TCP Theory

Mark Handley
Professor of Networked Systems
UCL Computer Science Dept.

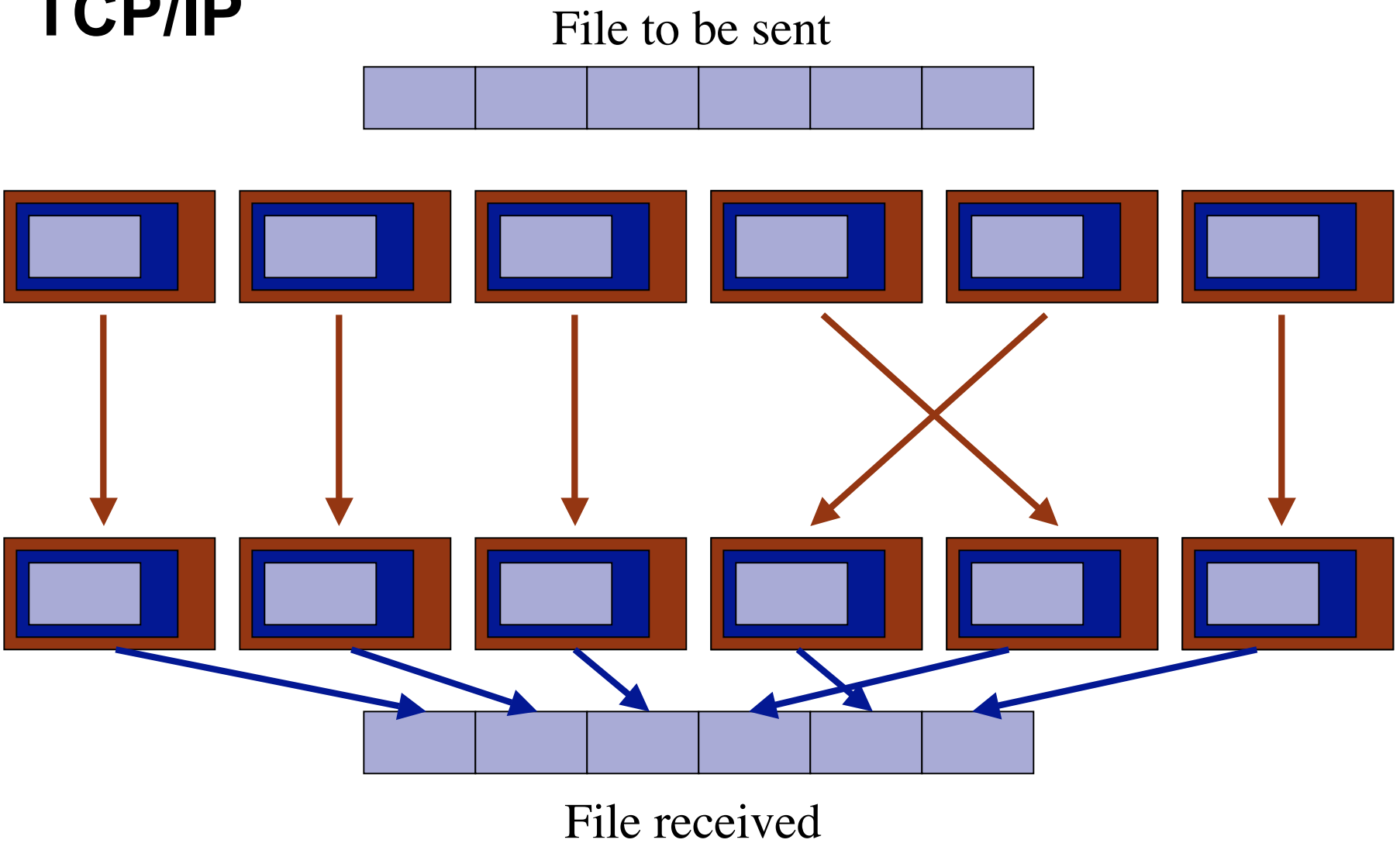
What is TCP?

- Transmission Control Protocol
  - Standardized in 1981 in RFC 793
  - Tweaked endlessly ever since.
- Provides a *reliable bytestream abstraction* end-to-end across an unreliable packet network.
- Transfers the vast majority of the traffic in the Internet.
  - Email, ftp, WWW, ssh, telnet, Kazaa, BGP routing, X windows, and many others are all built on top of TCP.

TCP/IP

- IP handles addressing (and some other stuff).
  - Routers look at the IP headers to move packets from sender A to receiver B.
  - Sometimes the routers will break, or get congested, or re-route your traffic over a piece of wet string, and then they'll drop packets.
- TCP packets are carried in IP packets.
  - The routers don't look at TCP.

TCP/IP



So what can go wrong?

- Packets can get corrupted.
- Packets can get lost.
- Packets can get delayed.
- Packets can arrive in a different order than the order in which they were sent.
- Sender tries to send faster than receiver can receive.
- End systems try to send more than the available network capacity.

So what can go wrong?

- Packets can get corrupted.
  - TCP checksum discards bad packets.
  - But beware: it's not very strong!
- Packets can get lost.
  - TCP will retransmit.
- Packets can get delayed.
  - TCP uses adaptive timers to try and avoid retransmitting unnecessarily.
  - Get this wrong, and you have congestion collapse.

