4

Internet Protocols

The predecessor of today's Internet was the ARPANET (Advance Research Project Agency Network). In autumn 1969, the first computer was connected to a node at the University of California and by the end of the year, the ARPANET consisted of four connected computers with different operating systems. The ARPANET grew continuously and soon the TCP/IP suite was adopted as the official protocol suite. TCP/IP was used by other networks to link to ARPANET since 1977, which led to a rapid growth.

In 1989, the World Wide Web (WWW) was invented. Mosaic, the first graphical Web browser was released in 1993 and the first search engine – 'Yahoo' (Yet Another Hierarchical Officious Oracle) – went online in 1994. The WWW led to the Internet becoming 'attractive' for ordinary people, which led to an even higher growth rate. Today, more than 1 billion people are connected to the Internet and this number is still rapidly growing.

The Federal Networking Council passed a resolution defining the term Internet in 1995 (see Federal Networking Council (FNC)):

Internet refers to the global information system that

1. is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons;
2. is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and
3. provides, uses or makes accessible, either publicly or privately, high-level services layered on the communications and related infrastructure described herein.

In this chapter, we look at the TCP/IP suite. First, we discuss the Internet protocol stack and the layer model. After that we discuss the most important Internet protocols.

4.1 The Internet Protocol Stack

Throughout this book when protocol stack layers are mentioned, the five layer reference model of the Internet is used; it is shown in Figure 4.1. It is described for example by Tanenbaum (2002). It is a hybrid of the OSI reference model of Zimmermann (1980) and

The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design
Oliver Heckmann © 2006 John Wiley & Sons, Ltd. ISBN: 0-470-01293-5

5	Application layer
4	Transport layer
3	Network layer
2	Data link layer
1	Physical layer

Figure 4.1 Hybrid 5 Layer Reference Model of the Internet

the TCP/IP reference model of Clark (1988); Leiner *et al*. (1985). Each layer consists of a set of *protocols* for communication with another entity on the same layer and of *communication services* that are offered to the next higher layer. The layers can be distinguished as follows:

- The *physical layer* defines the mechanical, electrical and timing interfaces of the network.
- The *data link layer's* main task is to transform the raw layer 1 transmission facility into a line free of undetected transmission errors between two *directly connected systems*, typically by using the concept of data frames.
- The *network layer* is concerned with forwarding and routing of packets from sender to receiver *end systems*. The basic network layer protocol of the Internet is IP (Internet Protocol); it offers a connection-less datagram forwarding service.
- The *transport layer* uses the network layer to provide sender to receiver *application* communication. The most important transport layer protocols of the Internet are the connection-oriented virtual error-free TCP (Transmission Control Protocol) and the connection-less UDP (User Datagram Protocol).
- The *application layer* contains high-level protocols such as, for example, HTTP. It handles issues like network transparency, resource allocation and problem partitioning for an application. The application layer is not the application itself; it is a service layer that provides high-level services.

We now shortly address the basic network and transport layer protocols of the Internet (IP, TCP, UDP) and then the lower layer protocols that are relevant for ISPs. Some selected application layer protocols are discussed in Chapter 5 in the context of the applications that use them and the properties of the traffic they generate.

4.1.1 IP

4.1.1.1 IPv4

IP is the glue that holds the Internet together. It is a connection-less unreliable layer 3 protocol. 'Unreliable' in this context means that there are no guarantees that an IP datagram will be successfully delivered. IP is the least common denominator on top of the different layer 2 technologies that are used as infrastructure in the different autonomous systems that form the Internet. This explains why it is a relatively simple protocol. In fact, a basic design principle of the Internet is the *end-to-end principle* that is described by Saltzer *et al*. (1984). The authors argue that processing by intermediate systems can be made simpler, relying on the end-system processing to make the system work. This leads

to the model of the Internet as a 'dumb network' that has smart end systems (terminals), a completely different model to the previous paradigm of the smart network with dumb terminals like the traditional telephony networks.

The transport layer takes data streams and breaks them up into datagrams that are transported with the IP to the target end system. Packets (datagrams) are transmitted independently from each other. IP datagrams can be as large as 64 kB, but in reality they are typically limited by a MTU (Maximum Transmission Unit) of 1500 byte[1]. The MTU determines the largest possible IP datagram size that can be transmitted by an IP router without it needing to be fragmented.

The IP version 4 (IPv4) header is depicted in Figure 4.2. It has a size of 20 bytes if no options are used. IPv4 options should be avoided because IP packets with options are often processed on the slow path of an IP router and this can lead to additional delay and performance problems. The *source* and *destination address* fields contain the sender's and receiver's IP address. Information about the sender/receiver port is transport layer information and is therefore not contained in the IP header but in the TCP/UDP header instead.

In practice, the *time to live* field is used as a hop counter used to limit packet lifetimes. It is decreased at each intermediate router, and the packet is discarded if it reaches zero. The default start value for the time to live is 64 according to RFC 1700. Some operating systems, for example, older versions of Microsoft Windows, use a value of 32, which is widely considered too small today.

The *protocol* field contains information about the transport layer protocol that generated the packet. For UDP, the field is 17 and for TCP, it is 6; see RFC 1700 for other protocols. The *header checksum* verifies the IP header only. It has to be recomputed at each hop because of the changing *time to live* field.

