1
Introduction and Overview

In this chapter we first give a brief tour of some basic concepts of IP networking. Then we introduce the SCTP protocol and discuss the rationale behind its development. In this discussion we will talk about SCTP's various features and its uniqueness to other IP transport protocols such as TCP and UDP. In the rest of the chapter we will recount briefly the history of SCTP's development, and the major issues encountered and decisions made during its development.

As in any other technical book nowadays, throughout this book the reader will inevitably encounter a lot of technical terms, abbreviations, and acronyms that are either general definitions for computer networking or more specific terms related to SCTP. For the convenience of the reader, we provide at the end of the book a list of all used definitions in the Glossary and Abbreviations section.

1.1 IP Networking Basics

One basic yet very important kind of communications in an IP network is so-called **connection-oriented communications**. In such a communication session, before any user data can be exchanged between two parties in the IP network, the two parties must go through a communication setup procedure and establish themselves in the appropriate state. Normally one party initiates the communication. This relationship usually starts with both parties confirming their willingness to engage in the communication. The involved parties will also need to agree, up front, upon the protocol or language to be used in the communication, the nature of the communication, etc. Once this state is exchanged, the relationship is established and reliable user data communication can begin.

In this section we will first discuss the basic addressing model used in an IP network, that is, how a message is addressed and how it is delivered to the intended receiver. This is essential to an understanding of how the communication parties find and exchange information with each other. Then we will discuss in detail how the endpoints in a communication, as well as the communication relationship itself, are logically defined.

1.1.1 How Messages Are Delivered in an IP Network

1.1.1.1 IP Address and IP Datagram Routing

In an IP network, application programs running on different machines communicate with each other usually by sending and receiving user messages. These user messages are translated into IP datagrams for transport across the IP network. In order to identify each individual machine, the network administrator assigns a unique IP address[1] to each of the machines connected to the network. This IP address assignment is instantiated in the machines either via manual configuration or via an automated address-management protocol such as the Dynamic Host Configuration Protocol (DHCP) (Droms 1993).

In many cases the assignment of an IP address to a machine is permanent; that is, the machine's IP address will remain unchanged for a significant period of time. An example of such a case is a server providing Web pages on the Internet; its IP address will stay the same until some rare circumstances occur and the network administrator decides to reconfigure the network and assign a new IP address to the server.

However, there are other occasions when the IP address assigned to a machine may change very frequently. An example of such a case is when, using a modem, you dial up to your Internet service provider from your home PC to set up a PPP[2] connection. Your Internet service provider will most likely assign a different IP address each time you dial in. However, once assigned, this IP address will normally stay unchanged for the entire duration of the PPP connection.

When the sender sends a message to the receiver, before the message gets on the network, the communication software inside the sender's machine will pack the message into one or more **IP datagrams**. The packing process includes the prefixing of a tag, called the **IP header**, to each IP datagram. The IP header will contain both the sender's machine's and the receiver's machine's IP addresses.

---

1. In some cases a machine may have more than one IP address assigned, based on various factors the administrator may deem appropriate. *See also* Section 1.1.3.

2. PPP (Simpson 1994) is a line protocol often used by Internet service providers to connect a home computer to the Internet.

Once an IP datagram leaves the sender's machine and gets onto the network, **IP routers**, which are essentially computers in the network that are specialized in forwarding IP datagrams, will take over and move the IP datagram toward the receiver machine. This process is called IP routing.

In its simplest form, IP routing requires each involved IP router to decide to which of its neighbors the IP datagram should be forwarded, based on its knowledge of the network topology and the receiver's IP address carried in the IP header of the IP datagram.[3] This process continues until an IP router eventually passes the IP datagram to the receiver machine.

1.1.1.2 IP Transport Port

When the IP datagrams arrive at the receiver's machine, the job of the routers in the network is finished. The rest is left to the communication software on the receiver's machine: to unpack the IP datagrams in order to reconstruct the original message, and finally to pass the message to the intended message receiver.

Since there may be other applications waiting for messages on the receiver's machine, there needs to be a way for the communication software at the receiver's machine to ascertain where the message is destined for before it can pass the message to the right receiver application. This need is solved by associating each message-receiving application on the destination machine with a different **transport port**. This assignment of transport ports is normally governed by the operating system on the machine.

When the sender's communication software packs the message into outbound IP datagrams, in addition to adding the IP header, it also prefixes a transport header. This transport header will contain both the sender's and receiver's transport ports. Once the IP datagrams arrive at the receiver's machine, the communication software there will identify the intended message-receiving application by examining the receiver's port number, which is carried in the transport header of each arrived IP datagram.