So what can go wrong?

- Packets can arrive in a different order than the order in which they were sent.
  - TCP uses sequence numbers to restore original order
  - But most implementations will perform badly with a lot of reordering: treat this as signal of loss (congestion).
  - Can be a big problem if you want to do load balancing.

So what can go wrong?

- Sender trys to send faster than receiver can receive.
  - A TCP receiver continuously signals to the TCP sender how much receiver buffer space is available.
  - Sender never sends more than the receiver can store.
- End systems try to send more than the available network capacity.
  - TCP does end-to-end congestion control.

TCP Congestion Control

Congestion Control

End-to-end congestion control serves several purposes:

- Divides bandwidth between network flows in a "reasonably fair" manner without requiring per-flow scheduling by routers.
- Prevents congestion collapse of the network by matching demand to supply to ensure overall goodput remains reasonably high.

Congestion Collapse

Congestion collapse occurs when the network is increasingly busy, but little useful work is getting done.

Problem: *Classical congestion collapse*:

Paths clogged with unnecessarily-retransmitted packets [Nagle 84].

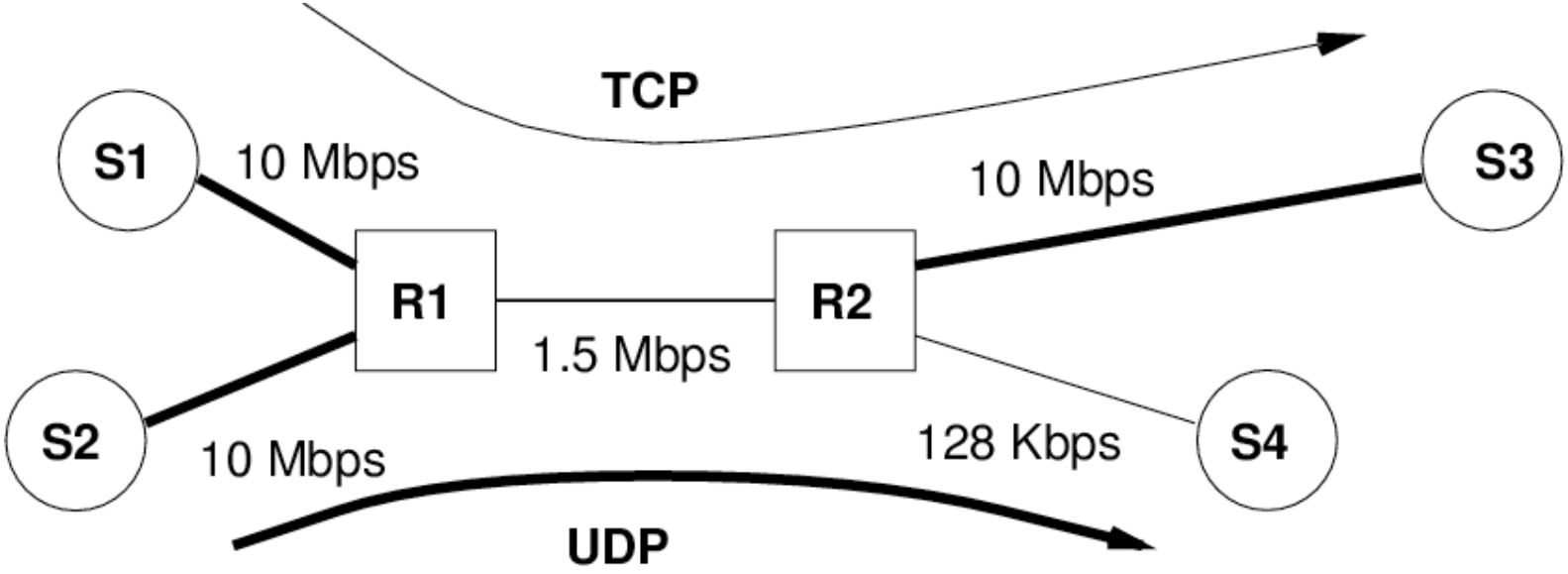
Fix:

Modern TCP retransmit timer and congestion control algorithms [Jacobson 88].

Congestion collapse from undelivered packets

Problem: Paths clogged with packets that are discarded before they reach the receiver [Floyd and Fall, 1999].

Fix: Either end-to-end congestion control, or a ``virtual-circuit'' style of guarantee that packets that enter the network will be delivered to the receiver.



Congestion Control

Since 1988, the Internet has remained functional despite exponential growth, routers that are sometimes buggy or misconfigured, rapidly changing applications and usage patterns, and flash crowds.