In Diffserv networks, the *type of service* field is redefined as the *Diffserv byte*, see Section 6.2.4. For best-effort networks, the *Type of Service (ToS)* field was intended to be used for selecting different types of service. However, as there was, for example, no way

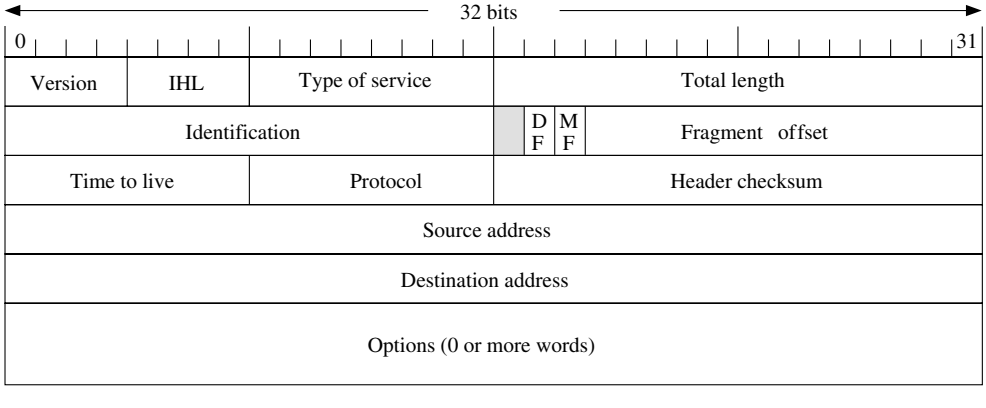

Figure 4.2 IPv4 Header

[1] This value comes from the maximum frame size of most Ethernet links on layer 2.

to stop end systems from always requesting the best possible service for all its packets, the *type of service field* is typically ignored by routers.

IP routers forward IP datagrams towards their destination. We discuss the different forwarding architectures in Section 6.3.

4.1.1.2 IPv6

IP version 6 (IPv6), also sometimes called IPng (IP next generation), is based on recommendations in RFC 1752. The core set of IPv6 protocols form an IETF Draft Standard since 1998. The protocol is described in Deering and Hinden (1998). Many other RFCs describe further details of IPv6 architectures and the transition from IPv4 to IPv6.

The main motivation behind the development of IPv6 was the predicted shortage of IPv4 addresses in the near future. IPv6 is therefore designed to 'never' run out of addresses. IPv6 addresses are 16-byte addresses which increase the address space drastically compared to the 4-byte IPv4 addresses. Multicasting is improved by a *scope* field as part of the multicast addresses and a new address type called *anycast* is introduced. Anycast addresses are group addresses where only one member of the group responds, for example, the member of the group that is closest to the source. Anycast addresses are potentially very interesting because the closest router or, for example, the closest name or time server can be accessed with that concept. For more details about the IPv6 addresses see Hinden and Deering (1998).

Besides this, IPv6 contains other improvements over IPv4. The most important ones are as follows:

- Simplification of the IP header. By comparing the IPv4 header in Figure 4.2 with the IPv6 header in Figure 4.3, one immediately recognises the streamlined header layout. Many IPv4 fields are dropped or made optional. This allows routers to process packets faster and can thus improve throughput.
- IPv6 header options are encoded differently compared to those of IPv4. This results in less stringent limits on the length of options and greater flexibility in introducing new options in the future.
- The IPv6 header contains a 20-bit *flow label* that can be used for marking packets belonging to certain flows to give them preferential treatment. The 8-bit *traffic class* field can be used as Diffserv byte similar to the *type of service* field in IPv4.
- Authentication, data integrity and data confidentiality are other important features of IPv6.

IPv6 maintains the good features of IPv4 and discards some of the bad ones. Owing to the increasing number of IP addresses needed in rapidly developing countries like China, internetworking with IPv6 networks will drastically increase in importance in the next years.

For more details on IPv6 see Loshin (2004) and Hinden and Deering (1998).

4.1.2 UDP

UDP (User Datagram Protocol) is the Internet's connection-less transport layer protocol. It is specified in RFC 768 (see Postel (1980)). UDP is a minimalistic protocol, its 8-byte

