### User Datagram Protocol (UDP)

* Lack of reliability
* Each UDP datagram has a length
* **Connectionless** service

### Transmission Control Protocol (TCP)

* **Connection**: TCP provides connections between clients and servers. A TCP client establishes a connection with a server, exchanges data across the connection, and then terminates the connection.
* **Reliability**: TCP requires acknowledgment when sending data. If an acknowledgment is not received, TCP automatically retransmits the data and waits a longer amount of time.
* **Round-trip time** (RTT): TCP estimates RTT between a client and server dynamically so that it knows how long to wait for an acknowledgment.
* **Sequencing**: TCP associates a sequence number with every byte (**segment**, unit of data that TCP passes to IP.) it sends. TCP reorders out-of-order segments and discards duplicate segments.
* **Flow control**
* **Full-duplex**: an application can send and receive data in both directions on a given connection at any time.

### Stream Control Transmission Protocol (SCTP)

Like TCP, SCTP provides reliability, sequencing, flow control, and full-duplex data transfer.

Unlike TCP, SCTP provides:

* **Association** instead of "connection": An association refers to a communication between two systems, which may involve more than two addresses due to multihoming.
* **Message-oriented**: provides sequenced delivery of individual records. Like UDP, the length of a record written by the sender is passed to the receiving application.
* **Multihoming**: allows a single SCTP endpoint to support multiple IP addresses. This feature can provide increased robustness against network failure.

**Multipath TCP** (**MPTCP**) is an ongoing effort of the Internet Engineering Task Force's (IETF) Multipath TCP working group, that aims at allowing a Transmission Control Protocol (TCP) connection to use multiple paths to maximize resource usage and increase redundancy.

## Message-based multi-streaming

SCTP applications submit their data to be transmitted in messages (groups of bytes) to the SCTP transport layer. SCTP places messages and control information into separate *chunks* (data chunks and control chunks), each identified by a *chunk header*. The protocol can fragment a message into a number of data chunks, but each data chunk contains data from only one user message. SCTP bundles the chunks into SCTP packets. The SCTP packet, which is submitted to the Internet Protocol, consists of a packet header, SCTP control chunks (when necessary), followed by SCTP data chunks (when available).

One can characterize SCTP as message-oriented, meaning it transports a sequence of messages (each being a group of bytes), rather than transporting an unbroken stream of bytes as does TCP. As in UDP, in SCTP a sender sends a message in one operation, and that exact message is passed to the receiving application process in one operation. In contrast, TCP is a stream-oriented protocol, transporting streams of bytes reliably and in order. However TCP does not allow the receiver to know how many times the sender application called on the TCP transport passing it groups of bytes to be sent out. At the sender, TCP simply appends more bytes to a queue of bytes waiting to go out over the network, rather than having to keep a queue of individual separate outbound messages which must be preserved as such.

The term *multi-streaming* refers to the capability of SCTP to transmit several independent streams of chunks in parallel, for example transmitting web page images together with the web page text. In essence, it involves bundling several connections into a single SCTP association, operating on messages (or chunks) rather than bytes.

TCP preserves byte order in the stream by including a byte sequence number with each segment. SCTP, on the other hand, assigns a sequence number or a message-id to each *message* sent in a stream. This allows independent ordering of messages in different streams. However, message ordering is optional in SCTP; a receiving application may choose to process messages in the order of receipt instead of in the order of sending.

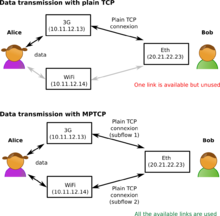
## Features

Features of SCTP include:

* Reliable transmission of both ordered and unordered data streams.
* Multihoming support in which one or both endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths.
* Delivery of chunks within independent streams eliminate unnecessary head-of-line blocking, as opposed to TCP byte-stream delivery.
* Explicit partial reliability.
* Path selection and monitoring to select a primary data transmission path and test the connectivity of the transmission path.
* Validation and acknowledgment mechanisms protect against flooding attacks and provide notification of duplicated or missing data chunks.
* Improved error detection suitable for Ethernet jumbo frames.

The designers of SCTP originally intended it for the transport of telephony (Signaling System 7) over Internet Protocol, with the goal of duplicating some of the reliability attributes of the SS7 signaling network in IP. This IETF effort is known as SIGTRAN. In the meantime, other uses have been proposed, for example, the Diameter protocol and Reliable server pooling (RserPool).

### Simplified description



Difference between TCP and MPTCP

The core idea of multipath TCP is to define a way to build a connection between two hosts and not between two interfaces (as standard TCP does).

For instance, Alice has a smartphone with 3G and WiFi interfaces (with IP addresses 10.11.12.13 and 10.11.12.14) and Bob has a computer with an Ethernet interface (with IP address 20.21.22.23).

In standard TCP, the connection should be established between two IP addresses. Each TCP connection is identified by a four-tuple (source and destination addresses and ports). Given this restriction, an application can only create one TCP connection through a single link. Multipath TCP allows the connection to use several paths simultaneously. For this, Multipath TCP creates one TCP connection, called sub flow, over each path that needs to be used.

The purpose of the different protocol operations (defined in RFC 6824) are:

* to handle when and how to add/remove paths (for instance if there's a connection lost of some congestion control)
* to be compatible with legacy TCP hardware (such as some firewalls that can automatically reject TCP connections if the sequence number aren't successive)
* to define a fair congestion control strategy between the different links and the different hosts (especially with those that doesn't support MPTCP)

Example of a full MPTCP session

Multipath TCP adds new mechanisms to TCP transmissions:

* The subflow system, used to gather multiple standard TCP connections (the paths from one host to another). Subflows are identified during the TCP three-way handshake. After the handshake, an application can add or remove some subflows (subtypes 0x3 and 0x4).
* The MPTCP DSS option contains a data sequence number and an acknowledgement number. These allow receiving data from multiple subflows in the original order, without any corruption (message subtype 0x2)
* A modified retransmission protocol handles congestion control and reliability.

