Introduction to TCP Congestion Control

Lecture 1

Subir Varma

Book

“Internet Congestion Control” by Subir Varma, 1rst Edition, Morgan Kaufman (2015).

Pre-Requisites

- Introductory Course on Computer Networking

TCP Congestion Control

- TCP’s congestion control algorithm has been described as the largest man-made feedback controlled system in the world.
- It enables hundreds of millions of devices, ranging from huge servers to the smallest PDA, to coexist together and make efficient use of existing bandwidth resources.
- It does so over link speeds that vary from a few kilobits per second to tens of gigabits per second, networks as varied as cellular wireless, data center networks and satellite links.

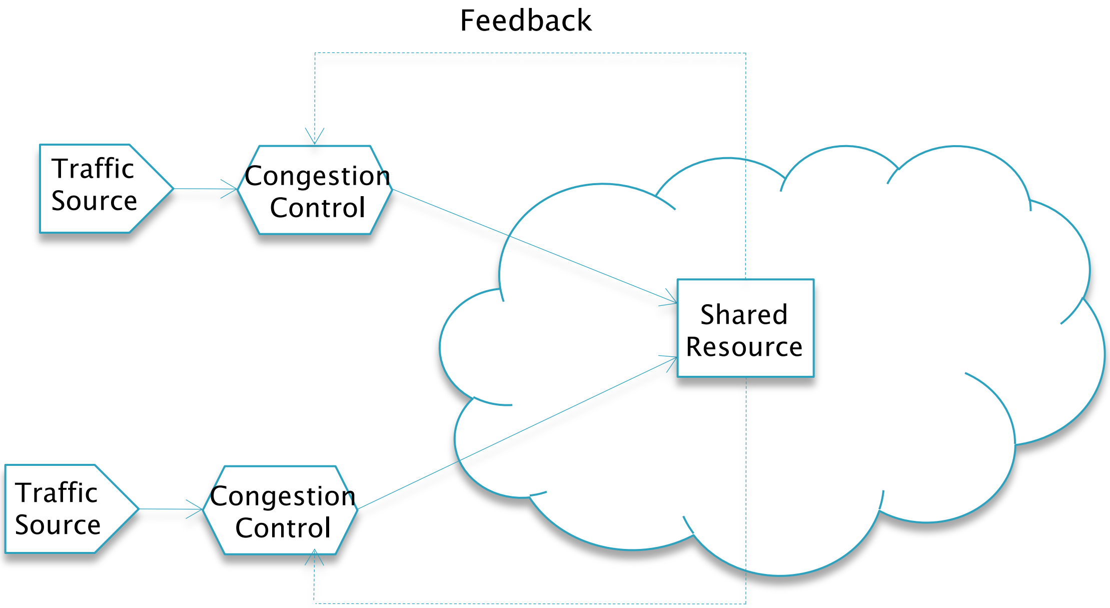
Subject of this Course: How and how well is it able to do this?

History of TCP Congestion Control

- Pre-1987: TCP used a simple static-window controlled algorithm. This led to a catastrophic congestion collapse in 1986.
- TCP congestion control went through several important enhancements in the years since 1986, starting with Van Jacobsen's invention of the Slow Start and Congestion Avoidance algorithms, described in his seminal paper*
- As a result, a congestion control algorithm called TCP Slow Start (also sometimes called TCP Tahoe) was put into place in 1987 and 1988, which in its current incarnation as TCP Reno, remains one of the dominant algorithms used in the Internet today.
- Since then, the Internet has evolved and now features transmissions over wireless media, link speeds that are several orders of magnitudes faster, widespread transmission of video streams, and the recent rise of massive data centers.
- The congestion control research and development community has successfully met the challenge of modifying the original congestion control algorithm and has come up with newer algorithms to address these new types of networks

https://ee.lbl.gov/papers/congavoid.pdf

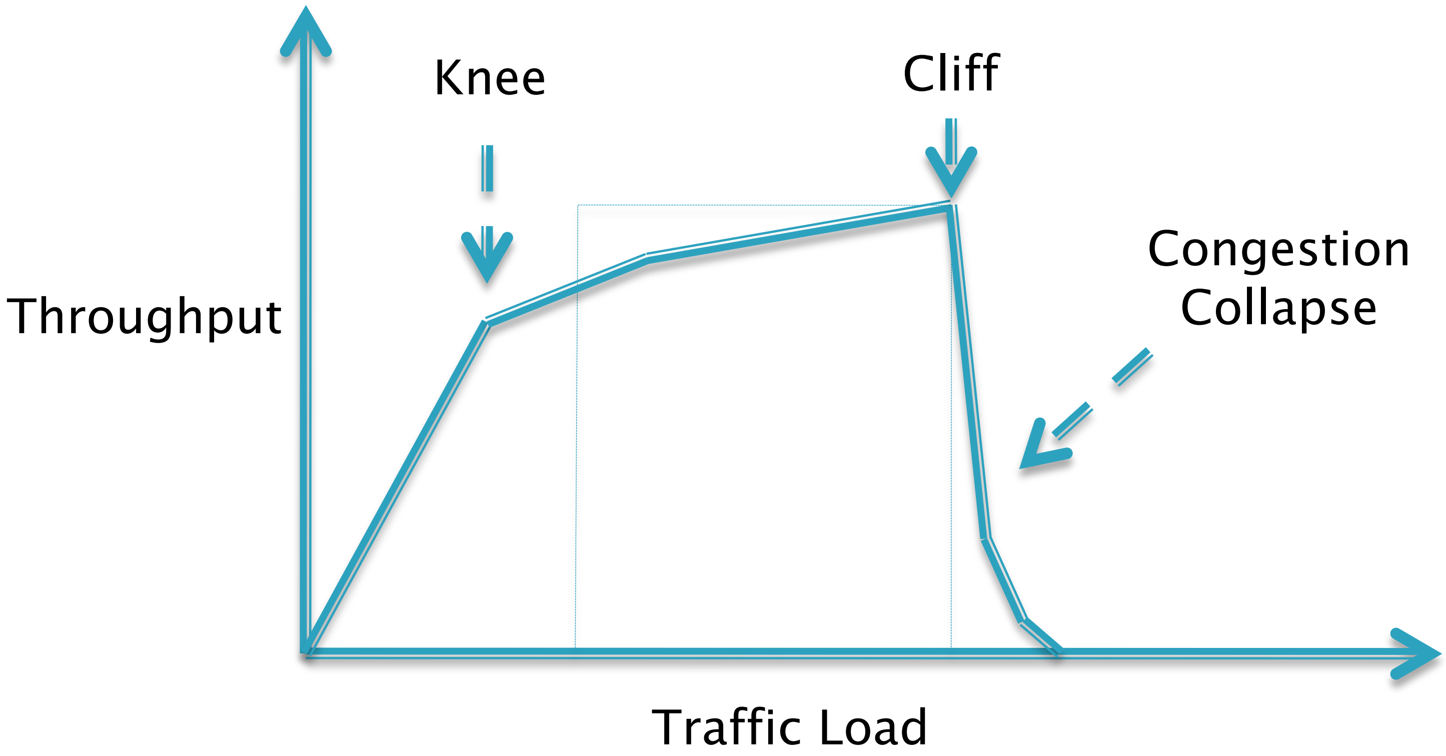
The Congestion Control Problem



The Congestion Control Problem

- The problem of congestion control arises whenever multiple distributed agents try to make use of shared resources.
- It arises widely in different scenarios, including traffic control on highways or air traffic control. This book focuses on the specific case of packet data networks,
- The high-level objective of congestion control is to provide good utilization of network resources and at the same time provide acceptable performance for the users of the network.

