LECTURE 11

Transport Layer Issues

Transmission Control Protocol (TCP)

- Reliable byte stream.
- Connection oriented -->
  - Guarantees reliable in-order delivery of a stream of bytes.
  - Has flow control i.e., the receiver can limit the amount of data that the source sends.
  - Does the demultiplexing that UDP does.
  - Includes a congestion control mechanism -- throttle the rate of sending to avoid overloading the network.

TCP basics

- A sliding window protocol is at the heart of TCP.
- TCP establishes an explicit logical connection between a client and a server.
- There is an explicit connection establishment phase (similar to dialing a connection) -- the two sides agree to exchange data.
  - The two parties establish some shared state to enable the sliding window algorithm to begin.
- There is a explicit teardown phase -- the connection is torn down.

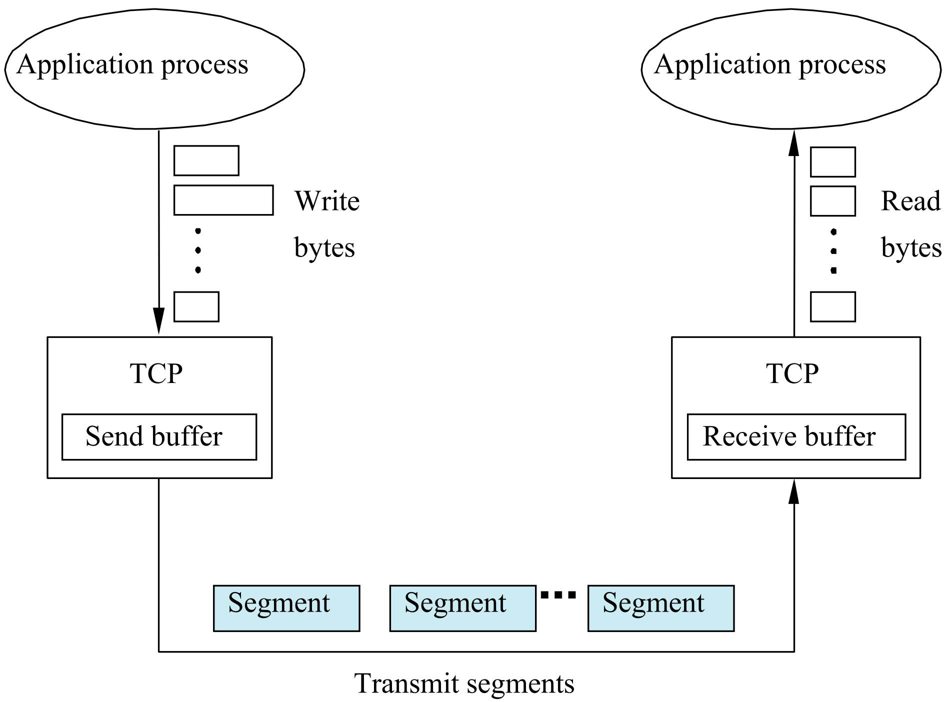
Packet re-ordering and Sequence Nos.

- Packets could be re-ordered when the traverse the Internet.
- Sequence numbers are used to ensure that they arrive in order.
  - How far out of order ? (to be determined)
- IP has a TTL
  - TCP uses this information to define a maximum segment lifetime (defined as MSL).
  - Current recommended setting for MSL is 120 seconds (it is a conservative estimate).

Segments

- TCP is a byte oriented protocol -- This means that sender writes "bytes" into a TCP connection and receiver retrieves bytes.
- But TCP does not really "directly" transmit bytes.
- Buffers enough to fill a reasonably sized data unit called segment and sends it to receiver.
- Receiver retrieves bytes and stores in buffer.