This is largely because most applications use TCP, and TCP implements *end-to-end congestion control.*

TCP Congestion Control

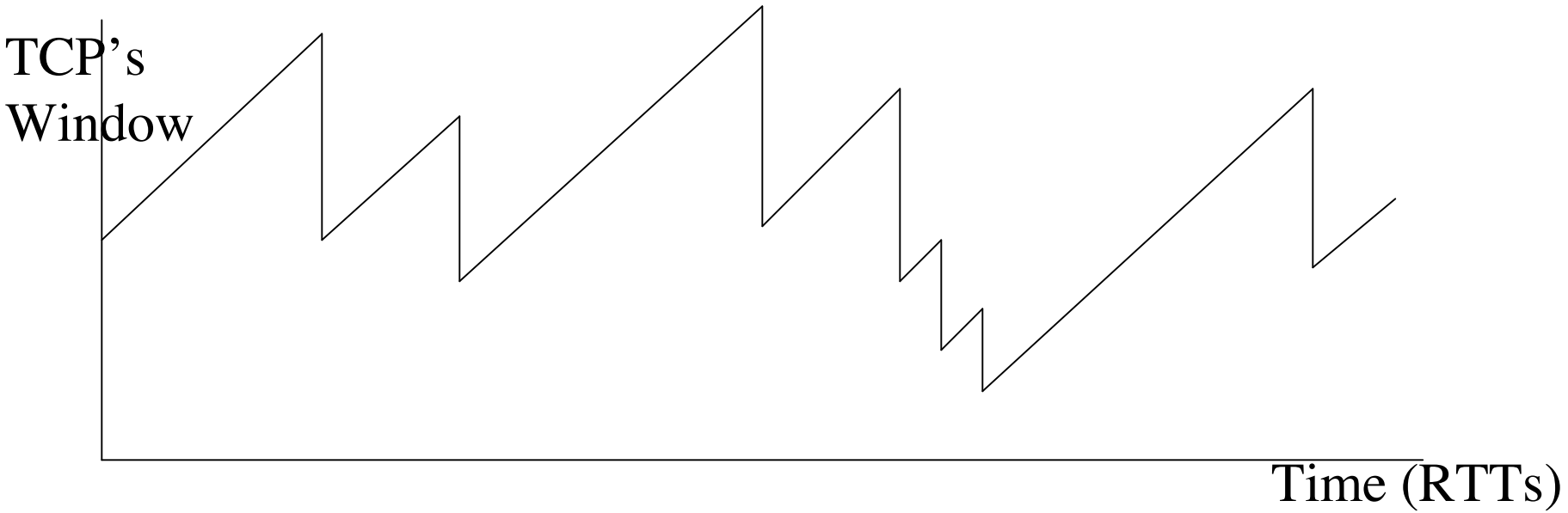
- TCP is *ack-clocked.*
  - When an Acknowledgement arrives back at the sender, it indicates a packet has left the network.
  - The sender can then send another packet.

- TCP does this by keeping a *window* of the packets currently in the network.
  - This is *inherently stable*: can only send as quickly as packets can get through the bottleneck link.
  - Should something go wrong, and a queue start increasing, TCP's constant window will cause the transmission rate to decrease.

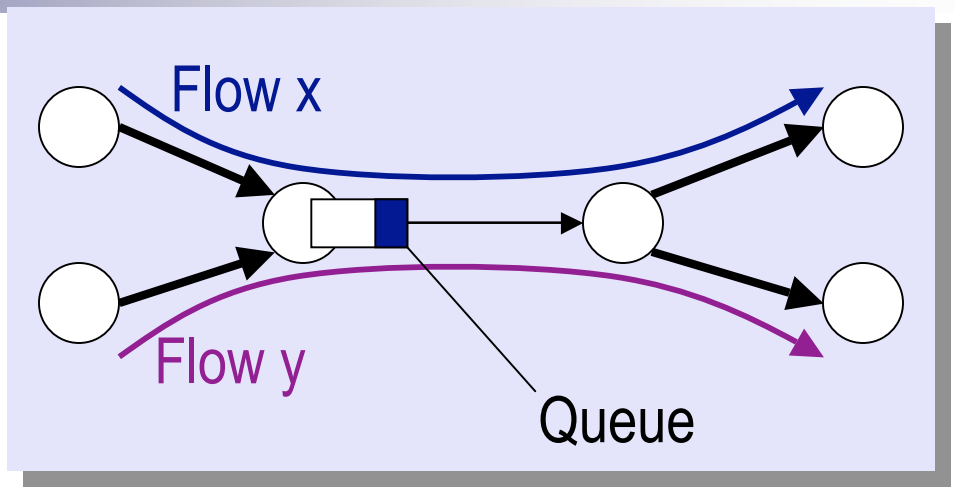
TCP Adaptive Congestion Control

Basic behaviour: Additive Increase, Multiplicative Decrease.

- Maintain a *window* of the packets in flight:
  - Each round-trip time, increase that window by one packet.
  - If a packet is lost, halve the window.