Problem Definition



Throughput vs Load

Problem Definition

- If the load is small, the throughput keeps up with the increasing load. After the load reaches the network capacity, at the “knee” of the curve, the throughput stops increasing. Beyond the knee, as the load increases, queues start to build up in the network, thus increasing the end-to-end latency without adding to the throughput. The network is then said to be in a state of congestion.

- If the traffic load increases even further and gets to the “cliff” part of the curve, then the throughput experiences a rather drastic reduction. This is because queues within the network have increased to the point that packets are being dropped. If the sources choose to retransmit to recover from losses, then this adds further to the total load, resulting in a positive feedback loop that sends the network into congestion collapse.

Problem Definition

- To avoid congestion collapse, each data source should try to maintain the load it is sending into the network in the neighborhood of the knee.
- An algorithm that accomplishes this objective is known as a congestion avoidance algorithm; a congestion control algorithm tries to prevent the system from going over the cliff.
- This task is easier said than done and runs into the following challenge:
  - The location of the knee is not known and in fact changes with total network load and traffic patterns.
- Hence, the problem of congestion control is inherently a distributed optimization problem in which each source has to constantly adapt its traffic load as a function of feedback information it is receiving from the network as it tries to stay near the knee of the curve.

Problem Definition

- The original TCP Tahoe algorithm and its descendants fall into the congestion control category because in the absence of support from the network nodes, the only way they can detect congestion is by filling up the network buffers and then detecting the subsequent packet drops as a sign of congestion.

- The trend in the last ten years has been to move away from buffer filling type algorithms, and towards algorithms that are able to detect congestion by looking at other types of feedback, such as delay. This is an active area of research.

Design Decisions

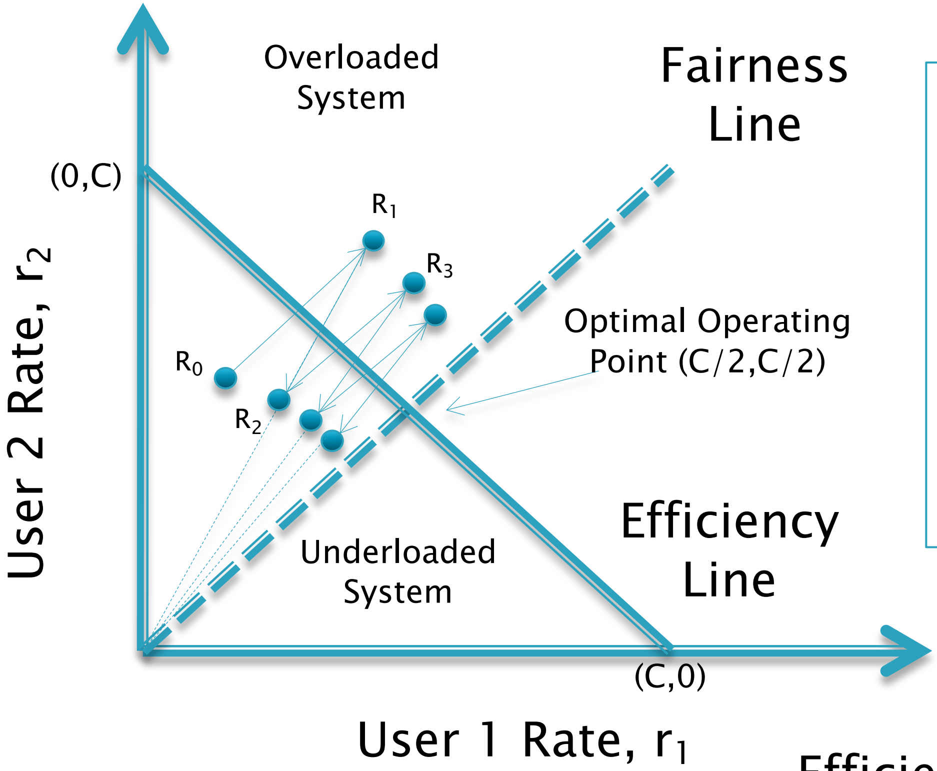
- How does the network signal to the end points that it is congested (or conversely, not congested)?
- What is the best Traffic Rate Increment rule in the absence of congestion?
- What is the best Traffic Rate Decrement rule in the presence of congestion?
- How can the algorithm ensure fairness among all connections sharing a link, and what measure of fairness should be used?
- Should the congestion control algorithm prioritize high link utilization of low end-to-end latency?
- Should the congestion control be implemented on an end-to-end or on a hop-by-hop basis?
- Should the traffic control be exercised using a windows-based mechanism or a rate-based mechanism?

A Simple Rate Allocation Problem

- Consider a network with 2 data sources and define $r_i(t)$ to be the data rate (i.e., load) of the i^{th} data source.
- Also assume that the system is operating near the knee of the curve so that the network rate allocation to the i^{th} source after congestion control is also given by $r_i(t)$.
- We assume that the system operation is synchronized and happens in discrete time so that the rate at time $(t + 1)$ is a function of the rate at time t, and the network feedback received at time t.
- We will also assume that the network congestion feedback is received at the sources instantaneously without any delay.

Chiu DM, Jain R. Analysis of increase and decrease algorithms for congestion avoidance in computer networks. Comput Netw ISDN Syst 1989;17(1).

Rate Allocation

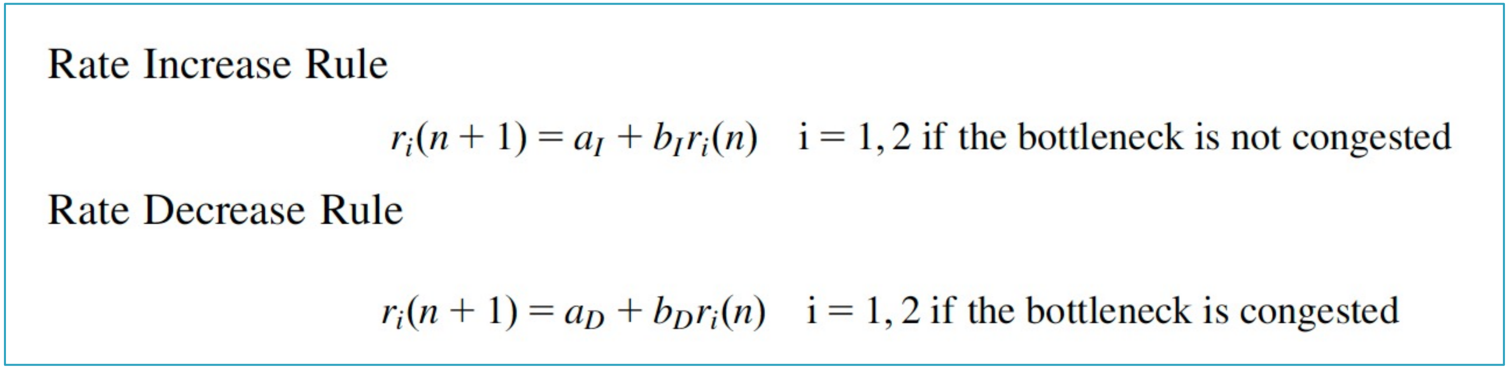


Any rate allocation (r_1,r_2) can be represented by a point in a two-dimensional space.
The objective of the optimal congestion control algorithm is to bring the system close to the optimal point (C/2, C/2) irrespective of the starting position.