In summary, in order to uniquely identify a message receiver in an IP network, we need both an IP address of the machine on which the receiver resides and the transport port assigned to the receiver on the machine. The layered reference model of the IP-based communication application shown in Figure 1–1 is a conve-

3. While most of the time only the receiver's IP address is used for routing, in modern IP networks, sometimes routers may also examine the sender's IP address as well as the transport layer ports carried in the transport header. For example, the sender's address may be used for ingress filtering, while the ports may be used for packet filtering for security and load balancing purposes.

   Moreover, in the case of source-based routing, both the sender's and receiver's addresses will be considered in making the routing decision. Source-based routing is still a research topic and is not generally available on the Internet, although it may be used by some Internet service providers for maintenance and troubleshooting.

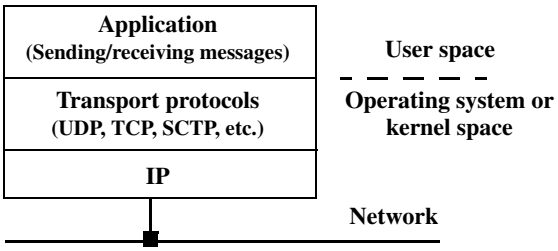

Figure 1–1 *IP communication application reference model*[a]

a. The transport protocols can also reside in the user space, depending on the specific implementation strategy in use. Moreover, in some special circumstances such as when the platform is built upon a very small device, there may be no separation between the user space and kernel space at all.

nient way of showing this collaboration. The IP address is used in the IP layer to identify the machine (sometimes also called a **communication node** in the network) on which the application is running, and the transport port is used in the transport layer to identify the application on the machine.

Figure 1–2 shows an example of two communicating applications on different machines in a network. In this example, each machine is attached to the network through its network interface (NI). Each network interface has been assigned a unique IP address by the network administrator. Application 1 on machine A has been assigned port number 100 by its operating system; similarly, application 2 on machine B has been assigned port number 200.

When application 1 sends a message to application 2, the IP datagrams that carry the message will have in their IP headers machine A's address (160.15.82.20) as the sender's IP address, and machine B's address (128.33.6.12) as the receiver's IP address. Similarly, the transport header in each IP datagram will indicate 100 as the sender's port and 200 as the receiver's port.

Conceptually, in a simplified view the IP address and port number form a division of labor that accesses the destination application. The IP address serves as the mechanism to route the IP datagrams through the network to the destination machine.[4] The port number serves as a de-multiplexing agent for the destination machine's communication software to use to find the individual receiver application that is to receive the message.

4. When moving the IP datagram through the network, some special network devices, such as some application-level packet filters, also look at the port information carried in the transport layer header to make routing decision.

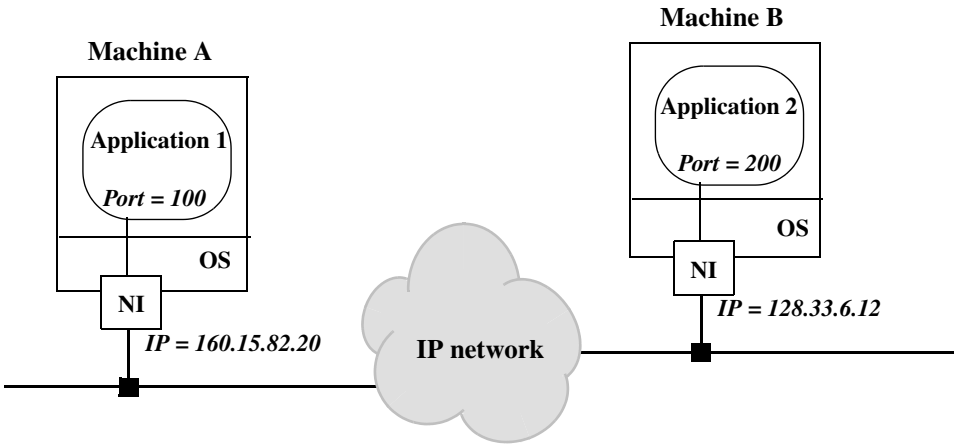

Figure 1–2 *Example of two communicating applications*

1.1.2 IP Protocols

Communication nodes in a network can only communicate with each other if they all follow the same communication rules or conventions. These rules are normally called **protocols** and are defined and maintained by various standardization bodies such as the IETF and the International Telecommunications Union (ITU). For IP networking, the IETF has defined almost all the protocols in use today.

In order to manage the complexity of the modern communication systems better, protocol designers have used a layering principle that divides the communication software design into layers. The behaviors and rules of each of the layers are in turn defined in separate protocols.