TCP Fairness



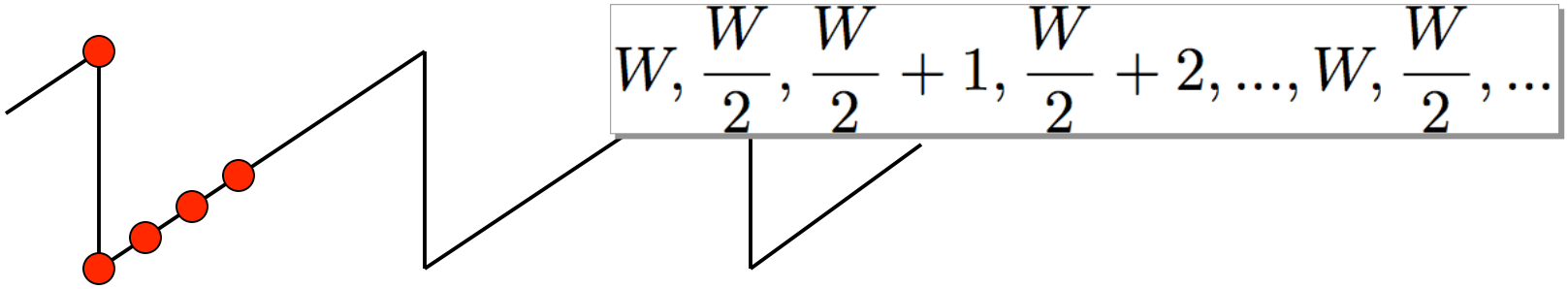
TCP (Details)

- TCP congestion control uses AIMD:
  - Increase the congestion window by one packet every round-trip time (RTT) that no packet is lost.
  - Decrease the congestion window by half every RTT that a packet loss occurs.
- In heavy congestion, when a retransmitted packet is itself dropped or when there aren't enough packets to run an ACK-clock, use a retransmit timer, which is exponential backed off if repeated losses occur.
- Slow-start: start by doubling the congestion window every roundtrip time.

TCP Modelling: The "Steady State" Model

The model: Packet size *B* bytes, round-trip time *R* secs, no queue.

- A packet is dropped each time the window reaches W packets.
- TCP's congestion window:



- The maximum sending rate in packets per roundtrip time: W
- The maximum sending rate in bytes/sec: $W\,B\,/\,R$
- The average sending rate T: $T = (3/4)W\,B\,/\,R$
- The packet drop rate p: $p = \dfrac{1}{\frac{3}{8}W^2}$
- **The result:** $T = \dfrac{\sqrt{6}B}{2R\sqrt{p}} = \dfrac{\sqrt{3/2}B}{R\sqrt{p}}$

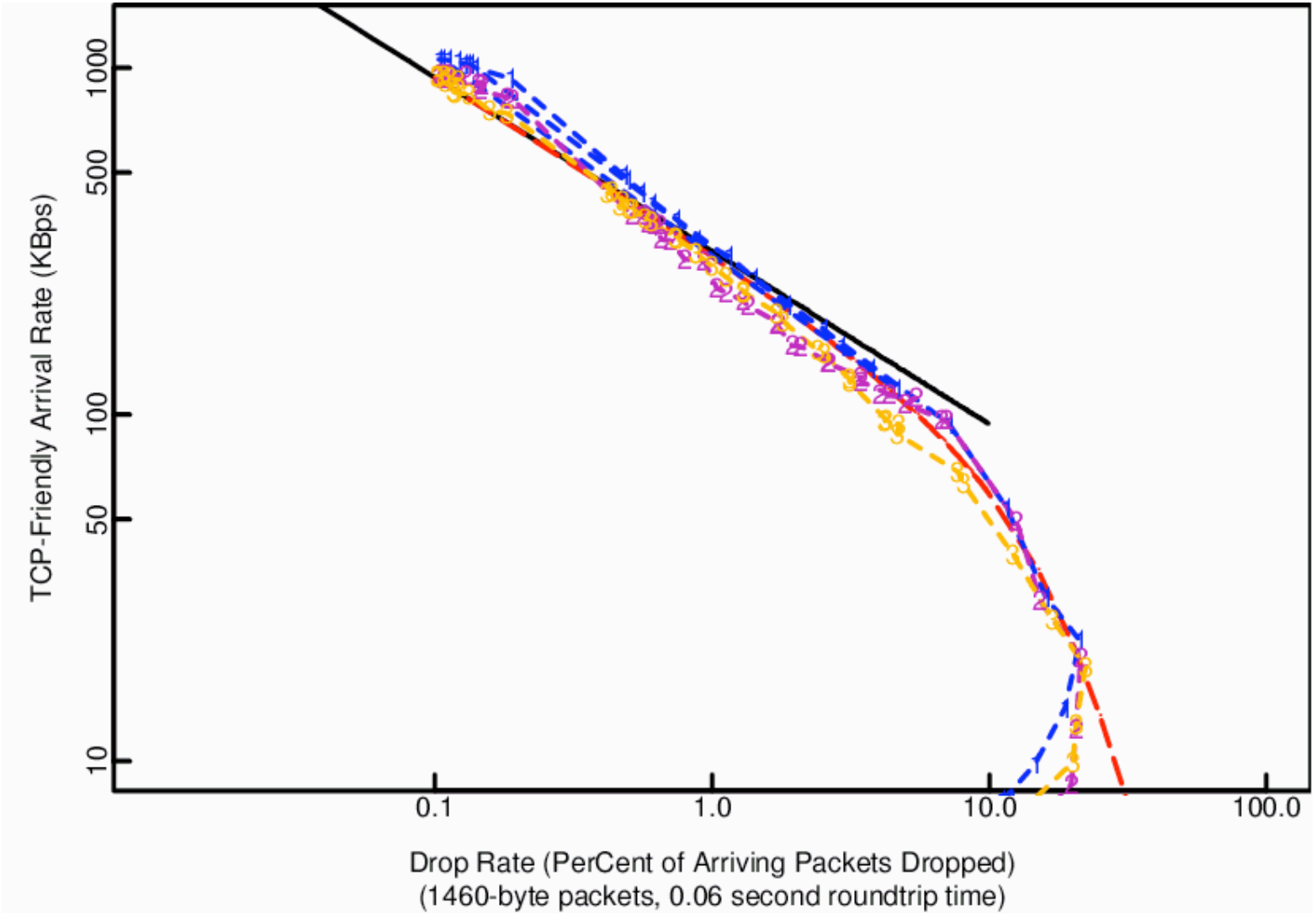
An Improved "Steady State" Model

A pretty good improved model of TCP Reno, including timeouts, from Padhye et al, Sigcomm 1998:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