Efficiency: r_1+r_2 should be close to C

Fairness: $F = \frac{(r_1+r_2)^2}{2(r_1^2+r_2^2)}$ should be close to 1

Rate Allocation Rules



Rate Increase Rule

$$r_i(n+1) = a_I + b_I r_i(n) \quad \text{i} = 1, 2 \text{ if the bottleneck is not congested}$$

Rate Decrease Rule

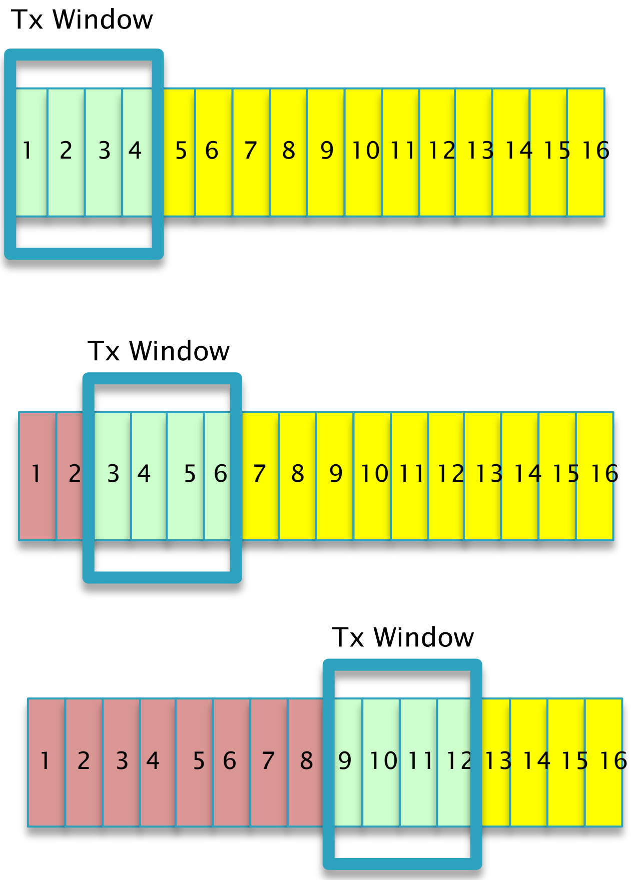
$$r_i(n+1) = a_D + b_D r_i(n) \quad \text{i} = 1, 2 \text{ if the bottleneck is congested}$$

- $a_I = a_D = 0$, $b_I > 1$ and $0 < b_D < 1$ corresponds to multiplicative increase/multiplicative decrease (MIMD) controls.
- $b_I = b_D = 1$ corresponds to additive increase/additive decrease (AIAD) controls.
- $b_I = 1$, $a_D = 0$ and $0 < b_D < 1$ corresponds to AIMD controls.
- $a_I = 0$, $b_D = 1$ corresponds to multiplicative increase/additive decrease (MIAD) controls.

- The AIMD Rule converges towards the optimal point on the equifairness line
- The AIAD and MIMD Rules oscillate at non-optimal parts of the graph
- The MIAD Rule is not stable

Window based Congestion Control

Window based Congestion Control



🟨 Not Yet Sent

🟩 Sent but not ACK’d

🟥 Sent and ACK’d

Transmit Window Operation

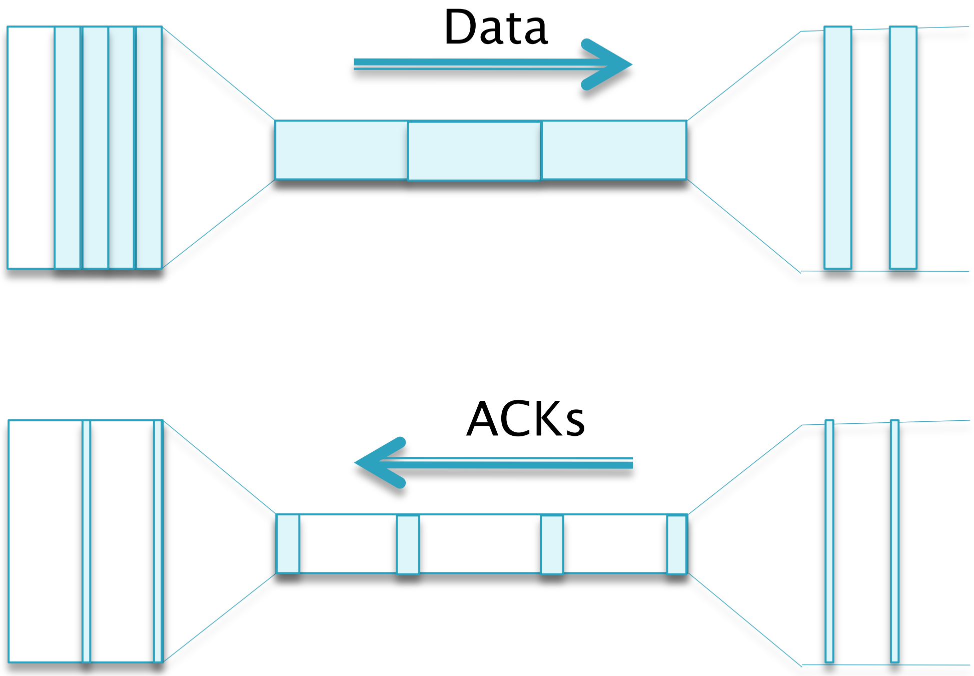
Window based Congestion Control

- Assume that the packets are numbered sequentially from 1 and up. When a packet gets to the destination, an ACK is generated and sent back to the source. The ACK carries the sequence number (SN) of packet that it is acknowledging.

- The source maintains two counters, Ack'd and Sent. Whenever a packet is transmitted, the Sent counter is incremented by 1, and when an ACK arrives from the destination, the Ack'd counter is incremented by 1. Because the window size is N packets, the difference un_ack = (Sent - Ack'd) is not allowed to exceed N. When unAck'd = N, then the source stops transmitting and waits for more ACKs to arrive.

- ACKs can be of two types: An ACK can either acknowledge individual packets, which is known as Selective Repeat ARQ, or it can carry a single SN that acknowledges multiple packets with lower SNs. The latter strategy is called Go Back N.

Self Clocking Property



- If a node is congested, then the rate at which ACKs are sent back is precisely equal to the rate at which the congested node is able to serve the packets in its queue.
- As a result, the source sending rate gets autoregulated to the rate at which the bottleneck node can serve the packets from that source.
- This is also known as the “self-clocking” property and is one of the benefits of the window-based congestion control over the rate-based schemes.

Issues with Static Window based Schemes

1. When the algorithm is started, there is no mechanism to regulate the rate at which the source sends packets into the network. Indeed, the source may send a whole window full of packets back to back, thus overwhelming nodes along the route and leading to lost packets. When the ACKs start to flow back, the sending rate settles down to the self-clocked rate.

2. Given a window size of W and a round trip time of T, the average throughput R is given by the formula R = W/T in steady state. If the network is congested at a node, then the only way to clear it is by having the sources traversing that node reduce their sending rate. However, there is no way this can be done without reducing the corresponding window sizes. If we do introduce a mechanism to reduce window sizes in response to congestion, then we also need to introduce a complementary mechanism to increase window sizes after the congestion goes away.

3. If the source were to vary its window size in response to congestion, then there needs to be a mechanism by which the network can signal the presence of congestion back to the source.

Rate Based Congestion Control

- Instead of using a window to regulate the rate at which the source is sending packets into the network, the congestion control algorithm may alternatively release packets using a packet rate control mechanism.
- This can be implemented by using a timer to time the interpacket intervals.
- Examples of rate based mechanisms:
  - Asynchronous Transfer Mode (ATM) Forum Available Bit Rate (ABR) traffic management scheme
  - Some proposed alternatives to TCP Reno, such as TCP FAST, BBR etc
  - The IEEE802.1Qau algorithm, also known as Quantum Congestion Notification (QCN)

Issues with Rate Based Control

- Window-based schemes have a built-in choke-off mechanism in the presence of extreme congestion when the ACKs stop coming back. However, in rate-based schemes, there is no such cut-off, and the source can potentially keep pumping data into the network.

