

Transport Control Protocol (TCP) In Heterogeneous Networks

Objectives

- Principles of TCP congestion control (AIMD)
- TCP Performance over heterogeneous networks
 - Long haul delay products
 - Lossy links
 - RTT fairness
 - Asymmetry bandwidth
- TCP enhancements over the past decade

View on the Internet Traffic

- About 80-90% of the Internet traffic comes from TCP flows.
- The rest of the traffic accounts mainly to UDP (about 10-15%) – with some leftover from protocols like ICMP (about 1%).

Metric	TCP (in %)	UDP (in %)	Others (e.g. ICMP in %)
Bytes	83 (±11)	16 (±11)	1 (±1)
Packets	75 (±12)	22 (±11)	3 (±2)
Flows	56 (±15)	33 (±10)	11 (±7)

Traffic aggregates by protocols (Grizzard et al., 2005)

Reliable Transport Protocol

- TCP Flow Control: receiver window TCP (*rwnd*)
- Congestion Control: AIMD window-based
 - Additively increase sending rate (1 segment per *rtt*) to probe the available bandwidth.
 - Multiplicatively halving the *cwnd* at the occurrence of segment loss.
 - Implements four major functional modules: slow start, congestion avoidance, fast transmission and fast recovery.
 - Maintain fairness among the co-existing flows.
 - Support network stability.
- TCP categories
 - Dropping based, e.g. Reno, New Reno, SACK
 - Delay based, e.g. Vegas
 - Mix of dropping and delay, e.g. FAST TCP
 - Explicit notification based, e.g. ELN, ECN
 - Rate based, e.g. TCP Westwood, TCP Real

Dropping based TCP

- Packet dropping is the only indication signal for network congestion (Maybe?)
- Different transmission condition causes severe throughput degradation
 - Segment losses can be to reasons other than congestion!
 - What happens in the occurrence of multiple simultaneous segment losses?
 - How about having many segment losses in one sending window?

Delay based TCP

- Packet delay is known to be more accurate indication of network congestion
 - In high speed networks, packet dropping is rare event (10^{-9} to 10^{-12}).
 - multi-bit information vs. one bit information
- Introducing extra delay at the transmission due to path change, link layer recovery, ..etc
 - Reduces the TCP throughput
 - Affect TCP throughput while determining the desired throughput
 - Causes TCP to fall into false network congestion

EN based TCP

- Uses a single bit to explicitly notify the source of a congestion or a packet loss.
 - ECN is set by the routers
 - ELN is set by the intermediate TCP agent
- Current network lacks of a feedback mechanism!
- Network is heterogeneous, not all routers or hosts can issue ECN or ELN
- Congestion at certain link does not necessary mean the existence of a network wide-range congestion.
- ECN has the potential to increase the congestion in the presence of higher network transmission in the next round.
- Security issues (MITM that falsely sets/unsets the ECN), which can flood a connection in a very short time.

TCP over Heterogeneous Networks

- Several TCP enhancements are proposed to handle different network transmission conditions
 - TCP over High bandwidth-delay products, e.g. optical networks with high bandwidth e.g. 10Gb/s
 - TCP with none congestion losses, e.g. lossy links in wireless environments
 - TCP and RTT fairness.
 - TCP and bandwidth Asymmetry e.g. satellite vs. dial-up connections

TCP over High Bandwidth-Delay Product

- WRN looks like an optimal transmission environment for TCP? Maybe!
 - Guaranteed high bandwidth allocation
 - Very low queuing delay vs. high propagation delay
 - Very low packet dropping (10^{-9} to 10^{-12})
- TCP over WRN suffers from
 - Slow convergence
 - RTT fairness at bottleneck
- Spare bandwidth is available \Rightarrow TCP increases by 1 packet per RTT even if spare bandwidth is huge
- When a TCP starts, it increases exponentially \Rightarrow Too many drops \Rightarrow Flows ramp up by 1 packet per RTT, taking forever to grab the large bandwidth

To increase the TCP window from half to full utilization of 10Gbps with 1.5 Kbyte packets, we need 1 hour with 100ms RTT and $p < 10^{-9}$

TCP for High Bandwidth-Delay Product

- **HSTCP** ACK: $W \leftarrow W + a(w)/W$
AIMD($a(w)$, $b(w)$) Loss: $W \leftarrow W - b(w) W$
- **STCP** ACK: $W \leftarrow W + 0.01$
MIMD(a , b) Loss: $W \leftarrow W - 0.125 W$
- **FAST** RTT: $W \leftarrow W \cdot \frac{\text{baseRTT}}{\text{RTT}} + \alpha$
- **SABUL** Combines UDP + TCP
- **TCP Westwood** Sender side Rate based

Round Trip Time

- TCP clock is based on RTT.
- Longer RTT reduces the transmission rate.
- Proposed solutions
 - TCP level:
 - TCP starts with larger *cwnd* e.g. 4
 - Count the received bytes instead of acks.
 - Application level:
 - XFTP, GridFTP: using many parallel TCP connections
 - Transfer objects at higher rate during congestion avoidance
 - Network level:
 - TCP segmentation e.g. TCP for satellite networks STP

