



Internet transport protocols

188lecture6.ppt

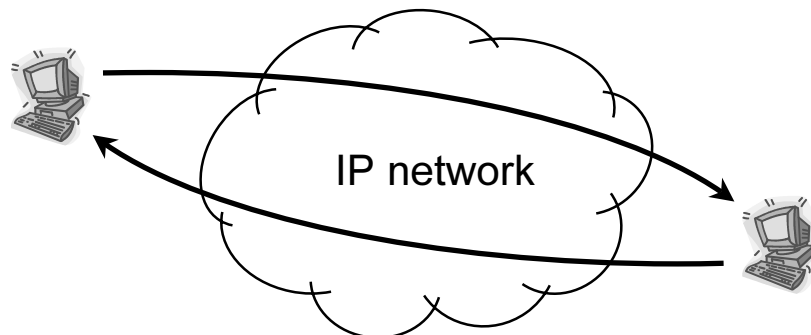
© Pasi Lassila

1

S-38.188 - Computer Networks - Spring 2004

Problem

- IP can be used to connect together heterogenous networks
 - IP network offers only best effort packet delivery (with no guarantees)
- Applications need end-to-end (process-to-process) communication
 - "logical channels" through the network
 - ⇒ **transport layer** protocols on top of **IP layer**



2

Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

3

Requirements of an end-to-end protocol (1)

- An end-to-end transport protocol is shaped
 - from above, by application requirements, and
 - from below, by limited capabilities of the network layer
- Common application requirements on transport protocols
 - guarantee message delivery
 - deliver messages in the same order they are sent
 - deliver at most one copy of each message
 - support arbitrarily large messages
 - support synchronization
 - allow the receiver to flow control the sender
 - support multiple application processes on each host
- Note(!), security is not in list above
 - implemented above transport layer

4

Requirements of an end-to-end protocol (2)

- **Best-effort networks (Internet) have limited capabilities and can**
 - drop messages
 - re-order messages
 - deliver duplicate copies of a given message
 - limit messages to some finite size
 - deliver messages after an arbitrarily long delay
- **Challenge:**
 - to develop protocols that use best-effort network, but can provide high (sufficient) level of service

5

Internet transport protocols

- **Traditional IP transport protocols**
 - UDP: simple multiplexing/demultiplexing
 - TCP: reliable byte stream
 - covered in this course
- **New/emerging IP transport protocols**
 - SCTP: reliable message transport protocol
 - briefly covered at the end of lecture
 - DCP: (proposed) transport protocol for streaming media
 - “TCP-friendly” congestion control but without retransmissions

6

Outline

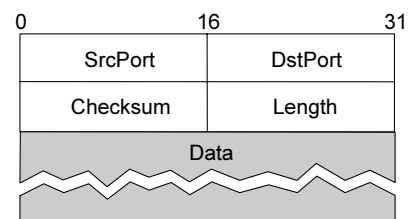
- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

7

Simple demultiplexor (UDP)

- UDP = User Datagram Protocol
 - used in real time services, question-reply protocols
 - traditionally not much UDP traffic in Internet
 - increasing amount of real time services/applications \Rightarrow more UDP traffic
- Basic features
 - unreliable and unordered datagram service
 - only adds multiplexing to best-effort
 - greedy: no flow or congestion control
- Endpoints identified by ports
 - servers have well-known ports
 - ex. DNS uses port 53 etc.
 - port only 16 bits (hostwide)
 - server's full address (IP addr, port nr)
- Optional checksum
 - pseudo header + UDP header + data