- Rate-based schemes require a more complex implementation at both the end nodes and in the network nodes. For example, at the end node, the system requires a fine-grained timer mechanism to control the rates.

Hop-by-Hop Congestion Control

- Hop-by-hop congestion control typically uses a window-based congestion control scheme at each hop and operates as follows: If the congestion occurs at the nth node along the route and its buffers fill up, then that node stops sending ACKs to the (n-1)rst node. Because the (n-1)rst node can no longer transmit, its buffers will start to fill up and consequently will stop sending ACKs to the (n-2)nd node. This process continues until the backpressure signal reaches the source node.

- Benefits:
  - Hop-by-hop schemes can do congestion control without letting the congestion deteriorate to the extent that packets are being dropped.
  - Individual connections can be controlled using hop-by-hop schemes because the window scheme described earlier operated on a connection-by-connection basis. Hence, only the connections that are causing congestion need be controlled, as opposed to all the connections originating at the source node or traversing a congested link.
  - Hop-by-Hop reacts to congestion much faster than end-to-end schemes, which typically take a round trip of latency or longer

- Issues: The benefits require that the hop-by-hop window be controlled on a per-connection basis. This can lead to greater complexity of the routing nodes within the network

Network Feedback

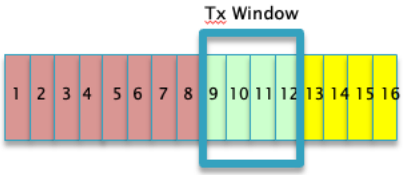
- The source can either implicitly infer the existence of network congestion, or the network can send it explicit signals that it is congested. Both of these mechanisms are used in the Internet.
- Regular TCP is an example of a protocol that implicitly infers network congestion by using the information contained in the ACKs coming back from the destination
- TCP Explicit Congestion Notification (ECN) is an example of a protocol in which the network nodes explicitly signal congestion

TCP Reno

Loss Based Congestion Notification

TCP Reno

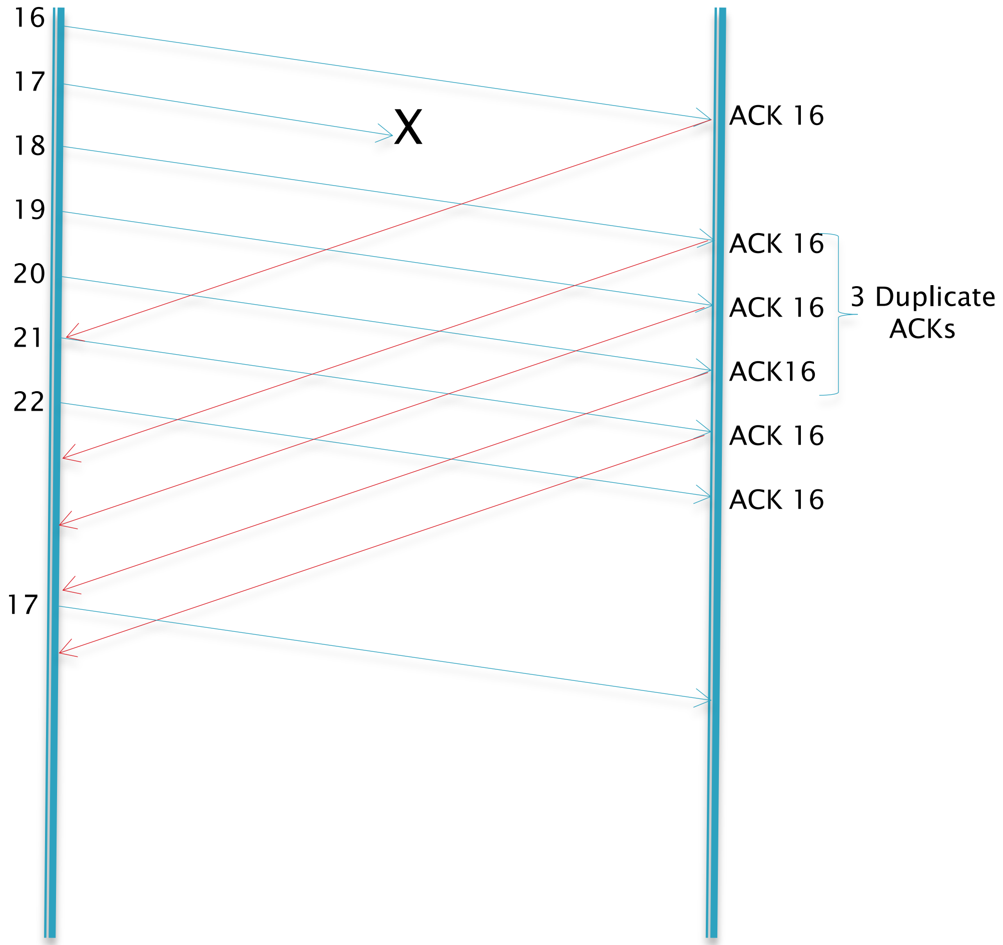
- TCP congestion control can be classified as a window-based algorithm that operates on an end-to-end basis and uses implicit feedback from the network to detect congestion.
- The sender maintains two counters:
  - Transmit Sequence Number (TSN) is the SN of the next byte to be transmitted.
  - Ack'd Sequence Number (ASN) is the SN of the next byte that the receiver is expecting.
  - To maintain the window operation, the sender enforces the following rule: $(SN - ASN) \leq W$
- The receiver maintains a single counter:
  - This is the SN of the next byte that the receiver is expecting.



