Both error-control and congestion-control mechanisms take into account intrinsic media flow characteristics, network QoS and application time constraints. The QoS-aware error-control mechanisms are intended to provide partially ordered and partially reliable (PO/PR) services. A differentiated and partially reliably service (D-PR) is proposed to satisfy specific reliability requirements of scalable multimedia streams. In order to satisfy specific time requirements, the service

is enhanced to provide a time-constrained and differentiated, partially reliable (TD-PR) service. The TFRC congestion-control mechanism allows enhanced transport protocols QoS awareness provided by the network, while at the application level the TD-TFRC congestion-control mechanism takes into account the time and reliability constraints. The composition of error- and congestion-control mechanisms aimed at supporting a large set of transport services is finally presented.

5.2 State of the Art of Transport Protocols

The different services available at the transport layer include the conceptual OSI transport layer model.

The Open System Interconnection (OSI) reference model describes how information from an application in one computer moves through a communication medium to an application in another computer. The OSI reference model is a conceptual model composed of seven layers, which details specific communication functions. The model was developed by the International Standardisation Organisation (ISO) in 1984, and it is now considered the base model for interconnected computers. Indeed, while layers 5 to 7 of this model have been considered as too complex for the service they provide, layers 1 to 4 represent the very basic parts of communication systems.

The transport layer in the OSI model is the lowest layer that operates on an end-to-end basis between two or more communicating hosts [91]. This layer is located at the boundary between the host applications and the network layer. A “service” is defined as the abstraction capability offered by an OSI model layer to a higher one. Transport services enable applications to abstract from the details of the available network services. A transport user generally performs three phases of operation: connection establishment, data transfer and connection termination. A connection-oriented transport service provides primitives for the three operations and a connectionless service supports only the data transfer phase. Moreover, a connection-oriented service maintains state information about the connection (i.e., messages sequence number, buffer sizes, etc.). Transport services and protocols can differ in several characteristics:

- Message and byte-stream services: In the case of a message service, users send messages or service data units (SDUs) having a specified maximum size and message boundaries are preserved. In the byte-stream service the data sent by the user are transported as a flow of bytes and message boundaries are not preserved.
- Reliability: comprising no-loss, no-duplicates, ordered and data integrity transport service.
- Blocking and nonblocking: A nonblocking service allows the sender to submit data and continue operating without taking into account the transport layer buffering capabilities or the rate at which the user receiver consumes data. A blocking service avoids overloading the transport layer with incoming data.

- Multicast and unicast: A multicast service enables a sender to deliver data to one or more receivers. A unicast service limits the data delivery to exactly one user receiver.
- Priority: A priority service enables a sending user to indicate the relative importance of various messages.
- Security: authentication of transport users, access control to resources, confidentiality and integrity of data, etc.
- Feedback or status-reporting service: allows a transport user to obtain specific information about the transport entities (i.e., performance, addresses, current timer values, statistics, etc.).
- Quality of service: A transport layer that explicitly provides QoS allows the quality specification of the required transmission service by a sender (i.e., delay, jitter, order, reliability, throughput, security, priority, etc.).

5.2.1 TCP and UDP

Transmission Control Protocol (TCP) offers a reliable and in-sequence end-to-end data transfer service between two interconnected systems [92]. TCP is a connection-oriented and byte-stream-oriented service. The User Datagram Protocol (UDP) has been proposed to offer a light transport service for messages or datagrams [93]. This UDP light transport service is implemented without the (time-consuming) connection phase and without the (resource-consuming) errors, congestion- and rate-control mechanisms. Therefore, UDP offers a connectionless and message-oriented service with no order and no reliability guarantees. Both TCP and UDP implement a multiplexing mechanism to support different applications transmitting several data flows using the same IP address. Both protocols implement mechanisms for the detection of corrupted data (i.e., checksum), where if a given set of bit errors is detected at the receiving side, the data packet is discarded. TCP implements error reporting and recovery mechanisms in order to provide a fully reliable service. Moreover, TCP implements flow and congestion-control mechanisms in order to avoid exceeding receiver buffers capacities and network congestion. Error, flow and congestion-control mechanisms implemented by TCP may induce transmission delay and variable throughput. Sometimes these effects are not compatible with application requirements such as multimedia applications, which demand throughput and delay guarantees. As a result, these applications have been implemented using UDP in combination with other protocols, i.e. Real-Time Transport Protocol (RTP)/Real-Time-Control Protocol (RTCP) [94], in order to obtain a more suited transport service.

5.2.2 TCP Evolution

TCP is at present the most widely used transport protocol in the Internet. Nevertheless, while the generic name has remained the same, the current TCP version used is different from the first TCP version in the early 1980s by Postel [92]. In this section, we describe four main TCP variants and the reasons that have led to the definition

of these different versions: TCP Tahoe [95], TCP Reno [96], TCP Vegas [97] and TCP New Reno [98] with SACK [99]. Finally, we present the latest version of TCP algorithms that has better performance over high-throughput networks and wireless networks.