For the IP network, the two most important communication layers are the **IP layer** and the **transport layer**, as shown in Figure 1–1.

The IP layer provides a basic connectionless datagram delivery service. The transport layer, which is built upon the IP layer and uses the datagram delivery service provided by the IP layer, provides a more sophisticated end-to-end data transportation service to the application.

Currently, two major IP layer protocols, the IPv4 (Postel 1981a) and IPv6 (Deering and Hinden 1998) protocols, are defined by the IETF. IPv4 has been widely deployed in the field and is the backbone of today's global Internet, while the deployment of IPv6 has just started.

At the transport layer, the best-known and most widely deployed protocols are the Transmission Control Protocol (TCP) (Postel 1981b) and the User Datagram Protocol (UDP) (Postel 1980).

1.1.3 IP Multi-homing

Another fundamental IP concept that we will need to understand in order to understand SCTP is **IP multi-homing**.

In the IP terminology, a communication node or host is called **multi-homed** if it can be addressed by (and thus "owns") multiple IP addresses.

Multi-homing is defined in technical detail in Braden (1989). But in a simplified view, multi-homing is usually the result of the host machine (where the communication node resides) being installed with either of the following:

- Multiple network interface cards, with each assigned a different IP address and/or
- A single network card to which multiple IP addresses are assigned[5]

However, for the purposes of our discussion, we will take a simple view in which multiple interfaces are installed and each interface has only one IP address. The example in Figure 1–3 shows such a multi-homed machine.

In the example, machine C has three network interfaces (NI-1, NI-2, and NI-3), each of which is assigned a different IP address. Normally the operating system on the machine is capable of sending and receiving IP datagrams through any of the three network interfaces. The applications running on a multi-homed machine can communicate with the outside through one or more of the available interfaces.

It should be noted that when there are multiple IP addresses on a machine, **there will be a unique transport port number space for each of the IP addresses**. In other words, on a multi-homed machine the transport port number space is defined on a per-IP-address basis, which means that we can have a unique port 100 for each of IP1, IP2, and IP3.

1.2 What Is SCTP?

Similar to TCP and UDP, the Stream Control Transmission Protocol (SCTP) is another general-purpose transport protocol for IP network data communications. It was published as *RFC2960* (Stewart et al. 2000) by the IETF in October 2000 as a Proposed Standard.

5. This is often the case in IPv6 networks where each network interface card is often assigned with a global address, a site-local address, and a link-local address. It is also allowable for a network card to have more than one of each of these types of addresses. The usefulness of the various addresses in IPv6 multi-homing will only be significant if the additional addresses can be used for path diversity.

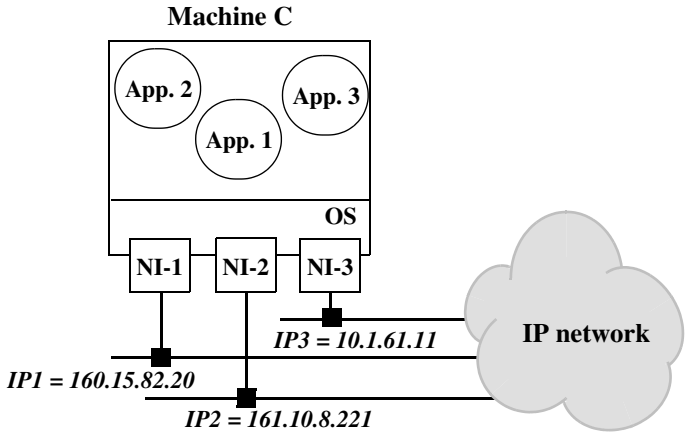

Figure 1–3 *A multi-homed machine with multiple network interfaces*

The primary purpose of SCTP is to provide a reliable end-to-end message transportation service over IP-based networks.

1.2.1 Where Does SCTP Fit in the IP Architecture?

Like TCP and UDP, SCTP belongs to the transport layer in the IP architecture, as shown in Figure 1–4.

Both SCTP and TCP provide a reliable data transportation service between two communication nodes or endpoints that are connected to each other by an IP-based network. However, the SCTP data transportation service is **message-oriented,** while the TCP's is **byte-oriented**.

This means that a sending application can construct a message out of a block of data bytes and then instruct SCTP to transport **the message** to a receiving application. SCTP will not only guarantee the delivery of this data block (message) in its entirety to the receiver; it will also indicate to the receiver both the beginning and end of the data block. This is sometimes also termed **conservation of message boundaries**.

TCP operates very differently in this regard; it simply treats all the data passed to it from the sending application as a sequence or stream of data bytes. It will deliver all the data bytes to the receiver in the same sequential order as they were passed down from the application. However, there will be no conservation of any data block boundaries by TCP.