UDP header format



8

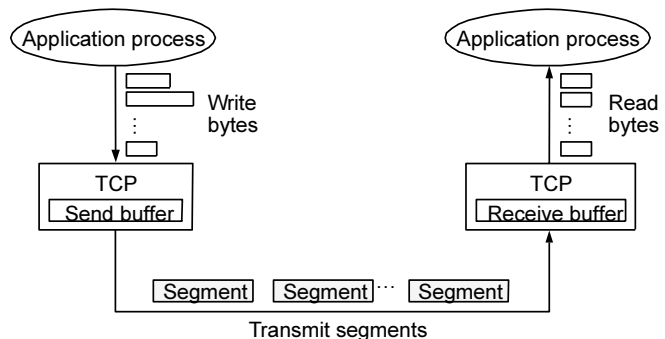
Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

9

TCP overview

- Connection-oriented
- Byte-stream
 - application writes bytes
 - TCP sends segments
 - application reads bytes
- Full duplex (pair of byte streams)



- Traffic management
 - Flow control:
 - keep sender from overrunning receiver
 - Congestion control:
 - keep sender from overrunning network
- Protocols that use TCP
 - majority of Internet traffic still generated by TCP
 - Telnet, FTP, Simple Mail Transfer Protocol (SMTP), POP (reading of e-mails), IMAP (reading of e-mails), HTTP, X Window System (X11, decentralized window system)

10

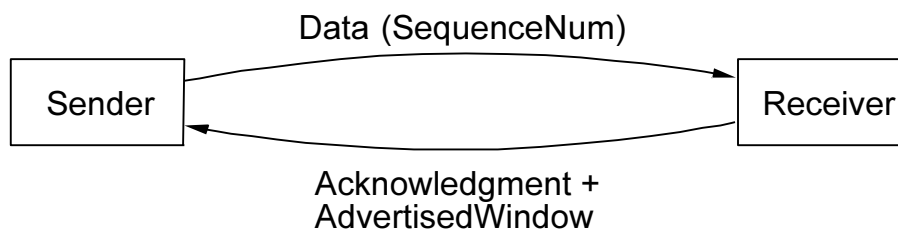
End-to-end issues

- TCP implements a sliding window protocol
 - similar to the one covered in lecture 2 for point-to-point links
- Issues that complicate design of an end-to-end sliding window protocol
 1. Connects many different hosts/applications
 - needs explicit connection establishment and termination
 2. Different RTTs
 - needs adaptive timeout mechanism (variations in RTTs)
 3. Long delays in network (packets reordered)
 - each TCP packet has **Maximum Segment Lifetime** (MSL, 120 s)
 - need to be prepared for arrival of very old packets
 4. Different/varying capacity at destination (delay x bandwidth)
 - accommodate very different node capacities
 - for a given node, amount of resources (buffer space) available changes with number of simultaneous TCP connections
 5. Different/varying network capacity (TCP has no link info)
 - need to be prepared for network congestion

11

Sending data

- TCP byte oriented
 - sender application writes and receiver application reads bytes
- ... but still TCP sends segments
 - MSS = Maximum Segment Size
- 3 mechanisms to trigger segment transmission
 - send segment once MSS bytes received
 - sender invoked operation (push)
 - timer that periodically fires
- Segment transmission:
 - data has sequence number, receiver acknowledges data and includes info on current buffer space (AdvertisedWindow)



12

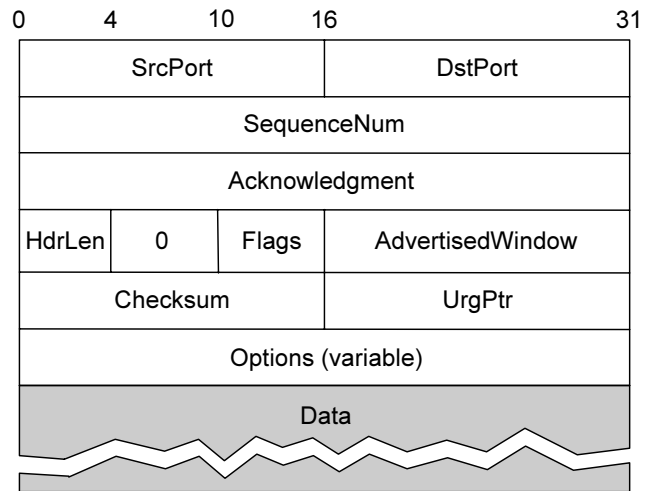
Segment format

- Unique identification
 - <SrcPort, SrcIPAddr, DstPort, DstIPAddr>

- For sliding window algorithm
 - SequenceNum: number of first byte carried
 - Acknowledgment & AdvertisedWindow: info on flow on reverse direction