Pictorial View of the Process

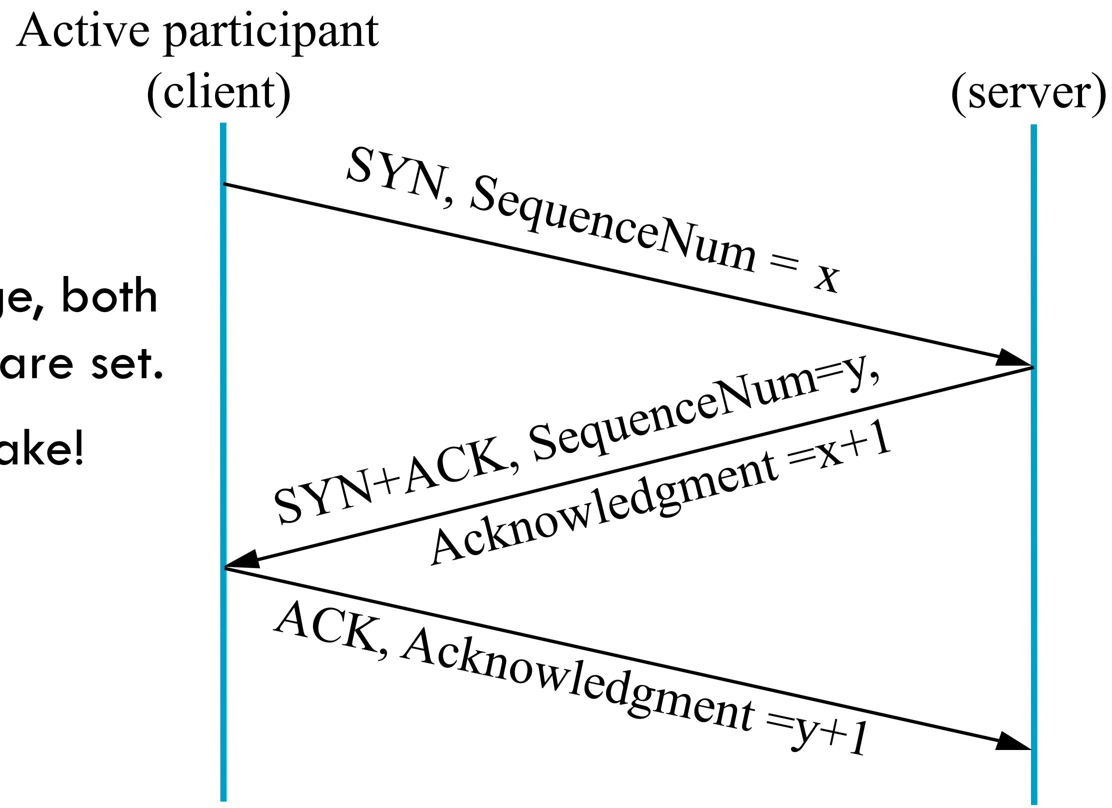


Flags in TCP header

- There are 6 bits for flags.
- SYN flag -- connection establishment
- FIN flag -- connection termination
- ACK flag -- Acknowledgement field is valid -- bytes are being acknowledged, so the receiving TCP entity should pay attention to that field.
- URG flag --segment contains urgent data.
- PUSH flag -- sender invoked PUSH -- send data to application right away.
- RESET flag : confusion -- abort connection.

Connection Establishment

- Client is the caller
- Server is the callee
- In the SYN+ACK message, both the SYN and ACK flags are set.
- It is a three way handshake!

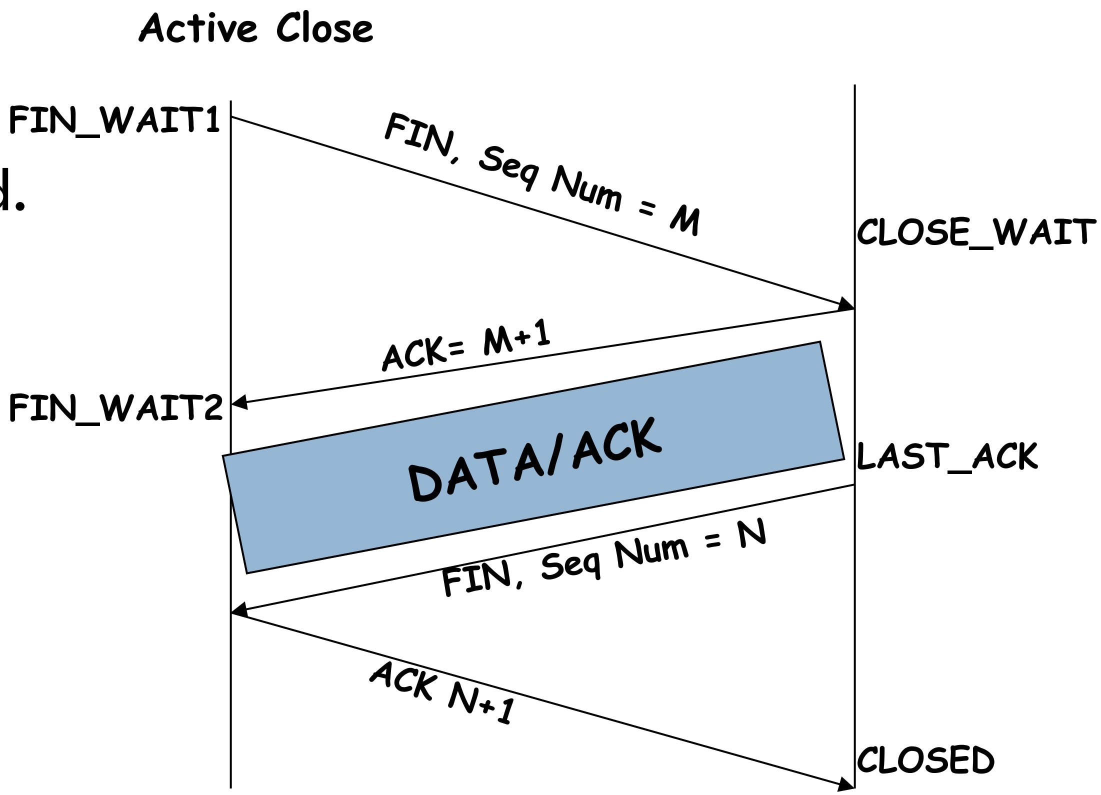


The Three Way Handshake

- If SYN+ACK is lost, then server is left hanging -- does not know that the client did not get it and therefore might have aborted.
- If ACK gets lost on the other hand, it is ok -- the sender sends the first segment and so on -- so the connection survives.

Connection Termination

- Note that after the server receives the FIN_WAIT_1, it may still have messages -- thus, connection not yet closed.



Goal of TCP Congestion Control

- Goal of TCP is to determine the available network capacity and prevent network overload.
  - Depends on other connections that share the resources.
- Typically, in discussions, First in First Out queues are assumed; however, congestion control mechanisms work with other queuing techniques (fair queuing) as well.