Detailed specification

## Congestion control

Several congestion control mechanisms have been defined for Multipath TCP. Their main difference with classical TCP congestion control schemes is that they need to react to congestion on the different paths without being unfair with single path TCP sources that could compete with them on one of the paths. Four Multipath TCP congestion control schemes are currently supported by the Multipath TCP implementation in the Linux kernel.

* The Linked Increase Algorithm defined in RFC6356
* The Opportunistic Linked Increase Algorithm
* The wVegas delay based congestion control algorithm
* The Balanced Linked Increase Algorithm

DCCP

In computer networking, the **Datagram Congestion Control Protocol** (**DCCP**) is a message-oriented transport layer protocol. DCCP implements reliable connection setup, teardown, Explicit Congestion Notification (ECN), congestion control, and feature negotiation. The IETF published DCCP as RFC 4340, a proposed standard, in March 2006. RFC 4336 provides an introduction.

DCCP provides a way to gain access to congestion-control mechanisms without having to implement them at the application layer. It allows for flow-based semantics like in Transmission Control Protocol (TCP), but does not provide reliable in-order delivery. Sequenced delivery within multiple streams as in the Stream Control Transmission Protocol (SCTP) is not available in DCCP. A DCCP connection contains acknowledgment traffic as well as data traffic. Acknowledgments inform a sender whether its packets have arrived, and whether they were marked by Explicit Congestion Notification (ECN). Acknowledgements are transmitted as reliably as the congestion control mechanism in use requires, possibly completely reliably.

DCCP is useful for applications with timing constraints on the delivery of data. Such applications include streaming media, multiplayer online games and Internet telephony. In such applications, old messages quickly become useless, so that getting new messages is preferred to resending lost messages. As of 2017 such applications have often either settled for TCP or used User Datagram Protocol (UDP) and implemented their own congestion-control mechanisms, or have no congestion control at all. While being useful for these applications, DCCP can also serve as a general congestion-control mechanism for UDP-based applications, by adding, as needed, mechanisms for reliable or in-order delivery on top of UDP/DCCP. In this context, DCCP allows the use of different, but generally TCP-friendly congestion-control mechanisms.

DCCP has the option for very long (48-bit) sequence numbers corresponding to a packet ID, rather than a byte ID as in TCP. The long length of the sequence numbers aims to guard against "*some blind attacks, such as the injection of DCCP-Resets into the connection*

*The DCCP generic header takes different forms depending on the value of X, the Extended Sequence Numbers bit. If X is one, the Sequence Number field is 48 bits long, and the generic header takes 16 bytes, as follows.*

DCCP generic header

| ***Offsets*** | **Octet** | **0** | | | | | | | | **1** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Octet** | **Bit** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **0** | **0** | Source port | | | | | | | | | | | | | | | |
| **2** | **16** | Destination port | | | | | | | | | | | | | | | |
| **4** | **32** | Data Offset | | | | | | | | CCVal | | | | CsCov | | | |
| **6** | **48** | Checksum | | | | | | | | | | | | | | | |
| **8** | **64** | Reset | | | Type | | | | X=1 | Reserved | | | | | | | |
| **10** | **80** | Sequence Number (high bits) | | | | | | | | | | | | | | | |
| **12** | **96** | Sequence Number | | | | | | | | | | | | | | | |
| **14** | **112** | Sequence Number (low bits) | | | | | | | | | | | | | | | |

If X is zero, only the low 24 bits of the Sequence Number are transmitted, and the generic header is 12 bytes long.

| ***Offsets*** | **Octet** | **0** | | | | | | | | **1** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Octet** | **Bit** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **0** | **0** | Source port | | | | | | | | | | | | | | | |
| **2** | **16** | Destination port | | | | | | | | | | | | | | | |
| **4** | **32** | Data Offset | | | | | | | | CCVal | | | | CsCov | | | |
| **6** | **48** | Checksum | | | | | | | | | | | | | | | |
| **8** | **64** | Res | | | Type | | | | X=0 | Sequence Number (high) | | | | | | | |
| **10** | **80** | Sequence Number (low bits) | | | | | | | | | | | | | | | |

Source port (16 bits)

Identifies the sending port

Destination port (16 bits)

Identifies the receiving port

Data Offset

(8 bits): The offset from the start of the packet's DCCP header to the start of its application data area, in 32-bit words.

CCVal (4 bits)

Used by the HC-Sender CCID

Checksum Coverage (CsCov) (4 bits)

Checksum Coverage determines the parts of the packet that are covered by the Checksum field.

Checksum (16 bits)

The Internet checksum of the packet's DCCP header (including options), a network-layer pseudoheader, and, depending on Checksum Coverage, all, some, or none of the application data

Reserved (Res) (3 bits)

Senders MUST set this field to all zeroes on generated packets, and receivers MUST ignore its value

Type (4 bits)

The Type field specifies the type of the packet

Extended Sequence Numbers (X) (1 bit)

Set to one to indicate the use of an extended generic header with 48-bit Sequence and Acknowledgement Numbers

Sequence Number (48 or 24 bits)

Identifies the packet uniquely in the sequence of all packets the source sent on this connection