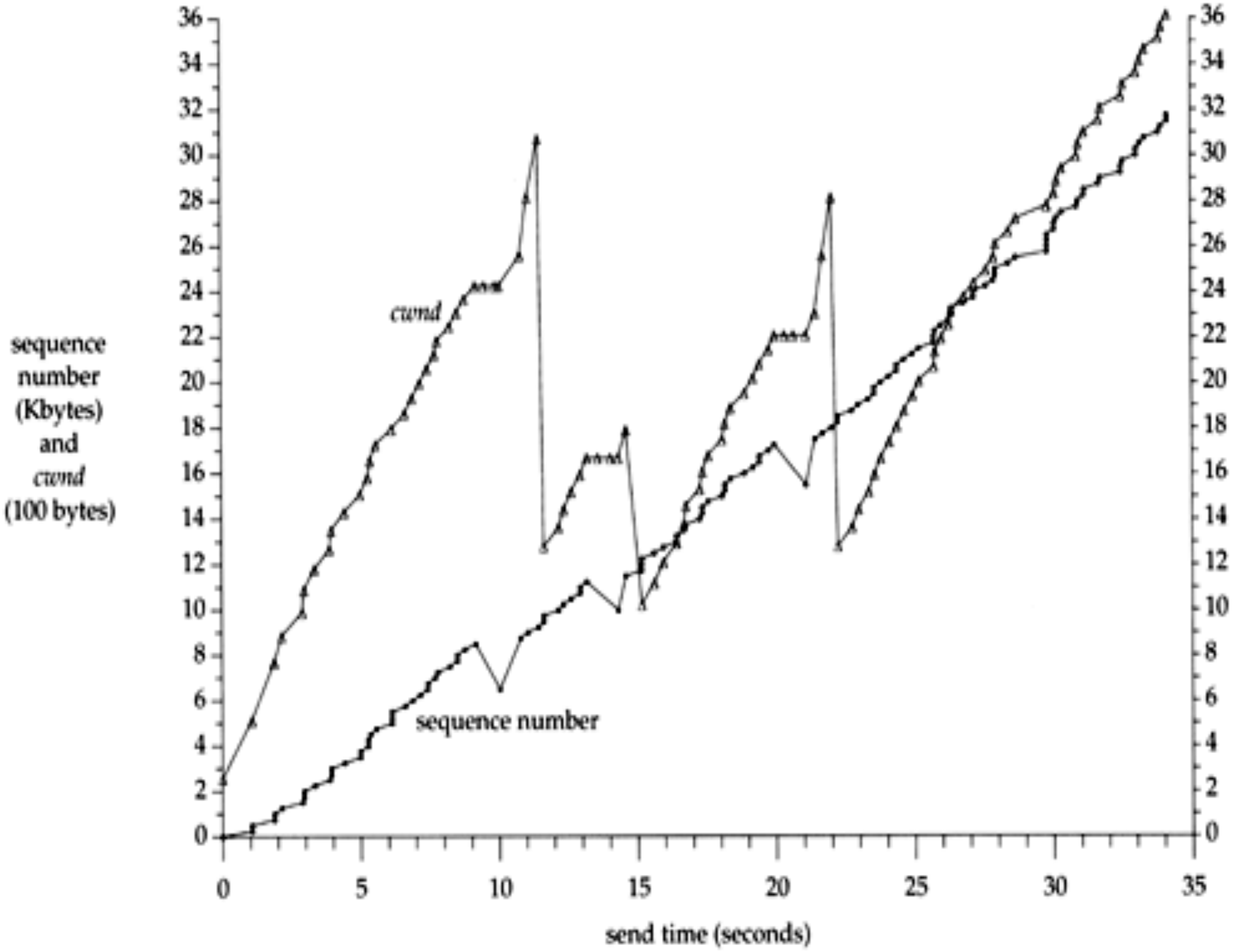
TCP Reno

- The receiver inserts the NSN value into the ACK field of the next packet that is being sent back.
- Note that ACKs are cumulative, so that if one of them is lost, then the next ACK makes up for it. Also, an ACK is always generated even if SN of a received packet does not match the NSN.
- In this case, the ACK field will be set to the old NSN again. This is a so-called Duplicate ACK
- The presence of Duplicate ACKs can be used by the transmitter as an indicator of lost packets, and indirectly, network congestion.

TCP Reno: Fast Re-Transmit



TCP New Reno: Fast Recovery



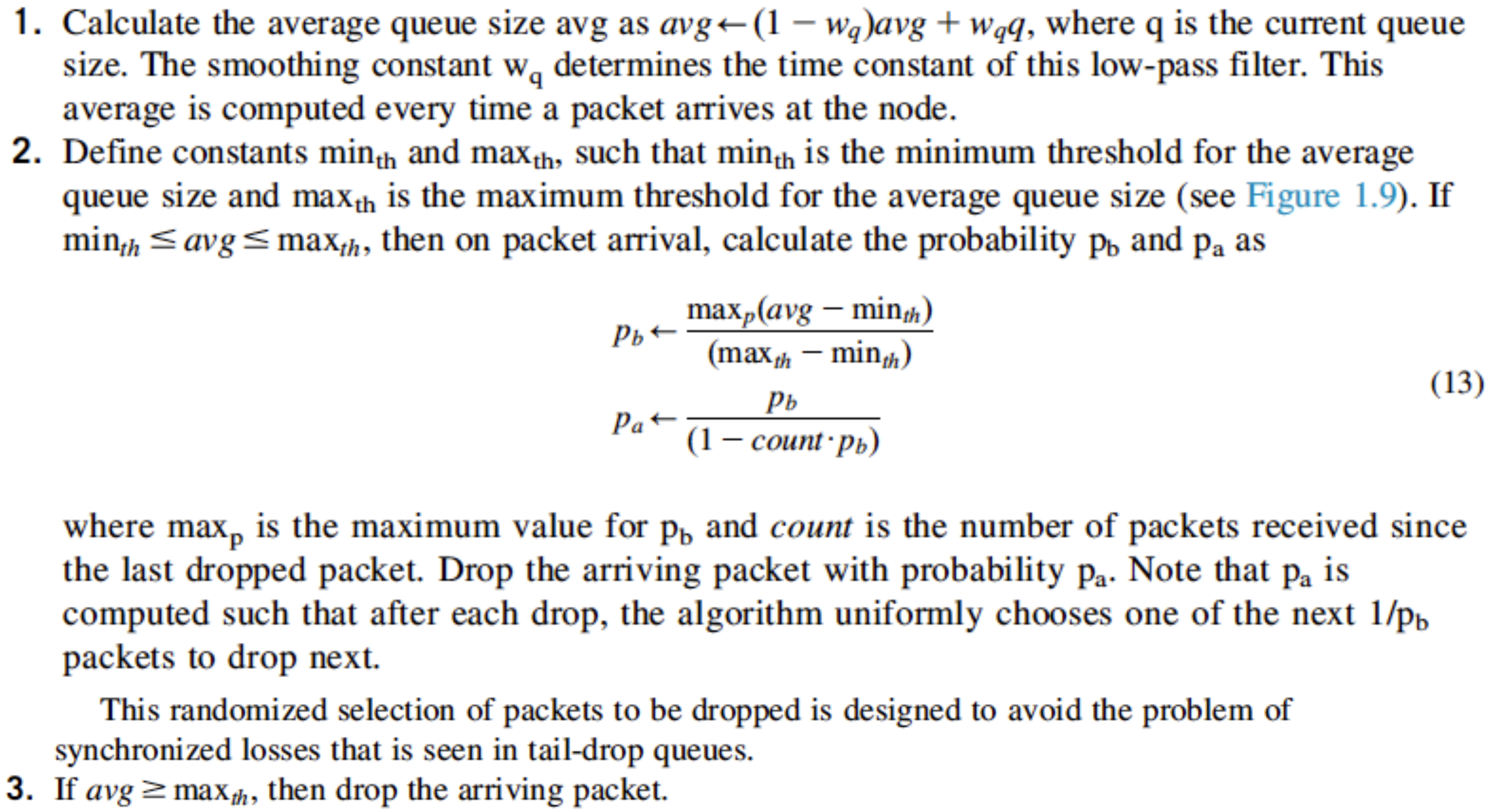
- On detection of Duplicate ACKs, set ssthresh ← W/2, W ← W+3MSS and transmit next 3 packets
- On detection of every Duplicate ACK thereafter increase W ← W +1 and transmit additional packets
- When the first lost packet is finally ACK'd them set W ← ssthresh

Effect of Fast Recovery:
- If multiple packets are lost then avoids Duplicate ACKs for each one of them
- Avoids burst transmission of A window full of packet when the first lost packet is finally ACK'd

Network Feedback

- Sophistication of algorithm constrained by the processing power available in the switch or router, and also the link speed.
- Passive Queue Management: Tail Drop
- Issues with Tail Drop:
  - Can lead to excess latencies across the network if the bottleneck node happens to have a large buffer size. Note that the buffer need not be full for the session to attain full link capacity; hence, the excess packets queued at the node add to the latency without improving the throughput
  - If multiple TCP sessions are sharing the buffer at the bottleneck node, then the algorithm can cause synchronized packet drops across sessions. This causes all the affected sessions to reduce their throughput, which leads to a periodic pattern, resulting in underutilization of the link capacity.

