CONTENT of this CHAPTER

- Fundamental Goals of the Transport Layer
- Concepts for the Transport Layer
- Standard Transport Layer Protocols
  - UDP
  - TCP
  - MPTCP
  - SCTP
  - DCCP

Motivation and Objectives

Current State

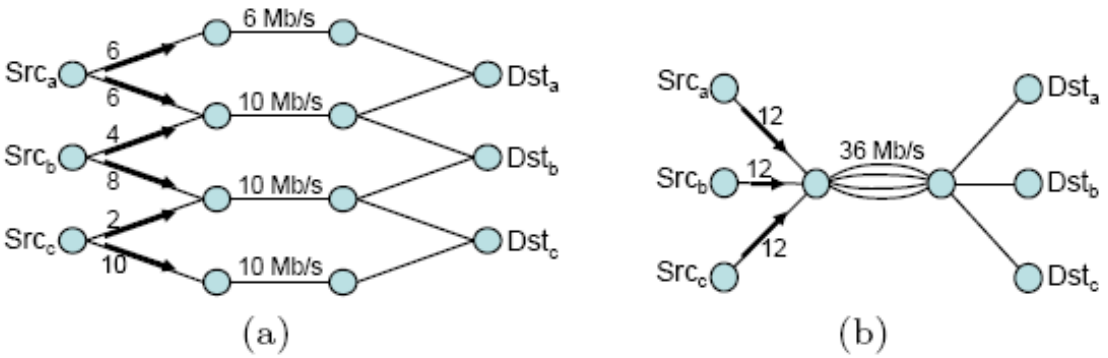
- Groups of end systems pool resources (e.g., CDNs)
- Mobiles are usually equipped with multiple interfaces (e.g., UMTS, WiFi)

Objectives

- Increase robustness, efficiency, and flexibility
- Exploit multiple paths
- Quickly move traffic away from congested or failed links in favor of uncongested links

General Idea: Resource Pooling

- Making a collection of resources behave like a single pooled resource



On which layer should we implement resource pooling?

Highly recommended paper: D. Wischik, M. Handley, M. Bagnulo Braun: "The Resource Pooling Principle", In: ACM CCR, 38(5), pp. 47-52, 2008.

Multipath TCP (MPTCP)

Idea

- Resource pooling on the transport layer
- Based on TCP principles
- Establishment of additional logical connections

Advantages

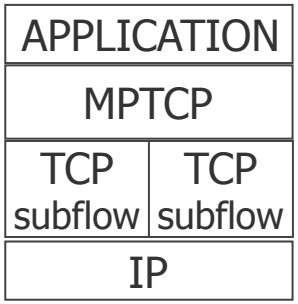
- TCP is the transport protocol of the Internet
  - Web, email, peer-to-peer
- TCP implements per se reliable data delivery and congestion control
- Backward-compatible with TCP
  - Allows for incremental deployment

Specifications

- RFC 6824, RFC 6897, RFC 6356

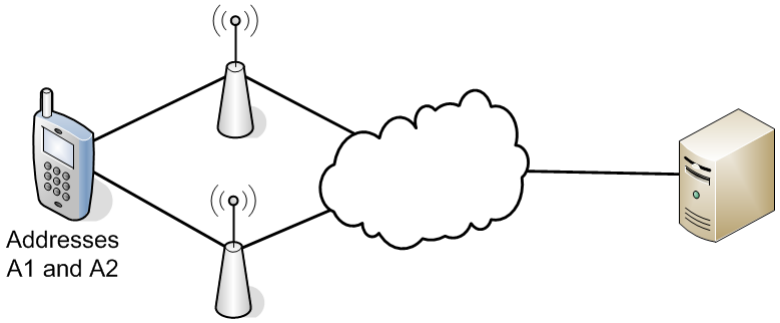
Design Concepts of MPTCP

- In case of TCP end points, MPTCP operates like TCP
  - MPTCP implementation provides MPTCP signaling
  - Advanced API provides additional control on application layer

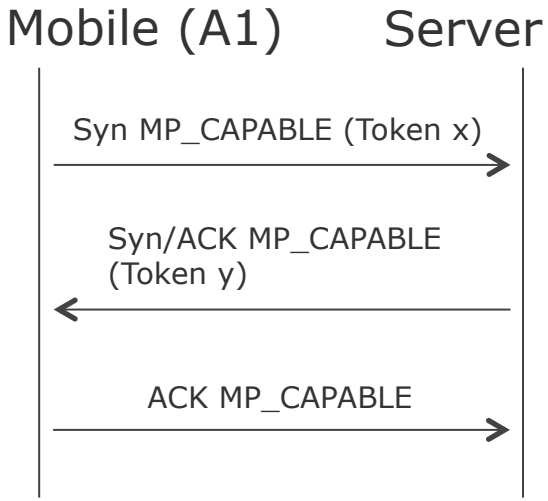


- MPTCP connection establishment uses TCP mechanisms
- MPTCP identifies multiple paths based on a host's different IP addresses
- In case of multiple paths: Creation of "subflows"
  - Subflows corresponds to TCP connections
  - Subflows are mapped to a single connection
- Different sequence number spaces for
  - Connections
  - Subflows

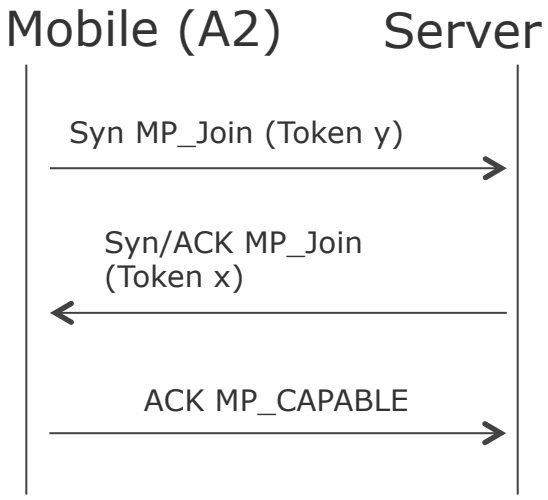
Basic Protocol Operations (1)



