Lecture 15. Wireless routing; Wireless TCP

1. Understand the link metric design (ETX and ETT) in Roofnet: How are they derived; why are they designed in this way; What are the limitations?

ETT:

- Link ETX: predicted number of transmissions
  - Calculate link ETX using forward, reverse delivery rates
  - To avoid retry, data packet and ACK must both succeed
  - Link ETX = 1 / (df × dr)

  df = forward link delivery ratio (data packet)
  dr = reverse link delivery ratio (ack packet)
- Path ETX: sum of the link ETX values on a path
- Limitation of ETX
  - ETX assumes all radios run at same bit-rate
  - But 802.11b rates: {1, 2, 5.5, 11} Mbit/s
  - Two links with the same ETX may have different bit-rates
- Solution: Use expected time spent on a packet, rather than transmission count
  - New metric: expected transmission time (ETT)

ETX:

- ACKs always sent at 1 Mbps, data packets 1500 bytes
- Nodes send 1500-byte broadcast probes at every bit rate b to compute forward link delivery rates $d_f(b)$
  - Send 60-byte (min size) probes at 1 Mbps → d_r
  - To represent reverse link (ACK) delivery ratio
- At each bit-rate b, ETX(b) = 1 / (df(b) × dr)
- For packet of length S, ETT(b) = (S / b) × ETX(b)
- Link ETT equals to the minimum ETT(b) among all b options
- ETT assumption
  - Total time of an end-to-end transmission equals the time spent on each link along the path
- Does ETT maximize throughput?
  No!
  - Underestimates throughput for long (≥ 4-hop) paths

  Distant links along the same path can send simultaneously (spatial reuse), instead of sequentially
  - Overestimates throughput when transmissions collide and are lost

- ETT assumption
  - Total time of an end-to-end transmission equals the time spent on each link along the path

- Does ETT maximize throughput?
  No!
  - Underestimates throughput for long (≥ 4-hop) paths

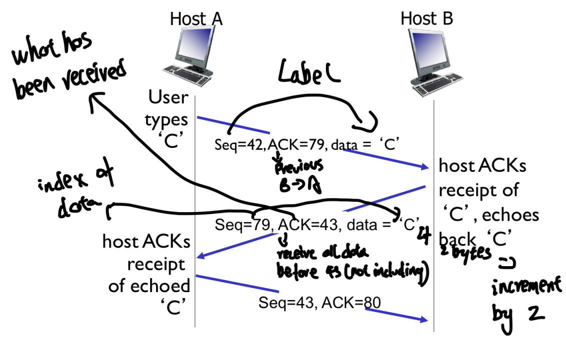
  Distant links along the same path can send simultaneously (spatial reuse), instead of sequentially
  - Overestimates throughput when transmissions collide and are lost

2. Work flow of TCP

TCP: Detect and recover from data loss

sequence numbers:
- Index of first byte in segment's data

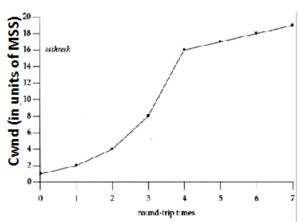
ACK:
- seq # of next byte expected from the other side
- cumulative ACK



simple TCP scenario

Fast retransmit
- Time-out period often relatively long: long delay before resending lost packet
- So instead, sender can detect lost segments via duplicate ACKs.
  - ✓ Sender often sends many segments back-to-back
  - ✓ If segment is lost, there will likely be many duplicate ACKs coming from receiver.
  - ✓ Sender retransmits immediately if it sees triple duplicate ACKs

- Two general phases of rate control
  - Slow start: a multiplicative increase (MI) algorithm
  - Congestion avoidance: an additive increase multiplicative decrease (AIMD) algorithm, to adjust rate according to congestion events



- Important variables
  - cwnd: congestion window size (i.e., sender's window size). Larger window means higher transmission rate
  - ssthresh: threshold between slow start and congestion avoidance phase; ideally should be set to half of estimated bandwidth (bandwidth: maximum cwnd the network can accept)

40

- Congestion indicated by loss. Examples: TCP Reno and TCP Tahoe
- TCP Reno case 1: Loss indicated by timeout:
  - cwnd reset to 1 MSS (maximum segment size);
  - window then grows exponentially (as in slow start) to threshold, then grows linearly
- TCP Reno case 2: Loss indicated by 3 duplicate ACKs
  - dup ACKs indicate network capable of delivering some segments
  - cwnd is cut in half, cwnd then grows linearly
- TCP Tahoe: set cwnd to 1 in both cases

Lecture 16. Wireless TCP

1. Understand the relationship between RTT, loss rate, and TCP bandwidth estimation

   - $W = \sqrt{8/3p} = (4/3) \times \sqrt{3/2p}$
   - Recall, estimated bandwidth $B = (\frac{3}{4}W \times \text{packet size}) / RTT$

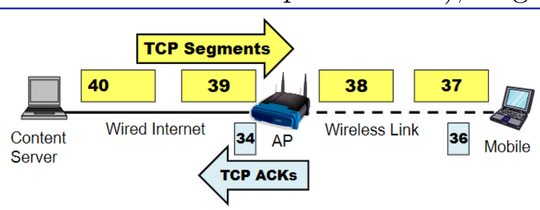
   $$\boxed{B = \text{packet size} / (RTT \times \sqrt{2p/3})}$$

   - Consequences:
     - ✓ Increased loss quickly reduces throughput
     - ✓ Flow with longer RTT achieves less throughput than flow with shorter RTT
     - ✓ Note: Don't get the wrong impression that bandwidth grows linearly with packet size! Packet size has a constant limit (equal to MSS).