Active Queue Management: The RED Algorithm



1. Calculate the average queue size avg as $avg \leftarrow (1 - w_q)avg + w_q q$, where q is the current queue size. The smoothing constant w_q determines the time constant of this low-pass filter. This average is computed every time a packet arrives at the node.
2. Define constants $\min_{th}$ and $\max_{th}$, such that $\min_{th}$ is the minimum threshold for the average queue size and $\max_{th}$ is the maximum threshold for the average queue size (see Figure 1.9). If $\min_{th} \leq avg \leq \max_{th}$, then on packet arrival, calculate the probability p_b and p_a as

$$\begin{aligned} p_b &\leftarrow \frac{\max_p(avg - \min_{th})}{(\max_{th} - \min_{th})} \\ p_a &\leftarrow \frac{p_b}{(1 - count \cdot p_b)} \end{aligned} \qquad (13)$$

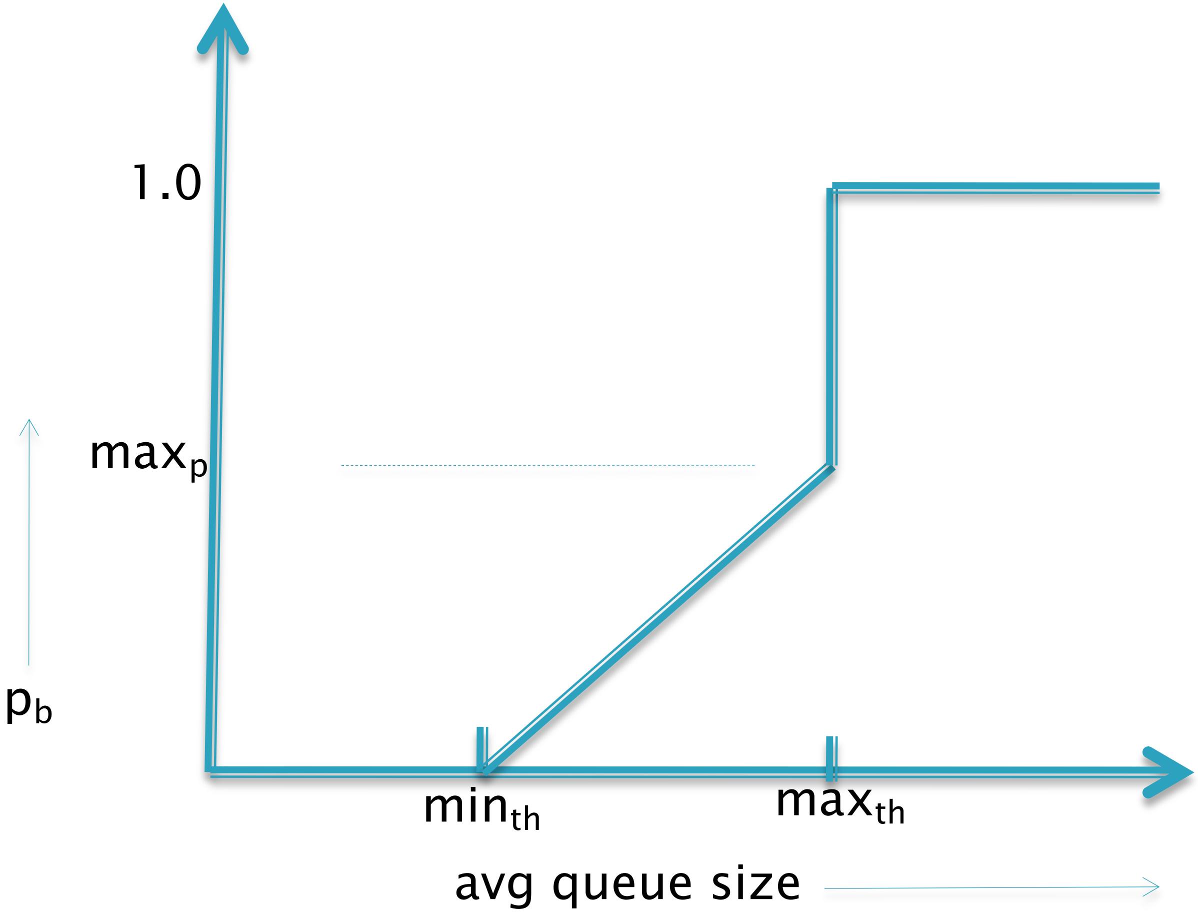
   where $\max_p$ is the maximum value for p_b and *count* is the number of packets received since the last dropped packet. Drop the arriving packet with probability p_a. Note that p_a is computed such that after each drop, the algorithm uniformly chooses one of the next $1/p_b$ packets to drop next.

   This randomized selection of packets to be dropped is designed to avoid the problem of synchronized losses that is seen in tail-drop queues.

3. If $avg \geq \max_{th}$, then drop the arriving packet.

Floyd S, Jacobsen V. Random early detection gateways for congestion avoidance. IEEE/ACM Trans Netw 1993;1:397-413.

Active Queue Management: The Random Early Detection (RED) Algorithm



Packet Discard Probability in RED

Issues with RED

- RED is not an ideal AQM scheme because there are issues with its responsiveness and ability to suppress queue size oscillations as the link capacity or the end-to-end latency increases.
- Choosing a set of RED parameters that work well across a wide range of network conditions is also a nontrivial problem, which has led to proposals that adapt the parameters as network conditions change.
- More recent variations on RED
  - ECN (Explicit Congestion Notification): Instead of dropping a packet, the switch marks the ECN bit instead. When it reaches the Source via the ACK, it activates congestion control and reduces window size.
  - Data Center Congestion Control Protocol (DCTCP, Lecture 8: Keeps the threshold K fixed and keeps track of the number of packets that encounter a queue size greater than K)

TCP Vegas

Delay Based Congestion Notification

TCP Vegas: A Delay based CC Algorithm

- TCP Vegas estimates the level of congestion in the network by calculating the difference in the expected and actual data rates, which it then uses to adjust the TCP window size. Assuming a window size of W and a minimum round trip latency of T seconds, the source computes an expected throughput R_E once per round trip delay, by

$$R_E = \frac{W}{T}$$

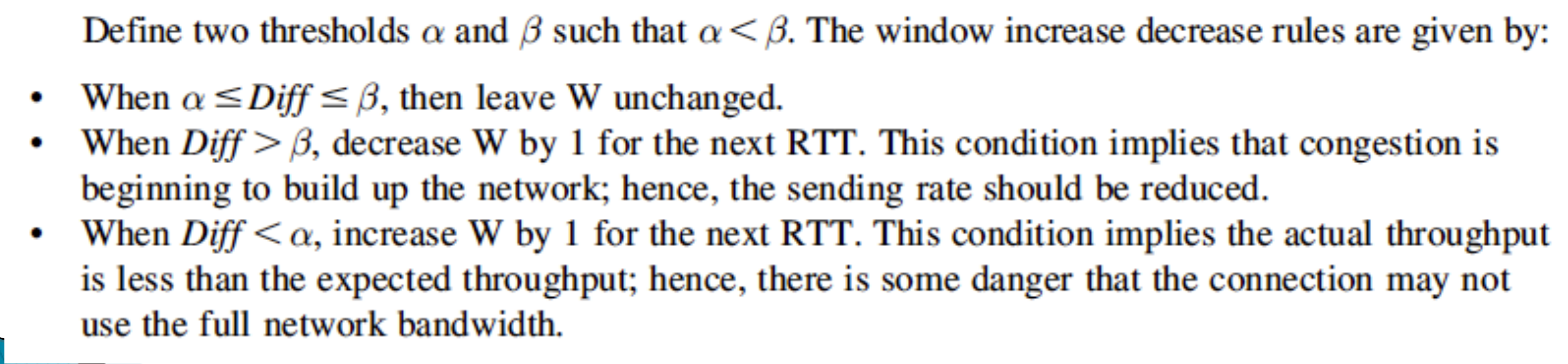
- The source also estimates the current throughput R by using the actual round trip time T_s according to

$$R = \frac{W}{T_s}$$

- The source then computes the quantity Diff given by

$$Diff = T(R_E - R) = R(T_s - T)$$

- By Little's law, $R(T_s–T)$ equals the number of packets belonging to the connection that are queued in the network and hence serves as a measure of congestion.