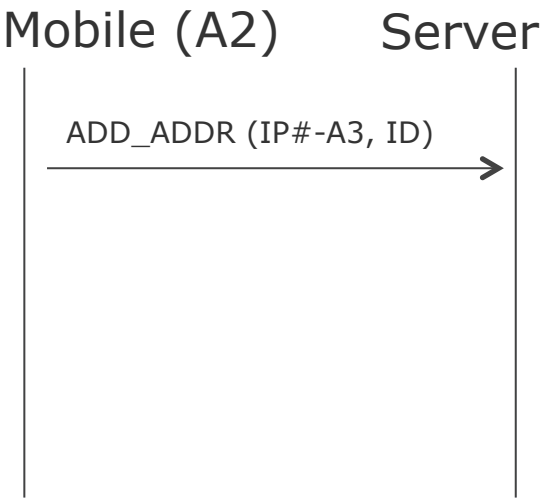
Connection Initiation



Starting a New Subflow



Advertisement of Another Address



Basic Protocol Operations (2)

Transmit Data

- Subflows have their own 32 bit sequence number
- Connections are identified based on 64 bit sequence number
- Send transmits TCP segments with DSN_MAP
  - Implements mapping between data sequence number and subflows
  - Receiver can assemble packet order

Acknowledgement (two types)

1. Regular TCP ACK confirms segments per subflow
2. DSN_ACK implements cumulative ACK per data sequence
   - Prevents deadlock in case of proactive ACKs from middleboxes

Congestion Control

- New calculation of congestion window
- Background: Fairness for TCP in case of a joint bottleneck

The Success of MPTCP (Until Now)

IETF History

- Mark Handley gives talk at IETF 72 (August 2008) about "Multipath Transport, Resource Pooling, and implications for Routing"
- Multipath TCP working group started 2009
- First RFCs came out 2011

Implementations

- (Almost) Complete implementation Linux (kernel and userland)
  - https://github.com/multipath-tcp
- Less complete FreeBSD implementation
- Commercial implementation: Anon and Citrix
- Android port (from CST research group ;)
- iOS 7 implements MPTCP (mid 2013)

MPTCP in action: watch
http://youtu.be/VWN0ctPi5cw

- MPTCP is a good example where research(ers) advances the Internet

CONTENT of this CHAPTER

- Fundamental Goals of the Transport Layer
- Concepts for the Transport Layer
- Standard Transport Layer Protocols
  - UDP
  - TCP
  - MPTCP
  - SCTP
  - DCCC

Stream Control Transmission Protocol (SCTP)

- Problems with TCP
  - TCP is too strict with in-order delivery (delays with head-of-line-blocking)
    - Not appropriate for some applications, such as signaling (e.g. SS7)
    - Degrades user experience in some other applications such as web browsing
  - Limitations of sockets wrt. highly-available data transfer using multi-homed hosts
  - Vulnerability of TCP wrt. DoS attacks such as SYN flooding
- Solution: SCTP
  - Reliable, connection-oriented transport protocol, message oriented
  - Multi-streaming and multi-homing support, heartbeats, 4-way-handshake
  - TCP friendly, more resistance in face of flooding/masquerade attacks
- Specifications
  - RFC 4960 (was 2960, updated by 3309, 6096, 6335)
    - Specification of the protocol, packet formats, options, features
  - RFC 3286
    - High-level introduction to SCTP
  - And some more, like RFC 5062 Security Attacks against SCTP…

