

## Advanced Networking

### TCP – Part III

Renato Lo Cigno  
Renato.LoCigno@dit.unitn.it

---

---

---

---

---

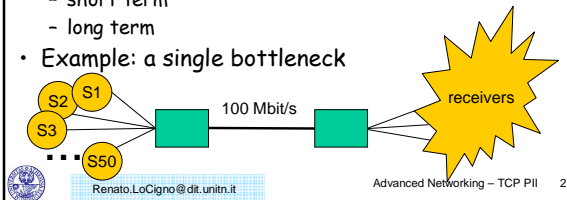
---

---

---

### TCP Congestion Control

- Retransmissions are useless if the network is congested ... but what is congestion?
- **Congestion**: a state of a network when the offered load (or traffic)  $\eta$  is larger than the network capacity  $C$ 
  - normalize  $\rho = \eta/C$  so that  $\rho > 1$  means congestion
  - short term
  - long term
- Example: a single bottleneck



---

---

---

---

---

---

---

---

### TCP Congestion Control Alternative Options

- Dynamic routing can alleviate congestion by spreading load more evenly
- But only effective for unbalanced loads and brief surges in traffic
- Indeed IP routing is dynamic only in face of failures or topology changes
- Load-dependant, or QoS routing is a topic discussed, researched-on and tested since 30 years, but never implemented

---

---

---

---

---

---

---

---

### TCP Congestion Control Alternative Options

- Congestion can only be controlled by limiting total amount of data entering network
  - I.E. making  $\rho < 1$
- ICMP source Quench message is crude and not effective ... and really not implements in hosts
- RSVP may help but not widely implemented
- No other Connection Admission Control Techniques available in the Internet



---

---

---

---

---

---

---

---

### TCP Congestion Control is Difficult

- IP is connectionless and stateless, with no provision for detecting or controlling congestion
- TCP only provides end-to-end flow control
- No cooperative, distributed algorithm to bind together various TCP entities
- No cooperation between IP and TCP
  - When links/routers are congested IP drops packets
  - TCP retransmit them ... **increasing the load!!!**



---

---

---

---

---

---

---

---

### TCP Flow and Congestion Control

- The rate at which a TCP entity can transmit is determined by rate of incoming ACKs to previous segments with new credit
- Rate of Ack arrival determined by round-trip path between source and destination
- Bottleneck may be destination or internet
- Sender cannot tell which
- Only the internet bottleneck can be due to congestion



---

---

---

---

---

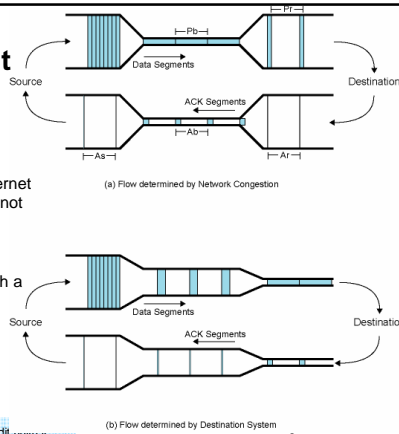
---

---

---

## TCP Segment Pacing

- Congestion in the Internet and at the receiver cannot be distinguished
- Both related with transmission window
- TCP handles both with a single window, which creates a lot of complexities



Renato.LoCigno@dit.unitn.it

---

---

---

---

---

---

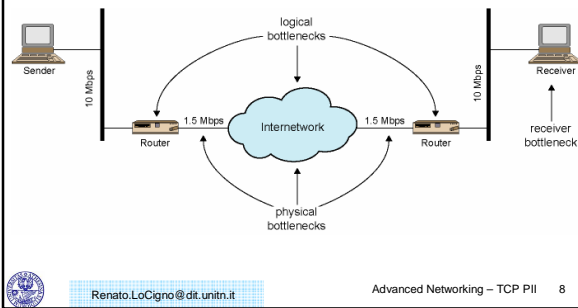
---

---

---

---

## Context of TCP Flow and Congestion Control



Renato.LoCigno@dit.unitn.it

Advanced Networking – TCP PII 8

---

---

---

---

---

---

---

---

---

---

## Retransmission Timer Management

- Congestion affects RTT → a technique to efficiently and reliably compute RTO is needed
- Three Techniques to calculate retransmission timer (RTO):
- RTT Variance Estimation
- Exponential RTO Backoff
- Karn's Algorithm



Renato.LoCigno@dit.unitn.it

Advanced Networking – TCP PII 9

---

---

---

---

---

---

---

---

---

---

## RTT Variance Estimation (Jacobson's Algorithm)

- 3 sources of high variance in RTT
- If data rate relative low, then transmission delay will be relatively large, with larger variance due to variance in packet size
- Load may change abruptly due to other sources
- Peer may not acknowledge segments immediately




