

Transport Protocols

ITS323: Introduction to Data Communications

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 23 May 2012
ITS323Y12S1L13, Steve/Courses/2012/s1/its323/lectures/transport.tex, r2334

Contents

Transport Protocols

User Datagram Protocol

Transmission Control Protocol

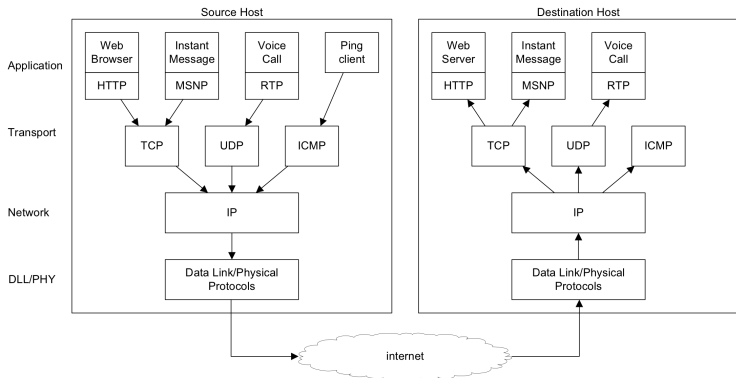
Transport Protocols

- ▶ Send data between application processes on source and destination hosts
- ▶ End-to-end (or host-to-host) communications
- ▶ Internet transport protocols: TCP, UDP, SCTP, DCCP (and other legacy or domain-specific protocols)
- ▶ **Transmission Control Protocol**: connection-oriented, error control, flow control, congestion control
- ▶ **User Datagram Protocol**: unreliable connection-less delivery (same as IP)
- ▶ Addressing is common to all transport protocols (TCP, UDP, SCTP, DCCP)
 - ▶ **Port** is abstract view of end-point for communications; actual end-point is process

Internet Applications

- ▶ Most Internet applications follow a client/server model of initiating communication:
 1. Client initiates communication
 2. Server waits for client to initiate communication
 3. Once the communication is initiated, data can flow in both directions (client to server and server to client)
- ▶ For client to initiate communication to server, the client needs to know IP address of server, and:
 - ▶ **Protocol number**: identifies transport protocol used by both hosts
 - ▶ 8-bit number; e.g. 6 = TCP, 17 = UDP; 1 = ICMP
<http://www.iana.org/assignments/protocol-numbers/>
 - ▶ **Port number**: identifies application process on a host
 - ▶ 16-bit number; 0–1023 well-known ports; 1024–49151 registered ports; 49152 dynamic/private ports
<http://www.iana.org/assignments/port-numbers/>

Multiple Applications, Multiple Transport Protocols



Contents

Transport Protocols

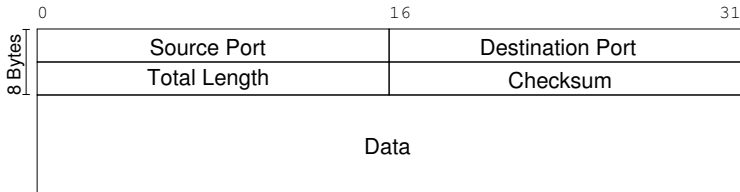
User Datagram Protocol

Transmission Control Protocol

User Datagram Protocol

- ▶ UDP is a unreliable connection-less transport protocol
- ▶ Takes Data from the application layer, attaches a UDP header, and delivers to IP
- ▶ UDP provides checksum over the packet
- ▶ UDP segments may be: lost, arrive out of order, duplicated, arrive in error (application that uses UDP must deal with this)
- ▶ UDP is simple (standard describes it in 4 pages)
- ▶ UDP is used by applications which:
 - ▶ Require simplicity, e.g. TFTP, network management in embedded devices
 - ▶ Don't require reliability, e.g. voice and video applications, network management
 - ▶ Require low overheads, e.g. voice and video applications (require low delay)

UDP Segment



- ▶ 8 Byte header plus data
- ▶ Length is count of bytes in header and data
- ▶ Checksum calculated over UDP header, UDP data, some parts of IP header (e.g. addresses, protocol)