Why prevent congestion ?

- Congestion is bad for the overall performance in the network.
  - Excessive delays can be caused.
  - Retransmissions may result due to dropped packets
    - Waste of capacity and resources.
  - In some cases (UDP) packet losses are not recovered from.
  - Note: Main reason for lost packets in the Internet is due to congestion -- errors are rare.

The Congestion Window

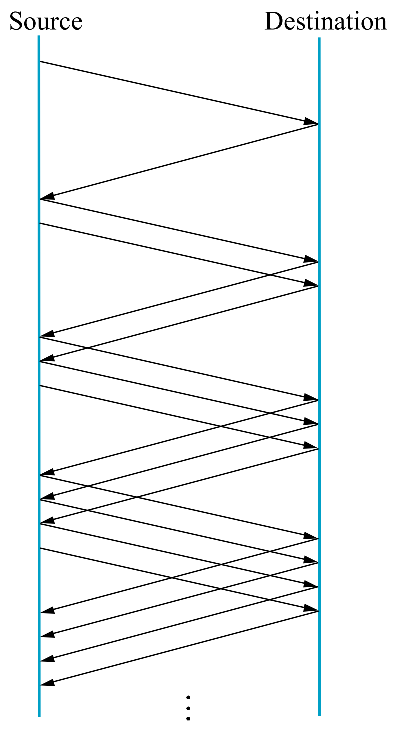
- In order to deal with congestion, a new state variable called "CongestionWindow" is maintained by the source.
  - Limits the amount of data that it has in transit at a given time.
- TCP sends no faster than what the slowest component -- the network or the destination host --can accommodate.

Managing the Congestion Window

- Decrease window when TCP perceives high congestion.
- Increase window when TCP knows that there is not much congestion.
- How ? Since increased congestion is more catastrophic, reduce it more aggressively.
- Increase is additive, decrease is multiplicative -- called the Additive Increase/Multiplicative Decrease (AIMD) behavior of TCP.

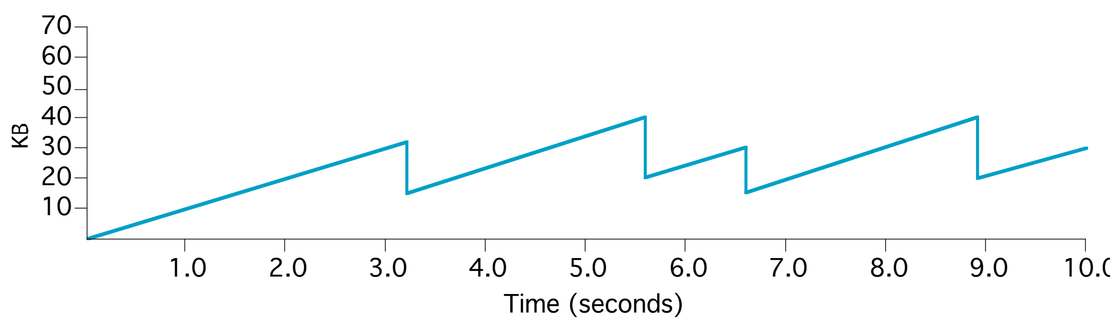
AIMD details

- Each time congestion occurs - the congestion window is halved.
  - Example, if current window is 16 segments and a time-out occurs (implies packet loss), reduce the window to 8.
  - Finally window may be reduced to 1 segment.
- Window is not allowed to fall below 1 segment (MSS).
- For each congestion window worth of packets that has been sent out successfully (an ACK is received), increase the congestion window by the size of a (one) segment.



More AIMD details

- TCP is byte oriented.
  - does not wait for an entire window worth of ACKs to add one segment worth to congestion window.
- Reality: TCP source increments congestion window by a little for each ACK that arrives.
  - Increment = MSS * (MSS/Congestion Window)
    - This is for each segment of MSS acked.
  - Congestion Window + = Increment.
- Thus, TCP demonstrates a sawtooth behavior !

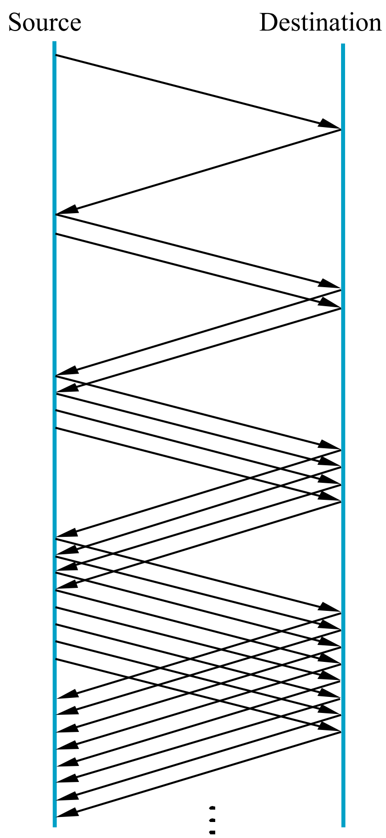


TCP Slow Start

- Additive Increase is good when source is operating at near close to the capacity of the network.
  - Too long to ramp up when it starts from scratch.
  - Slow start --> increase congestion window rapidly at cold start.
- Slow start allows for exponential growth in the beginning.

  E.g. Initially CW =1, if ACK received, CW = 2.