---

---

---

---

---

---

---

---

---

---

## Jacobson's Algorithm

- $SRTT(K + 1) = (1 - g) \times SRTT(K) + g \times RTT(K + 1)$
- $SERR(K + 1) = RTT(K + 1) - SRTT(K)$
- $SDEV(K + 1) = (1 - h) \times SDEV(K) + h \times |SERR(K + 1)|$
- $RTO(K + 1) = SRTT(K + 1) + f \times SDEV(K + 1)$
- $g = 0.125$
- $h = 0.25$
- $f = 2$  or  $f = 4$  (most current implementations use  $f = 4$ )




---

---

---

---

---

---

---

---

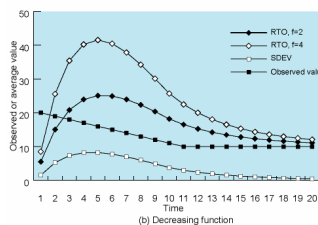
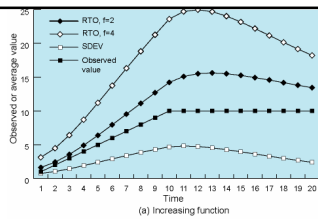
---

---

## Jacobson's RTO Calculation

Figures show the behavior when RTT increase from 0 to 10 or decreases from 20 to 10

The most used implementation grossly overestimates since has a large weight on variance




---

---

---

---

---

---

---

---

---

---

## Two Other Factors

- Jacobson's algorithm can significantly improve TCP performance, but:
  - ANSWER: exponential RTO backoff algorithm
- What RTO to use for retransmitted segments?
  - ANSWER: Karn's algorithm
- Which round-trip samples to use as input to Jacobson's algorithm?
  - ANSWER: Karn's algorithm



---

---

---

---

---

---

---

---

## Exponential RTO Backoff

- Increase RTO each time the **same segment** is retransmitted - backoff process
- Multiply RTO by constant:
  - $RTO = q \times RTO$
- $q = 2 \rightarrow$  binary exponential backoff



---

---

---

---

---

---

---

---

## Which Round-trip Samples?

- If an ack is received for retransmitted segment, there are 2 possibilities:
  - Ack is for first transmission
  - Ack is for second transmission
- TCP source cannot distinguish 2 cases
- No valid way to calculate RTT:
  - From first transmission to ack, or
  - From second transmission to ack?



---

---

---

---

---

---

---

---

## Karn's Algorithm

- Do not use measured RTT to update SRTT and SDEV
- Calculate backoff RTO when a retransmission occurs
- Use backoff RTO for segments until an ack arrives for a segment that has not been retransmitted
- Then use Jacobson's algorithm to calculate RTO



---

---

---

---

---

---

---

---

## Window Management

- Slow start
- Congestion Avoidance
- Dynamic window sizing on congestion
- Fast retransmit
- Fast recovery
- Limited transmit



---

---

---

---

---

---

---

---

## Slow Start

- $awnd = \text{MIN}[ \text{credit}, cwnd ]$ 
  - where
  - $awnd$  = allowed window in segments
  - $cwnd$  = congestion window in segments
  - $credit$  = amount of unused credit granted in most recent ack ( $rcwn$ )
- $cwnd = 1$  for a new connection and increased by 1 for each ack received, up to a maximum
- Most implementations are not compliant and start from 2 to "counter" delayed ACKs



---

---

---

---

---

---

---

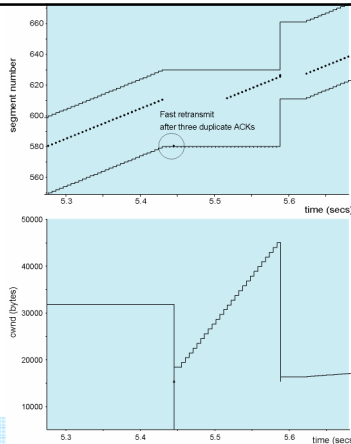
---







## Fast Recovery Example of window evolution (simplified)



Renato.LoCigno@dit.unitn.it

---

---

---

---

---

---

---

---

## Limited Transmit

- If congestion window at sender is small, fast retransmit may not get triggered,
  - e.g.,  $cwnd = 3$
- Under what circumstances does sender have small congestion window?
- Is the problem common?
- If the problem is common, why not reduce number of duplicate acks needed to trigger retransmit?



Renato.LoCigno@dit.unitn.it

Advanced Networking – TCP PII 29

---

---

---

---

---

---

---

---

## Limited Transmit Algorithm

- Sender can transmit new segment when 3 conditions are met:
  - Two consecutive duplicate acks are received
  - Destination advertised window allows transmission of segment
  - Amount of outstanding data after sending is less than or equal to  $cwnd + 2$  (i.e. the window was exactly 3)
- Rarely implemented, solves just a limited number of cases
- What about correlated losses?



Renato.LoCigno@dit.unitn.it

Advanced Networking – TCP PII 30

---

---

---

---

---

---

---

---