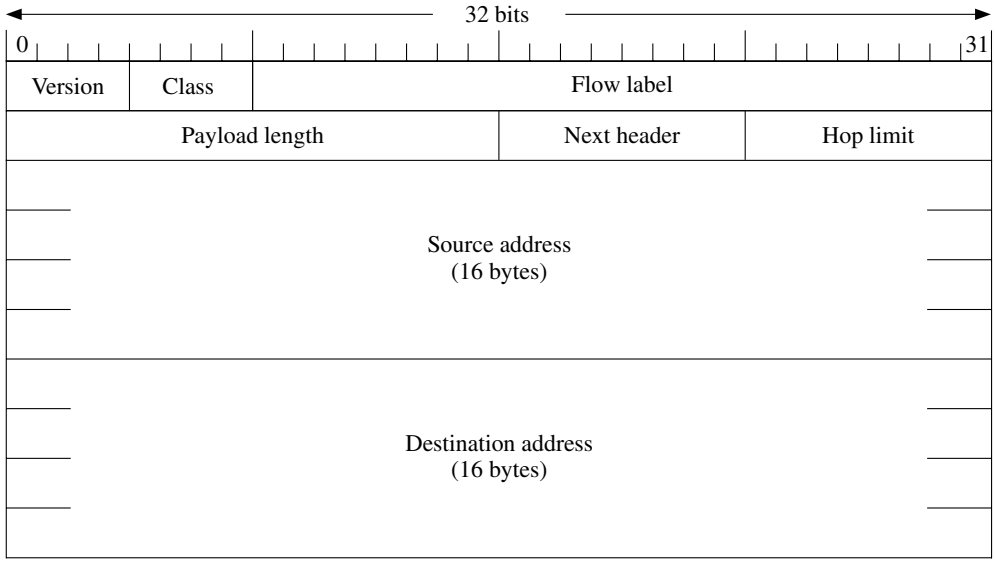

Figure 4.3 IPv6 Header

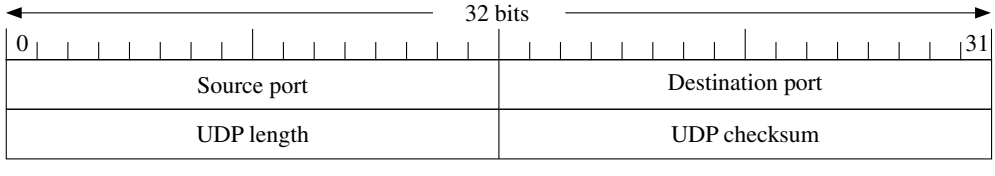

Figure 4.4 UDP Header

header is depicted in Figure 4.4. *Source* and *destination ports* identify the applications on the end systems (the source and destination IP addresses are in the IP and not the in UDP header). Calculating the *UDP checksum* is optional.

UDP does not support flow control or the reliable or even-ordered delivery of datagrams. UDP is mostly used for multimedia application protocols such as VoIP or video streaming. At the time of writing, a promising alternative to UDP for these applications is currently under development in the IETF. The Datagram Congestion Control Protocol (DCCP) (DCCP, see Kohler *et al.* (2005)) is a message-oriented transport layer protocol like UDP but has congestion control built in, like TCP, but without the TCP's in-order and retransmission features.

4.1.3 TCP

4.1.3.1 Introduction

TCP (Transmission Control Protocol) was designed to transmit a byte stream reliably using the unreliable IP datagram service. TCP is the most commonly used transport protocol today. It is best suited for application protocols such as SMTP, FTP or the HTTP that need

a reliable connection-oriented service. It is less suited for real-time streaming applications that do not need the retransmission of lost packets and that prefer to have more influence on the transmission rate. The main features of TCP are the following:

- TCP is a connection-oriented protocol. A TCP connection is a byte stream, not a message stream. This means that the TCP stack and not the application splits the byte stream into packets (called *TCP segments*) that are transmitted through the network.[2]
- TCP connections are full duplex (traffic can go in both directions) and point to point (no multicast or broadcast).
- TCP takes care of the reliable in-sequence delivery of the TCP segments. Lost packets are retransmitted and out-of-sequence segments are reordered at the end system.
- TCP's window-based flow control mechanisms allow a slow receiver to slow down a fast sending sender.
- TCP has congestion control mechanism that tries to detect congestion in the network and adapt the window size accordingly.

TCP was first formally described in RFC 793 and then clarified, extended and changed in several other RFCs; see Table 4.1 for a selection of important TCP-related RFCs.

The 20-byte TCP header is depicted in Figure 4.5. Besides the *ports,* one notices the *sequence number* of the transmitted data (measured in bytes) and with the *acknowledgement number* the next expected byte in the opposite directions. The *SYN* bit is used for the three-way handshake at connection setup and the *FIN* bit for closing a connection. The *window size* contains the receiver window that tells the opposite end system how many bytes it may maximally send starting with the byte indicated by the

Table 4.1 Selected RFCs Related to TCP

Name	Title
RFC 793	Transmission Control Protocol
RFC 1122	Requirements for Internet hosts – communication layers (contain TCP clarifications and bug fixes)
RFC 3782	The NewReno modification to TCP's fast recovery algorithm
RFC 2018	TCP Selective Acknowledgement (SACK) Options
RFC 2883	An extension to the Selective Acknowledgement (SACK) Option for TCP
RFC 2581	TCP congestion control
RFC 2988	Computing TCP's retransmission timer
RFC 3042	Enhancing TCP's loss recovery using limited transmit
RFC 3390	Increasing TCP's initial window
RFC 2861	TCP congestion window validation
RFC 3168	The addition of Explicit Congestion Notification (ECN) to IP (this also influences TCP)
RFC 1323	TCP extensions for high performance
BCP 28	Enhancing TCP over satellite channels using standard mechanisms

[2] There are, however, means for an application programmer to influence how the byte stream is split into segments.