- 6-bit Flags: control info btw. TCP peers
 - Syn, Fin: establish & terminate TCP connection
 - Ack: if Acknowledgment is valid
 - Urg: urgent data
 - Push: sender invoked push operation
 - Reset: receiver is confused

- Checksum:
 - TCP header, TCP data, pseudoheader



13

Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

14

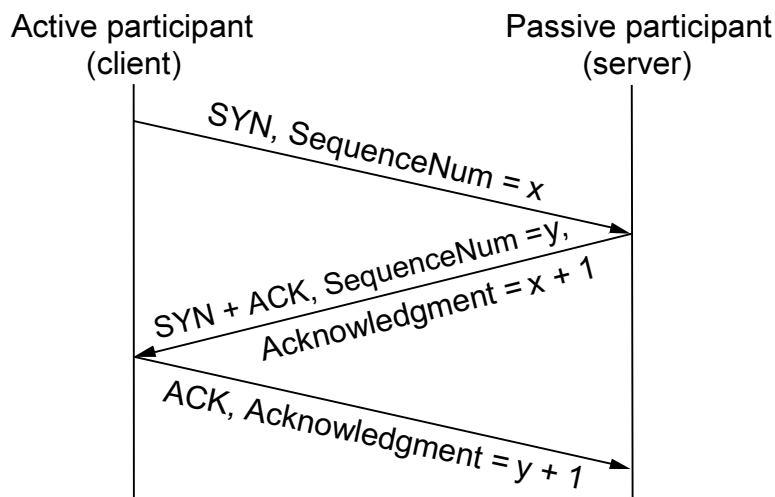
Connection establishment

- Connection setup:
 - done before any actual data is transmitted
 - asymmetric, active open (caller/client) & passive open (callee/server)
- Connection teardown:
 - symmetric (each side closes independently)
- 3-way handshake
 - algorithm for establishing a connection and connection tear down
 - idea during establishment: to agree on starting sequence numbers
- Why not fixed starting numbers?
 - TCP specification: random sequence numbers
 - protection against two incarnations of the same connection

15

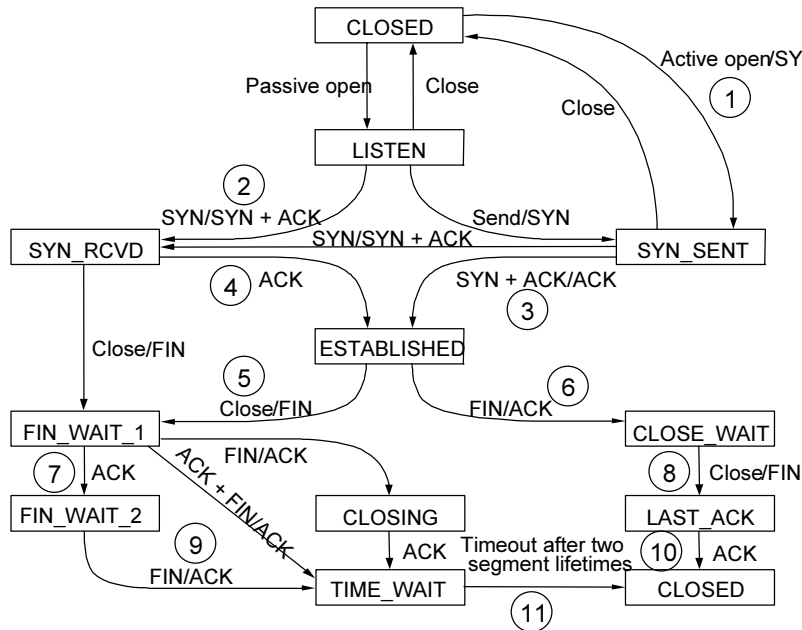
Timeline for 3-way handshake (establishment)