  If 2 ACKs are now received, CW = 4. If 4 ACKs are now received, CW =8 and so on.
- Note that upon experiencing packet loss, multiplicative decrease takes over.



Why Call it Slow Start ?

- The original version of TCP suggested that the sender transmit as much as the Advertised Window permitted.
- Routers may not be able to cope with this "burst" of transmissions.
- Slow start is slower than the above version -- ensures that a transmission burst does not happen at once.

Where does AIMD come in now ?

- Slow start is used to increase the rate to a "target window size" prior to AIMD taking over.
- What is this target window size ? -- Unclear
- In addition, we now have to do book keeping for two windows -- the congestion window and the "target congestion window" where Slow start ends and AIMD begins.

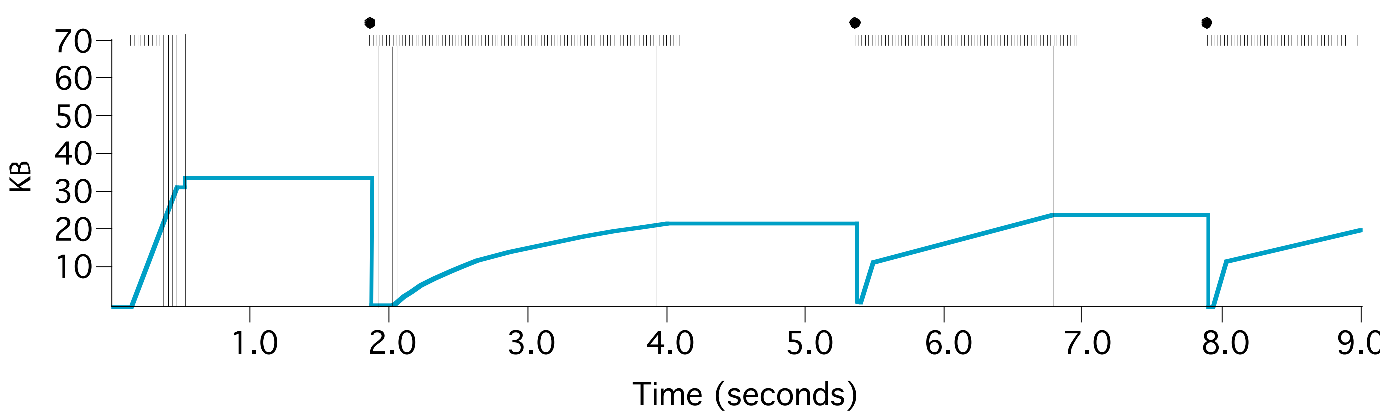
The Congestion Threshold

- Initially no target window -- when a packet loss occurs, divide the current CW by 2 (due to multiplicative decrease) -- this now becomes the target window.
- Define this to be the "Congestion Threshold".
- Reduce actual CW to 1.
- Use Slow Start to ramp up to the Congestion Threshold (or simply threshold). Once this is reached use AIMD.

Summary: TCP Tahoe

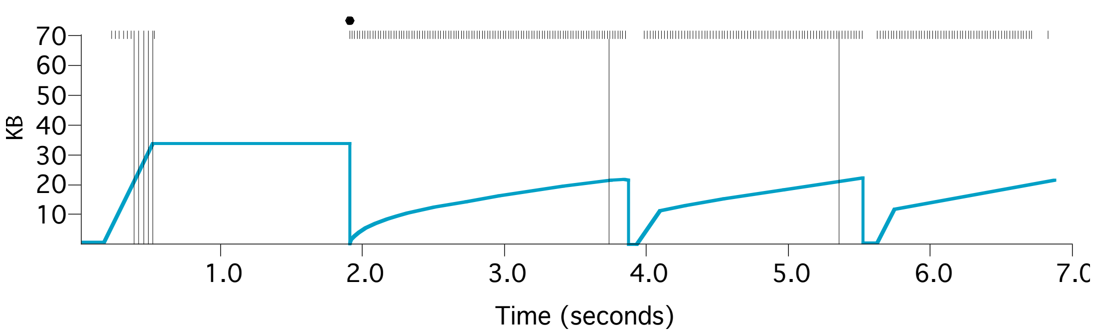
- Thus:
  - When CW is below the threshold, CW grows exponentially
  - When it is above the threshold, CW grows linearly.
  - Upon time-out, set "new" threshold to half of current CW and the CW is reset to 1.

  - This version of TCP is called "TCP Tahoe".