T : sending rate in bytes/second]
R : round trip time
p : fraction of packets lost
t_{RTO} : TCP retransmission timeout

Verifying the Models



TCP Variants

- Four mainstream TCP variants
  - TCP Tahoe
  - TCP Reno
  - TCP NewReno
  - TCP Sack
- Tahoe and Reno are obsolete.
- NewReno and Sack are current.
  - Sack sends selective acknowledgements, so much more robust to multiple losses in one window of data.

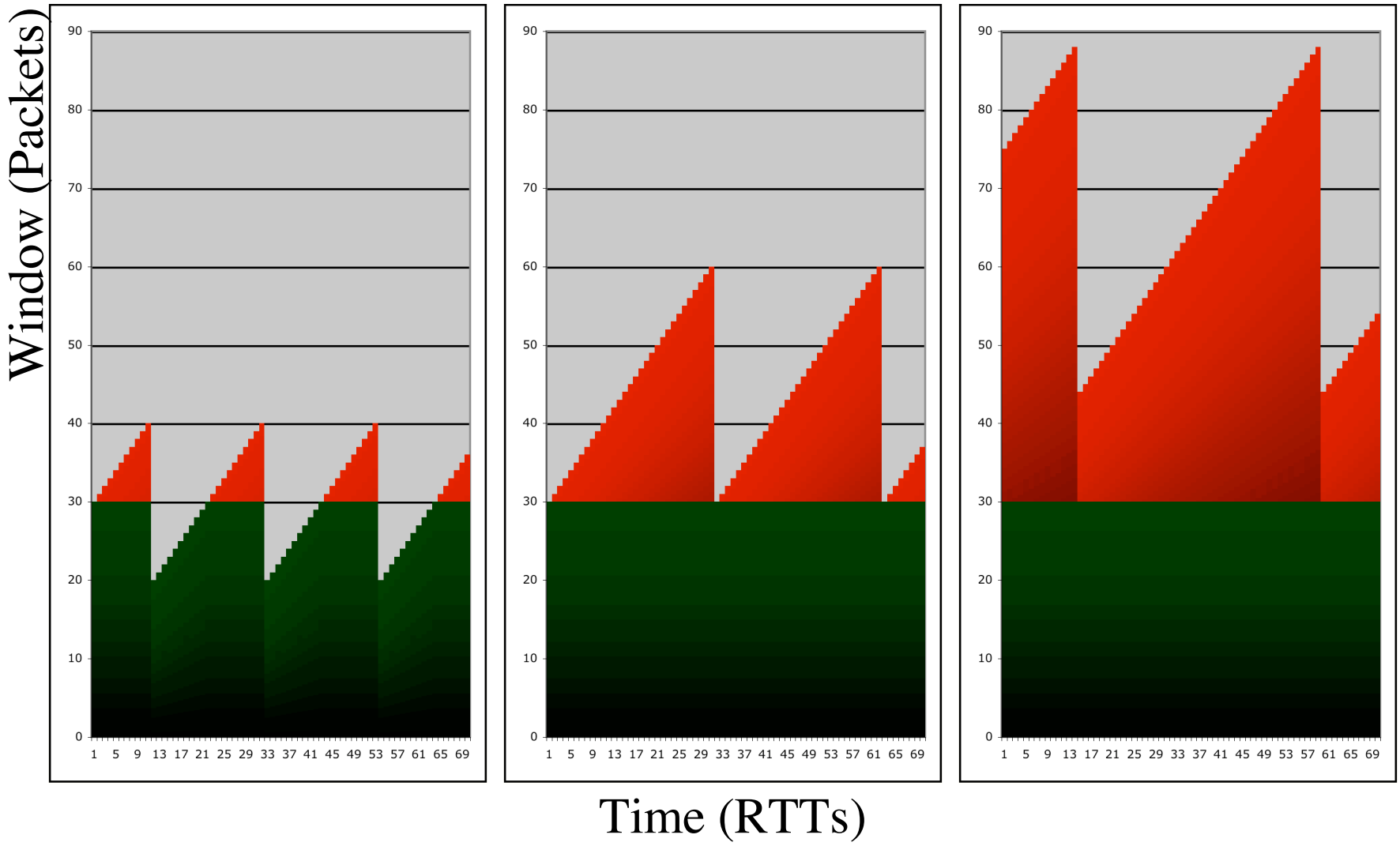
TCP and Network Queuing

Queuing

- The primary purpose of a queue in an IP router is to smooth out bursty arrivals, so that the network utilization can be high.