RTT Fairness Solutions

- RTT fairness and the bandwidth allocation at the bottleneck
- TCP level:
 - Faster *cwnd* growth for long RTT
 - Constant Rate algorithm: achieve a constant rate increase regardless of RTT.
- Network level:
 - Packet dropping policy
 - Drop Tail
 - RED, Flow RED
 - Class-based queues (CBQ)
 - Stochastic Fairness Queuing (SFQ)

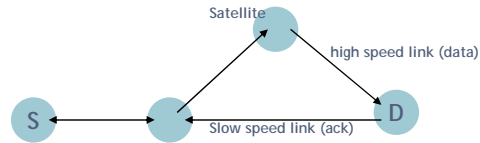
TCP over Lossy Channels

- TCP considers a segment loss as a signal for network congestion. Nothing new!
 - TCP TO \Rightarrow heavy network congestion
 - TCP TD \Rightarrow light network congestion
- Non-congestion losses caused by transmission errors:
 - packet corruption,
 - bad links,
 - channel fading,
 - hand offs.
- TCP over lossy links (wireless links) is mostly due to the misinterpretation of the wireless link losses as congestion losses.

Solutions for Non-congestion Losses

- Hiding the non-congestion losses
 - Local recovery of losses e.g. at the link level, such as using backup links, different routes
- Link-level
 - Retransmission (ARQ)
 - Efficient when losses are not frequent
 - Causes in-sequence delivery (TD)
 - Must be very fast $<$ RTT.
 - Forward Error Correction (FEC)
 - Only retransmit the corrupted parts
- TCP level
 - Retransmission at the input of the lossy link e.g. Indirect TCP, TCP SACK, Snoop Protocol such as rate-based Wireless TCP (WTCP), Mobile TCP (MTCP) using selective repeat, Freeze TCP using *rwnd*, if = 0, then no data is sent.
- End-to-End level
 - Explicit loss/congestion notification

Bandwidth Asymmetry



- High bandwidth at the forward direction (cable, satellite) while low speed channel (dial up) that carries the ACKs
 - Congestion/dropping at the slow link causes larger RTTs, slower growth of *cwnd*
 - Damage the TCP clocking mechanism
- Receiver side solution using compression
- Reducing the number of ACKs
- Reducing the TCP burstiness

Summary of TCP Implementations

- **TCP Reno**
 - Reno halve its *cwnd* every time it successfully retransmits one lost segment.
 - The most widely implemented TCP in current systems
 - Reno is expected to gain worse performance over links with multiple losses.
- **TCP New Reno**
 - Attempts to recover from multiple data segment losses without halving the *cwnd* each time when retransmitting a lost segment.
 - New Reno does not halve its *cwnd* until all the lost segments from that window have been successfully retransmitted.
 - Subject to several round trips until successfully recover the lost segments.
- **TCP Selective Acknowledgment (SACK)**
 - SACK maintains a number of SACK blocks, where each block contains a non-contiguous set of data received. SACK sender is able to send more than one lost segments at a time.
 - SACK block option helps improving the TCP performance links with multiple losses.

Summary of TCP Implementations cont.

- TCP Duplicate SACK (DSACK)
 - DSACK reports the receiving of duplicate packets such that the sender can constantly be informed of which segments are received by the destination.
 - In DSACK, the TCP sender can undo the halving of *cwnd* resulted from receiving two copies of a segment caused by burst behavior such as out-of-order delivery.
- TCP Forward Acknowledgment (FACK)
 - FACK maintains explicit measurements of the total number of bytes outstanding in the network.
 - The sender has the capability to determine the segment block that is dropped in the network.
- TCP Explicit Congestion/Loss Notifications (ECN/ELN)
 - ECN/ELN distinguish packet losses among congestion, contention, link failure, or other reasons.

Summary of TCP Implementations cont.

- TCP Delayed Congestion Response (DCR)
 - An *exponential back-off* delay is added before the TCP senders trigger any response to packet loss/out-of-order delivery.
 - The extra delay can also allow the link layer to perform the segment losses recovery if applicable.
 - DCR is expected to well perform when integrated with the data recovery using retransmission (ARQ) approach.
 - Delay can be severe in congestion, since acks are also delayed during transmission.
- TCP Eifel Algorithm
 - Eifel algorithm originally devised in wireless environments for detecting suspicious timeouts and retransmission ambiguity.
 - Eifel uses timestamps in the acknowledgements to overcome the problem of segment retransmission ambiguity.
- Burst TCP (BTCP) with BACK/BNACK
 - Proposed for detecting suspicious timeouts in OBS network.
 - TCP segments from the TCP sender to the OBS ingress nodes are acknowledged separately.
 - TCP agent located at each OBS edge or core node reports burst back negative ack (BNACK) to the TCP senders