5.2.2.1 TCP Tahoe

In the first TCP version described by Van Jacobson in [95], the congestion control mechanism is based on the estimation of losses by the sender, and a congestion window regulates the number of packets that can be sent over the network (i.e., emitted rate). The increase of this congestion window follows two different stages: the slow-start and the congestion-avoidance phase. In the slow-start phase, the congestion window grows exponentially until a certain threshold has been reached. Once the protocol has reached this value, it follows a congestion-avoidance phase where it only increases the congestion window by a value of one more segment.

This version of TCP suffered from numerous drawbacks. The first concerns the Go-back-N error-recovery mechanism. This mechanism is not efficient, mainly because it can only retransmit packets that have already been received. The second drawback of this TCP version concerns the method for the detection and recovery of losses. Loss detection in this version is done using a timer that is triggered for every packet and staying active, either until the reception of the corresponding acknowledgement packet or until a fast-retransmit occurs. Furthermore, when the loss is finally detected, the protocol goes back to the slow-start phase with a threshold value of the half of the actual congestion window.

In order to solve these problems, new mechanisms are required: the selective repeat mechanism for the problem of the error-recovery mechanism, and fast-recovery added to the fast-retransmit mechanism for the problem of the loss detection.

5.2.2.2 TCP Reno

TCP Reno took into account the drawbacks of TCP Tahoe in order to improve the protocol. It modifies the fast-retransmit algorithm that could have been implemented in the Tahoe version to integrate the fast-recovery algorithm. The fast-recovery algorithm halves the congestion window instead of going back to the slow-start algorithm. This congestion window is increased during this period by the number of duplicate ACKs. Furthermore, this version applies the selective repeat mechanism for the recovery of packets lost. Nevertheless, these mechanisms also contain some minor drawbacks: the successive fast-retransmit problem or the false fast-retransmit followed by a false-recovery problem. This version also suffers from performance problems when multiple packets are dropped in the same sending window.

5.2.2.3 TCP Vegas

TCP Vegas [97] proposes new algorithms for the slow-start phase, the estimation of the available bandwidth in the congestion-avoidance phase and loss detection.

In order to detect congestion in the network, TCP Vegas defines a *BaseRTT* as the minimum measured *RTT* and the *ExpectedRate* as the ratio of the congestion window to the *BaseRTT*. Furthermore, the sender measures the *ActualRate* based on the sample *RTT*. Then, if the difference between the *ExpectedRate* and the *ActualRate* is superior to an upper bound, the sender linearly decreases the congestion window during the next *RTT*. Otherwise, if this difference is lower than a lower bound, the sender linearly increases the congestion window. According to [97], this TCP version achieves a better rate than the Tahoe and Reno TCP versions. Nevertheless, this version was never deployed due to scalability and stability concerns identified later in [100, 101]. One of the main drawbacks of TCP Vegas concerns its poor performance when mixed with other TCP versions.

5.2.2.4 TCP New Reno

In order to improve the behaviour of TCP Reno when multiple packets are lost in the same window, TCP New Reno has been proposed. In this version, a modified version of the fast recovery algorithm has been integrated, where partial ACKs are used to indicate multiple losses in the same window. This new fast-recovery algorithm has been described in [98]. This version of TCP appears to be adequate for wired networks where the bandwidth-delay product is not too high, but performs poorly over high bandwidth-delay product and wireless networks. For instance, the problems over wireless networks are due to the interpretation of a lost packet, because the TCP loss-detection mechanism supposes that packets are lost because of congestion in the network. However, in a wireless network, these losses can be due to channel errors or bad transmission.

These facts have motivated new TCP enhancements that are presented in the following section.

5.2.2.5 TCP Variants for High-Throughput and Wireless Networks

Nowadays, networks offer increasing bandwidth capacities to the end user. Nevertheless, TCP alone cannot take advantage of these new services, because of the additive increase/multiplicative decrease (AIMD) algorithm, which makes TCP too slow to adapt its sending rate to the network's bottleneck. In order to solve this problem, two kinds of transport protocols, based on a variation of the TCP AIMD congestion avoidance phase algorithm, have been proposed. The first proposals remain close to TCP as they do not require routers to modify their internal algorithms and marking, such as in BIC, HSTCP or STCP protocols [86, 102, 103]. The second set of proposals, such as XCP [104] and VCP [105], performs better than both TCP New Reno, but requires the network to provide information about the actual network congestion level using an ECN-like mechanism [106].

Nowadays, another TCP problem concerns its poor performance over wireless networks. TCP has been designed to perform over wired networks where a packet lost means network congestion. As a result, the congestion-control mechanism—as described above with the fast-recovery and fast-retransmit mechanism—decreases

the congestion window, in order to reduce the congestion in the network. Nevertheless, in the case of wireless communications, losses can also be due to urban obstacles, mobility of devices or channel interferences. In this context, new versions of TCP congestion control have been proposed such as WTCP [107] or TCP Westwood [108]. Another solution, TCP Veno (Vegas + Reno), proposed to let TCP Vegas estimate the available bandwidth and to act as TCP Reno when a loss is detected due to congestion [109]. Nowadays, the current TCP used in the Internet are TCP New Reno (*BSD), TCP Westwood (Microsoft Windows Vista), and a TCP BIC variant (GNU/Linux) all with selective acknowledgement (SACK) enabled by default.