- But queues add delay and cause jitter.
  - Understanding and controlling network queues is key to getting good performance.

TCP Throughput and Queue Size

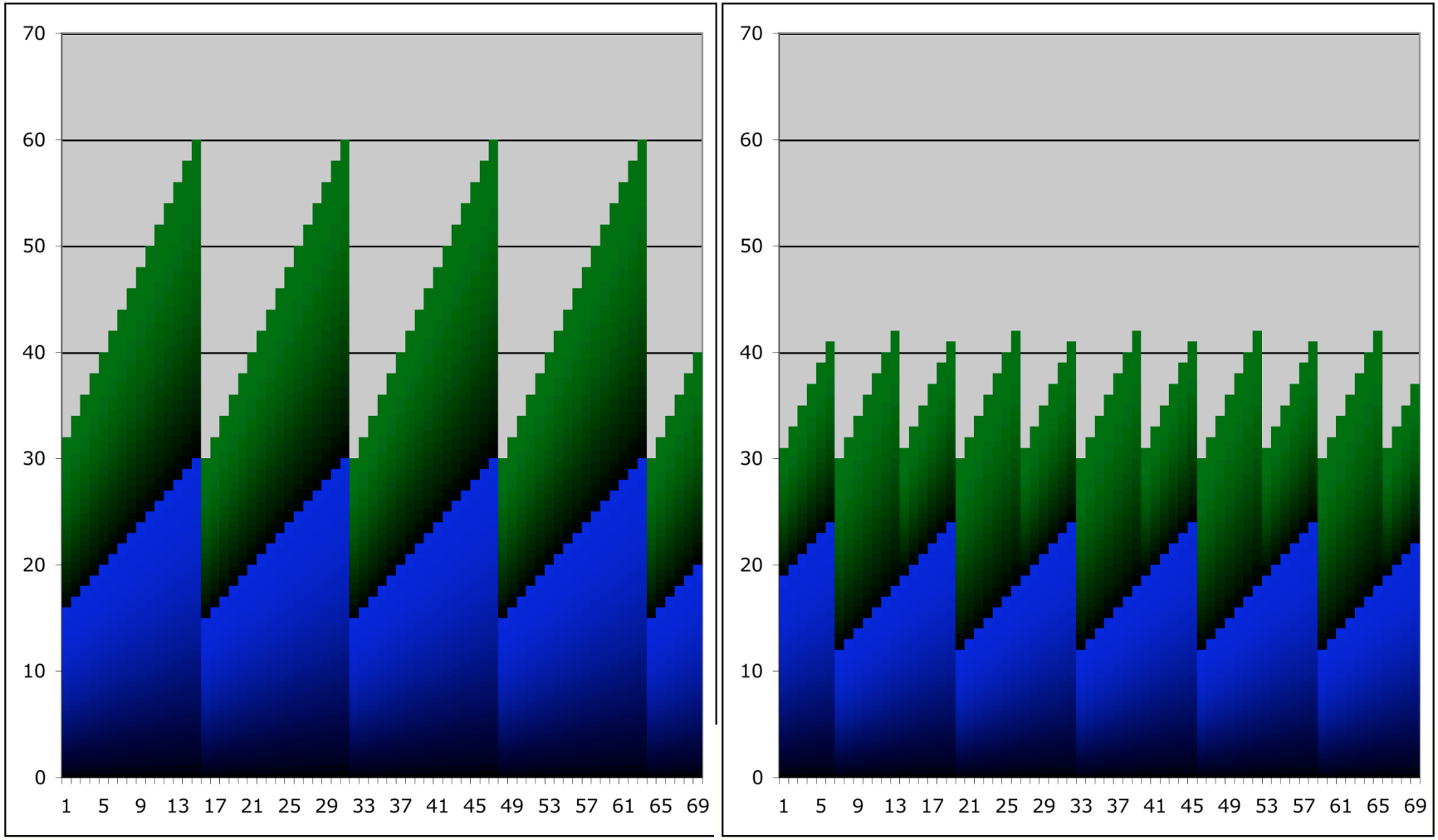


TCP and Queues

- TCP needs *one delay-bandwidth product* of buffer space at the bottleneck link for a TCP flow to fill the link and achieve 100% utilization.

- Thus, when everything is configured correctly, the peak delay is twice the underlying network delay.
  - Links are often overbuffered, because the actual RTT is unknown to the link operator.
  - Increased RTT => slower convergence.

Two TCP Flows (Effects of Phase)



Multiple TCP flows and Queues

- If multiple flows all back-off in phase, the router still needs a delay-bandwidth product of buffering.

- If multiple flows back-off out of phase, high utilization can be maintained with smaller queues.
  - How to keep the flows out of phase?

Active Queue Management

Goals of Active Queue Management

- The primary goal: Controlling average queuing delay, while still maintaining high link utilization.
- Secondary goals:
  - Improving fairness (e.g., by reducing biases against bursty low-bandwidth flows).
  - Reducing unnecessary packet drops.
  - Reducing global synchronization (i.e., for environments with small-scale statistical multiplexing).
  - Accommodating transient congestion (lasting less than a round-trip time).