1. Client sends SYN and own initial seq. number x
2. Server responds with SYN+ACK, where own initial seq. number = y , and next expected byte (acknowledgment) = $x+1$
3. Client responds with ACK, where next expected byte (acknowledgment) = $y+1$



16

State transitions (not all, no timeouts)



- Format: event/action
- Connection establishment (above "Established")
 - asymmetric
- Connection tear down (below "Established")
 - symmetric (both sides tear down independently)
- Client initiated establish (C=client, S=server)
 - Server is in state LISTEN
 - (C,1), (S,2), (C,3), (S,4)
- Server initiated tear down
 - (S,5), (C,6), (S,7), (C,8), (S,9), (C,10), (S,11)
 - transition 11 takes 240 s!
 - server does not know if client received last ACK ⇒ must wait max possible time for a retransmitted FIN from client

17

Outline

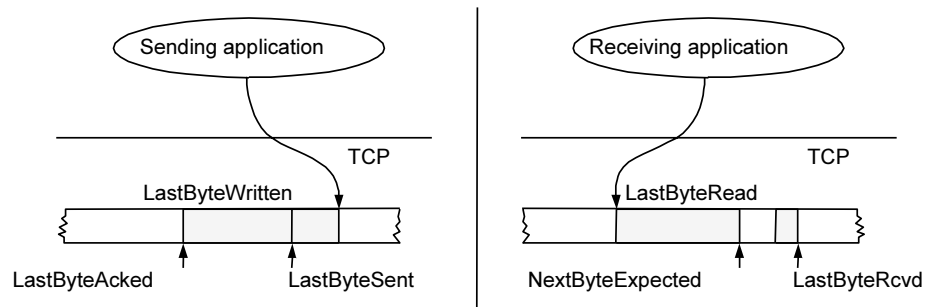
- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

TCP sliding window

- Purposes:
 - guarantees reliable delivery of data
 - ensured that data is delivered in order
 - enforces flow control between sender and receiver
- Receiver advertises a window size to sender
 - idea: prevent sender from overrunning receiver's buffer
- Both sides have (finite) buffers and 3 pointers

19

Sliding window buffers and pointers



- Sending side
 - $\text{LastByteAcked} \leq \text{LastByteSent}$
 - $\text{LastByteSent} \leq \text{LastByteWritten}$
 - buffer bytes between LastByteAcked and LastByteWritten
- Receiving side
 - $\text{LastByteRead} < \text{NextByteExpected}$
 - $\text{NextByteExpected} \leq \text{LastByteRcvd} + 1$
 - buffer bytes between LastByteRead and LastByteRcvd

20

Flow control

- Variables
 - Send buffer size: `MaxSendBuffer`
 - Receive buffer size: `MaxRcvBuffer`
- Receiving side
 - $\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}$
 - $\text{AdvertisedWindow} = \text{MaxRcvBuffer} - (\text{LastByteRcvd} - \text{LastByteRead})$
 - AdvertisedWindow = max amount of buffer space left
 - always send ACK in response to arriving data segment (even when receiving out of order segments \Rightarrow seq.number does not change)
- Sending side
 - $\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$
 - $\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$
 - amount of data that can be sent
 - $\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$
 - block sender if $(\text{LastByteWritten} - \text{LastByteAcked}) + y > \text{MaxSendBuffer}$
- Persist when $\text{AdvertisedWindow} = 0$
 - send periodically probe segments

21

Protection against wrap around

- 32-bit SequenceNum may wrap around
 - during connection's lifetime (not during one RTT)
- fast network \Rightarrow sequence numbers consumed fast
- Bandwidth Time Until Wrap Around