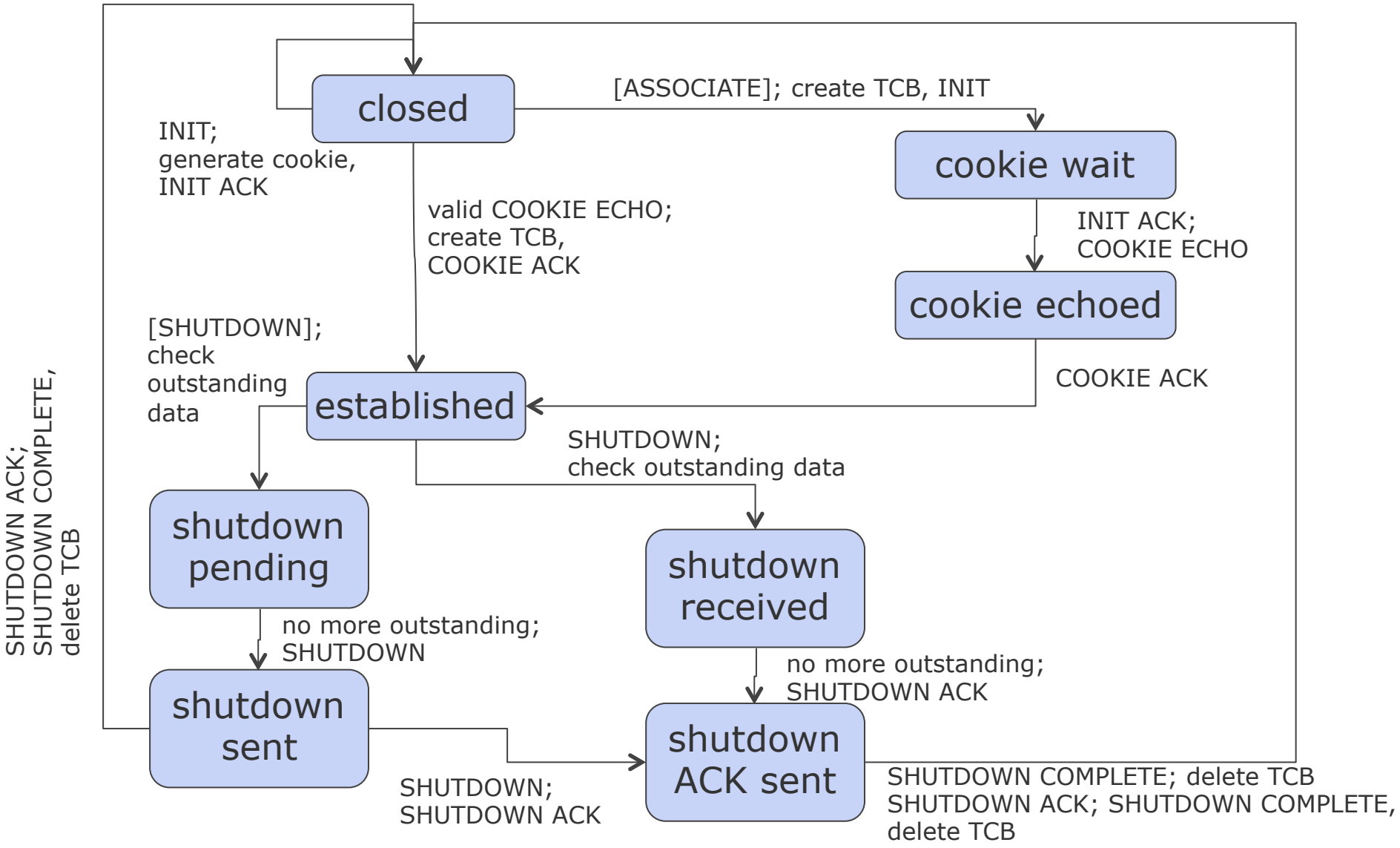
SCTP Multi-Streaming

- Allows partitioning of data into multiple streams within a single *association*
- Streams are independent wrt. sequenced delivery
  - i.e. loss in any one stream will only affect delivery within that stream
- Allows for partial ordering of e.g. objects of a web page
  - Each object is assigned to a stream, order of object delivery irrelevant
    - Even if an object does not get through, the others can go through. Better user experience.
  - However, all streams are subjected to a common flow and congestion control mechanism
- Two sequence numbers used
  - Transmission Sequence Number
    - Governs transmission of messages, detection of message loss
  - Stream Identification/Stream Sequence Number
    - Determines the sequence of delivery of received data
  - Goal: separate streams are affected by message loss from other streams

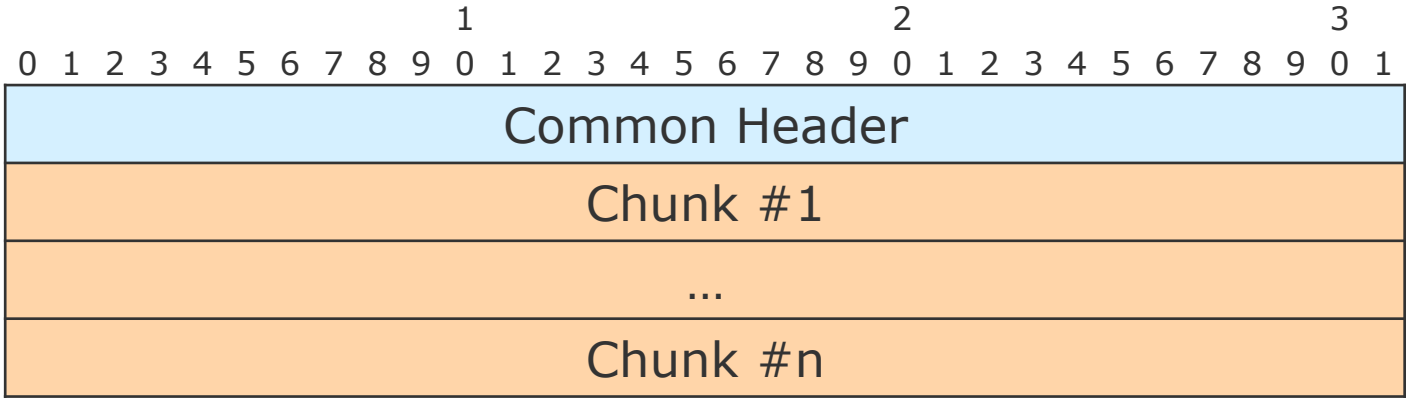
SCTP Multi-Homing

- Ability for a single endpoint to support multiple IP addresses
- Used for redundancy (not for load sharing up to now)
  - e.g. laptop could be connected via LAN, WLAN and UMTS to the Internet
  - Failure of one network will not cause a failure of the association
- Principle: one address chosen as primary IP address
  - Primary address is the destination for all DATA chunks for normal transmission
  - Retransmitted DATA chunks use alternate address(es)
  - Continued failure of primary address results in transmitting all DATA chunks to alternate address (until heartbeats can reestablish primary address)