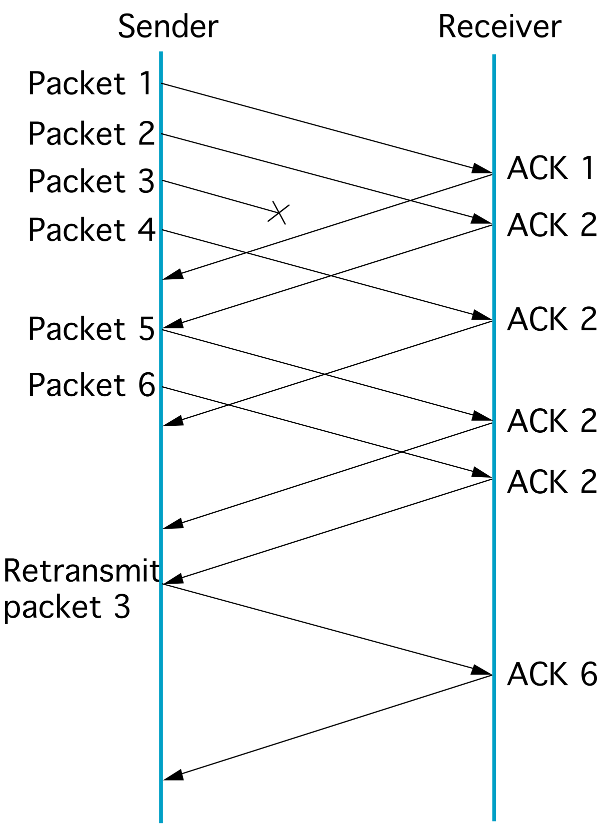
Fast Retransmit

- Coarse grained TCP time-outs sometimes lead to long periods wherein a connection goes dead waiting for a timer to expire.
- Fast Retransmit -- a heuristic that sometimes “triggers” the retransmission of a packet faster than permissible by the regular time-out.
- Every time a data packet arrives at a receiver, the receiver ACKs even though the particular sequence number has been ACKed.
- Thus, when a packet is received in out of order, resend the ACK sent last time -- a duplicate ACK!



Duplicate ACKs

- When a duplicate ACK is seen by the sender, it infers that the other side must have received a packet out of order.
  - Delays on different paths could be different -- thus, the missing packets may be delivered.
  - So wait for "some" number of duplicate ACKs before resending data.
  - This number is usually 3.



Fast Recovery

- When the fast retransmit mechanism signals congestion, the sender, instead of returning to Slow Start uses a pure AIMD.
  - Simply reduces the congestion window by half and resumes additive increase.
- Thus, recovery is faster -- this is called Fast Recovery.

TCP Reno

- The version of TCP wherein fast retransmit and fast recovery are added in addition to previous congestion control mechanisms is called TCP Reno.
  - Has other features -- header compression (if ACKs are being received regularly,omit some fields of TCP header).
  - Delayed ACKs -- ACK only every other segment.

Impact at transport layer

- Popular transport layer protocols for the Internet are UDP and TCP
  - UDP offers best effort delivery ; no reliability, flow control or congestion control
  - TCP offers all the above features
- The wireless channel, interference and mobility all induce packet losses.
- This impacts UDP – lossy flows
- The impact on TCP could be even more dramatic

TCP issues

- The main assumption that TCP makes is that packet losses are due to congestion
  - Not due to the other effects like wireless channel, interference or mobility induced failures.
- This can create problems – poor TCP performance.
- TCP repeatedly goes back to Slow Start
- Operates at low congestion windows
  - Congestion window → defines the number of outstanding packets at the sender side
    - Maximum number of packets that can be sent without causing congestion at intermediate routers.
- Inaccuracies in RTT estimates, out of order packet delivery.

Implications of Wireless and Mobility

- Error rates on wireless links are an order of magnitude higher than on fiber or copper links.
  - Packet losses much more common.
    - In spite of link layer retransmissions
- Mobility can cause delays and packet loss
  - Packets may still be forwarded to an old foreign agent
    - Nothing to do with wireless access but because of packet rerouting issues.
- TCP is unable to distinguish between these effects and losses due to congestion
  - Congestion losses exist but are not the primary cause of packet losses.

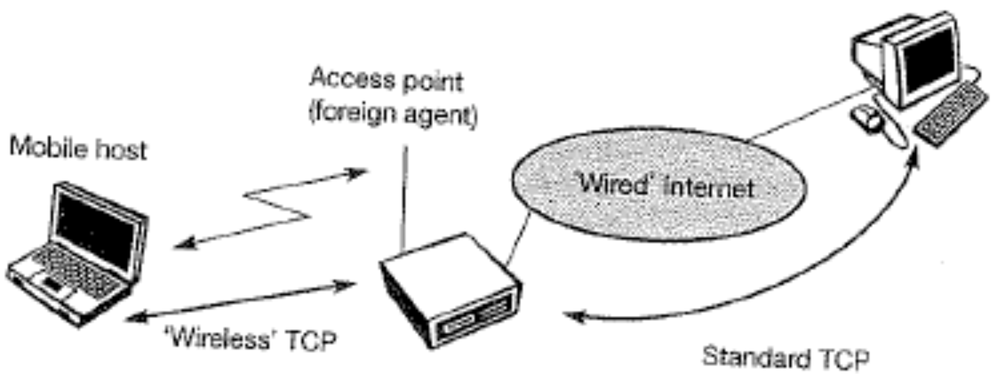
What could happen ?

- TCP could repeatedly go back to slow start
- Operate at very small congestion windows
- Unnecessary retransmissions end to end
  - Even though losses are likely to be on the wireless link.
- In spite of these issues, we need to retain TCP
  - Almost all devices on the Internet use TCP
    - Hard to change things.
- Need to make changes such that the solutions are backward compatible.

Indirect-TCP

- Based on two observations
  - Poor performance of TCP with wireless links
  - TCP cannot be changed in the Internet
- Basic idea is to split the connection into two parts
  - A wireless part
  - A wired part
- Standard TCP is used in the wired part
- It can be also used in the wireless part, but one can optimize this part.

