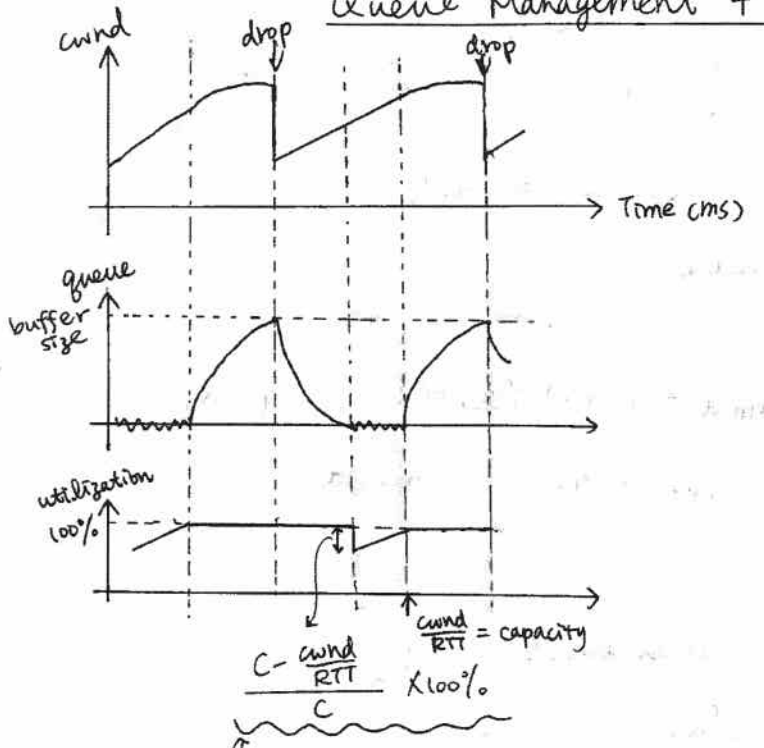
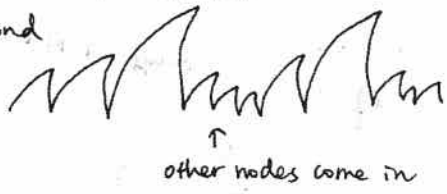
Queue Management + Congestion Control



1 TCP flow

throughput = $\frac{cwnd}{RTT}$ → ||| → C = capacity of link

In real TCP network:



↳ as small as possible ⇒ the queue has to be large, so that queue will never be empty

- If there's no queue, once $\frac{cwnd}{RTT} > C$ (even if $\frac{cwnd}{RTT}$ exceeds C a little), cwnd will drop to half ⇒ utilization also drops to half

Queue

Transient
- queue absorbs when TCP > capacity
  drains when TCP < capacity
- healthy

Persistent
- some queues that never drain
- unhealthy

Managing Queues

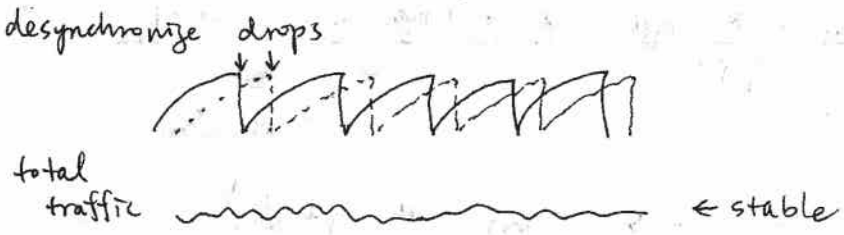
Drop Tail

Upon a packet arrival
- queue is full → drop
- queue is not full → queue packet

- pros: simple
- cons: synchronize sources

S1 →
S2 → ||| → all sources see drop around the same time
SN →
→ drop cwnd around the same time, oscillating together

Random Early Detection (RED)