Define two thresholds α and β such that $\alpha < \beta$. The window increase decrease rules are given by:

- When $\alpha \le Diff \le \beta$, then leave W unchanged.
- When $Diff > \beta$, decrease W by 1 for the next RTT. This condition implies that congestion is beginning to build up the network; hence, the sending rate should be reduced.
- When $Diff < \alpha$, increase W by 1 for the next RTT. This condition implies the actual throughput is less than the expected throughput; hence, there is some danger that the connection may not use the full network bandwidth.

Brakmo LS, Peterson LL. End-to-end congestion avoidance on a global Internet. IEEE JSAC 1995;13:1465-80.

TCP Vegas (cont)

Benefits:

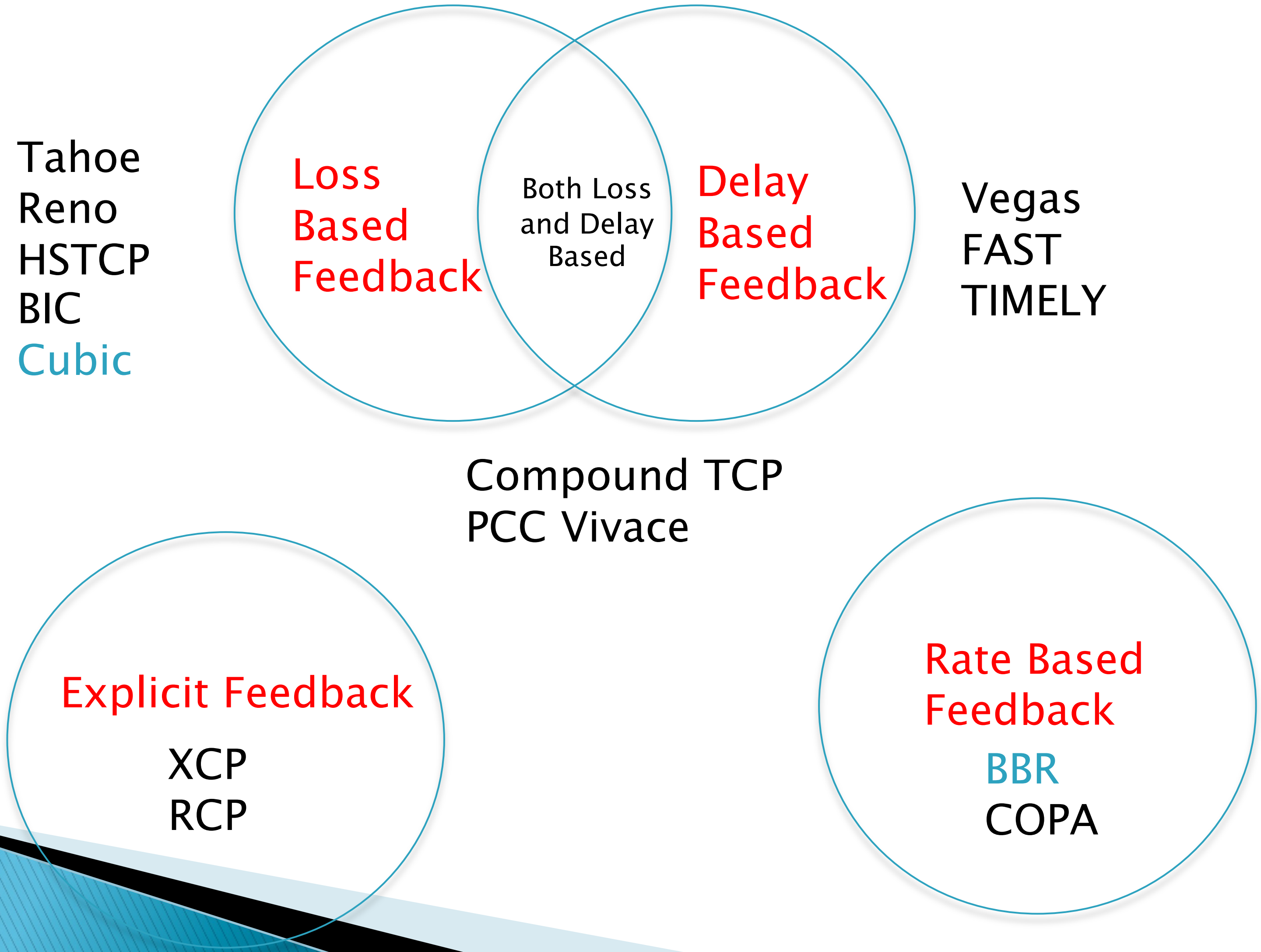
- TCP Vegas tries to maintain the “right” amount of queued data in the network. Too much queued data will cause congestion, and too little queued data will prevent the connection from rapidly taking advantage of transient increases in available network bandwidth. Note that this mechanism also eliminates the oscillatory behavior of TCP Reno while reducing the end-to-end latency and jitter because each connection tends to keep only a few packets in the network.

Issues:

- When TCP Vegas and Reno compete for bandwidth on the same link, then Reno ends with a greater share because it is more aggressive in grabbing buffers, but Vegas is conservative and tries to occupy as little space as it can.
- Problems with latency estimates: How can the system ensure that the minimum latency estimate is accurate? How to differentiate the ‘standing queue’ from transient bursts?

  Because of these reasons, Vegas is not widely deployed despite its desirable properties. Recent protocols such as BBR and Copa were designed with the objective of solving these problems.