I-TCP continued



- The point of termination is either an AP or the foreign agent.
- This acts as a proxy for the mobile device
- It can track where the mobile device moves – so it is the right location to split the connection.
  - One can also envision splitting the connection at the SGSN or the GGSN with GPRS

Role of the Proxy

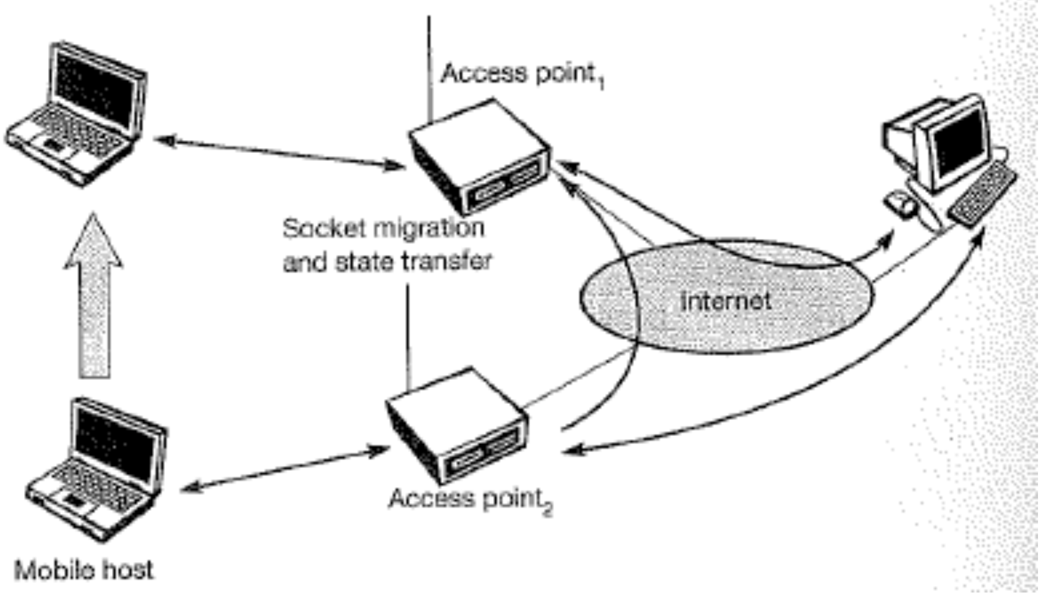
- The proxy relays the data in both directions.
- As far as either end-host is concerned, the proxy is the termination point of the TCP connection.
- If CN sends the packet, proxy ACKs – then takes responsibility of forwarding to MN.
- Similarly, if MN sends packet, it ACKs and takes responsibility of sending it to the CN.

Hand-off

- Various actions needed by I-TCP if there is a hand-off.
- The old proxy must now forward buffered data to new proxy.
  - Recall that the new foreign agent can inform the old one about the mobility of the MN – to facilitate forwarding.
- But this is not all.....

Socket migration

- The goal is to keep things transparent from the application.
- This requires a migration of of the sockets of the proxy to the new proxy
  - Leads to implementation challenges.
  - State needs to be transferred from old AP or FN to new AP or FN.
- New connection must not be established for the MN!



Advantages of I-TCP

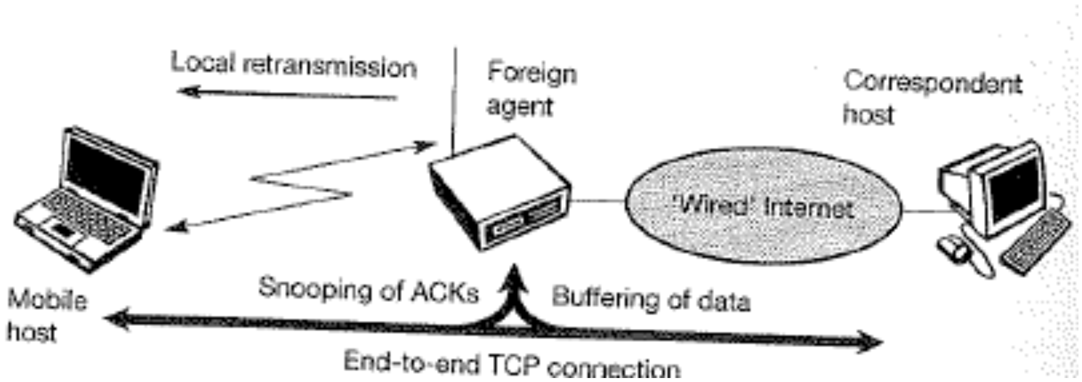
- No changes needed to TCP
- Transmission errors on wireless link cannot propagate into the Internet
- Changes to the wireless part possible without affecting TCP
- Header compression possible on wireless link
  - Fragmentation etc. are not an issue since the packets only traverse a single link.

Disadvantages of I-TCP

- Loss of e2e semantics a big issue
  - If proxy dies, then the end hosts have no way of knowing that the packets did not reach the destination.
- Hand-over is complex
  - If FA has a large number of packets to be forwarded this could be an issue
  - State migration leads to complexities in implementation.
- The proxy needs to be trusted !
  - Things like e2e encryption (IPSec) not possible.

Snooping TCP

- In order to fix the problem of I-TCP with regards to e2e semantics, Snooping TCP was invented
- Instead of terminating the connection at a proxy, proxy only monitors the connection and performs local retransmissions in cases of loss.