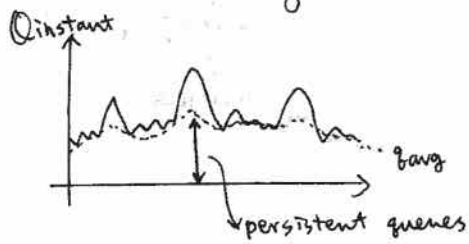


RED scheme

1) RED focuses on persistent queue

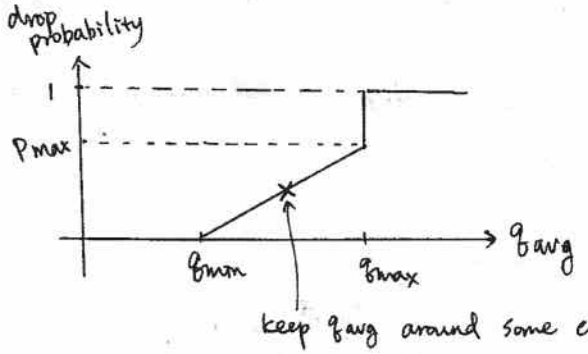
Upon arrival : $q_{avg}(i+1) = w\, q_{instant} + (1-w)\, q_{avg}(i)$. $0 < w \ll 1$

↑ a low-pass filter on $q_{instant}$



↓ persistent queues ⇒ try to reduce them

2) Smooth change in drop rate with congestion



make things smoother

→ detect the change of q_{avg}.

and adjust P_{drop} accordingly

q_{avg} ↑ . drop prob. ↑ (want to drop more)

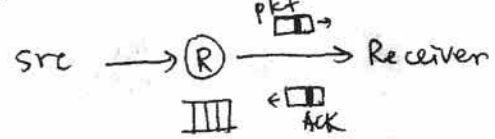
3) Spread dropping

drop prob. : spread the drops ⇒ desynchronize drops

Early Congestion Notification (ECN)

In RED: drop packet when there's still space in queue

ECN: If queue is not full, instead of dropping . mark the packet



src learns that there's congestion through the marked Ack.

RED Advantages over DT

- Smaller persistent Queues → smaller delays
- less dramatic oscillation

Disadvantages

- More complex
- Hard to pick parameters
  - ↳ depends on # flows in the network, the bottleneck ... etc.

If didn't pick well, the oscillation might be even worse.

	Cong. State		Feedback to the source
DT	No cong.	→	No drops
	low cong.	→	No drops
	severe cong.	→	High drops
RED	No cong. ($< q_{min}$)	→	No drops
	low cong.	→	low drop prob.
	severe Cong.	→	high drop prob.

Two scenarios

1) large # TCPs - only few TCP will see the drops. fine tuning
   smooth feedback → smooth rate/cwnd adaptation

2) Few TCPs (1 to 4) - don't see the smooth effect
   smooth feedback ↛ smooth change in total rate

DCTCP

1) Small #concurrent TCPs (1~3)

2) Delay is very important ⇒ Need to control queue build-ups and oscillations

Need smooth feedback!

compute feedback:

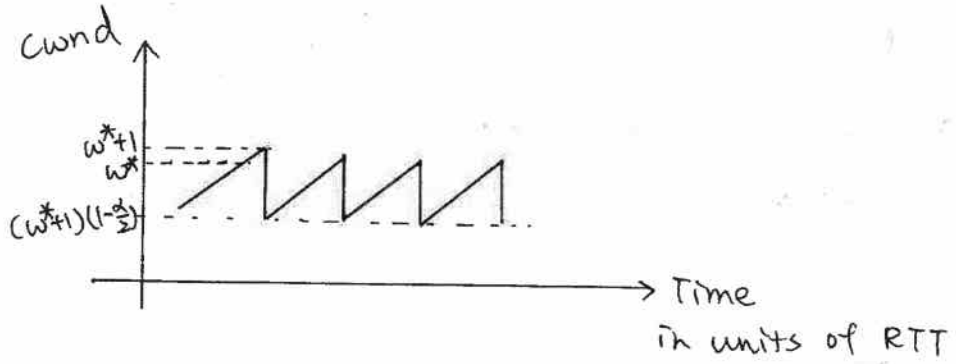
$\alpha(t)$ = fraction of marked pkts in cwnd(t)

(If $\alpha(t)\uparrow$, the degree of congestion ↑)

[Src] If no marks in RTT, cwnd = cwnd + 1

If mark in this RTT, cwnd = cwnd $(1 - \frac{\alpha}{2})$

(If α is small, cwnd changes only a little.)



Router Marks all packet when $Q \geq K$.