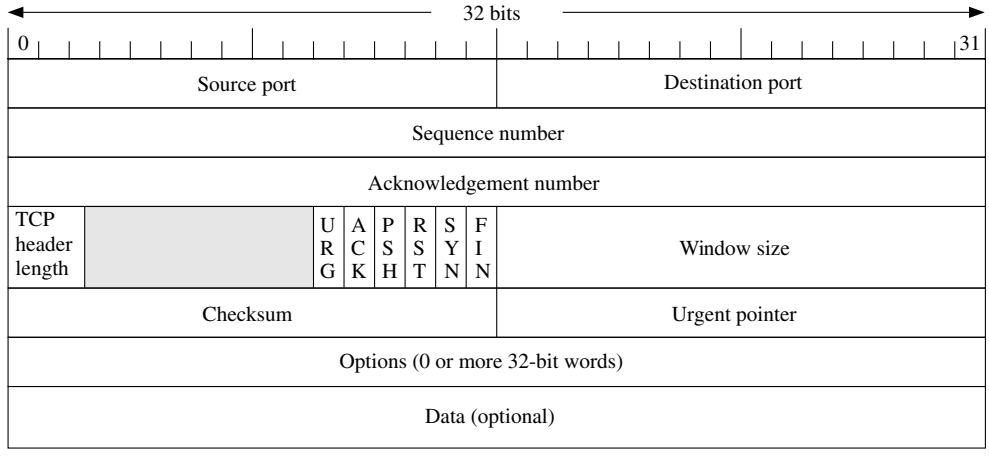

Figure 4.5 TCP Header

acknowledgement number. The window size is used for flow control. The 16-bit field for the window size is too small for high-bandwidth high-latency connections. Therefore, in RFC 1323 a window scale option was proposed, which is now widely supported by different TCP implementations. It allows shifting the window size field by up to 14 bits to the left, thus allowing windows of up to 2^{30} bytes. The original congestion control algorithm is also problematic for high-speed connections; there are several modifications for this under discussion, see for example, High-Speed TCP (HS-TCP) (see Floyd (2003)) or FAST TCP (see Jin *et al.* (2005)).

4.1.3.2 Flow and Congestion Control

TCP interprets packet loss as an indication for congestion in the network and reacts by decreasing its window size and, therefore, the number of packets it can have in the network at one point in time[3]. A TCP sender keeps track of two windows for sending data. The advertised window of the receiver (for flow control) and the congestion window (for congestion control). The maximum amount of data that can be sent unacknowledged at one point in time is given by the minimum of the receiver window and the congestion window.

TCP is a self-clocked algorithm. This means that the window size is adapted in intervals proportional to the round-trip time. TCP starts in a phase called *slow start*. The initial value of the congestion window $cwnd$ is one maximum segment size (MSS). Each time an ACK is received while in slow start, the congestion window is increased by one segment size. Therefore, if the sender receives its full window's worth of ACKs per RTT, $cwnd$ is doubled per RTT.

Another variable, the slow-start threshold $ssthresh$, is used at the sender to keep track of when to end the slow-start phase and enter the congestion avoidance phase. The names are misleading as the slow-start phase is actually the phase in which $cwnd$ is increased

[3] This behaviour creates problems for wireless networks, where a packet drop is not necessarily a sign of congestion.

faster. The default value of *ssthresh* is 64 kB (which corresponds to 42–44 segments). Short-lived connections that only transmit little information might therefore never leave the slow-start phase.

When *cwnd* is less than or equal to *ssthresh*, TCP is in slow start. Otherwise, it enters congestion avoidance and *cwnd* is modified the following way: Each time an ACK is received, *cwnd* is increased by $\text{MSS} \cdot \frac{\text{MSS}}{cwnd}$. This means that if a full window's worth of ACKs are received per RTT, *cwnd* is increased linearly by one MSS and not exponentially as in slow start.

There are two indications of congestion: a retransmission timeout[4] occurring and the receipt of three duplicate ACKs in a row. TCP can generate an immediate acknowledgement (a duplicate ACK) when an out-of-order segment is received as that is a sign that the previous segment could be lost. When three duplicate ACKs are received in a row, TCP performs an immediate retransmission of the missing segment without waiting for the retransmission timer to expire. This mechanism is called *fast retransmit*.

When congestion occurs, *ssthresh* is set to one-half of the current window size[5]. Additionally, if the congestion is indicated by a timeout, *cwnd* is set to one MSS and slow start is re-entered. If the congestion is indicated by duplicate ACKs, no slow start is performed. This is called *fast recovery*. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP that there is only moderate congestion, as one packet got lost but a later packet (and the ACK) got through.

To summarise, slow start continues until the window is halfway to where it was when congestion occurred. The behaviour of TCP is visualised in Figure 4.6. When analysing long-lived TCP connections with losses of 5% and lower, the slow-start phase is often

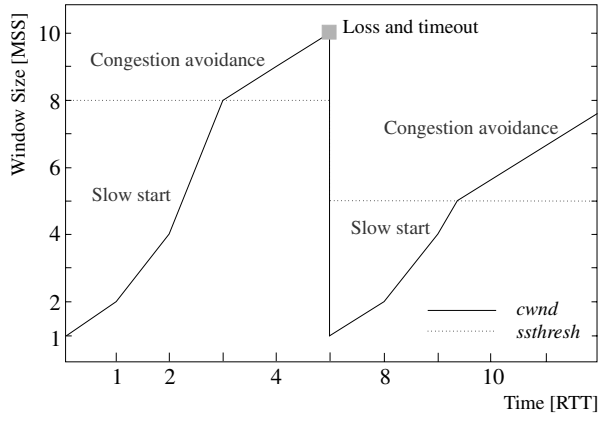

Figure 4.6 TCP Example

[4] The retransmission timeout is a function of the estimated round-trip time RTT. It is measured with Jacobson's and Karn's algorithm (Jacobson (1988); Karn and Partridge (1991)). In practice, the retransmission timeout is much greater than the actual RTT. Retransmissions based on the timer may therefore be slow. This is the reason for the duplicate ACKs as second congestion indicators.