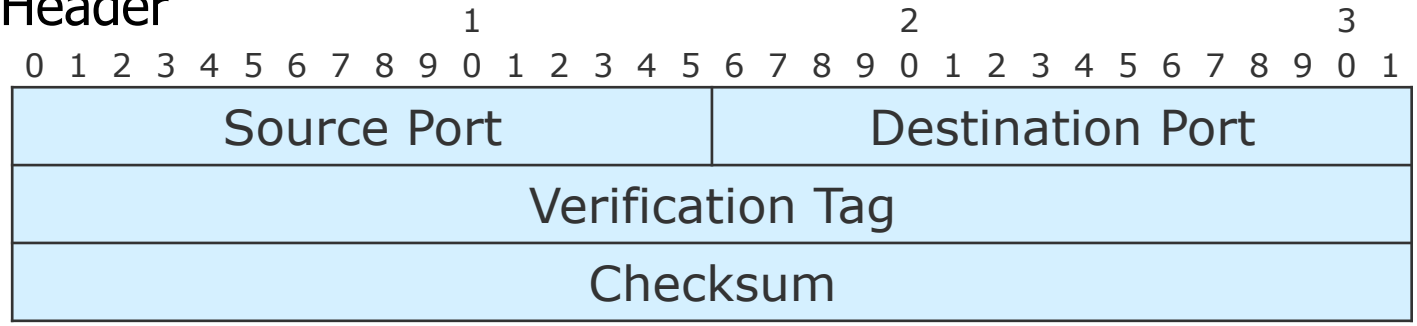
SCTP State Diagram (without timers, ABORT)



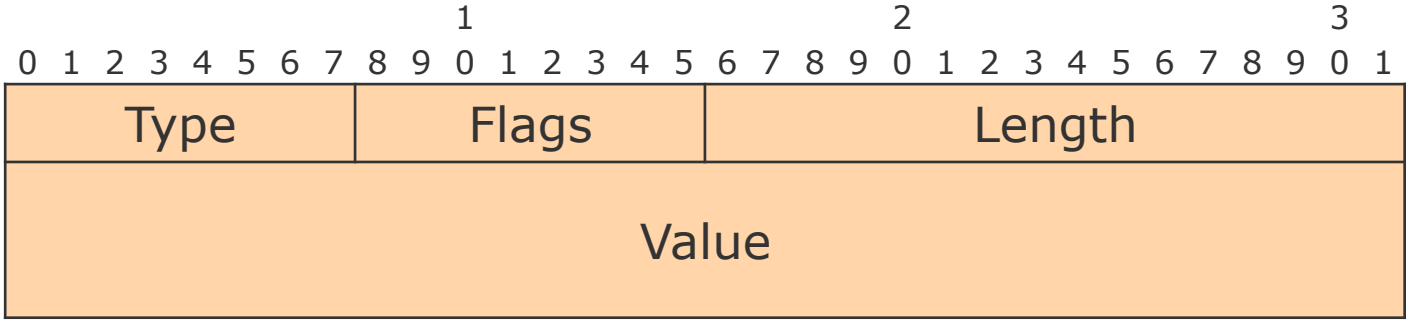
SCTP Packet Format



- Common Header



- Chunk



SCTP Common Header

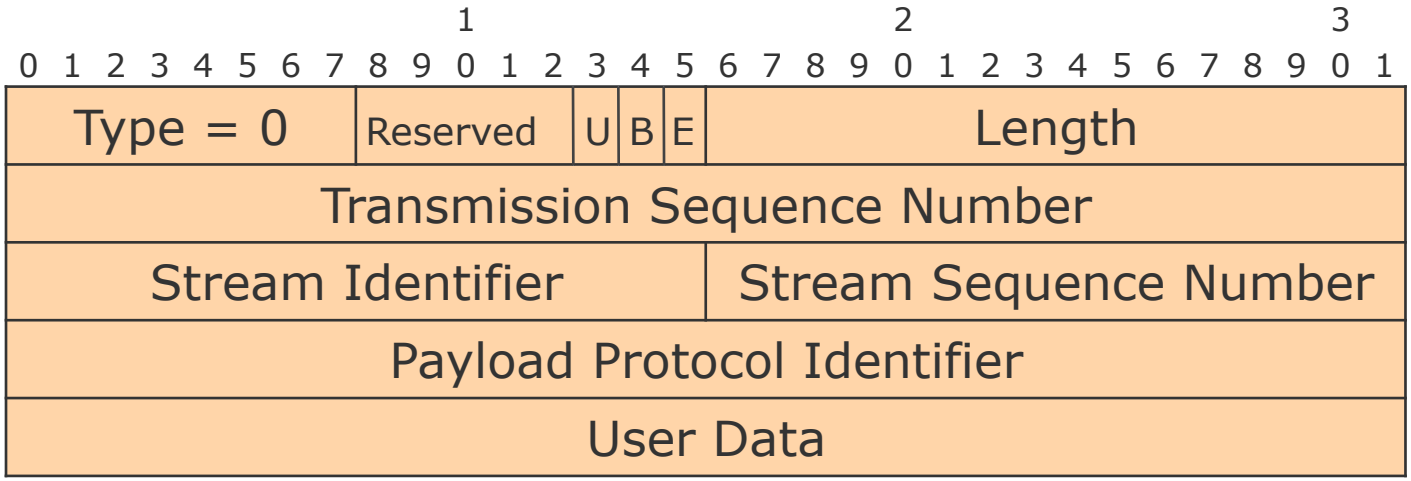
- Ports
  - 16 bit as used in TCP, UDP
- Verification Tag
  - Validation of the sender, per association
  - set to the value of the Initiate Tag received during association initialization
    - random number generated once at association initialization by each end-point
  - Set to 0 in packets with INIT chunk
  - See RFC for use during SHUTDOWN and ABORT
- Checksum
  - CRC32c checksum algorithm using the polynomial $x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1$
  - Calculated over the whole SCTP packet including common header (with checksum initialized to 0) and all chunks
  - Invalid packets are typically silently discarded