SCTP is also richer in functionality and more tolerant to network and component failures than TCP. In Section 1.3.3 and in Chapter 12 we will compare TCP and SCTP in more detail.

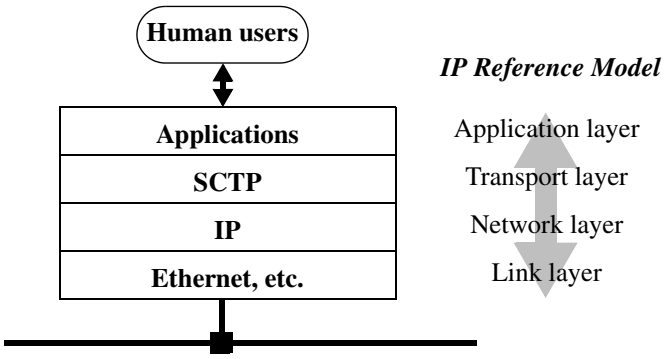

Figure 1–4 *SCTP in the IP reference model*

1.3 Motivation for Developing SCTP

Through its early academic development to its later explosive proliferation into commercial sectors, IP technology has relied successfully on both TCP and UDP as the workhorses of data transfer. However, as the desire of further exploring IP technology for a wider range of commercial applications grows (such as real-time multimedia and telephony applications), some have started to feel that the data transfer services offered by TCP and UDP are inadequate.

One particular application that best exemplifies many of the shortcomings of TCP and UDP and directly motivated the development of SCTP is the transportation of telephony signaling messages over IP networks.

Telephony signaling has rigid timing and reliability requirements that are often set forth through government regulations and must be met in order for the phone service provider to be qualified.

1.3.1 TCP Limitations

The following limitations of TCP make it hard to meet the rigid timing and reliability requirements of telephony signaling:

- TCP provides both reliable data transfer and strict order-of-transmission delivery of data. Telephony signaling applications often need reliable message transfer but not a globally strict sequence maintenance. What is often more

desirable for telephony signaling is a partial ordering of the data, such as maintaining ordering only within some subflows of the data.

The strict sequence maintenance in TCP not only makes such type of partial ordering of the data impossible, it can also potentially introduce unnecessary delay to the overall data delivery service. Loss of a single TCP segment can block delivery of all subsequent data in a TCP stream until the lost TCP segment is delivered. This is sometimes called **head-of-line blocking**. Excessive delays in telephony signaling may cause various types of service failure and hence must be closely controlled.

- The byte-oriented nature of TCP is often an inconvenience to the message-based telephony signaling. Applications must add their own record marking to delineate their messages, and they must make explicit use of the push facility to ensure that a complete message is transferred in a reasonable time.
- TCP has no built-in support for multi-homed IP hosts. This brings an immense challenge to the system designers who want to build link or path-level redundancy. Without link or path-level redundancy, the data transfer service in a network can become vulnerable to link failures. This is often unacceptable for a telephony signaling network, because providing highly available data transfer capability is one of its primary requirements.
- For telephony applications, especially commercial phone services, security against malicious attacks intended to cause failure or interruptions to the services is often a top priority. But TCP is known to be relatively vulnerable to some denial of service attacks, such as the **SYN flooding** attack.[6]

6. A SYN flooding attack is one of a number of denial-of-service attacks (Ferguson and Senie 1998) that have been used on the Internet. It is usually executed by a malicious host (the attacker) sending a targeted host (the victim) a large number of SYN messages. (The SYN message is the first setup message in a TCP connection, similar to SCTP's INIT message, as we will later see.)

   The attacker sends each SYN message as if it had come from some other machine; that is, it fakes and keeps changing (often randomly) the source IP address in each SYN message. Each time the TCP stack on the victim machine gets a SYN message for a port that is accepting connections, it allocates some kernel resource in preparation for a new connection and sends back a SYN-ACK message.

   The attack takes effect when a great number of such SYN messages flood into the victim machine in a short period of time and eventually deplete its resources. When this happens, the legitimate services provided by the victim machine may be disrupted. In severe cases, the victim machine can be completely knocked out of service by the attack.

   It should be noted that some modern TCP implementations do implement methods to make SYN flooding less effective by, for example, enhancing the system resource to sustain attacks with very high connection attempt rates or putting a limit on the number of connection requests that can be attempted over some period of time. Also, Internet providers now do ingress filtering so that invalid IP source addresses are not allowed from the network edge.