– T1 (1.5 Mbps)	6.4 hours
– Ethernet (10 Mbps)	57 minutes
– T3 (45 Mbps)	13 minutes
– FDDI (100 Mbps)	6 minutes
– STM-1 (155 Mbps)	4 minutes
– STM-4 (622 Mbps)	55 seconds
– STM-8 (1.2 Gbps)	28 seconds
- Protection against wrap around implemented by TCP options

22

Keeping the pipe full

- 16-bit AdvertisedWindow
 - maximum window size = 64 KB
- Delay x bandwidth worth of data can be transmitted, RTT of 100 ms assumed below
 - Bandwidth Delay x Bandwidth Product
 - T1 (1.5 Mbps) 18KB
 - Ethernet (10 Mbps) 122KB
 - T3 (45 Mbps) 549KB
 - FDDI (100 Mbps) 1.2MB
 - STS-3 (155 Mbps) 1.8MB
 - STS-12 (622 Mbps) 7.4MB
 - STS-24 (1.2 Gbps) 14.8MB
- Increased maximum window size through TCP options

23

Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

24

Adaptive retransmission

- Reliable delivery of data \Rightarrow retransmission needed
- ACK not received \Rightarrow timeout
- Timeout depends on RTT
 - in Internet RTT is a random variable (depends on random queuing delays and also routes may change)
- Choosing the right timeout (i.e., RTT):
 - RTT too short: too many retransmissions
 - RTT too long: unnecessary waiting
 - many algorithms to determine RTT

25

Adaptive retransmission (original algorithm)

- Measure SampleRTT for each segment/ACK pair
 - Note! for retransmissions SampleRTT includes also retransmission time
- Compute weighted average of RTT

$$\text{EstRTT} = a \times \text{EstRTT} + (1-a) \times \text{SampleRTT}$$

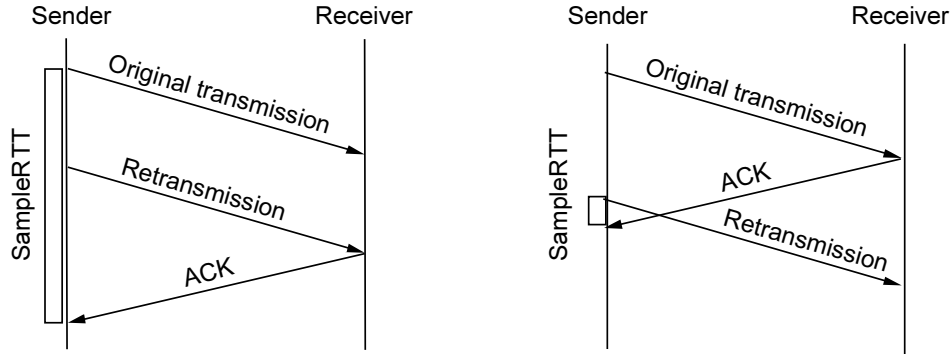
- where a between 0.8 and 0.9

- Set timeout based on EstRTT

$$\text{TimeOut} = 2 \times \text{EstRTT}$$

26

Karn/Partridge algorithm



- Problem:
 - impossible to know if ACK came from original packet or retransmission (see above)
 - timeout implies congestion, retransmissions increase congestion \Rightarrow reaction to timeout should be conservative
- Solution:
 - don't sample RTT when retransmitting!
 - double timeout after each retransmission (\Rightarrow exponential backoff)

27

Jacobson/ Karels algorithm

- Problem with earlier algorithms:
 - RTT is a random variable, moreover it has a variance
 - If variance is small, EstimatedRTT more reliable
 - If variance is large, timeout should not be “too much” based on EstimatedRTT
- New Calculations for EstRTT (i.e., average RTT)

$$\begin{aligned} \text{Diff} &= \text{SampleRTT} - \text{EstRTT} \\ \text{EstRTT} &= \text{EstRTT} + (d \times \text{Diff}) \\ \text{Dev} &= \text{Dev} + d \times (|\text{Diff}| - \text{Dev}) \end{aligned}$$