[5] Precisely, that is, the minimum of *cwnd* and the receiver's advertised window, but at least two segments.

abstracted from and the TCP window assumed to move in a sawtooth behaviour (see Figure 4.7). Therefore, TCP congestion control is also called *additive increase/multiplicative decrease* (AIMD).

For an excellent source of more information on TCP, we recommend Stevens (1994).

4.1.3.3 TCP Flavours

There are different 'official' TCP versions plus countless slightly different implementations of TCP stacks in the different operation systems. The original TCP version is specified in RFC 1122. TCP *Tahoe* was developed later and distributed with the 4.3 BSD Unix in 1988. Its main advantage is that it includes the fast *retransmit* mechanism that was discussed above. In the 1990 BSD Unix, TCP *Reno* was developed and implemented. It includes all mechanisms of Tahoe plus the fast *recovery* mechanism explained above. TCP Reno has problems with multiple losses, because it does not see duplicate ACKs and times out. This led to the development of TCP NewReno in RFC 3782. It contains an improved fast recovery algorithm that deals better with multiple losses.

A basic problem of the all these TCP flavours is that they use cumulative ACK: one ACK acknowledges the correct reception of the indicated byte plus all previous bytes. If in a sequence of segments (1, 2, 3, 4) segment 2 gets lost but the other segments arrive, only segment 1 but not segments 3 and 4 can be acknowledged with cumulative ACKs. In the *SACK* TCP flavour of RFC 2018, selective acknowledgements are supported that allow acknowledging out-of-sequence segments. In combination with a selective retransmission policy, this can lead to considerable performance improvements.

A radically different TCP flavour is TCP *Vegas*. Vegas tries to use packet delay rather than packet loss as a congestion indicator. It uses the difference between the expected and the actual flow rate to estimate the available bandwidth in the network. When the network is not congested, the expected and the actual rates will be very similar. If the network is congested, however, the actually achieved rate will be significantly smaller than the expected rate. The difference between the rates can be expressed with the difference between the window size and the actual acknowledged packets. The sending rate in Vegas is adapted according to this measured difference. For details, we refer to Brakmo and Peterson (1995). A problem of TCP Vegas is that when competing with other TCP versions, it does receive less than a fair share of the bandwidth.

4.1.3.4 TCP Rate Estimation

For many practical purposes, it is important to get a feeling for the rate or the throughput of a single TCP connection. The TCP rate depends on several parameters. The most important ones are the round-trip time and the loss probability. We next discuss and present three different approaches to estimating the throughput or the rate of a TCP connection. We start with the famous square root formula, then give a better approximation formula and finally end with discussing a formula for short-lived flows.

The Square Root TCP Rate Formula The so-called square root TCP rate formula can be easily derived, see Floyd (1991) and Lakshman and Madhow (1997). Assuming a long-lived TCP connection that is dominated by TCP's congestion avoidance phase, the

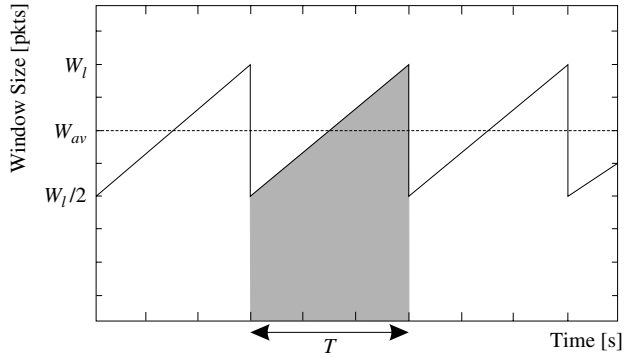

Figure 4.7 Square Root TCP Rate Formula

connection setup and the slow-start algorithm can be neglected. Assuming that losses occur with probability p in regular intervals every $1/p$ packets when the congestion window size has reached W_l, the window size over time will show the sawtooth behaviour depicted in Figure 4.7. If one ACK per packet is sent, the window size increases linearly by one per round-trip time RTT. Therefore, $T = \frac{W_l}{2} RTT$. The average window size W_{av} is $W_{av} = \frac{3}{4} \cdot W_l$ as follows from Figure 4.7. The average rate r_{av} in packets/second is therefore $r_{av} = W_{av}/RTT = \frac{3 \cdot W_l}{4 \cdot RTT}$. The area marked in Figure 4.7 equals the number of packets N sent in one cycle T. Therefore,

$$N = r_{av} T = 1/p \tag{4.1}$$

From this it follows that $\frac{3}{8} W_l^2 = 1/p$ and $W_l = \sqrt{\frac{8}{3p}}$. The average TCP rate is therefore $r_{av} = \frac{1}{RTT\sqrt{\frac{2}{3}p}}$ measured in packets/second or if expressed as r'_{av} in bytes/second $r'_{av} = \frac{MTU}{RTT\sqrt{\frac{2}{3}p}}$ with the packet size MTU. The MTU of TCP connections is typically 1500 bytes with and 1460 bytes without TCP/IP headers.