SCTP Chunks

- TLV format: Type-Length-Value
  - And flags that depend on type
- Type (Examples)
  - 0: Payload Data (DATA)
  - 1: Initiation (INIT)
  - 2: Initiation Acknowledgement (INIT ACK)
  - 3: Selective Acknowledgement (SACK)
  - 4: Heartbeat Request (HEARTBEAT)
  - 5: Heartbeat Acknowledgement (HEARTBEAT ACK)
  - 6: Abort (ABORT)
  - 7: Shutdown (SHUTDOWN)
  - 8: Shutdown Acknowledgement (SHUTDOWN ACK)
  - 9: Operation Error (ERROR)
  - 10: State Cookie (COOKIE ECHO)
  - 11: Cookie Acknowledgement (COOKIE ACK)
  - 12: Reserved for Explicit Congestion Notification Echo (ECNE)
  - 13: Reserved for Congestion Window Reduced (CWR)
  - 14: Shutdown Complete (SHUTDOWN COMPLETE)
- Length
  - Overall length of chunk in bytes
- Value
  - Actual content of the chunk

SCTP DATA Chunk Flags

- Example of chunk type 0 (Data)

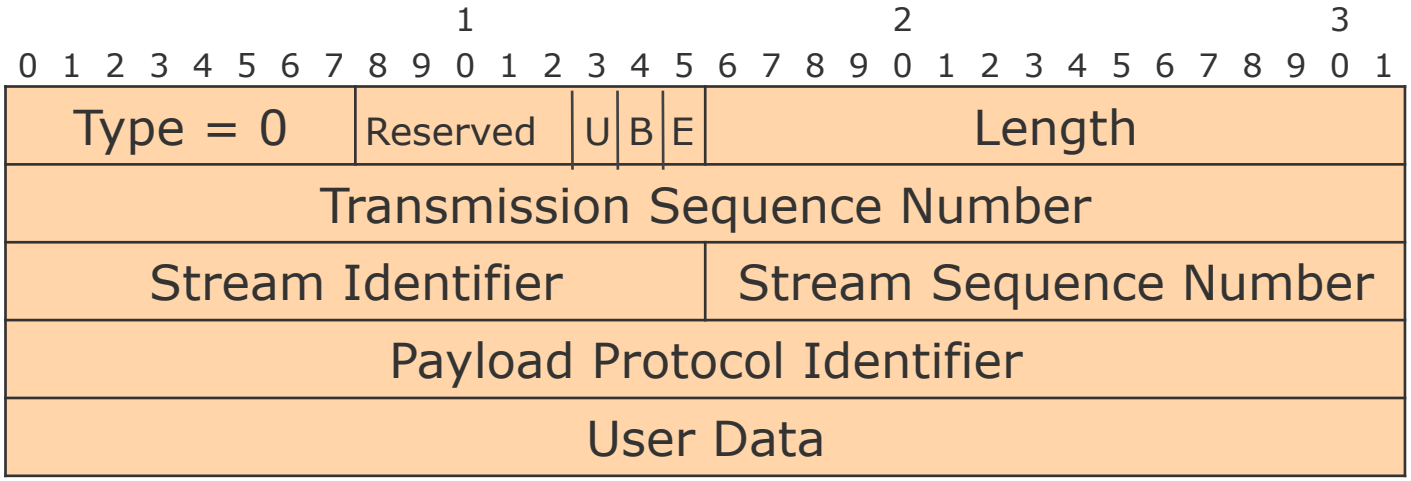


- U: Indicates unordered DATA chunk, ignore Stream Sequence Number
- B and E flags indicate :

B	E	
1	0	first fragment of a user message
0	0	middle fragment of a user message
0	1	last fragment of a user message
1	1	Unfragmented user message

SCTP DATA Chunk

- Transmission Sequence Number
  - Sequence number for this DATA chunk
- Stream identifier
  - Identification of data stream to which the following data belongs
- Stream Sequence Number (SSN)
  - Sequence number of the following user data within the stream
  - When a user message is fragmented all fragments must carry the same SSN
- Payload Protocol Identifier
  - Identifies application layer protocol (used by end or intermediate systems)