5.2.3 SCTP

The Stream Control Transmission Protocol (SCTP) is a reliable transport protocol, which operates on top of a connectionless packet network protocol, such as IP, and offers a reliable transport service [86]. SCTP is unicast and session oriented. An SCTP session is an association established between two end systems. When end systems present several network addresses, the address list is exchanged during the association establishment phase. Supporting several IP addresses within the same SCTP session is known as multihoming. The SCTP error control detects lost, disordered, duplicated or corrupted packets and uses retransmission schemes to implement the error-recovery mechanism. SCTP uses selective acknowledgements (SACK) in order to confirm data reception. Retransmission is done after a time-out or when the SACKs indicate that some SCTP packets have not been received. In contrast to TCP, SCTP is a message-oriented transport protocol. SCTP packets are composed of a common header and data chunks. Multiple chunks may be multiplexed into one packet up to the path-MTU size. A chunk may contain either control information or user data. SCTP offers a multistream service, which means that application data can be partitioned in multiple flows that can be delivered using several independent-ordered streams. SCTP does not enforce any ordering constraints between the different streams. It provides a full-ordered intrastream service and a full-unordered interstream service. This service guarantees that data delivery over the rest of streams is not affected if some loss or disordering is detected in a stream. In contrast, flow and congestion control are implemented on the association basis and not independently for every stream. These mechanisms are based on the TCP algorithms: the receiver informs the sender about the reception buffer capacity and a congestion window is maintained for the SCTP association. The slow-start, congestion-avoidance, fast-recovery and fast-retransmission mechanisms are implemented following the TCP algorithms, but using the SCTP packets as the acknowledgement unit. Applications accepting a partially ordered transport service can take advantage of the multistream service provided by SCTP. However, the effects of the fully reliable service of SCTP can be incompatible with multimedia applications presenting bandwidth, delay, jitter and synchronisation constraints. Currently, a partial reliability extension to SCTP has been proposed [110]. In this proposal only a timed reliability service has been specified. Timed reliability means that the user can specify the lifetime of a message. However, this mechanism could be insufficient for time-constrained applications pre-

senting specific ADU reliability requirements and interdependency constraints (e.g., I, P and B frames of MPEG video).

5.2.4 DCCP

The Datagram Congestion Control Protocol (DCCP) offers a nonreliable transport service for datagram flows [111]. DCCP provides a congestion-control mechanism in order to behave fairly with other TCP flows. DCCP uses a Congestion Control Identifier or CCID to identify the congestion control to be used for each direction of the DCCP connection. This congestion control can be configured according to two profiles numbered Congestion Control ID (CCID) 2 or 3. In CCID 2 [112], a congestion control similar to the window-based congestion control is provided. In CCID 3 [113], a rate-based congestion control is provided. This congestion control is based on the TCP-Friendly Rate Control (TFRC) [114].

DCCP is suited to applications currently using UDP. In order to avoid network congestion, applications that use UDP should implement their own congestion-control mechanism. DCCP aims to deliver a transport service that combines both the efficiency of UDP and the congestion control and network friendliness of TCP. DCCP allows the negotiation of some connection properties to be used such as the congestion-control mechanism.

DCCP can be used by applications presenting time constraints and which are able to adapt their transmission rate to the limitations imposed by the congestion-control mechanism. If a certain reliability is needed, DCCP designers propose to implement error-recovery mechanisms (i.e., data retransmission or redundancy) at the application level. Nevertheless, the implementation of these mechanisms at the user space increases development efforts and can be less efficient than transport mechanisms operating in the kernel space.

5.2.5 Discussion

In this section a survey of the main transport protocols has been presented. This study demonstrates that traditional and new generation transport protocols have been designed taking into account only a subset of the QoS requirements of multimedia applications. These protocols have mainly focused on the implementation of congestion-control mechanisms to save network resources (i.e., TCP, SCTP and DCCP) while providing either full order and full reliability or nonorder and non-reliability. Moreover, mechanisms intended to satisfy time constraints are not supported at the transport layer. A QoS-oriented transport service based on the delay, jitter, throughput and synchronisation constraints of the multimedia applications and taking into account the partial order and partial reliability tolerance, as well as the scalable characteristics of multimedia flows, has not yet been provided. This is the reason that led us to propose the design of a new generation transport protocol aimed at providing end-to-end QoS guarantees. These protocols have to be designed to be easily configured from the requirements of current and future multimedia applications. This means that, first, the services provided by enhanced transport protocols

should be transparently deployed for existing applications, minimising the adaptation efforts (i.e., preservation of standard transport API, transparent adaptation to legacy RTP streaming applications, etc.). Second, these specialised services should be accessible to new multimedia applications that are able to explicitly specify their QoS requirements (i.e., providing a QoS-enhanced transport API). Moreover, transport mechanisms taking into account underlying communication services (i.e., network services) could be implemented to provide soft or hard end-to-end QoS guarantees. A survey of common transport mechanisms implemented at the transport layer is presented in the next section.

5.3 Transport Mechanisms

The previous section presented services provided by common transport protocols. In this section, the basic mechanisms generally used to implement these common transport services are introduced. In particular, the congestion-control and error-control mechanisms are described.