2. Work flow, pros and cons of Snoop TCP

   Pros: Downlink works without modification to mobile or server; Preserves end-to-end principle. Crash does not affect correctness, only performance

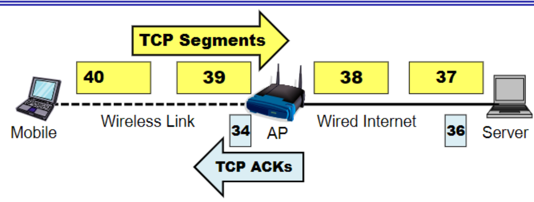
   Cons: Mobile host still needs to be modified at MAC and transport layers(Needed due to NACK scheme for uplink traffic); Slight violation of the end-to-end principle



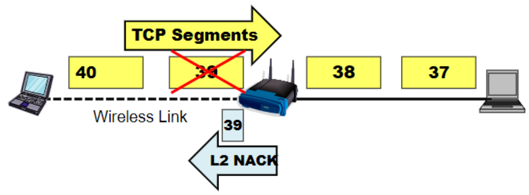
- AP buffers downlink TCP segments
  - Until it receives corresponding ACK from mobile
- AP snoops on uplink TCP acknowledgements
  - Detects downlink wireless TCP segment loss via time-out or duplicate ACKs (sent by mobile)

- When AP detects a lost TCP segment:
  - Locally, quickly retransmit that segment over the wireless link
  - Minimize duplicate ACKs flowing back to server
- Goal: server unaware of wireless loss and retransmission
  - No unnecessary reduction in cwnd

TCP Snoop: uplink traffic case



- Buffer & retransmit TCP segments at AP? Not likely useful
  - Uplink loss can only be fixed by Mobile's retransmissions



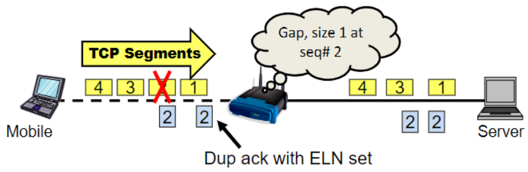
- AP detects wireless uplink loss via missing sequence numbers
- AP immediately sends MAC-layer negative ACK (NACK) to mobile
  - Mobile quickly & selectively retransmits data
  - Requires modification to AP and mobile's link layer

3. Understand why/how ELN improves wireless TCP
Pros:Simpler, easier to implement than Split or Snoop
Cons:Still requires modifications at the AP (very hard to implement in cellular networks)

Explicit loss notification (ELN)

- Basic idea
  - Notify the TCP sender that a wireless link (not congestion) caused a certain packet loss
  - Upon notification, TCP sender retransmits packet, but doesn't reduce congestion window

- When AP sees a duplicate ACK
  - AP compares the ACK seq# with its recorded gaps
  - If match: AP sets ELN bit in the duplicate ACK and forwards it
- When mobile receives dup ack with ELN bit set:
  - Resends packet, but doesn't reduce congestion window



4. Understand the principles and pros/cons of other TCP optimization mechanisms: time-out freezing, selective retransmission, performance enhancement proxies

Other TCP optimization for wireless: transmission/timeout freezing

- Motivation
  - Mobile hosts can be disconnected for a longer time
  - no packet exchange possible, e.g., in a tunnel, disconnection due to overloaded cells or multiplexing with higher priority traffic
  - TCP disconnects after time-out completely

Other TCP optimization for wireless: transmission/timeout freezing

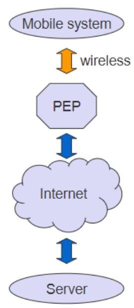
- Transmission/timeout freezing
  - MAC layer is often able to detect interruption in advance
  - MAC can inform TCP layer of upcoming loss of connection
  - TCP stops sending, but does not assume a congested link
  - MAC layer signals again if reconnected
- Advantage
  - scheme is independent of data
- Disadvantage
  - TCP on mobile host has to be changed, mechanism depends on MAC layer

- Motivation
  - Batch ACK: ACK *n* acknowledges correct and in-order reception of packets up to *n* (not including *n*)
  - if single packets are missing quite often, a whole packet sequence beginning at the gap has to be retransmitted, thus wasting bandwidth