- where d is a factor between 0 and 1
- Idea: consider variance when setting timeout value

$$\text{TimeOut} = m \times \text{EstRTT} + f \times \text{Dev}$$

- where $m = 1$ and $f = 4$
- Notes
 - algorithm only as good as granularity of clock (500ms on Unix)
 - accurate timeout mechanism important to congestion control (later)

28

TCP extensions

- Implemented as header options
 - store timestamp (32 bit) in outgoing segments (to improve RTT measurement accuracy by measuring it from the packet)
 - extend sequence space with 32-bit timestamp (protection against sequence number wrap around)
 - shift (scale) advertised window (larger window sizes)
 - use of Options negotiated during TCP connection establishment
- Nagle algorithm (RFC896)
 - built in most TCP implementations
 - sender holds a partial segment's worth of data (even if PUSHed) until either a full segment accumulates or the most recent outstanding ACK arrives (= all data has been received correctly by receiver)
 - small packet problem: Telnet can generate 1B + 40B packets (4000% overhead)
- Delayed ACKs (RFC813)
 - to limit generation of small ACK packets
 - receiver waits until there is data to transmit on reverse path and "piggybacks" ACKs on TCP data segment
 - timer guards that receiver does not wait "too long"
 - ACKs must still be generated at least after receiving 2*MSS bytes (every 2 segments)

29

Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

30

Stream Control Transmission Protocol (SCTP)

- Suitable transport protocol for signaling traffic (SS7) over IP
 - may be used for traditional Internet services such as those based on HTTP and SIP
 - sometimes called the “Next Generation TCP”
 - Internet standard RFC2960
- Service model
 - unicast transport protocol
 - provides **reliable, message-oriented** data delivery
 - SCTP sends “complete” messages whereas TCP is byte oriented and does not preserve any structure in the byte stream
 - reliable = lost/corrupted messages are retransmitted
- Other features
 - functionality is TCP-like with modifications due to message oriented principle
 - TCP-like sliding window, flow control and congestion control
 - uses selective acknowledgements (SACKs) to report out of sequence data
 - multistreaming: data stream can be partitioned into multiple streams, each controlled independently
 - multihoming: for redundancy, an SCTP endpoint can be associated with multiple IP addresses

31

Outline

- Overview of end-to-end protocols
- UDP
- TCP
 - Connection establishment / termination
 - Sliding window and flow control
 - Adaptive timeout + TCP extensions
- SCTP
- About Remote Procedure Calls (RPC)

32

About remote procedure calls

- Request/reply paradigm, but UDP message going in one direction and UDP message back is not enough
 - messages can be lost or reordered
 - use of TCP overkill
- Protocol family Remote Procedure Call (RPC)
 - RPC more complex than local procedure call:
 - network between calling process and called process is complex
 - computer at each end may have different architectures and data representation
 - semantics:
 - at-most-once = for every request message, only one copy of message is delivered to the server
 - zero-or-more = remote procedure call invoked zero or more times
 - examples: SunRPC, DCE RPC
- Internet society view
 - RPC carried on top of UDP \Rightarrow RPC is not a transport protocol according to Internet architecture

33

Some remarks

- Getting a transport protocol right is hard, changing circumstances make it harder
- In Internet, transport protocols have a crucial role
 - IP network is unreliable
 - IP design principle has usually been that end systems implement all intelligent functions \Rightarrow transport layer is (first) end-to-end layer
- Protocols can and do change
 - for example TCP timers and congestion control
 - networks do change
 - How to change a protocol in the Internet?
 - Must always consider interoperability...
 - How to supply the level of service required by applications (that also changes all the time)
 - New requirements by real time applications \Rightarrow DCP
 - Other emerging requirements...

34