5.3.1 Overview

In [115], the authors gave the following definitions of the basic transport layer mechanisms. They focused particularly on error and congestion, which are the two pillar mechanisms of a transport protocol.

- Error-control techniques protect against loss or damage of user data and control information. Error control is performed in two phases: error detection, and error reporting and recovery. Error detection identifies lost, disordered, duplicated and corrupted Transport Protocol Data Units (TPDUs). Error reporting is a mechanism where the transport receiver informs the sender about errors detected. Error reporting may be implemented by positive acknowledgement of data received (ACK) or negative acknowledgement of errors detected (NACK). Error recovery mechanisms are used by the sender or receiver in order to recover from errors. They can be implemented by retransmission strategies (i.e., Automatic Repeat request or ARQ) or redundancy mechanisms (i.e., Forward Error Correction or FEC).
- Flow and congestion control: Flow control is a mechanism implemented by the transport layer to limit the rate at which data are sent over the network in order to avoid exceeding the capacity of the transport receiver entity. Congestion-control mechanisms are intended to preserve network resources in order to avoid network congestion.
- Multiplexing and demultiplexing are mechanisms implemented by a transport protocol in order to associate several Transport Service Access Points (TSAPs) to a single Network Service Access Point (NSAP). In other words, this mechanism enables supporting several transport layer connections using the same network

layer connection. The use of protocol port numbers to perform the multiplexing/demultiplexing operations allows multiple transport users to use the same network address.

- Segmentation and reassembly: When the size of the Transport Service Data Units (TSDUs) is bigger than the allowed size of the Network Service Data Units (NSDUs), the TSDUs have to be segmented into smaller TPDUs by the transport sender. The transport receiver reassembles the TPDUs in order to rebuild the TSDUs to be delivered to the user receiver.
- Other mechanisms: Some transport protocols implement other specialised mechanisms, such as concatenation/separation and splitting/recombining. Concatenation combines several TPDUs into one NSDU in order to reduce the number of NSDUs submitted to the network layer. Separation of concatenated TPDUs will be performed by the transport receiver entity. The splitting operation is opposite to the multiplexing functions. Several NSAPs can support a single TSAP, thus providing additional resilience against network failures and increasing the throughput capabilities.

The next paragraphs will present a detailed study related to congestion-control and error-control mechanisms intended to evaluate the most adequate mechanisms for enhanced transport protocols.

5.3.2 Congestion-Control Mechanisms

The Internet protocol architecture is based on a connectionless end-to-end packet service using the IP protocol. These characteristics offer advantages in terms of flexibility and robustness, but a careful design is also required to provide good service under a heavy load. In fact, lack of attention to the dynamics of packet forwarding can result in severe service degradation. This phenomenon is technically called “congestion collapse” [116]. Network congestion is characterised by excessive delay and losses in delivering data packets. Congestion-control mechanisms are intended to avoid network congestion and its consequences. In [117] congestion-control mechanisms are classified into rate-control, rate-shaping and rate-adaptive encoding. Rate-control and rate-shaping mechanisms can be implemented as transport layer functions. In contrast, rate-adaptive encoding is based on compression techniques, being suitable for implementation as an application layer function. In this work, only rate-control and rate-shaping mechanisms able to be implemented at the transport layer will be studied.

5.3.2.1 Window-Based Congestion Control

Window-based congestion control was originally proposed for the TCP transport protocol [118]. This mechanism probes the available network bandwidth by slowly increasing a congestion window limiting the data being inserted into the network by the source. Detection of packet loss is considered as an indication of network congestion and the congestion window is reduced in order to avoid network collapse.

TCP-like congestion-control mechanisms are typically coupled with error-correction mechanisms (i.e., retransmissions), which can increase data delivery delay. For this reason, these mechanisms are considered as being noncompliant with the time constraints of multimedia flows.

5.3.2.2 Rate-Based Congestion Control

The rate-based congestion control mechanisms are characterised by the use of an estimation of the available network bandwidth as the allowed transmission rate. Rate-based mechanisms may use information about losses in order to calculate the transmission rate. This characteristic makes the rate-based mechanisms more suitable for controlling network congestion when full reliability is not required. Rate control can be performed by the source, the receiver or a combination of both [117]. The available network bandwidth can be estimated following a probe-based or a model-based approach.

5.3.3 Reliability Mechanisms

Packet loss is one of the consequences of network congestion in traditional IP networks. During data transmission, packets are temporarily stored in intermediate nodes before being forwarded to their final destination. When storage capacities are exceeded, some packets are dropped. As explained in previous sections, the degree of tolerance of packet loss depends on the type of application. Some applications are able to tolerate a certain degree of packet losses (i.e., audio, video, images, etc.), while for other applications packet loss is unacceptable (i.e., data files, text, etc.). Two main error-control mechanisms have been proposed for implementation at the transport layer. The first mechanism is known as Forward Error Correction or FEC [117]. FEC is performed by the data source and it is based on adding redundant information to the data packets to be transmitted. Redundant information will be used by the receiver to recover lost packets. The second error-control mechanism, called Automatic Repeat Request or ARQ, is based on the retransmissions of lost packets. Retransmissions can be demanded by the receiver when losses are detected or be performed by the source if the packets are not acknowledged after some time. Other mechanisms, such as error resilience and concealment, have been proposed at the application layer. Error-resilience mechanisms are intended to prevent error propagation or to limit the consequences of packet losses in the compressed data flow (i.e., synchronisation marks, data partitioning, etc.) [119]. Error-concealment mechanisms are performed by the receivers in order to hide the consequences of losses to the final user (i.e., spatial or temporal interpolation, replacement of lost frames by the previous ones, etc.) [120]. Hereafter, only the error-control mechanisms capable of implementation at the transport level will be studied.