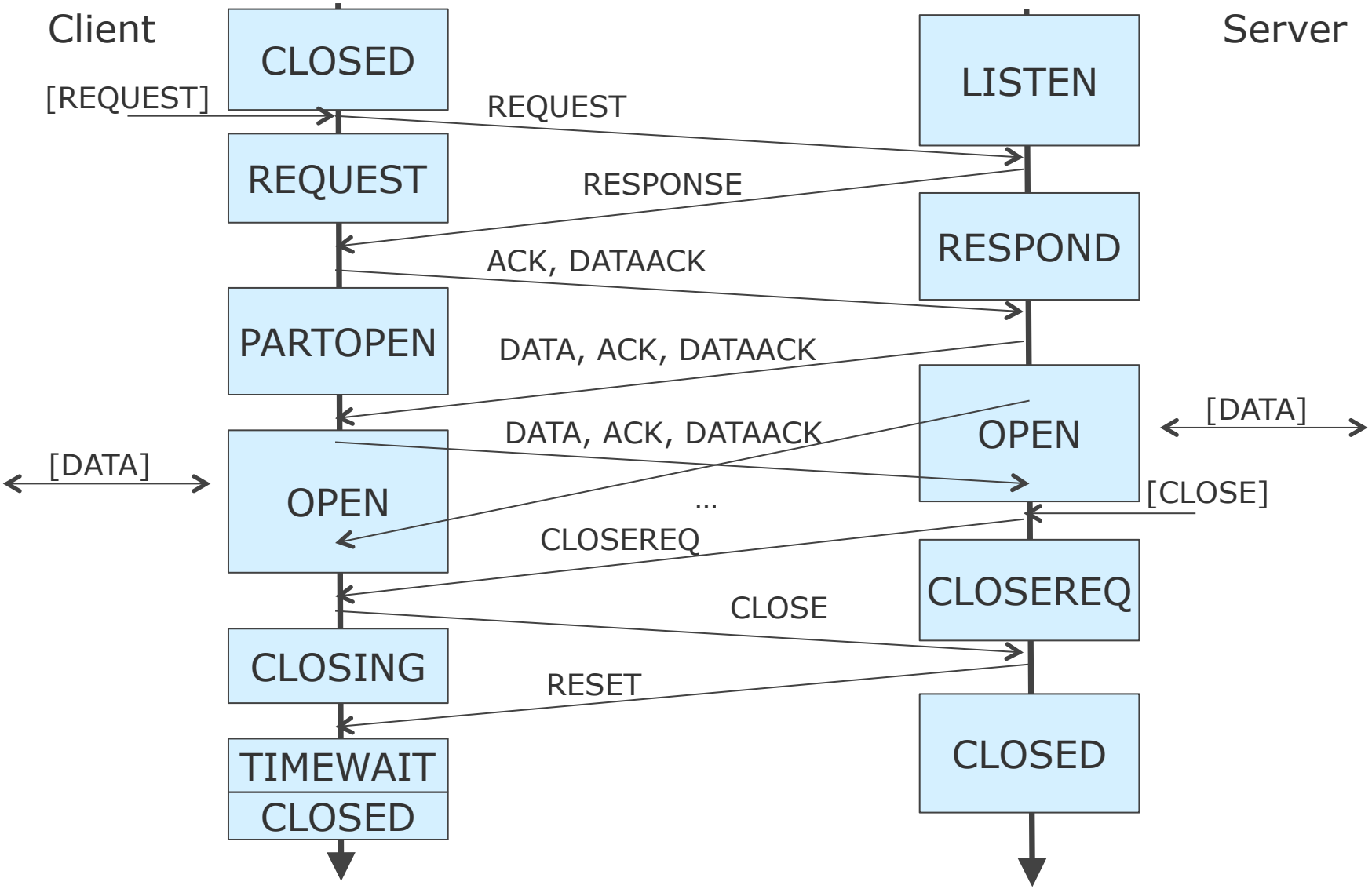
CONTENT of this CHAPTER

- Fundamental Goals of the Transport Layer
- Concepts for the Transport Layer
- Standard Transport Layer Protocols
  - UDP
  - TCP
  - MPTCP
  - SCTP
  - DCCC

Datagram Congestion Control Protocol (DCCP)

- Problem:
  - rapid growth of applications using UDP
    - streaming media, on-line games, internet telephony
    - apps preferring timeliness instead of reliability or short start-up delay
  - non TCP-friendly behavior of UDP poses threat to the health of the Internet
- Requirements:
  - clean, understandable, low-overhead, minimal protocol => like UDP
  - TCP-friendliness => like TCP
- Solution: DCCP
  - unreliable flow of datagrams with end-to-end congestion control
  - also: simpler firewall traversal, parameter negotiation
- Spefications
  - RFC 4340, updated by 5595, 5596, 6335
  - RFC 4336: Problem Statement for the Datagram Congestion Control Protocol

DCCP Time Sequence Diagram (Incomplete Example)