1.3.2 UDP Limitations

UDP has its own shortcomings when it is being considered for carrying telephony signaling data. First, UDP only provides an **unreliable** data transfer service to the application; in other words, an application using UDP cannot know whether data sent to a peer application is received by the other end or not. Moreover, for those that reach the destination, there is no guarantee on the ordering of the data, and the receiver may get duplicated copies of the same data. Secondly, UDP has no built-in congestion management mechanism to detect path congestion and consequently to throttle back its data transmission. This can have two very undesirable effects:

- Injecting more data into an already congested network is unfriendly to the network; it often will worsen the network congestion and adversely affect other applications communicating using the same network.
- Data sent over an already congested network will likely be discarded by the network. This in turn makes UDP data transfer service more unreliable when the network is congested.

By itself, due to the above shortcomings, UDP cannot meet the data reliability requirements and thus is unsuitable for telephony signaling applications. However, because UDP is message-oriented and generally considered a lightweight protocol that has a relatively small overhead, there have been attempts to make up in the application itself what is lacking in UDP in order to meet more stringent timing and data reliability requirements. This normally means that the application will need to build its own mechanisms to do the following:

- Detect and retransmit any lost messages
- Correct out-of-order and duplicated messages
- Detect and deal properly with network congestion

This may not always be a good solution though, because the added complexity to the application may be considered excessive, and the mechanisms needed are not trivial to design and implement correctly.

1.3.3 SCTP Enhancements over TCP and UDP

In order to address the above shortcomings and limitations of TCP and UDP for these types of applications, the Signaling Transport (SIGTRAN) working group in the IETF developed SCTP. While the development of SCTP was directly motivated by the transportation of the Public Switched Telephone Network (PSTN)

signaling messages across the IP network, SIGTRAN ensured that the design is also a good match for other applications with similar requirements.

Two major new capabilities are designed into SCTP: the support for multi-homed hosts and the support for multiple streams in a single SCTP association.

The built-in support for multi-homed hosts allows a single SCTP association to run across multiple links or paths, hence achieving link/path redundancy. With this capability, an SCTP association can be made to achieve fast failover from one link/path to another with little interruption to the data transfer service.

The multiple stream mechanism is designed to solve the **head-of-the-line blocking** problem of TCP. This feature gives the user of SCTP the ability to define subflows inside the overall SCTP message flow and to enforce message ordering only within each of the subflows. Therefore, messages from different subflows will not block one another.

Besides the two major new features just described, there are other enhancements designed into SCTP. Table 1–1 gives a more detailed feature comparison between SCTP and TCP and UDP.

1.4 A Short History of SCTP Development

SCTP has been formed over a number of years that involved a lot of research and experiments. Some of this research and these experiments the authors of this book were involved with, and much of it we were not. This section details some of the early work that has culminated in the current SCTP protocol.

1.4.1 Early Works Before the IETF and MDTP

The work that has become SCTP began quite a number of years ago with the realization that TCP had several key weaknesses in dealing with telephone call control. The first realization came in 1991 when a network broke while testing and many minutes transpired before the TCP socket gave an error indication. This was quite unacceptable and began the quest (at least by the authors) to build something better.

Three consecutive works were started by this incident, each experimenting with methods of putting together reliable communications that used UDP. Each one attempted to escape some of the deficiencies of TCP. One of these early implementations used a continual three-way handshake, while another used a modification of this. Each improved on the other until the last, Multi-network Datagram Transmission Protocol (MDTP), began in 1997. After getting most of the general concepts together and having a working implementation, the authors decided to submit it to the IETF for consideration in 1998.

Table 1–1 Feature Comparison Between SCTP, TCP, and UDP

Protocol Feature	SCTP	TCP	UDP
State required at each endpoint	yes	yes	no[a]
Reliable data transfer	yes	yes	no
Congestion control and avoidance	yes	yes	no
Message boundary conservation	yes	no[b]	yes
Path MTU discovery and message fragmentation	yes	yes[b]	no
Message bundling	yes	yes[b]	no
Multi-homed hosts support	yes	no	no
Multi-stream support	yes	no	no
Unordered data delivery	yes	no	yes
Security cookie against SYN flood attack	yes	no	no
Built-in heartbeat (reachability check)	yes	no[c]	no

a. With UDP a node can communicate with another node without going through a setup procedure or changing any state information. This is sometimes called **connection-less**, but in reality each UDP packet has the needed state within it to form a connection so that no ongoing state needs to be maintained at each endpoint.

b. Because TCP treats all the data passed from its upper layer as a formatless stream of data bytes, it does not preserve any message boundaries. However, due to its byte-stream-based nature, TCP can automatically resize all the data into new TCP segments suitable for the Path MTU before transmitting them.

c. Most TCP implementations do implement a "keep-alive" mechanism. This mechanism is very similar to the SCTP heartbeat, with the main difference being the time interval used. In TCP the "keep-alive" interval is, by default, set to two hours. The goal of this "keep-alive" is long-term state cleanup, which is in sharp contrast to SCTP's much more rapid heartbeat, which is used to aid in fast failover.