5.3.3.1 Automatic Repeat Request

Automatic Repeat Request (ARQ) is an error-control mechanism based on the retransmission of packets considered as lost or damaged. This mechanism has been

used to provide reliability in a number of communication protocols. Loss detection and recovery signalling techniques are specific to each communication protocol. The next paragraphs introduce some of these protocols and show how the ARQ mechanism has been implemented.

Stop-and-Wait ARQ offers the simplest reliability service at the transport level. A sender using the Stop-and-Wait protocol transmits a TPDU and then waits for a response. The receiver waits for a TPDU and sends an Acknowledgement (ACK) if the TPDU has been correctly received or a Negative Acknowledgement (NACK) otherwise. In practice, the receiver may not be able to detect lost TPDUs, and the sender needs to implement a timer to retransmit the TPDU when no response has been received from the receiver. If an ACK has been received, the transmitter can start sending the next TPDU. In order to detect duplicated TPDU or ACKs, each message has to be uniquely identified using, for instance, a sequence number.

In contrast to the Stop-and-Wait mechanism, Go-Back-N ARQ allows the simultaneous transmission of multiple TPDUs, as allowed by the transmission window size. Each TPDU must be identified uniquely by a sequence number. The source must keep in memory all the TPDUs that have been sent, but have not been yet acknowledged. The receiver must keep in memory the highest TPDU sequence number correctly received. When a packet loss is detected, the receiver sends a negative acknowledgement packet and all the received packets with a sequence number higher than the lost packet are discarded. The source restarts the Go-Back-N retransmission, since the TPDU corresponding to the sequence number indicated in the NACK packet.

Selective Repeat is a more complex, but more efficient error-control mechanism. This scheme is employed by the TCP transport protocol. Similar to the Go-Back-N mechanism, the retransmission is performed in response to the selective repeat feedback sent by the receiver. However, the sender retransmits only the TPDU for which the selective repeat has been indicated. This feature reduces the number of retransmissions, but increases the complexity at the sender and receiver. Indeed, each TPDU must be acknowledged individually, and the receiver must keep in memory packets received out of sequence.

In [95] a TCP extension was proposed, implementing the concept of Selective Acknowledgement (SACK). By sending selective acknowledgements, the receiver of data can inform the sender about all segments that have arrived successfully, so the sender needs to retransmit only the segments that have actually been lost. In [121] an implementation of this Selective Acknowledgement mechanism combined with a selective repeat retransmission policy was proposed.

5.3.3.2 Flow Control

In order to prevent the receiver from dropping packets because its receiving buffer is full, the receiver needs to inform the sender about the available space in its buffer. Usually this mechanism is performed based on the exchange of information concerning the available buffer space at the receiver and the update of a variable representing

this information at the receiver each time a packet is sent. As a result, the sender will stop sending data when it supposes that the buffer is full—even if this is not correct.

5.3.4 Discussion

In this section, a survey of different congestion and error control mechanisms has been presented. Congestion-control mechanisms including transport- and application-based approaches have been presented. Within these mechanisms, a rate-based congestion control seems to be more suitable. Indeed, rate-based mechanisms have been designed for implementation at the transport level and independently of error-control mechanisms. Furthermore, these mechanisms could be enhanced in order to be more compliant with time-constrained applications. This enhancement can be done using the scalable characteristics of media flows (i.e., using a rate-shaping approach) in order to take into account not only the available network resources, but also intrinsic time constraints as well as partial ordering and partial reliable tolerance of applications. Regarding error-control mechanisms, FEC and ARQ have been introduced. These mechanisms could be enhanced in order to be more compliant with time-constrained applications.

5.4 Enhanced Transport Protocol Mechanisms

5.4.1 TFRC and gTFRC, a QoS-Aware Congestion Control

TCP-Friendly Rate Congestion Control (TFRC) is an equation-based rate-control mechanism, which aims at reproducing the behaviour of TCP congestion control. The TCP equation presented in [122] and used in TFRC is

$$X = \frac{s}{\left(RTT \cdot \sqrt{\frac{p \cdot 2}{3}} + RTO \cdot \sqrt{\frac{p \cdot 27}{8}} \cdot p \cdot (1 + 32 \cdot p^2)\right)}. \tag{5.1}$$

The use of an equation instead of the AIMD algorithm (used by window-based congestion-control mechanisms) in order to estimate the sending rate produces smoother throughput variations. Furthermore, the TFRC congestion control is based on a datagram-based communication instead of the stream-based TCP connection.