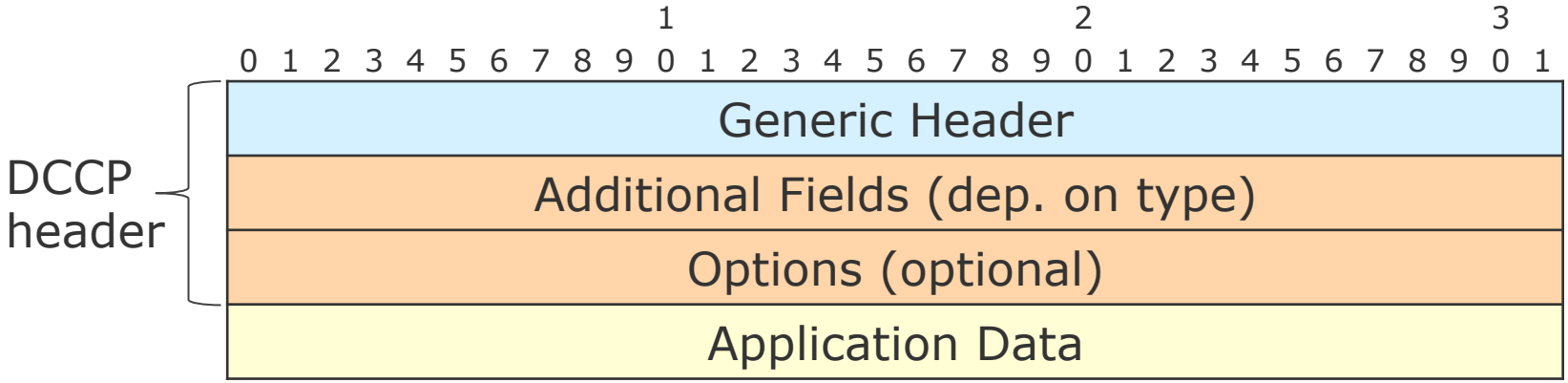


DCCP vs. TCP/UDP

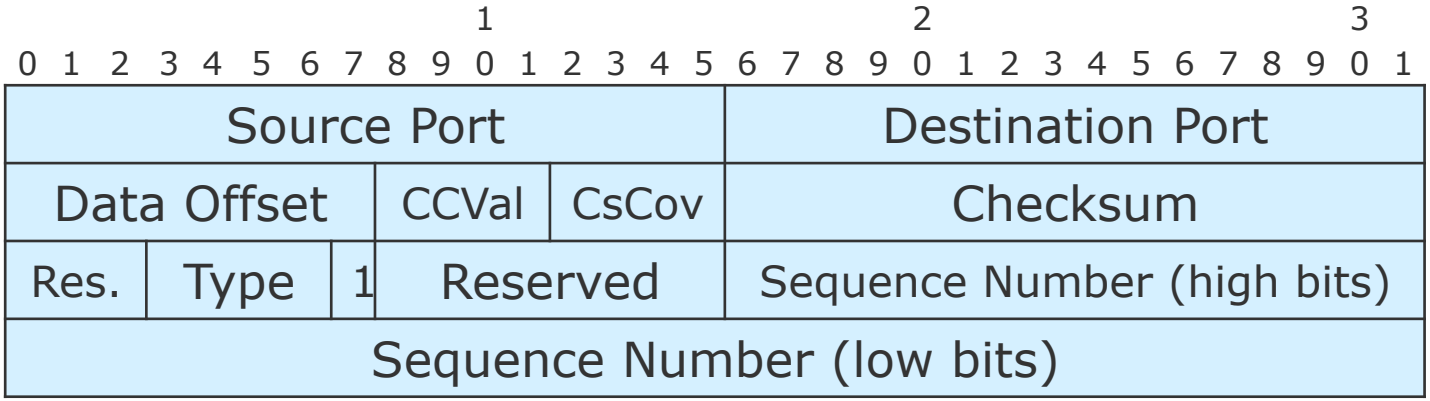
- Very roughly:
  - DCCP = TCP – bytestream semantics - reliability
  - DCCP = UDP + congestion control + handshake + acknowledgements

- DCCP specific features:
  - Choice of congestion control mechanisms per half-connection (eg TCP-like, TCP-friendly Rate Control etc.)
  - Feature negotiation mechanism
  - Per packet sequence numbers
  - Different ACK formats (Still: no reliability! Up to the app to decide!)
  - DoS protection (e.g. cookies against flooding)
  - Distinguishing of different kinds of loss
    - Corruption, receiver buffer overflow…
  - No receive window, simultaneous open, half-closed states…
  - …

DCCP Packet Format



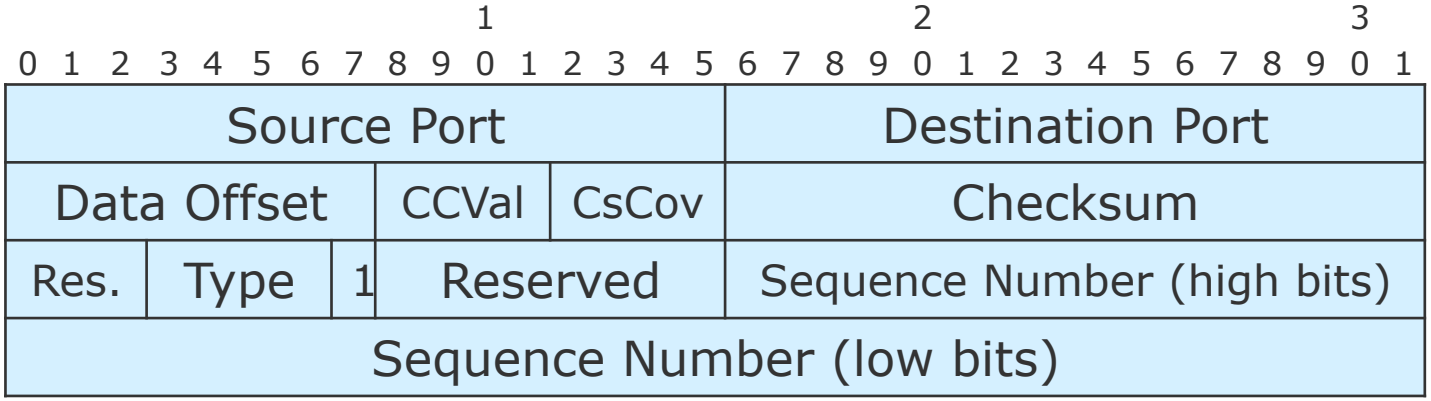
- Generic Header
  - 16 byte version, increased protection against wrapping sequence numbers



- There is also a shorter 12 byte Generic Header

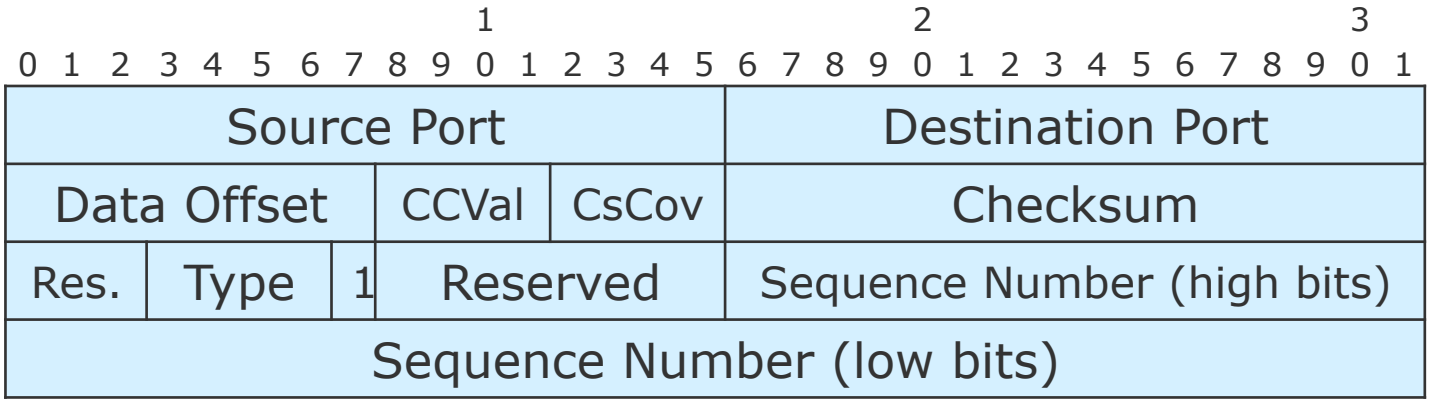
DCCP Generic Header

- Ports
  - 16 bit as used in TCP, UDP
- Data Offset
  - Start of application data area in 32 bit words
- CCVal
  - Determines congestion control method
- CsCov
  - Determines the checksum coverage (header and options always included)
- Checksum
  - Based on TCP/IP checksum algorithm
  - Covers all, some, or none of application data depending on CsCov