Other TCP optimization for wireless: Selective retransmission

- Selective retransmission as one solution
  - Allows for acknowledgements of single packets, not only acknowledgements of in-sequence packet streams without gaps
  - Sender can now retransmit only the missing packets
- Advantage
  - much higher efficiency
- Disadvantage
  - more complex software in a receiver, more buffer needed at the receiver
  - Might be a problem in low-profile devices…

- Performance enhancement proxies
  - On transport layer: Local retransmissions and acknowledgements (similar to snoop TCP)
  - On application layer: Content filtering, compression, picture downscaling
- Advantage
  - Better performance, esp. seen at applications (video, Web, etc.)
- Disadvantage
  - Violates end-to-end principle



Lecture 17. Mobile and wireless applications

1. Understand the challenges and solution principles of mobile Web loading

- Poor HTTP ←→ TCP interaction when running in wireless networks
  - TCP: performs best when there is a steady stream of data packets to drive the cwnd to quickly converge to true network capacity
  - HTTP: generates short, bursty flows; flow often ends before cwnd converges to true network capacity
  - Problem is amplified in wireless networks
    - ✓ Wireless networks (e.g., 4G LTE) have long RTT→ TCP takes longer time to converge
    - ✓ Mobility causes the network capacity to vary quickly → TCP cannot easily keep track of the true network capacity
    - ✓ Wireless link loss further disturbs TCP convergence

Improving HTTP over wireless

- Key principle: make TCP converge fast!
  - Let the PHY layer of cellular link directly estimate the true network capacity, and differentiate link loss from congestion loss
  - Mobile receiver (e.g., a smartphone) directly informs TCP sender (i.e., HTTP server) about the optimal congestion window
  - TCP cwnd converges to network capacity in one RTT!
- Key challenge: how to estimate the true network capacity without changing the MAC/PHY standards?
  - Solution algorithm: take PHY layer logs from smartphones to estimate the frequency/time usage on the cellular link
  - Available frequency/time resources → available bandwidth for the link

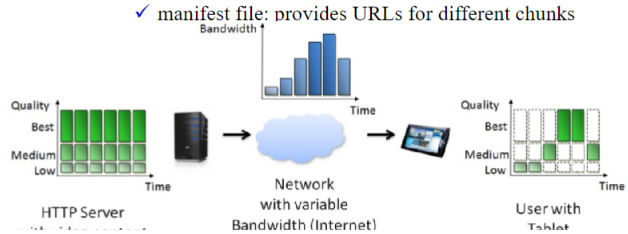
2. Understand the challenges and solution principles of mobile video streaming

Streaming stored video: challenges

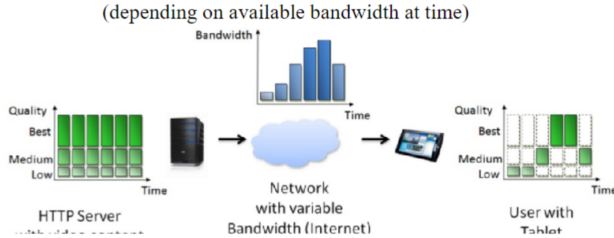
- Continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- Client interactivity: pause, fast-forward, rewind, jump through video
- Video packets may be lost, and retransmitted

DASH: Dynamic Adaptive Streaming over HTTP

- Industry standard application-layer protocol, for streaming stored video over HTTP/TCP
  - Server
    - ✓ divides video file into multiple chunks
    - ✓ each chunk stored, encoded at different rates
    - ✓ manifest file: provides URLs for different chunks



  - Client
    - ✓ periodically measures server-to-client bandwidth
    - ✓ consulting manifest, requests one chunk at a time
    - ✓ chooses maximum coding rate sustainable given current bandwi
    - ✓ can choose different coding rates at different points in time (depending on available bandwidth at time)



Solution principles for video streaming over wireless

- Solution 1: smart buffering
  - Video receiver uses a large buffer to smooth out the network capacity variation (play out only after receiving a sufficient number of video frames)
  - But a large buffer is unsuitable for interactive video (need immediate playback within a few hundred milliseconds)
  - What should be the proper buffer size?
    - ✓ A hard decision, as hard as choosing the optimal video bit-rate itself
    - ✓ No widely recognized usable solution yet

- Solution 2: active bandwidth probing
  - Video server periodically sends some dummy data to estimate the end-to-end network capacity, and then chooses the video bit-rate that fits the network capacity
  - Challenges:
    - ✓ Need to modify the video streaming application itself (e.g., adding active probing mechanism into DASH)
    - ✓ Probing frequently: costs extra network resource; probing infrequently: hard to keep track of the true network capacity

- Solution 3: physical layer informed mobile video streaming
  - Wireless link (cellular or WiFi) is often the bottleneck link along the end-to-end path
  - So it suffices to estimate capacity of the wireless link
  - Estimate the wireless link capacity based on PHY layer statistics (signal strength, time/frequency resource utilization, etc.)
  - Challenges
    - ✓ Need the wireless link to provide PHY layer statistics (good news: already available in many WiFi and cellular devices)