Even if the knowledge of the guaranteed bandwidth could be provided to the transport level, the AIMD principle integrated into TCP does not use the instantaneous throughput as an input value for its congestion control. Only acknowledgements and time-out analysis allow TCP to act on the rate control. On the contrary, the TFRC mechanism makes use of the instantaneous throughput in conjunction with the flow RTT and loss events. These parameters are used in order to compute the controlled rate. In the following, we assume a network that is well-provisioned and that the whole in-profile traffic does not exceed the resource allocated to the considered class of service; for instance, the AF class. In case of excess bandwidth in the network, the application can send more than its given target rate, say g, so the network

should mark its excess traffic out-of-profile. If the network becomes congested, this out-of-profile traffic is predisposed to be lost first. In such a case, the optimal rate estimated by TFRC still can be below the target rate g needed by the application and provided by an underlying CoS; for instance, a DiffServ network. In such a case, TCP would react in the same manner by halving its congestion window. As for TCP in the AF class, the TFRC flow is not aware that the loss is corresponding to an out-of-profile packet and that it should not decrease its emitted throughput below the target rate.

In contrast to TCP, the usage of the TCP equation allows the direct usage of the actual sending rate in conjunction with the flow RTT and loss event values. A new resulting congestion control mechanism, called gTFRC, can be thus made aware of the target rate negotiated by the application with the DiffServ-like network. Thanks to this knowledge, the application flow is sent in conformance with the negotiated QoS while staying TCP-friendly in its out-of-profile traffic part. This is achieved by computing the sending rate as the maximum between the TFRC rate estimation and the CoS target rate as given in (5.2):

$$G = \max(g, X). \tag{5.2}$$

G is the transmission rate in bytes/s; g is the target rate in bytes/s and X is the transmission rate in bytes/s computed by the TCP throughput algorithm. The rest of the gTFRC mechanism follow entirely the TFRC specification in [114].

gTFRC requires the knowledge of the underlying bandwidth guarantee provided by the DiffServ/AF network service to the session. This information is available to the mechanism at socket-creation time, directly by the application. This parameter is given to the application, after it has been previously negotiated in an end-to-end basis by the signalling mechanism provided by the QoS architecture (as an example see the EuQoS architecture in Chap. 6).

5.4.2 Application-Aware Transport Mechanisms

In this section, the enhancement of specific transport mechanisms in order to make them more compliant with the application's QoS requirements while taking into account constrained network services (i.e., Best-Effort service) is proposed. This enhancement is done by a cross-layer communication between the transport protocol and the application layers.

5.4.2.1 Application Profile-Aware Congestion Control

Currently, most commercial multimedia applications (i.e., streaming and conferencing applications) do not implement congestion-control mechanisms. The implementation of a congestion-control mechanism for these kinds of applications is required in order to reduce the risks of future congestion collapse of the Internet, due to flows that do not use end-to-end congestion control [116]. These applications are usually implemented using RTP/UDP protocols to transmit media flows. Real-time flows

are either transmitted using a near-constant rate or an adjustable rate based on the feedback obtained from the receiving application (i.e., RTCP messages). But even when applications are able to adapt their sending rate, it is usually done in long time scales. Some studies have been conducted in order to propose congestion control mechanisms adapted to the characteristics of these applications [123]. As we have seen, one of these mechanisms is the TCP-Friendly Rate Control (TFRC).

The rate-control mechanism of TFRC is based on a delaying policy aimed at adapting the flow to the allowed sending rate. This mechanism can penalise applications with strict delay constraints. For these applications, received packets could be discarded if they arrive too late to be presented. An alternative to the delaying policy implemented by TFRC may be a quality adaptation policy. Quality adaptation mechanisms can be performed by applications (i.e., adaptive encoding, switching between multiple preencoded version, etc.). But usually these mechanisms are executed in long time scales. We propose performing quality adaptation at the transport level. This requires that QoS information describing the multimedia flows must be available at the transport layer. This information must include at least the time constraints associated to every ADU as well as specific QoS information aimed at performing the quality adaptation (i.e., ADU priorities, dependency, order, etc.). The delaying strategy of TFRC is based in a computation of the interpacket interval time (IPIT) for every data packet to be transmitted. TFRC calculates this IPIT value as follows:

$$IPIT = \frac{s}{r}, \tag{5.3}$$

where s is the packet size and r is the allowed sending rate. The IPIT value represents the time to delay the current data packet in order to respect the allowed sending rate. If the QoS information associated to data packets include the delivery time stamp of every packet to the receiving application, then the feasibility of this delaying strategy could be checked, taking into account the end-to-end delay of the applications. The one-way delay must be known in order to perform this temporal validation. The one-way delay can be estimated using the RTT estimated in TFRC:

$$oneWayDelay = \frac{RTT}{2}. \tag{5.4}$$

Using the one-way delay, the delivery time stamp of the current data packet can be calculated as follows:

$$eDeliveryTimestamp = now + IPIT + oneWayDelay, \tag{5.5}$$

where now is the current time. Data packets can be considered as obsolete by the receiving application if the following equation is valid:

