ALTERNATIVE TRANSPORT PROTOCOLS

OVERVIEW OF IMPORTANT ALTERNATIVE TRANSPORT PROTOCOLS BESIDES TCP AND UDP

Peter R. Egli
peteregli.net

Contents

1. SCTP Stream Control Transmission Protocol RFC2960 (1/4):

SCTP was initially developed as a signaling transport protocol for voice networks. It has the potential to become a successor for TCP.

SCTP corrects some of the shortcomings of TCP which make it the choice for reliable, message-oriented and resilient communication between applications.

Shorter timers:
TCP recovers slowly from problems (retransmission timers, keepalive timers etc.). SCTP has much shorter timers than TCP for faster recovery from problems.

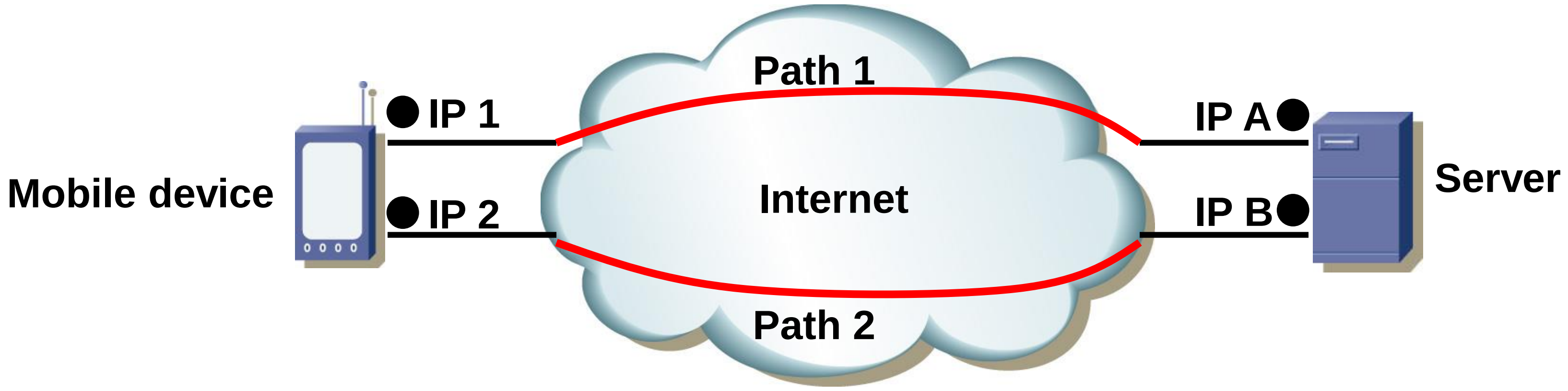
Flooding immunity:
SCTP is better protected against flooding with incoming connection requests.
TCP is vulnerable against SYN-flooding (excessive number of incoming SYN packets each allocating data = TCB Transport Control Block). SCTP also allocates a TCB (contains sequence numbers etc.) but sends it as a ‚cookie' back to the sender and then discards the TCB. If the connection request is spoofed (wrong source IP) the answer will be sent back to nowhere but since the TCB is discarded no resources are allocated on the server. SCTP uses a 4-way handshake for synchronizing the sequence numbers.

© Peter R. Egli 2017

1. SCTP Stream Control Transmission Protocol RFC2960 (2/4):

Multihoming support for more robust communication:

A TCP connection is defined as the quadruple {src IP, dst IP, src port, dst port}. Thus multihoming (multiple IP addresses per TCP port/socket) is not possible. SCTP allows multi-homing as exemplified below:



Both mobile device and server have 2 interfaces to the network with 2 IP addresses each. On both the server and mobile device the SCTP socket is bound to both local IP addresses. If one path fails communication is still possible through the other path since SCTP can send/receive from both addresses. SCTP monitors each path and switches to an alternate path if necessary (all peer transport addresses are monitored constantly).

1. SCTP Stream Control Transmission Protocol RFC2960 (3/4):

Message oriented data transfer:

TCPs stream oriented way of transferring data is unsuitable for most applications, that is the applications must introduce a „framing“ to find the message boundaries (TCP does not provide message boundary preservation). SCTP is message oriented (data chunks), i.e. the receiving application receives data messages as they were sent by the sending application.

Support for fragmentation:

SCTP supports fragmentation in order to reduce the size of data chunks when otherwise the IP layer would fragment the data. Fragmentation in SCTP is better since data needs to be copied around anyway (buffering) while fragmentation in the IP layer adds additional processing load.

No support for half-close:

TCP allows to close only one of the 2 simplex connections while the other remains open. Half-close does not provide functionality that could not otherwise be implemented and is thus not supported by SCTP. This results in a leaner and simpler protocol.

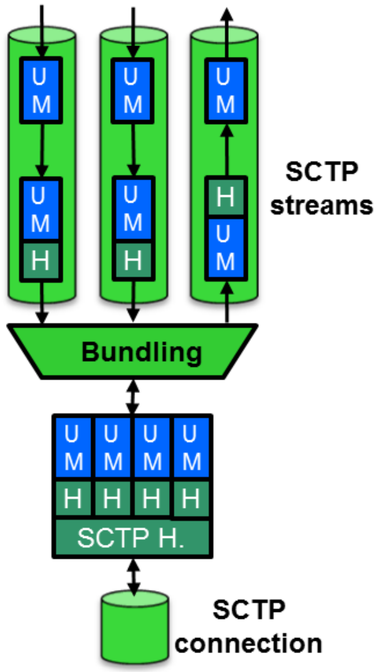
1. SCTP Stream Control Transmission Protocol RFC2960 (4/4):

Support for bundling:

SCTP allows packing multiple data chunks (messages) from multiple streams into one SCTP packet thus decreasing overhead.