Measurements of packet traces have shown that the introduction of a proportional factor of 1.22 leads to a better approximation. This results in the well-known square root TCP formula

$$r = 1.22 \frac{MTU}{RTT\sqrt{\frac{2}{3}p}} \tag{4.2}$$

Measurements have also shown that the assumptions in this formula are not valid for loss rates of $p = 5\%$ and higher.

A Better Approximation The square root formula (4.2) is a simple model of the congestion avoidance phase of a long-lived TCP connection. Padhye *et al.* (1998) derive a better steady-state model for the TCP rate which takes timeouts as well as duplicate ACKs as loss indication into account. A good approximation of their model is given with Model 4.1. It is based on the Reno flavour of TCP. The rate of a long-lived TCP connection can be limited by congestion avoidance algorithm and also by the maximal

Model 4.1 Advanced TCP Rate Approximation Model

Parameters

W_{max}	Maximum receiver window size [pkts]
b	Number of packets acknowledged by a received ACK (typically $b = 2$)
p	Loss probability
T_0	Retransmission timeout [s] (initially $T_0 = 3s$, adapted according to RFC 793)
RTT	Round-trip time [s]
r_{av}	TCP rate [pkts/s]

Equation

$$r_{av} = \min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1,\ 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}\right) \quad (4.3)$$

congestion window size as advertised by the receiver. This maximal window size W_m is given by the buffer limit set aside for the connection on the receiver side. It is taken into account in the first part of the main *min* term in (4.3). The assumptions made for the derivation of the formula are that the effect of the fast recovery algorithm can be neglected and that the time spent in the slow-start phase is negligible. The latter holds true only for long-lived TCP connections. Among other assumptions, it is assumed that the round-trip time *RTT* is not affected by the window size. This assumption is acceptable for most connections except when the connections go through an extreme bottleneck such as a low-bandwidth modem line with a large buffer. The rate r_{av} is specified in packets/second. To obtain the rate in bytes/second, r_{av} has to be multiplied with the average packet size, which is typically around 1500 bytes.

For a derivation of the formula and more details, see Padhye *et al.* (1998).

Short-lived TCP Flows The aforementioned models predict the rate of long-lived TCP connections. Many HTTP transfers of web pages, however, are short-lived TCP connections that spend little or no time in the congestion avoidance phase of TCP. For these connections, the slow-start phase has to be taken into account. A detailed model for short-lived TCP flows is presented in Cardwell *et al.* (2000). Here we present a simplified version of that model assuming that no losses occur until the transfer is finished. The results are summarised in Model 4.2.

If only a few packets are transmitted, the initial handshake for TCP connection setup cannot be neglected. The total duration D of the short-lived TCP transfer consists of the

Model 4.2 TCP Latency Model

Parameters

D	Total duration of the TCP transfer [s]
r_{av}	Average TCP rate [pkts/s]
L	Duration of the TCP connection establishment [s]
T	Duration of the data transfer itself [s]
d	Number of data segments to be transferred [pkts]
RTT	Round-trip time [s]
W	Unconstrained window size at end of transfer [pkts]
γ	Slow-start growth rate ($\gamma = 1.5$ if one ACK per two data segments is sent)
w_1	Initial congestion window size [pkts] (typically $w_1 = 2$)
W_{max}	Maximum receiver window size [pkts]

Equations

$$D = L + T \tag{4.4}$$

$$L = RTT \tag{4.5}$$

$$W = \frac{d(\gamma - 1)}{\gamma} + \frac{w_1}{\gamma} \tag{4.6}$$

$$T = \begin{cases} RTT \cdot \left(\log_\gamma \left(\frac{W_{max}}{w_1} \right) + 1 + \right. \\ \left. \frac{1}{W_{max}} \left(d - \frac{\gamma W_{max} - w_1}{\gamma - 1} \right) \right) & \text{when } W > W_{max} \\ RTT \cdot \left(\log_\gamma \left(\frac{d(\gamma-1)}{w_1} \right) + 1 \right) & \text{otherwise} \end{cases} \tag{4.7}$$

$$r_{av} = \frac{d}{D} \tag{4.8}$$

connection setup time L and the time T for the data transfer itself. The connection setup consists of a three-way handshake; (4.5) shows the expected duration of the handshake, taking into account that the last ACK of the handshake already carries data (and is therefore part of T).

It is assumed that in total d segments are to be transmitted (d is approximately the number of bytes to be transmitted divided by the MSS which again is 1460 bytes in most cases). During slow start, there are two possibilities: If the maximum congestion window W_{max} is very large, the window size will be W at the end of the transfer. Duration T is then given by the second part of (4.7). Otherwise, the maximum congestion window W_{max} is reached during the transfer and T is expressed by the first part of (4.7).