$$eDeliveryTimestamp - timestamp > MAXDELAY. \tag{5.6}$$

The time stamp is the scheduled delivery date. MAXDELAY expresses the delay tolerance of the application (i.e., 400 ms for interactive application). These obsolete packets will be generally discarded by receiving applications. However, if the

temporal validation is performed by the source, discarding could be anticipated in order to avoid wasted bandwidth. Nevertheless, this basic discarding policy could seriously affect the QoS perceived by the final user if important Application Data Units (ADUs) are discarded. Selective frame-discarding methods based on ADU-related QoS information can be used to optimise the QoS provided to the user, while preserving network resources and respecting the application delay constraints. This selective frame-discarding method can be applied if the medium has been encoded using specific compression and ADU segmentation techniques, which facilitate the implementation of this method at the transport layer (i.e., Application Level Framing/ALF approach for the segmentation of flows such as MPEG, H.263, MJPEG, etc.). This method could also be implemented at the multimedia transport level, where the TFRC sending rate can be shared between the flows comprising the multimedia session. In this case, scalable flows could grant part of the allowed rate to other flows presenting poor adaptive capabilities (i.e., audio flows). Similarly to the multilayered multicast flows, for transport-level quality adaptation strategy, several quality layers could be defined using the QoS description of the ADUs composing the multimedia flows. For instance, for an MPEG flow composed by I, P and B images, three quality layers could be defined:

- Layer 2: I, P and B images
- Layer 1: only I and P images
- Layer 0: only I images

If the TFRC sending rate is lower than the rate demanded for the layer 0, other layers composed by subsets of I images and presenting lower rates could be defined. For a multimedia session, the same scheme can be used. For instance, for a session composed by a H.263 video with I and P pictures and a GSM audio flow, the following quality layers could be defined:

- Layer 2: Video (I+P) + Audio
- Layer 1: Video(I) + Audio
- Layer 0: Only Audio

Likewise, new layers could be defined for this multimedia session using intermediate quality levels placed between the specified layers. The definition of these differentiated layers allow us to propose an enhancement to the TFRC algorithm (TD-TFRC) intended to provide a rate control compatible with the time constraints and the intrinsic characteristics of multimedia flows. The next algorithm describes this specialisation of the TFRC mechanism:

```
currentLayer=0 join layer(currentLayer) while (sessionIsActive)
{
/* When feedback received or noFeedBack timeout : estimation of
TFRC parameters and compute r */

  /* filtering */
  if (currentPacket.layer > currentLayer)
      currentPacket.discard = true
  else {
```

```
// inter-packets interval
IPIT = currentPacket.size / r;
// estimation of time of data delivery
eDeliveryTimestamp = (now + t_ipi + RTT/2 + delta)
// estimation of presentation delay eDelay =
eDeliveryTimestamp - currentPacket.timestamp

// quality adaptation action in response to the
// estimated delay
if (eDelay <= MinDelayThreshold) // i.e. 50 ms
    action=increase
else {
      //i.e. 400 ms action = decrease
      if (eDelay >= MaxDelayThreshold)
      // quality adaption decrease action
      if (action == decrease)
          if (currentLayer == MIN_LAYER && eDelay \
                                < MinDELAY)
              drop layer(currentLayer) STOP
          else
              if (currentLayer > MIN_LAYER)
              currentLayer = currentLayer - 1;
              currentPacket.discard = true
      // quality adaption increase action
      if (action == increase && currentLayer < \
                                MAX_LAYER)
          currentLayer = currentLayer + 1;
} //scheduling of current packet
transmission if not currentPacket.discard
scheduleTransmission
currentPacket,t_ipi
}
```

Now is the current time, RTT is the round-trip time and delta is a tolerance constant including error in time estimations. In order to avoid abrupt changes in the QoS provided to the final user, quality layer increase and decrease actions have been proposed to be tailored by the `MinDelayThreshold` and `MaxDelayThreshold` obtained from the QoS requirements.

5.4.2.2 QoS-Aware Error Control Mechanism

As previously explained, some multimedia applications present some preference for timeliness over order and reliability. Actually, many of these applications do not require a fully ordered and fully reliable transport service when the delay incurred by this service is not compatible with their time constraints. For this reason, most multimedia applications have been designed to use the UDP protocol without any guarantees of order and reliability. In some cases, these applications have to implement ad-hoc error control mechanisms to satisfy their requirements. In this section we present an error control mechanism based on the partial ordering and partial reliability constraints of multimedia flows aimed at improving the QoS delivered to the multimedia applications.

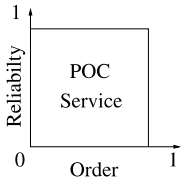

Fig. 5.1. Space of solution of the POC service

Transport protocols offering a partially ordered and/or a partially reliable (PO/PR) service have been proposed in several research works. The Partial Order Connection protocol (POC) was one of the first works proposing a partially ordered transport service for multimedia applications [124]. Unlike classic transport protocols that deliver objects either in the exact order transmitted or according to no particular order, POC provides a partial order service where some, but not all, objects have to be received in the order transmitted. This protocol has been designed to provide a partially reliable service, which accepts a subset of transmitted objects to be lost. A POC service specification can be defined by a subspace inside the whole space of partial ordered and reliable services, which can be realised for the delivery of a given set of Application Data Units (see Fig. 5.1).