SCTP allows multiple streams per connection:

TCP only allows 1 stream per TCP connection. Multiplexing of multiple streams into the same TCP connection is not possible. SCTP allows multiple independent streams of data to be multiplexed into the same SCTP connection. If one stream is blocked due to retransmissions, the other streams are still able to send/receive data.

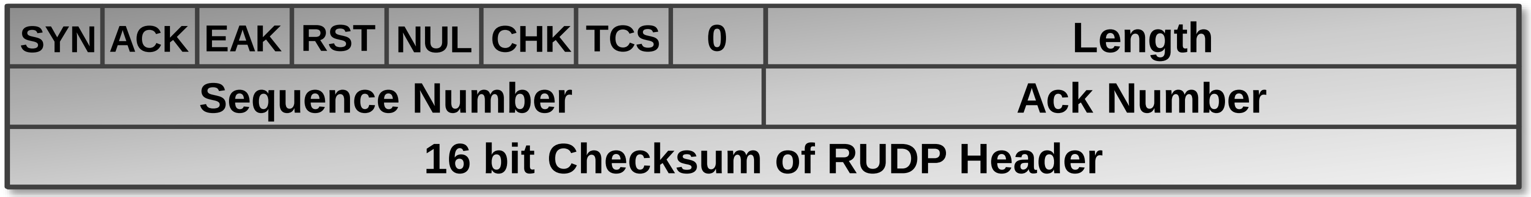


2. Reliable UDP

RUDP introduces some of the algorithms used by TCP to add some of the guarantees of TCP to the UDP protocol.

1. **Client acknowledgment of server data**
2. **Windowing and congestion control (prevent sender from flooding receiver)**
3. **Retransmissions in case of packet loss**

RUDP introduces a new header:



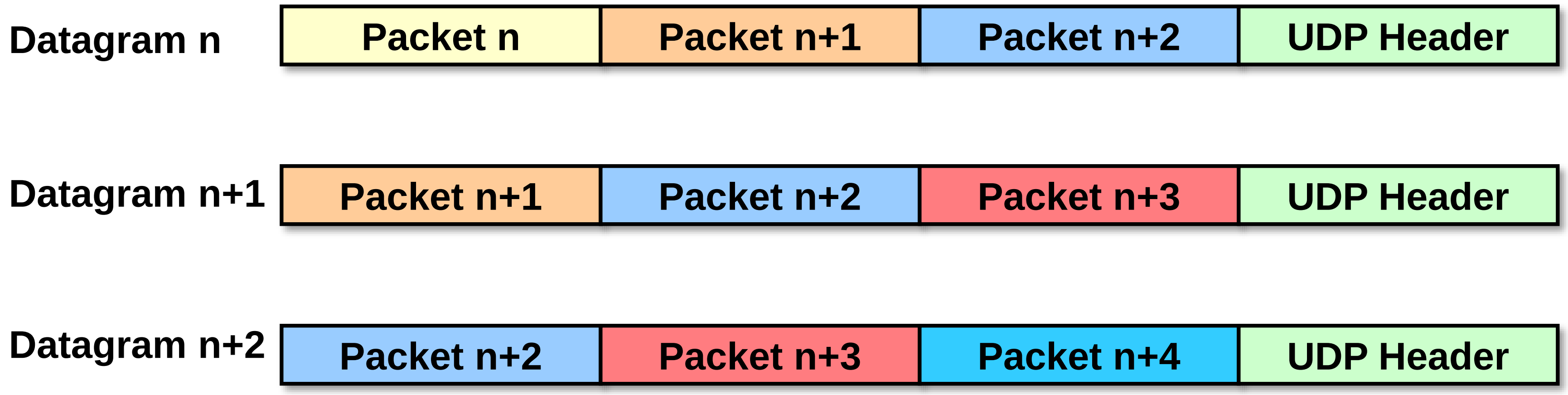
SYN:	**Indicates a synchronization segment.**
ACK:	**Indicates that Ack Number in header is valid.**
EACK:	**Extended Ack.**
RST:	**Reset segment.**
NUL:	**Null segment.**
CHK:	**Indicates if checksum is calculated over RUDP header or header and body (data).**
TCS:	**Transfer Connection State segment.**
Length:	**Indicates beginning of user data in packet.**
Sequence Number:	**Packet based sequence number (incremented by 1 per packet, not per byte).**
Ack Number:	**Indicates the last in-sequence packet correctly received.**
Checksum:	**Checksum of RUDP header or header + data.**

3. UDP Redundancy:

UDP redundancy is a simple mechanism to increase reliability of the data transfer. It just sends data multiple times thus increasing the chance that data arrives correctly (at least once).

This mechanism is not a new transport protocol but part of the application layer right on top of the socket interface.

In order to reduce the overhead, m packets are packed into 1 UDP datagram (e.g. m=3) as follows:


Datagram n
Packet n
Packet n+1
Packet n+2
UDP Header
Datagram n+1
Packet n+1
Packet n+2
Packet n+3
UDP Header
Datagram n+2
Packet n+2
Packet n+3
Packet n+4
UDP Header