Contents

Transport Protocols

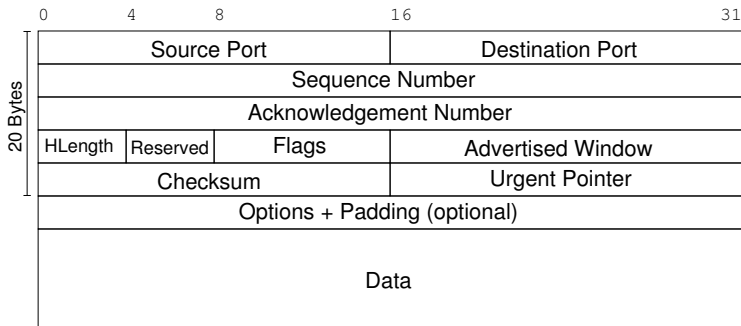
User Datagram Protocol

Transmission Control Protocol

Transmission Control Protocol

- ▶ Most commonly used transport protocol today
 - ▶ Web browsing, email, file sharing, instant messaging, file transfer, database access, proprietary business applications, some multimedia applications (at least for control purposes), . . .
- ▶ Services provided by TCP:
 - ▶ Stream-oriented: TCP treats data from application as continuous stream of bytes
 - ▶ Connection-oriented data transfer
 - ▶ Buffered transfer
 - ▶ Full duplex connection
 - ▶ Reliability (error control)
 - ▶ Flow control
 - ▶ Congestion control

TCP Segment



- ▶ Header contains 20 bytes, plus optional fields
- ▶ Optional fields must be padded out to multiple of 4 bytes

TCP Segment Fields

- ▶ Source/Destination port
- ▶ Sequence number of the first data byte in this segment (or ISN)
- ▶ Acknowledgement number: sequence number of the next data byte TCP expects to receive
- ▶ Header Length: Size of header (measured in 4 bytes)
- ▶ Window: number of bytes the receiver is willing to accept (for flow control)
- ▶ Checksum: error detection on TCP segment
- ▶ Urgent pointer points to the sequence number of the last byte of urgent data in the segment
- ▶ Options: such as maximum segment size, window scaling, selective acknowledgement, ...

TCP Segment Flags

- ▶ Flags (1 bit each, if 1 the flag is true or on):
- ▶ CWR: Congestion Window Reduced
- ▶ ECE: Explicit Congestion Notification Echo
- ▶ URG: segment carries urgent data, use the urgent pointer field; receiver should notify application program of urgent data as soon as possible
- ▶ ACK: segment carries ACK, use the ACK field
- ▶ PSH: push function
- ▶ RST: reset the connection
- ▶ SYN: synchronise the sequence numbers
- ▶ FIN: no more data from sender

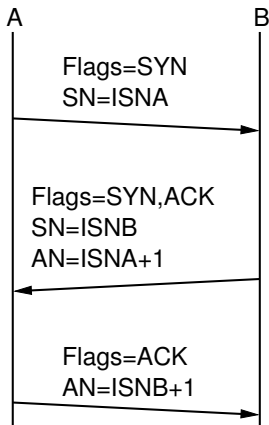
TCP Sequence Numbers

- ▶ TCP uses 32-bit sequence numbers to keep track of data sent and received (error control, flow control, congestion control)
- ▶ Upon connection establishment, **Initial Sequence Numbers** are chosen by each side
- ▶ ISN does not have to be 0; usually larger than last sequence number used in previous connection
- ▶ Sequence numbers used in each direction are independent
- ▶ Each Byte of data has a sequence number relative to the ISN