Random Early Detection (RED)

- As queue builds up, randomly drop or mark packets with increasing probability (before queue gets full).
- Advantages:
  - Lower average queuing delay.
  - Avoids penalizing streams with large bursts.
  - Desynchronizes co-existing flows.

RED Algorithm

```
for each packet arrival
    calculate the new average queue size q_avg
    if min_th < q_avg < max_th
        calculate probability p_a
        with probability p_a:
            mark/drop the arriving packet
    else if max_th > q_avg
        drop the arriving packet
```

Variables:

q_{avg} : average queue size

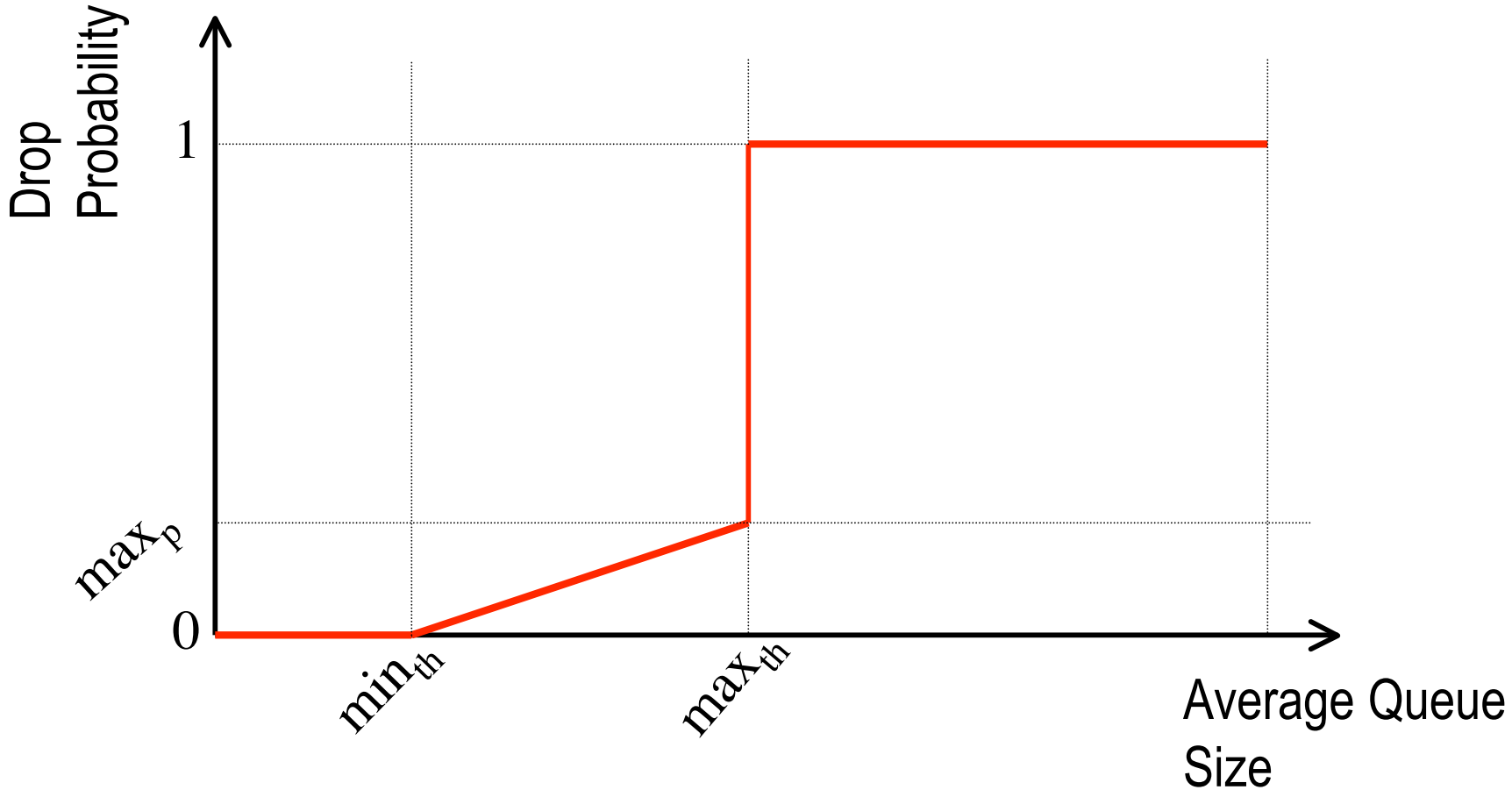
p_a : packet marking or dropping probability

Parameters:

min_{th} : minimum threshold for queue

max_{th} : maximum threshold for queue

RED Drop Probabilities



The argument for using the *average* queue size in AQM

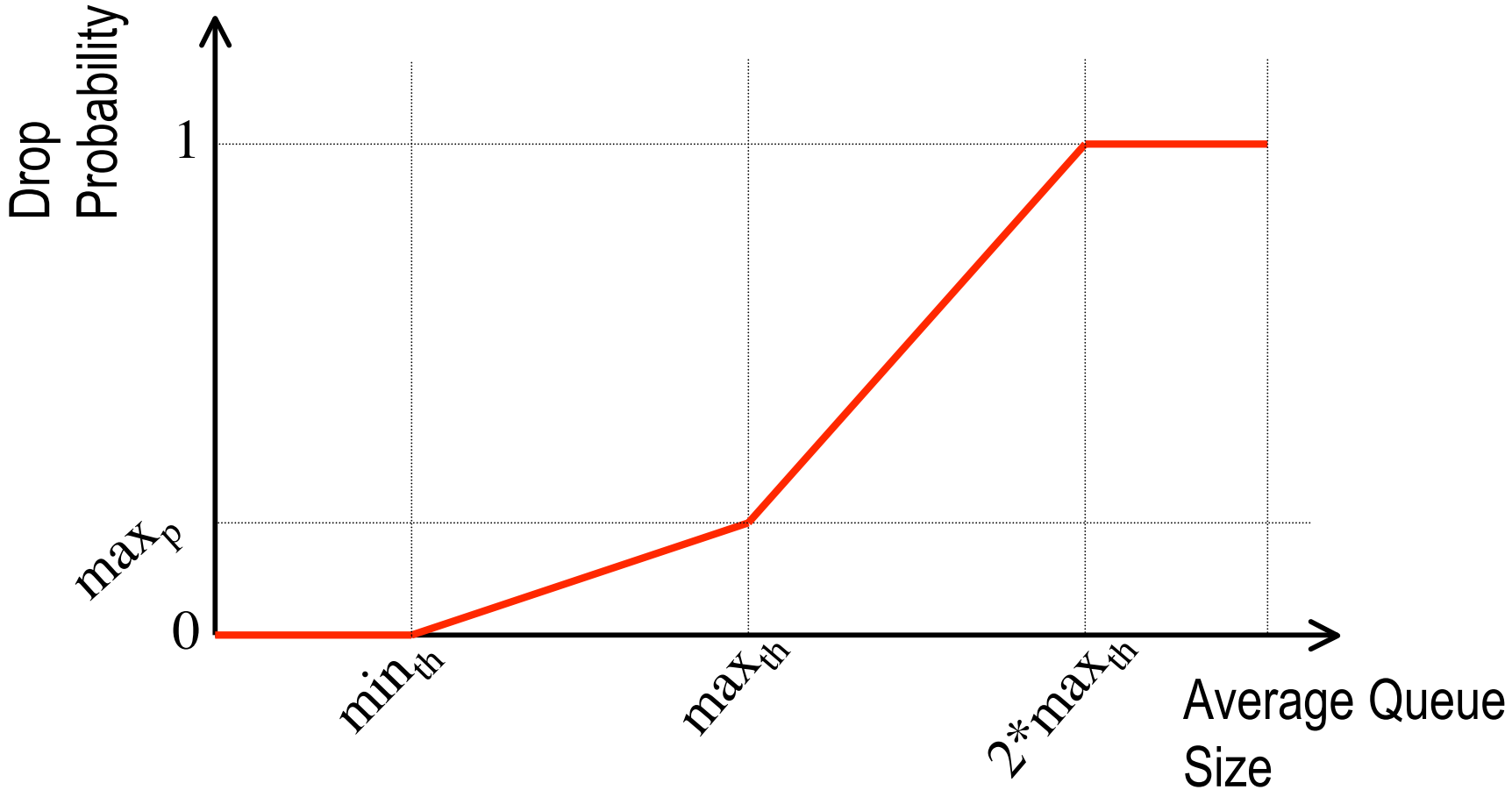
To be robust against transient bursts:

- When there is a transient burst, to drop just enough packets for end-to-end congestion control to come into play.
- To avoid biases against bursty low-bandwidth flows.
- To avoid unnecessary packet drops from the transient burst of a TCP connection slow-starting.

The problem with RED

- Parameter sensitivity
  - How to set min_{th}, max_{th} and max_p?
- Goal is to maintain mean queue size below the midpoint between min_{th} and max_{th} in times of normal congestion.
  - max_{th} needs to be significantly below the maximum queue size, because short-term transients peak well above the average.
  - max_p primarily determines the drop rate. Needs to be significantly higher than the drop rate rfequired to keep the flows under control.
- In reality it's hard to set the parameters robustly, even if you know what you're doing.

RED Drop Probabilities (Gentle Mode)



Other AQM schemes.

- Adaptive RED (ARED)
- Proportional Integral (PI)
- Virtual Queue (VQ)
- Random Exponential Marking (REM)
- Dynamic-RED (DRED)
- Blue
- Many other variants... (a lot of PhDs in this area!)

Explicit Congestion Notification

Explicit Congestion Notification (ECN)

- Standard TCP:
  - Losses needed to detect congestion
  - Wasteful and unnecessary
- ECN:
  - Routers mark packets instead of dropping them.
  - Receiver returns marks to sender in ACK packets.
  - Sender adjusts it's window as it would have done if the packet had been dropped.
- Advantages:
  - Bandwidth up to bottleneck not wasted.
  - No delay imposed by retransmission.

TCP Summary

- TCP performs a number of roles.
  - Need to get everything right to work well.
- Adaptive congestion control is critical.
  - Even if you think you don't need it, when something breaks you'll be glad it's there.
- Interaction between TCP and Queuing is important.
- TCP response function isn't set in stone.
  - A lot of current research as to how to do better with high delay-bandwidth products.
  - No clear consensus on how to agree what the future of congestion control is.