The submission of MDTP coincided with another telephony-signaling-over-IP initiative also being started in the IETF. That initiative resulted in the forming of the SIGTRAN working group in the Transport Area. At that time, the goal of SIGTRAN was to move existing telephone signaling protocols, including ISUP, DSSI, etcetera, onto a pure IP-based network.

The requirements for telephony signaling transport and the modular architecture developed by SIGTRAN found a good fit with MDTP's design concept. This began a host of modifications of the protocol in SIGTRAN, improving and refining the original design.

1.4.2 IETF Refinements

In the IETF the original MDTP design grew and changed through the input of many on the SIGTRAN mailing list and from the various face-to-face design team meetings that were held between 1998 and 2000. Somewhere along the way the name of the protocol was changed from MDTP to SCTP.

This name change evoked much discussion on the SIGTRAN mailing list as well as within the design team. The name change in many ways was more significant than many thought. It not only symbolized the substantial design improvement that the working group had performed; it signified an expansion of scope and functionality of the protocol. This expansion was what eventually led to SCTP being moved from a protocol running over UDP to one that directly runs over IP.

The following list describes some of the major changes that were invoked by the working group and the Internet Engineering Steering Group (IESG):

- *Multi-stream concept*—The working group made the decision to separate the data reliability function from the message ordering function. This in effect enabled multiple ordered subflows within a single reliable connection and provided a solution to the "head-of-line blocking" problem that the original MDTP design was not able to solve.
- *Congestion control enhancements*—The original MDTP design addressed congestion control incorrectly. The working group corrected this by completely redesigning most parts of the MDTP congestion control function, using both the TCP experiences learned from the past and new research results on TCP congestion control in the IETF and IRTF communities.
- *Four-way secure association initiation sequence*—The original design used a modified three-way handshake initiation sequence similar to that of TCP. This was improved by the working group via the addition of the fourth "leg" to the handshake sequence and the introduction of an encrypted **state cookie** to fend off potential security attacks.
- *Selective acknowledgment (SACK) improvements*—The SACK function was improved and enhanced from its original design so it more closely paralleled the TCP SACK extension.
- *Message bundling improvements*—The original MDTP message bundling was replaced with a very efficient and flexible chunk-based message bundling mechanism by the working group.
- *Path MTU discovery*—This was added to SCTP by the working group as a mandatory function in order to make SCTP more adaptive to various network conditions.

- The large message fragmentation function was redesigned.
- *Extensibility improved*—The chunk-based design and the extension mechanism were introduced to allow the IETF to add new features to the protocol in the future.
- An enhancement was made to allow user data to be piggybacked on the third and fourth legs of the initial opening handshake sequence.

After about two years of designing and reviewing and about twenty revisions on the draft document, SCTP finally became an IETF Proposed Standard in October 2000 and was published as *RFC2960* (Stewart et al. 2000). Since then, the continued work on SCTP has been handed over from SIGTRAN to the Transport Area working group (TSVWG) in the IETF.

If you would like to plumb the history of SCTP and dig through the 4,000+ e-mails that were generated during SCTP's birth, you can consult the IETF Web pages to find the current pointer to the e-mail archives of both SIGTRAN and TSVWG at *http://www.ietf.org*.

1.5 Major General SCTP Issues Debated in the IETF

In this section we give a recount of some important architectural discussions that took place in the IETF during the process of developing SCTP. Some of those discussions eventually led to SCTP design decisions of fundamental importance.

1.5.1 Do We Really Need a New Transport Protocol?

When the SIGTRAN working group was formed in 1998, one of the first questions it was chartered to find the answer to was whether TCP was good enough for telephony signaling applications. At that time, IP telephony was still a relatively new concept to the IETF, and most of the experience on IP telephony was gained from experiments with personal computers equipped with low-cost sound cards passing sampled speech to each other. However, people quickly realized the potential of providing telco- or carrier-grade telephony services over an IP-based infrastructure on a much larger scale than a few connected PCs with sound cards.

To provide large-scale commercial telephone services, a reliable way of transporting telephony signaling messages across IP networks had to be defined first.

People from the telephone industry, the so-called **Bell heads**, were well aware of the stringent timing and reliability requirements of transporting telephony signaling messages and had specifically created the Signaling System No. 7 (SS7)

network to meet those requirements. They had serious doubts about whether TCP, the only reliable data transport protocol in the IP world, was up to the task.