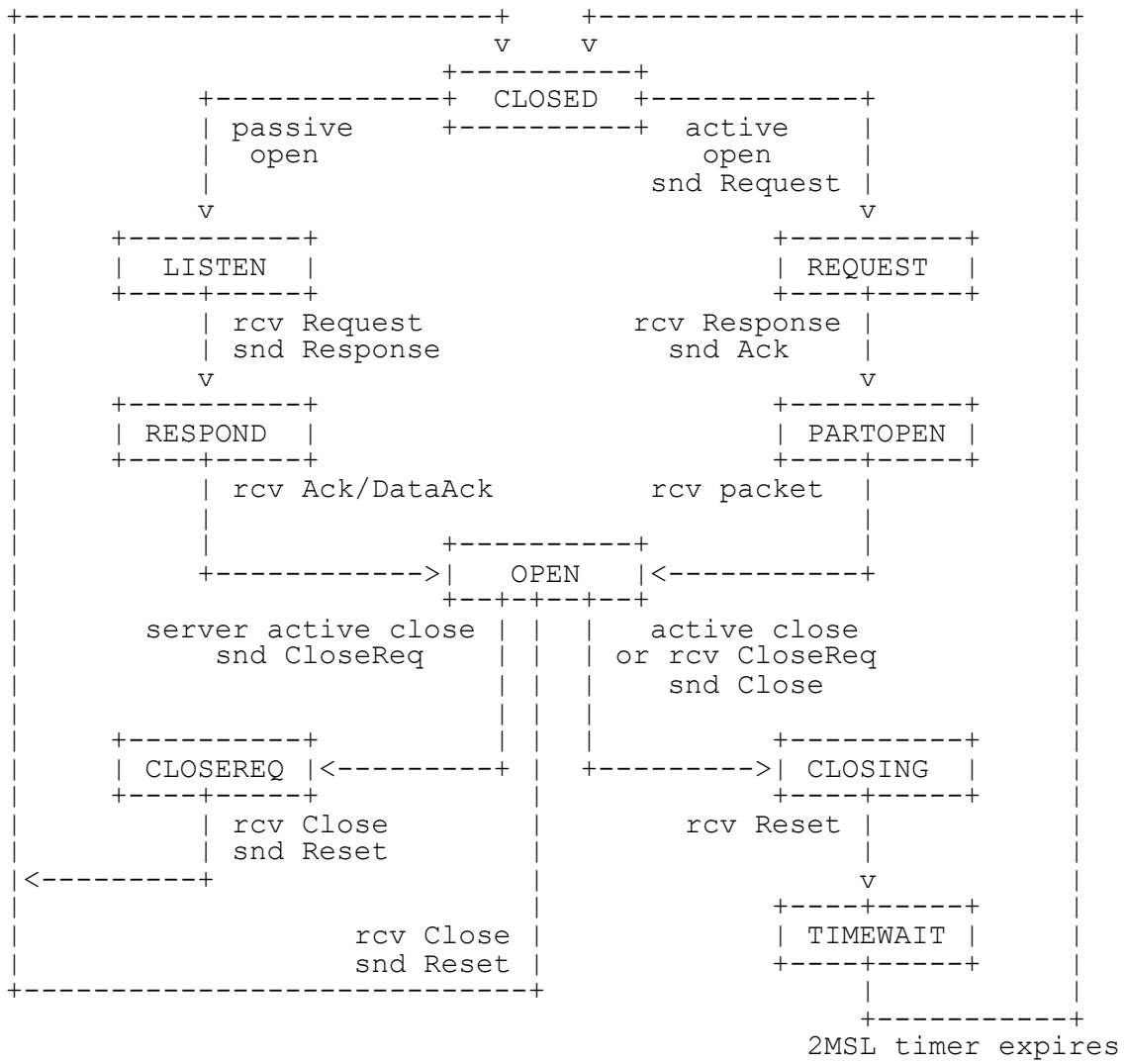


DCCP Generic Header

- Reserved/Res
  - All bits set to 0
- Type
  - Type of packet: REQUEST, RESPONSE, DATA, ACK, DATAACK, CLOSEREQ, CLOSE, RESET, SYNC, SYNCACK, reserved (10-15)
- Sequence Number
  - 48 bit (extended sequence number as shown) or 24 bit
  - Counts every packet, including ACKs



DCCP State Diagram



Transport Layer Summary

- The TCP/IP reference model has only two protocols on the transport layer
  - UDP for connectionless, unreliable, but lightweight protocol
  - TCP for connection oriented and reliable communication, but really complex
    - Connection establishment
    - Flow control
    - Congestion control
    - Connection termination
    - Fairness
- New applications motivate new transport layer protocols
  - MPTCP to leverage multipath
  - SCTP to support signaling messages (and support multi-stream/multi-home too)
  - DCCP offers unreliable datagram service, but with congestion avoidance
- Many more protocols exist – but will they be used?
  - RUDP (Reliable UDP), UDP Lite, …
  - See http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml for some more as IPv4 (protocol) and IPv6 (next header) point also to layer 4 protocols