In [125] the use of this family of protocols in the transport of video streams was studied. It was demonstrated that POC connections do not only fill the conceptual gap between TCP and UDP, but also provide real performance improvements for the transport of multimedia streams such as MPEG video. Other approaches, based on the POC concept and proposing the use of QoS requirements in order to offer a PO/PR services, have been presented. For instance, the partial reliability extension to SCTP (PR-SCTP) introduces the concept of timed reliability to implicitly regulate the error-control mechanism. However, a solution integrating the time, order, reliability and synchronisation constraints has not yet been proposed.

This section presents some error-control mechanisms aimed at providing PO/PR services. These mechanisms could be explicitly configured by specific order and reliability QoS requirements or implicitly deduced from the application requirements (i.e., delay, jitter, synchronisation, etc.). In both cases, specific QoS information describing the ADU characteristics has to be provided by applications in order to ensure that adequate order and reliability policies are applied by the error control mechanism.

5.4.2.2.1 Partially Ordered Service (PO)

A partially ordered service can be provided by the error-control mechanism at two levels:

- Medium or intraflow level: specific QoS information could be used by the transport protocol delivery mechanism in order to offer a partially ordered delivery service for the ADUs composing a flow. For instance, packets comprising the JPEG2000 or MJPEG2000 flows could be delivered by a partially ordered ser-

vice deduced from the ADU description (i.e., resolution layers, region of interest, etc.).

- Multimedia or interflow: ETP can use the interflow synchronisation constraints to schedule the transmission and delivery of data packets for every flow, permitting a certain degree of out-of-order delivery. For instance, a session composed of two audio and video flows may be delivered with a partially ordered service between the audio and video ADUs, but respecting the interflow synchronisation constraints (e.g., 80 ms for lip synchronisation).

Transport mechanisms oriented to provide a PO service could be controlled by the source or receiver transport entities. Source control could be implemented following sending rate constraints imposed by congestion-control mechanisms in order to optimise the ADU transmission (i.e., sending first ADU with higher priorities). A partially ordered service could be scheduled taking into account intrinsic characteristics of ADUs (i.e., time stamp, dependency, etc.). A receiver controlled mechanism could optimise the delivery of data packets to the receivers when a partial order is permitted (i.e., ADUs corresponding to independent segments or tiles of JPEG2000 images). Furthermore, both PO source-based and receiver-based mechanisms could be implemented for unicast or multicast protocols.

5.4.2.2.2 Partially Reliable Service (PR)

A partially reliable service can be implemented based on the explicit reliability requirements of the application, or implicitly from intrinsic time constraints. FEC and ARQ mechanisms may be enhanced in order to provide this specialised error control mechanism. FEC mechanisms have been described as being very dependent on the traffic-loss characteristics. Furthermore, these schemes are usually designed to offer a full ordered and full reliable service when implemented at the transport layer. Partial order and partial reliability services using FEC have not been widely studied.

5.4.2.2.3 Partially Reliable, Differentiated and Time-Constrained ARQ (D-PR & TD-PR)

ARQ error-control mechanisms work as follows: when a loss is detected, the receiver sends a feedback message to ask the source to retransmit the message. This means that a retransmitted packet arrives at least three one-way delays after the transmission of the original packet. Sometimes, this delay could exceed the delay constraints of the application. However, if the one-way delay is short, this mechanism could be efficiently used to recover the losses. Time-constrained ARQ mechanisms could be implemented for unicast connections by receiver or source-based methods:

- Receiver-based: The objective of this mechanism is to avoid the request of retransmission demands, which would not arrive on time for their presentation. When a loss is detected before demanding its retransmission, the following condition has to be checked:

```
if (now + RTT < presentationTime)
    ask for retransmission
```

Now is the current time and RTT is the round-trip time. The problem with this method is that the receiver has to know the scheduled presentation time of lost packets.

- Source-based: This error-control method is intended to avoid retransmissions of packets that will arrive too late to be presented. The retransmissions can be demanded by the receiver when losses are detected. The source will check the following condition before performing the retransmission:

```
if (now + RTT/2 < presentationTime)
    retransmission of packet
```

This mechanism can be easily implemented by the source if QoS information related to the time presentation is available at the transport layer. Indeed this method can be used to provide a differentiated and partially reliable service, taking into account the notion of differentiated layers previously introduced.

5.5 Conclusions

The mechanisms discussed in this chapter include congestion-control mechanisms that are intended to preserve the resources of networks by providing a Best-Effort service. An error-control mechanism was presented that is intended to provide a partially ordered (PO) and partially reliable (PR) service, explicitly or implicitly configured from the application requirements. This error-control mechanism has also been enhanced in order to provide a differentiated and partially reliable service (D-PR). Both error- and congestion-control mechanisms have been enhanced in order to take into account intrinsic application time constraints (TD-PR & TD-TFRC). The composition of error- and congestion-control mechanisms to provide a large set of transport services has been discussed.

As it will be seen in the next chapter, the full design and implementation of the corresponding multi-QoS transport architecture and protocols was done within the framework of the EuQoS project.