4.1.3.5 TCP Root Cause Analysis

While the previous section described the theoretical throughput of a TCP connection as a function of the network parameter's loss and delay, TCP root cause analysis is concerned with determining reasons that limit the throughput of an actually measured TCP connection. This is important for end users as well as INSPs, as the network is not necessarily always the limiting factor for TCP connections. The possible rate-limiting factors as described in Zhang *et al*. (2002) are as follows:

- Opportunity (lifetime): Many short-lived flows do not transmit long enough to reach a high throughput.
- Bandwidth and Congestion: Packet loss limits the TCP throughput owing to TCP's congestion control as discussed above. Packet loss can be caused by the TCP rate approaching the total bottleneck bandwidth (e.g. the upload bandwidth of a ADSL connection) or by other flows competing for bandwidth in the network.
- Transport: The sender might be in congestion avoidance without experiencing loss.
- Receiver window: The sending rate can be limited by the advertised receiver window. This is called *flow control* and is used by slow receivers to throttle fast senders.
- Sender window: The sender window is constrained by the buffer space at the sender. It limits the amount of unacknowledged data outstanding at any time.
- Application: The rate with which an application produces data can of course also be an important limit of the actual throughput.

Zhang *et al*. (2002) introduce the tool T-RAT. On the basis of the backbone packet level traces and summary flow level statistics, they conclude that the dominant rate-limiting factors are congestion and receiver window limits. Siekkinen *et al*. (2005) discuss some drawbacks of the tool T-RAT and provide some new root cause analysis algorithms based on certain time series, for example, of the interarrival time of acknowledgements or of the number of unacknowledged bytes. These time series can be extracted from bidirectional packet header traces.

4.1.4 Lower Layer Protocols

INSPs connect their POPs (points-of-presence) typically by leasing lines from carriers. There are several layer 2 technologies available for carriers that can be employed by INSPs to run their IP overlay network over. In this section, we focus on the high-speed layer 2 technologies currently favoured by carriers.

4.1.4.1 Synchronous Optical Networking (SONET) / Synchronous Digital Hierarchy (SDH)

SONET (Synchronous Optical Networking) and SDH have replaced the Plesiochronous Digital Hierarchy (PDH) and are the most common link-layer technologies for today's high-speed wide area networks (WAN). SONET is a standard for optical communication,

providing framing, as well as a rate hierarchy and optical parameters for interfaces ranging from 51 Mbps (OC-1) up to 9.8 Gbps (OC-192) and higher[6]; see ANSI T1.105 (1995); ANSI T1.119 (1995). SONET has been adopted as a standard for North America by the American National Standards Institute (ANSI), while a slightly advanced version – SDH, see ITU Recommendation G.707 (1996) – has been adopted by the International Telecommunication Union/Telecommunication Standardisation Sector (ITU-T) and is used in the other parts of the world.

SONET/SDH use time division multiplexing with a scan time interval of 125 μs, indicating the background of SONET/SDH, the telecommunication market where the standard sampling rate is 8000 samples/second for voice. SONET/SDH needs a tightly synchronised clocking environment for the synchronous transmission of the data streams.

Table 4.2 lists the line and payload rates of the optical SONET/SDH circuit hierarchy. The overhead carries information that provides 'Operations, Administration, Maintenance and Provisioning' capabilities such as framing, multiplexing, status, trace and performance monitoring. Higher-speed circuits are formed by successively time multiplexing multiples of slower circuits; for example, four OC-3 circuits can be aggregated to form a single OC-12 circuit.

For the transport of IP packets over SONET/SDH, the IP datagrams are typically encapsulated into Point-to-Point Protocol (PPP) packets. PPP provides link error control and initialisation. The PPP-encapsulated datagrams are then framed using high-level data link control (HDLC) and sent over a SONET/SDH circuit to the next hop. RFC 2615 describes this process, see Malis and Simpson (1999).

Simplified Data Link (SDL) is a very low overhead alternative to the HDLC-like encapsulation that avoids HDLC's byte-stuffing expansion and is designed for rates at OC-192 and above. It is described in RFC 2823, see Carlson *et al*. (2000).

4.1.4.2 10-Gigabit Ethernet (10GE)

Beginning with the 1-Gigabit Ethernet standard IEEE 802.3z, Ethernet is deployed not only in local area networks (LAN), the traditional domain of Ethernet, but also in metropolitan area networks (MAN). With the 10-Gigabit Ethernet (10GE) standard IEEE

Table 4.2 SONET/SDH Data Rates

SONET	SDH	Line Rate	Payload Rate	Overhead Rate
OC-1	–	51.840 Mbps	50.112	1.728
OC-3	STM-1	155.520 Mbps	150.336	5.184
OC-12	STM-4	622.080 Mbps	601.344	20.736
OC-48	STM-16	2488.320 Mbps	2405.376	82.944
OC-192	STM-64	9953.280 Mbps	9621.504	331.776
OC-768	STM-256	39813.120 Mbps	38486.016	1327.104

[6] SONET is also specified for non-optical digital circuits but because high data rates usually require fibre optic cable, we concentrate on OC (optical carrier) circuits here.

802.3ae, Ethernet can now be expanded also into wide area networks (WAN), the traditional domain of SONET/SDH. 10GE can work over SONET links and also without SONET as end-to-end Ethernet.

10GE is based entirely on the use of optical fibre and only full-duplex mode is supported. Two end systems can be connected directly; for more end systems, a switch has to be used. 10GE still shares the MAC (Media Access Control) protocol and the frame format with the slower Ethernet standards. But because there are only point-to-point connections rather than the multipoint connections that were used in the classic Ethernet networks (IEEE standard 802.3 before 802.3ae), the classic Ethernet collision detection mechanism CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is no longer necessary.