A team led by Telcordia (formerly Bellcore) engineers was formed in SIGTRAN to investigate the adequacy of using TCP/IP to transport telephony signaling. They came to the conclusion that the main problem with using TCP/IP for providing commercial-grade telephony signaling was the lack of control over the TCP retransmission timers. Some of the working group members quickly pointed out that this lack of user-level timer control was merely an implementation problem of the TCP stack, instead of a limitation on the protocol itself.

However, the analysis also pointed out that the "head-of-the-line blocking" imposed by TCP was also an issue. In high call-volume application scenarios, such as call processing centers, this "head-of-the-line blocking" could become a major problem due to delays imposed on user messages waiting at the receiver for a lost TCP segment to arrive. These delays would impose a domino effect on all calls, escalating the delay to an unacceptable level.

With the submission of MDTP and the ensuing discussions, the working group also quickly agreed on another major shortcoming of TCP: its lack of path-level redundancy support. This was viewed by many telco engineers as a major weakness of TCP/IP compared to SS7, because the latter was designed with full support of link-level redundancy. (For example, in the field, the data connection between two SS7 signaling nodes is often deployed over a set of physical T1 links, commonly referred to as a **link-set**, that may consist of up to 16 separate T1s, thus providing considerable link redundancy.)

Because of its packet-switched nature, a connection in an IP network normally does not confine itself to a specific physical link. Rather, what is equivalent to a link in the circuit-switched SS7 network is a **path** in an IP network. In order to achieve a degree of link-level fault resilience similar to that offered by the SS7 network, the working group reached the consensus that the IP transport protocol for telephony signaling must provide support for path redundancy.

There were also concerns that the three-way handshaking procedure used during the TCP setup might introduce too much delay at call setup.

The option of modifying or enhancing TCP to meet those new requirements was briefly discussed in the working group but was quickly abandoned. The working group was hesitant to take that approach, probably because other similar IETF investigations on transport issues—for example, the Requirements for Unicast Transport/Sessions (RUTS) BOF[7] and the Support for Lots of Unicast Multiplexed Sessions (SLUMS) BOF—had already pointed out the difficulty of modifying TCP.

7. BOF (Birds of a Feather) is an informal meeting format used in the IETF to gauge the technical interest of the engineering public on a specific topic prior to the creation of a full working group.

In the end, the working group reached the conclusion that a new transport protocol that would run over UDP was needed. MDTP was adopted as the baseline design of the new protocol. The working group also decided to insert a signaling protocol adaptation layer between the new transport protocol and the telephony signaling protocols. This decision effectively separated the transport function from any specifics of the upper-layer telephony signaling protocols. This separation paved the way for further transformation of MDTP into a generic IP transport protocol.

1.5.2 Over UDP Versus Over IP

As stated above, MDTP was designed to run on top of UDP for reasons that include implementation convenience (you do not need to write kernel code) and the desire of having tight control over the retransmission timing. This approach was unchanged after the working group took on the protocol development.

The work on MDTP had been moving forward smoothly and the protocol design was becoming stable when the working group received word from the IETF Transport Area Directorate in December 1999 that some important design decision should be revisited. The TAD had been discussing whether to change the new protocol (renamed SCTP already) to sit directly over IP instead of over UDP.

Soon afterward the IESG and the TAD made the recommendation of moving SCTP to run directly on top of IP. The reason was that the IESG and the TAD saw the value and significance of the new protocol and recognized the great potential of SCTP becoming a major transport protocol. SCTP was thought to be useful to a much wider range of applications than telephony signaling transport. Moving SCTP directly over IP with its own port number space was viewed as the architecturally correct solution.

Initially the working group was very reluctant to accept this recommendation. The main concern was that this change would almost certainly mean that SCTP would have to be implemented in the operating system kernel. It would normally take most operating system vendors a long time, up to several years in some cases, to make a new protocol commercially available in their operating system kernels. It was thought that the telephone industry just could not wait that long for the SIGTRAN protocol. In addition, some were concerned that to have the protocol sitting in the operating system kernel would make it much harder to keep tight control over the retransmission timers. Without this control, the stringent timing requirements of telephony signaling would be difficult to meet.

However, the IESG insisted that the working group should look at the bigger picture beyond signaling transport as well. The long-term benefits of putting SCTP in an architecturally correct position in the IP stack far outweighed the short-term deployment delay the change would incur on the telephony signaling transport applications. Eventually the working group agreed.