Congestion Control Algorithms Taxonomy: Network Feedback



Tahoe
Reno
HSTCP
BIC
Cubic

Loss Based Feedback

Both Loss and Delay Based

Delay Based Feedback

Vegas
FAST
TIMELY

Compound TCP
PCC Vivace

Explicit Feedback

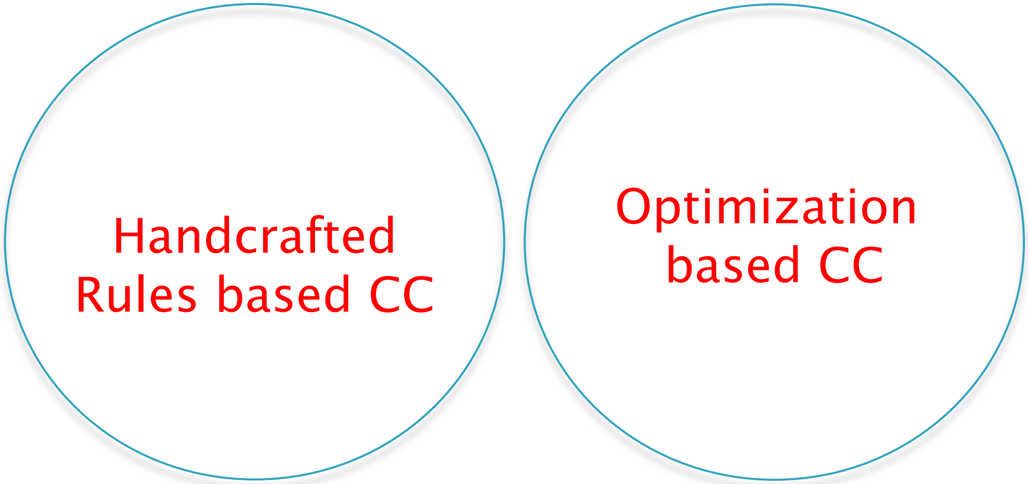
XCP
RCP

Rate Based Feedback

BBR
COPA

Congestion Control Algorithms Taxonomy: Algorithm Type

Tahoe
Reno
Cubic
Vegas



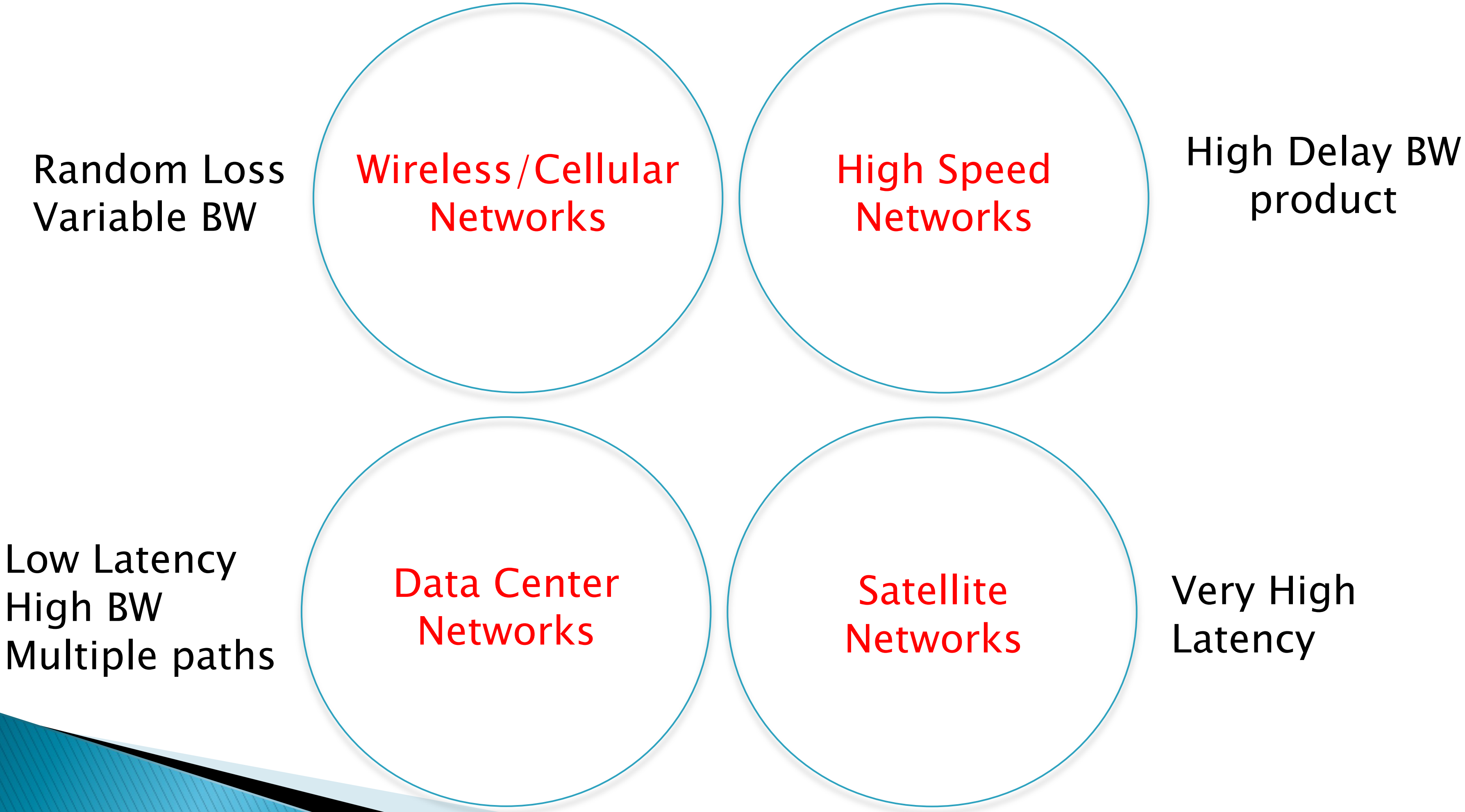
Handcrafted Rules based CC

Optimization based CC

Remy
PCC Allegro
PCC Vivace
Deep Reinforcement Learning based CC

- Carry out actions that increase Utility
- Shows influence of Machine Learning on CC Algorithms

Congestion Control Algorithms Taxonomy: Applications



Random Loss
Variable BW

Wireless/Cellular Networks

High Speed Networks

High Delay BW product

Low Latency
High BW
Multiple paths

Data Center Networks

Satellite Networks

Very High Latency

Lecture Schedule

- **Lecture 1 – Introduction to TCP Congestion Control**: Objectives of Congestion Control, Types of CC Algorithms, TCP Reno, Network Feedback, TCP Vegas.
- **Lecture 2 – Congestion Control Models Part 1**: TCP Throughput Analysis, Effect of Buffer Size, Window Size and Link Error Rate on throughput, Fluid Flow Model for CC, Throughput analysis for more general models.
- **Lecture 3 – Congestion Control Models Part 2**: Network wide models of CC using Optimization Theory, Stability Analysis of TCP using Classical Control Theory, The Proportional and Proportional + Integral Controllers, The Averaging Principle.
- **Lecture 4 – Congestion Control in Broadband Wireless Systems**: Split Connection TCP, Congestion Control over Lossy Links, TCP Westwood, Link Level Error Correction, the Bufferbloat Problem, CoDel Active Queue Management.
- **Lecture 5 – Congestion Control in High Speed Networks**: Design Issues in High Speed Networks, Analysis using Response Functions, RTT Fairness, High Speed TCP (HSTCP), TCP BIC
- **Lecture 6 – Congestion Control in High Speed Networks**: TCP Cubic, Compound TCP, Yeah TCP, Compatibility with TCP Reno, Throughput analysis

Lecture Schedule (cont)

- **Lecture 7 – Congestion Control in High Speed Networks**: TCP BBR (Bottleneck BW and Rate Control), BDP Estimates, RTT Probing, Multiple flows case, Interaction with loss based protocols, XCP (Express Control Protocol), RCP (Rate Control Protocol)

- **Lecture 8 – Congestion Control in Data Center Networks**: Data Center Network Architecture and Traffic Patterns, Data Center TCP, Deadline Aware Congestion Control, Multipath TCP, the Incast Problem.

- **Lecture 9 – Delay Bounding Congestion Control Algorithms**: TCP Vegas, TCP FAST, TCP COPA, TIMELY, Bounds on performance of delay bounding algorithms.

- **Lectures 10 – Optimization based Congestion Control Algorithms**: Global Optimization based algorithms, Remy and Indigo Congestion Control, Local Optimization based algorithms, Performance Oriented Congestion Control (PCC), PCC Allegro, PCC Vivace, Deep Reinforcement Learning based algorithms

- **Lecture 11 – Video Delivery over TCP**: Video traffic characteristics, Adaptive Streaming, Video Rate Control, The DASH framework, Adaptive Bit Rate (ABR) Algorithms, Rate based and Buffer size based algorithms, ABR/TCP interaction.

Further Reading

- Chapter 1 of Internet Congestion Control
- Chapters 20 and 21 of “TCP/IP Illustrated, Vol 1” by Fall and Stevens