TCP Connection Establishment

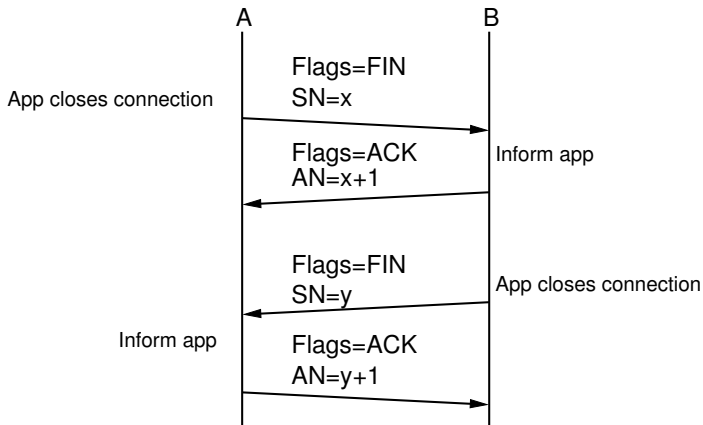
- ▶ What is the purpose of connection establishment?
 - ▶ Allows each end to assure that the other exists
 - ▶ Allows exchange or negotiation of optional parameters, specifically synchronise initial sequence numbers
 - ▶ Triggers the allocation of resources for the connection (e.g. allocate buffer space in memory)
- ▶ TCP uses a **three-way handshake**: SYN-SYN/ACK-ACK
- ▶ (A two-way handshake: duplicate segments from old connections may cause errors)

TCP Three-Way Handshake



- ▶ Initiator A selects an **Initial Sequence Number**, *ISNA*
- ▶ B acknowledges *ISNA* and also chooses its own *ISNB*
- ▶ Data transfer can start after *ISNB* is ACKed
- ▶ Optionally, 3rd segment can contain data

TCP Connection Close

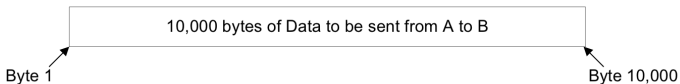


- ▶ Both sides should independently close the connection
- ▶ E.g. A closes connection, B can still send data to A
- ▶ Other approaches are possible, e.g. abort using RST flag

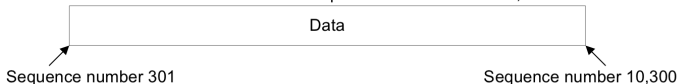
TCP Data Transfer

- ▶ TCP uses sliding window mechanism for efficient data transfer
- ▶ TCP specifies algorithms for error, flow and congestion control
- ▶ Some details are implementation dependent
- ▶ E.g. when to send data? when to send an ACK? when is data delivered to application?
- ▶ TCP offers stream-oriented data transfer service
 - ▶ Application delivers multiple messages to TCP during connection
 - ▶ TCP treats all messages as continuous stream of bytes
 - ▶ TCP implementation chooses when to send the bytes in a segment

TCP Segment Example



Since the Initial Sequence Number is 300, the bytes of Data to be sent can be seen as sequence number 301 to 10,300



TCP decides to send this data as 2 x 1000 byte blocks plus
2 x 4000 byte blocks of data

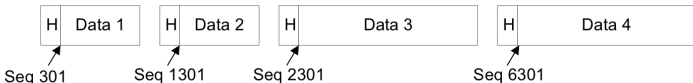
Data 1 1000 B	Data 2 1000 B	Data 3 4000 B	Data 4 4000 B
------------------	------------------	------------------	------------------

TCP Segment Example (continued)

TCP decides to send this data as 2 x 1000 byte blocks plus
2 x 4000 byte blocks of data

Data 1 1000 B	Data 2 1000 B	Data 3 4000 B	Data 4 4000 B
------------------	------------------	------------------	------------------

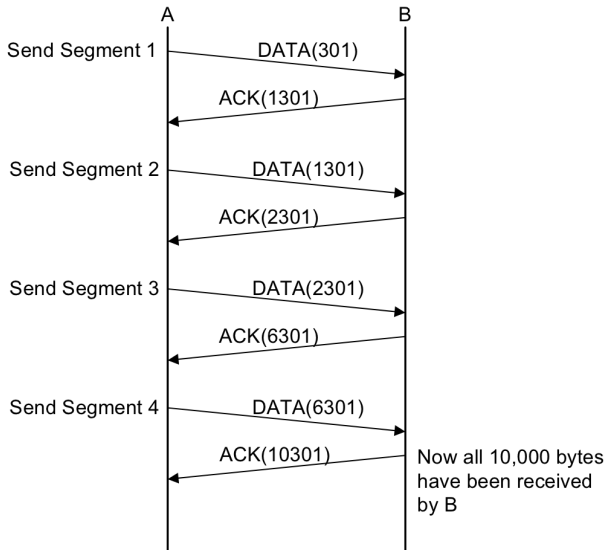
TCP includes a TCP header for each piece of data to make 4 segments
Each header gives the Sequence Number of the first byte of data



TCP sends these segments separately. That is, it sends segment 1, then segment 2
and so on.