1.6 Organization of this Book

The authors' intention is to make this book useful for two different groups of people: readers who are seeking an understanding of the protocol and wish to have something easier to read than the protocol specification itself, and serious protocol implementers who are looking for insights into protocol design and detailed discussions on how various protocol features work or should be implemented.

To achieve this goal, we documented some important issues and items of debate that occurred while the protocol was being developed, giving the reader a bird's eye view of how the current protocol took its form in the IETF.

We also provided many examples to illustrate and explain the internal mechanisms and features of SCTP.

Whenever conflicting needs must be accommodated, a designer must make various compromises. SCTP's design is no exception. In this book we try to explain the reasoning behind those compromises, discussing the pros and cons of those features and how best to apply them in various environments.

For implementers we include an annotated version of the open-source user-space SCTP reference implementation. We explain in detail the software architecture and how the overall protocol features are implemented in the reference implementation.

Particularly, in Chapter 2 we provide a detailed analysis of some basic terms and definitions used in SCTP, such as **transport address**, **endpoint**, **association**, etc. Those terms and concepts are essential to understanding the design and operation of SCTP described in the rest of the book.

Chapter 3 describes in detail the SCTP packet formats—the bits and the bytes that appear on the wire. This description is more detailed than you can find in *RFC2960* (Stewart et al. 2000). It will help new readers get a grasp of what is on the wire, and it will provide a valuable reference to protocol developers. Readers whose goal is to gain a basic understanding of the protocol can refer to this chapter as needed but do not need to read it in detail.

Chapter 4 walks the reader through the details of setting up an SCTP association. It gives a high-level overview of the association establishment process, followed by a detailed summary. A reader who only wants to understand the basic concept of and steps for setting up an association can examine the high-level overview and move on, while those who want to implement the protocol may find helpful the detailed information in the rest of the chapter on how and why the protocol reacts the way it does during setup.

In Chapter 5 we discuss how the user data transfer happens within SCTP. For the convenience of the readers, an overview of the SCTP data transfer operation is given at the beginning of the chapter. The rest of the chapter provides a great deal of detail on

- How the SCTP sender transforms user messages into DATA chunks
- How the chunks are put together to form SCTP packets
- How fragmentation is done
- How bundling is done
- How the retransmission timer operates
- How the receiver acknowledges received DATA chunks
- How lost data is handled
- How SCTP deals with multi-homed hosts, etc.

Chapter 6 talks about SCTP congestion control and congestion avoidance algorithms. For those who plan to implement SCTP, this chapter is very important to understand.

The ability to detect failures and recover from them is one of the important design objectives of SCTP. Chapter 7 discusses how this is accomplished. The reader will find detailed discussions on the types of failures SCTP is capable of detecting, the internals of the failure detection and recovery algorithms, how the multi-homing and route arrangements interplay with SCTP failure detection and recovery, and so on.

In Chapter 8 we discuss some auxiliary functions within SCTP, particularly the handling rules for the out-of-the-blue (OOTB) SCTP packets and the rules dealing with the verification tags. Readers can skip this chapter if they only want to understand SCTP basics. But those who implement SCTP must obtain a full understanding of all of these rules.

Chapter 9 gives a detailed account of how an SCTP association is closed or terminated. It covers both the graceful shutdown case and the ungraceful abort of an association. This chapter is important for all readers to understand.

Chapter 10 talks about the procedures for creating future extensions of SCTP and assigning well-known SCTP port numbers through IANA.

In Chapter 11 we describe the proposed application programming interface (sockets API) for SCTP. Some readers may not find this chapter immediately useful, but implementers and application programmers who work on SCTP will benefit from the material in this chapter.

Chapter 12 discusses in detail some important differences between SCTP and TCP.

Chapter 13 gives examples to illustrate how the SCTP stream feature can be used.

In Chapter 14 we discuss the open-source SCTP reference implementation. We explain the architecture and design of the implementation and describe each of the main SCTP internal functions and the functional flow within the implementa-

tion. Implementers may find this chapter helpful if they wish to delve a bit deeper and look into the implementation itself.

1.7 Summary

In this chapter we first reviewed some basic concepts and the background of IP networking. We then introduced the SCTP protocol and described its basic features and its role in the IP architecture. We also talked about the motivation behind the creation of SCTP and provided a detailed feature comparison for SCTP, TCP, and UDP. The history of the SCTP protocol development was recounted, and some important IETF debates over SCTP design were summarized.

1.8 Questions

1. Why is SCTP considered a message-oriented transport protocol?

2. Why is UDP alone inadequate for transporting telephony signaling messages over an IP network?

3. List a few advantages and disadvantages of running SCTP directly on top of IP.

4. What is the limitation of TCP that SCTP's multi-stream feature intends to solve?