Local retransmissions

- Snoops packet flows in both directions
  - Recognize acknowledgements.
- Foreign agent buffers packets until it receives ACK from the mobile node.
- If no ACK is received within a certain period, it assumes that either the packet or ACK is lost.
- It performs local retransmissions.
- If duplicate ACKs are received (Reno), this again shows the loss of a packet
  - Local retransmissions are again performed.
- Advantage: Local retransmission much faster – reflects the delay of a single hop + processing time.

Transparency

- In order to remain transparent, the FA does not ACK data to the end-host.
- If it does, the end-host is mislead into believing that the mobile node received the packet.
  - End-to-end semantics violated.
- However, foreign agent can filter duplicate ACKs to avoid unnecessary retransmissions from end-host.

What if foreign agent fails ?

- If the foreign agent crashes, the end-host will time out.
- It may lose some duplicate ACKs but things still work.
- After the time out, it retransmits the packets.
- If the foreign agent comes back up and has state information on which packets were correctly received on the wireless link, it can discard those packets.

Advantages of Snooping TCP

- End-to-end semantics preserved.
- No changes to correspondent end-host – all changes to foreign agent.
- No changes to mobile host either!
- Backward compatible – if foreign agent does not use Snooping TCP, TCP progresses as is.

Disadvantages of Snooping TCP

- The errors are not as well isolated from the wired client.
  - If delays on wireless link are large, these still cause the end-host to time out.
- It doesn't work if there is end to end encryption at a layer below TCP (IPSec)
  - Foreign agent cannot see the packets and thus, cannot determine which packets to locally retransmit.
    - Note : SSL will still work (port nos. sequence nos. visible).

Mobile TCP -- Goals

- In previous cases, the goal was to handle high error rates on wireless links.
- Mobile TCP : Goal is to handle mobility
- In particular, a mobile node may get disconnected when it is on the move.
- TCP however, being unaware of the disconnected state, continues to time-out and retransmit packets.
  - Retransmits an unacknowledged packet every minute and will give up after 12 retransmissions.
- If connectivity is back interim, in the worst case, the sender has to still wait for 1 minute.
  - Moreover, it has probably returned to slow start.

Why don't the other approaches work ?

- With I-TCP – the proxy buffers more and more data
  - Longer the disconnection, more the buffering.
- If there is a handover, (typical after a disconnection), all data and state has to be transferred to the new proxy.
- With Snooping TCP – mobile does not ACK packets – there is little the foreign agent can do.
  - TCP behaves as it does in the default case.

Mobile TCP (M-TCP) approach

- Splits TCP connection into two parts as with I-TCP.
- The intermediate proxy is referred to as the "supervisory host" or SH.
- TCP between the connecting host (in the Internet) and the SH is unchanged.
- An optimized TCP is used between the SH and the mobile host (MH).
- M-TCP assumes a low bit error rate on wireless link
  - So it does not perform caching/retransmissions via the SH.
  - If packet is lost, it needs to be sent end-to-end again.
- The goal of M-TCP is primarily to handle disconnections.

M-TCP Approach (Continued)

- The SH monitors packets sent to the MH and the ACKs returned from the MH
- If it does not receive an ACK in a long time, it assumes that the end host is disconnected.
- It then chokes the sender !
  - Set the sender's advertised window to 0.
  - What does this do ?

Sender side control

- The window size of `0' forces the sender to go into the persistent mode
  - The state of the sender is fixed for the duration of the disconnection.
- Sender does not try to send new or retransmit data.
- As soon as the SH (either the old one or a new SH) detects connectivity again, it reopens the window and the sender returns to the old value.
- Sender can continue to send at full speed
  - Slow start is disabled

Advantages of Mobile TCP

- Mechanism does not need changes to sender side TCP.
- Recovery from packet loss faster.
- Maintains end-to-end semantics – SH does not send ACKs itself.
- If MH is disconnected it avoids useless retransmissions, slow starts or breaking connections.
- No buffering as with I-TCP – so no need to forward packets to a new SH.

Disadvantages of M-TCP

- Since SH does not act as a proxy, packet loss percolates to the sender.
- Slow start is disabled – this requires a new bandwidth manager implementation.

Moving to a new foreign agent

- When a MN moves to a new foreign agent, there can be packet loss.
- Again – TCP concludes congestion and goes to slow start.
- There is no congestion though !
- How can we deal with this ?
- Exploit the fast retransmit/fast recovery mechanisms.

Exploiting fast retransmit/fast recovery

- Idea: Artificially force the fast retransmit behavior on both the Mobile node and the correspondent host node.
- When mobile host registers with a foreign agent, it sends duplicate ACKs to correspondent hosts (CH).
  - Send three of these.
  - Causes the CH to go into fast retransmit mode and not to slow start.
  - The CH continues to send at the same rate as prior to hand off.
- In addition, put the MH in fast retransmit mode when it discovers the foreign agent (need Mobile IP help)
  - It retransmits all unACKed packets using the current congestion window size without going to slow start.

Advantages and Disadvantages

- Advantages
  - Simplicity of the approach is the main advantage
  - Minor changes in mobile host's software
  - No changes to foreign agent or correspondent host.
- Disadvantages
  - If handover takes a long time, hosts may still time out and perform retransmissions.
  - Packet losses due to wireless link errors not considered.
  - Cooperation with Mobile IP at the MH needed.