10GE is, in contrast to SONET/SDH, an asynchronous protocol and differently clocked domains are interlinked by switches and bridges that buffer and re-synchronise the data. Therefore, 10GE requires less complexity and is generally cheaper than the SONET/SDH equipment. This was in fact one of the design goals of 10GE; see IEEE 802.3 High Speed Study Group (2002). Another cost-saving factor is that 10GE is capable of using lower-cost uncooled optics and multimode fibre for short-distance connections.

Over single-mode fibre, 10GE can bridge distances of 40 km and can therefore be used to build a pure Ethernet WAN. For compatibility with the existing SONET/SDH network, 10GE can also be operated on top of SONET OC-192/SDH STM-64 connections that only have a slightly slower transmission rate (see Table 4.2). This operation is described by the 10 GE WAN PHY (physical layer) specification. 1-Gigabit Ethernet (1GE) is already quite commonly found as the foundation for MAN[7].

Both Ethernet and SONET/SDH have their individual advantages, which are summarised in Table 4.3.

4.1.4.3 Wavelength-division Multiplexing (WDM)

Wavelength-division Multiplexing (WDM) is the generic name for frequency-division multiplexing in the optical domain. It is best understood as a fibre-multiplication technology: it allows multiple optical circuits to share a single physical fibre strand without interfering with each other as their signals use different carriers occupying non-overlapping parts of the frequency spectrum (virtual fibres).

The number of optical signals multiplexed within a window is limited only by the precision of the optical equipment. WDM can therefore increase the optical fibre bandwidth many folds without expensive re-cabling. However, electronic switching gear is commonly used at the ends of the optical circuits and forms the bottleneck in today's backbones. Optical switching technology promises to also remove this bottleneck and decrease the costs for bandwidth even further.

Because WDM operates at the photonic level, it allows different framing and transmission technologies to be used on each wavelength. Considering the vast investment carriers have made in SONET and SDH equipment and their experience with it, the integration of SONET/SDH and WDM seems a reasonable and likely step on the way to all optical

[7] See, for example, the services of Yipes (www.yipes.com), Cogent Communications (www.cogentco.com) and OnFibre (www.onfibre.com).

Table 4.3 Comparison of SONET and Long-distance Ethernet

	SONET/SDH	Ethernet
Historical traffic	Voice traffic	Data traffic
Historical network type	WAN	LAN
Standardisation gremium	ANSI/ITU-T	IEEE
Supported network types	MAN, WAN	LAN, MAN, WAN (10GE)
Link protocol	Synchronous	Asynchronous
Bandwidth scalability	52 Mbps to 40 Gbps	1 Mbps to 10 Gbps
Advantages	Survivable (50 ms restoration time with APS (automatic protection switching, 99.999% reliability; see Goralski (2002))	Lower equipment costs
	Optimised for voice traffic	Optimised for data traffic
	Widely deployed in WANs	Widely deployed in LANs; MANs/WANs can be connected to LANs without reframing
Annotations		Solutions exist for running Ethernet over existing SONET/SDH infrastructure

transport networks; see Cavendish (2000). Thus, one solution for IP over WDM is running IP over PPP/SDL over SONET/SDH over a WDM link.

It is possible to simplify the protocol stack by removing the complexity of SONET/SDH and send IP directly over WDM links using Multi-Protocol *Lambda* Switching (MPλS), a variation of the Multi-Protocol *Label* Switching (MPLS)(for MPLS, see Section 6.3.2) approach that uses wavelengths instead of labels. Packet-switching MPLS and wavelength-switching MPλS are subsumed[8] under the GMPLS (Generalised MPLS) framework that provides a generalised signalling control protocol standard for multiple types of switching. More information can be found in Banerjee *et al*. (2001); Berger (2003); Durresi *et al*. (2001).

With a set of tests over four testbeds in Finland, France, Sweden and Switzerland, Rodellar (2003) compares and evaluates three approaches for IP over WDM: (1) IP – Packet over SONET/SDH – WDM, (2) IP – native 1GE – WDM, (3) IP – DPT (Dynamic Packet Transport[9]) – WDM.

Among other things, the study shows that none of these solutions has a clear technical advantage over the other. The feasibility of real-time applications such as IP telephony or video across a 1GE link over a WDM network has been verified. The study, however,

[8] In the literature, MPλS and GMPLS are sometimes used as synonyms. This is not technically correct as GMPLS explicitly also addresses other kinds of switching besides wavelength switching, as for example switching in the time domain (time division multiplexing).

[9] DPT is a proprietary layer 2 switching solution from Cisco for transporting IP packets over ring networks.

also found that it is still necessary to separate the traffic of these real-time applications from other low-priority traffic. We investigate methods for doing this in Part II.

4.2 Summary and Conclusions

In this chapter, we discussed the basic network and transport layer protocols of the Internet: IP, TCP and UDP. TCP is the most commonly used transport protocol in the Internet and uses a complex congestion and flow control mechanism that was discussed in this chapter. Methods for estimating the throughput of a TCP connection for different network conditions were discussed as well. Finally, different lower layer technologies for ISPs and carriers were shortly discussed towards the end of this chapter.