TCP Data Transfer Example

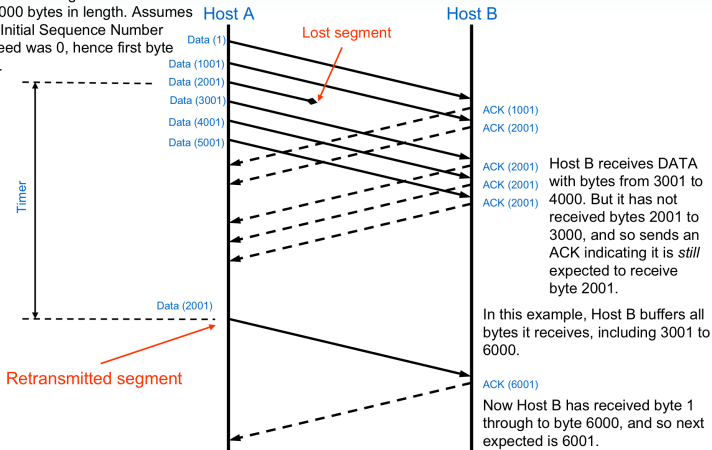


TCP Error Control

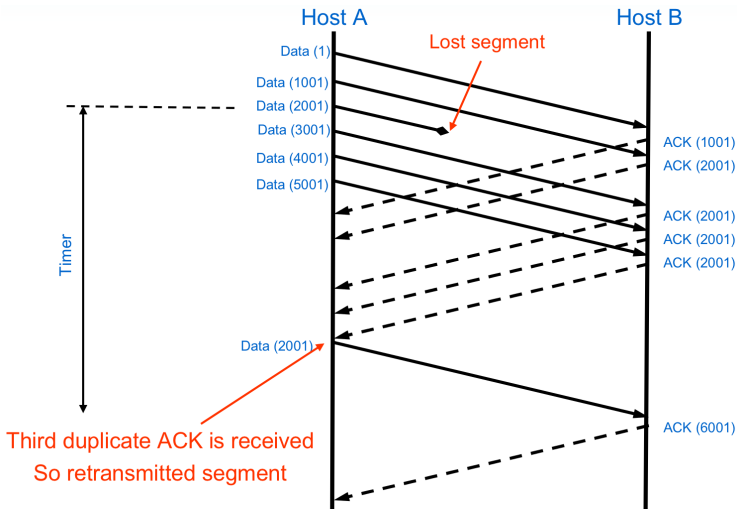
- ▶ TCP uses selective-retransmission style ARQ
- ▶ ACK numbers indicate next in-order sequence number expected
- ▶ Receiver buffers out-of-order data received
- ▶ If data segment sent is not implicitly ACKed within timeout, retransmit segment
- ▶ Cumulative ACKs can be used; ACK number implicitly acknowledges multiple received segments
- ▶ Optimization of normal retransmission: **Fast Retransmit**
 - ▶ Timeouts generally have to be long to handle varying round trip times (RTTs)
 - ▶ Waiting for a timeout may lead to poor efficiency
 - ▶ If 3 duplicate ACKs are received, retransmit segment

TCP Basic Retransmit Example

Host A sending DATA which is 1000 bytes in length. Assumes the Initial Sequence Number agreed was 0, hence first byte is 1.



TCP Fast Retransmit Example



TCP Flow Control

- ▶ Aim: prevent sender from overflowing buffers of receiver
- ▶ Must consider variable end-to-end round trip times (RTT)
- ▶ Uses sliding-window flow control
- ▶ Receiver notifies sender amount of buffer space available in **Advertised Window** field
- ▶ Sender cannot send more than advertised window of bytes

TCP Congestion Control

- ▶ Aim: prevent sender from overflowing the routers
- ▶ Assume segment loss is indicator of congestion
- ▶ Decrease sending rate if segment loss detected
- ▶ Increase sending rate if ACK received
- ▶ Various algorithms/options for how to decrease/increase sending rate