MAC Layer assistance

- The approaches so far cannot handle long interruptions.
  - Loses connection in a tunnel
  - Cell without capacity
- TCP will disconnect after a long time out.
- But the MAC layer has detected disconnection!
  - It does not assume congestion as TCP does.
- It can inform the TCP layer that this is the case
  - Referred to as "Cross Layer Design" – breaks layer barriers.

What can TCP do ?

- TCP can now stop sending and "freezes" the current state of its congestion window.
- If the link breakage is detected quickly, both the mobile side and the correspondent host side can be informed and can do this.
- As soon as MAC layer detects connectivity, it signals TCP.
- TCP resumes operations at exactly the same point where it had stopped.
  - For TCP – time had stopped in between – no timers expire.

Advantages and Disadvantages

- Advantages
  - Can handle long interruptions.
  - Independent of TCP mechanisms (slow start, ACKs, sequence numbers etc.)
- Disadvantages
  - Major changes needed at both the mobile host and AP side.
  - The approach is not agnostic to the MAC layer – meaning the MAC has to specially be designed to assist TCP.
  - Need resynchronization after interruption
    - State consistency etc.

Impact of packet losses on TCP

- Recall:
  - TCP ACKs are cumulative – in order receipt of packets up to a certain packet.
  - If a single packet is lost, everything starting from the lost packet has to be retransmitted.
- Wastes capacity especially in mobile settings.
- An extension of TCP that may find use in mobile settings is the use of selective retransmissions.

Selective Retransmissions

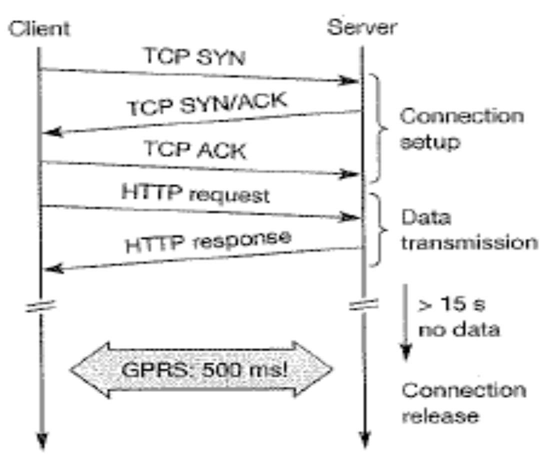
- RFC 2018
- TCP can indirectly request a selective retransmission of packets.
- ACK single packets – not simply trains of in sequence packets.
  - E.g. Packets X and X+2 are ACKed
- Sender can determine which packet is lost and retransmit that packet.

Advantages and Disadvantages

- Advantage
  - Saving bandwidth –sender resends only lost packets selectively.
  - Note : gain in efficiency not limited to wireless settings.
- Disadvantage
  - More complex software on the receiver side
  - Need to re-sequence data, wait for gaps to be filled.

Impact of delays on TCP

- TCP has an elaborate set up phase
  - 3 way handshake
  - Application request response.
  - In systems with delay, this can be profound
    - A lot of time to establish connections.



- GPRS web scenario
  - Three way Handshake
  - HTTP request transmitted
  - GPRS establishment (ask for channel, get assigned)

  - Total of at least 3+2+2 before transaction

Transaction-oriented TCP

- RFC 1644.
- T/TCP combines packets for connection establishment and connection release for user data packets.
- Reduces the number of set up messages.
  - Reduction in overhead
- Not designed for original TCP
  - So requires changes to both the mobile host and the corresponding hosts.

Ad hoc networks and TCP

- How does TCP cope with link breakage ?
- Sometimes when routes break due to mobility , routing protocol needs to rediscover route
  - The rediscovery phase takes time.
- If TCP is used as is, it can induce continuous retransmissions, time-outs and return to the slow start phase as in the earlier settings.

Mobility affects TCP performance

- Presence of stale routes
  - Possible with protocols like AODV
  - Caching with DSR also results in stale routes
- Asymmetry in routes – double trouble
  - Not only data gets lost, but ACKs can get lost – not symmetric
- ARP failures to neighbors who have moved away

Use of Explicit Feedback

- The Idea is similar to the use of explicit notifications is not new – ECN or Explicit Congestion Notification in the Internet to inform source about congestion.
- A similar scheme can be thought of which can provide the source about an explicit notification about the failure of a link.
- This message may be called ELFN or Explicit Link Failure Notification.
- Upon receiving this message, a TCP source can infer that packet losses are due to link failures rather than congestion, and therefore act differently.

How can one implement ELFN ?

- Simplest way : ICMP message to indicate that host is unreachable.
- Second possibility : Piggyback this to TCP on the Route Failure Message.
- When the TCP layer at the source receives this message it disables the congestion control mechanisms.
- What does it need to do ?

TCP response to ELFN

- Enter a mode called the standby mode.
- Disable the retransmission timers.
- In this mode a packet is sent at periodic intervals to probe whether the route has been established.
- If an ACK is received, it leaves the stand-by mode and restores its retransmission timers, and continues as normal.
- Another possibility is to generate an explicit route restored notification – but how ?

Other problems we won't go into

- Longer connections have more problems
  - More likely losses, more likely TCP will back off etc.
- Out of order packet delivery
  - Because of route changes
  - Can trigger fast retransmit/recovery
- There are specialized version of TCP protocols designed for ad hoc networks – but do not solve all problems categorically.