Optimization and Control of Networks

Duality Model of TCP/AQM

Lijun Chen

02/08/2016

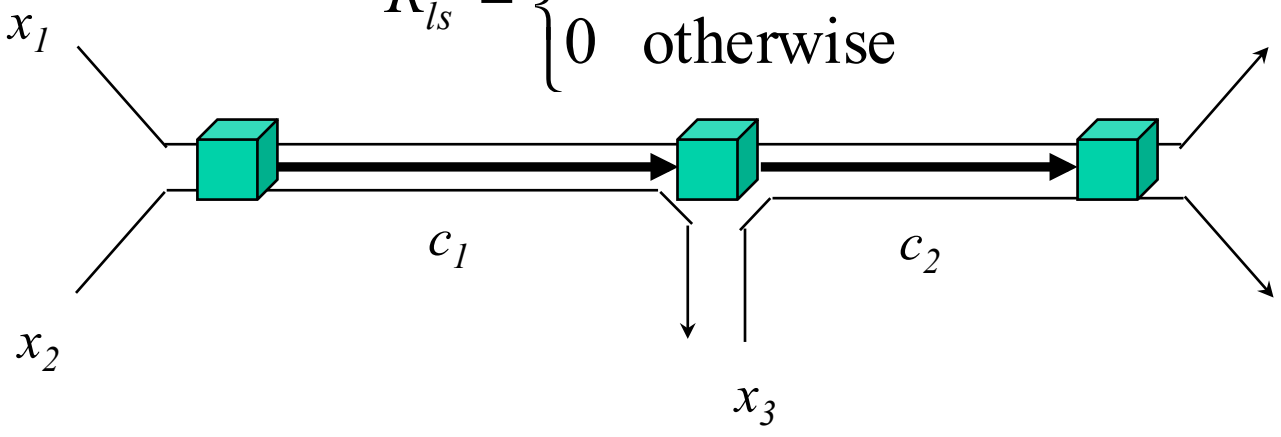
Agenda

- Utility maximization and dual decomposition
- An introduction to TCP congestion control
- A general dynamical model of TCP/AQM
- The duality model of TCP/AQM

Network model

- A network modeled as a set L of links with finite capacities $c = \{c_l, l \in L\}$
- Shared by a set S of sources
- Each source s uses a subset $L_s \subseteq L$ of links, which defines a routing matrix

$$R_{ls} = \begin{cases} 1 & \text{if } l \in L_s \\ 0 & \text{otherwise} \end{cases}$$



$$R = \begin{pmatrix} 1\,1\,0 \\ 1\,0\,1 \end{pmatrix}$$

Utility function

- Each source s attains a utility $U_s(x_s)$ when transmitting at rate x_s
  - Non-decreasing function
- Given the set X of possible alternatives, a function

$$U : X \rightarrow R$$

is a utility function representing preference relation among alternatives, if for all $x, y \in X$,

$$\text{"}x \text{ is at least as good as } y\text{"} \Leftrightarrow \mathrm{U}(x) \geq U(y)$$

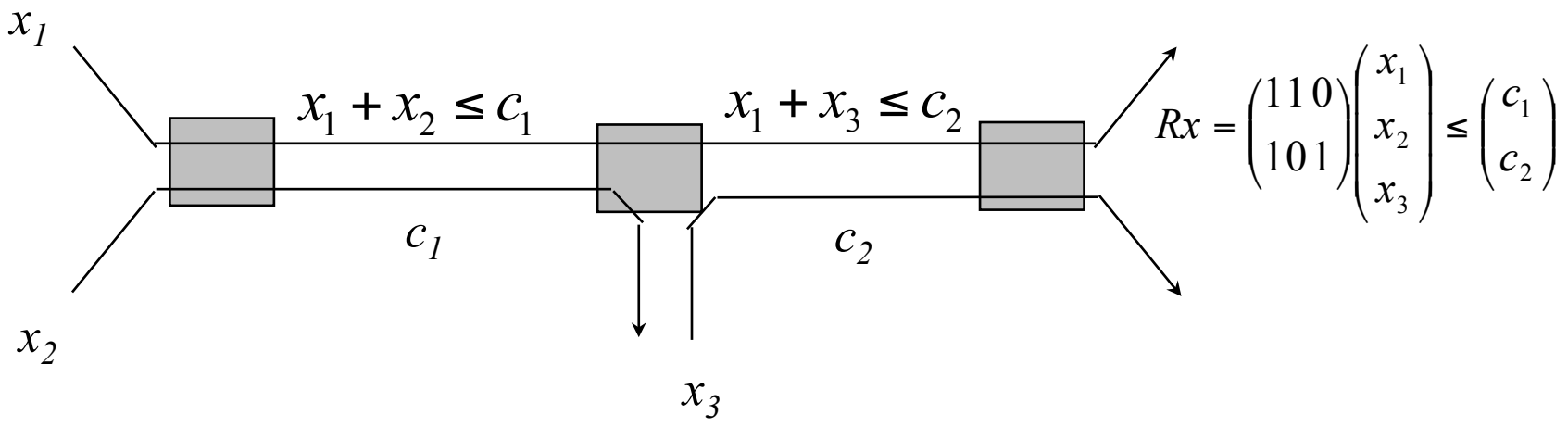
- To use utility function to characterize preferences is a fundamental assumption in economics

Resource allocation problem

- Each source s attains a utility $U_s(x_s)$ when transmitting at rate x_s
- Utility maximization (Kelly '98)

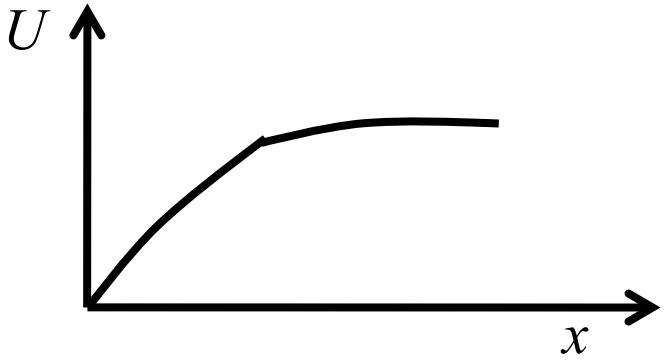
$$\max_x \sum_s U_s(x_s)$$

$$s.t. \quad Rx \le c$$



Fairness

- For elastic traffic, $U_s(\cdot)$ is assumed to be continuously differentiable, increasing and strictly concave
  - Diminishing return



- Ensure some kind of fairness
- Examples: $U_s(x_s) = x_s^{1-\alpha}/(1-\alpha), \quad \alpha > 0$
  - $\alpha \to 1$, $U_s(x_s) = \log x_s$, proportional fairness $\sum_s (y_s - x_s)/x_s \le 0$
  - $\alpha \to \infty$, max-min fairness

$$\max_{x} \sum_{s} U_s(x_s)$$

$$s.t. \quad Rx \leq c$$

- Polynomial-solvable, if all the utility and constraint information is provided. But impractical in real networks
- Have to seek decomposition to obtain distributed algorithm

Lagrangian dual

- Consider the dual problem (Low ’99)

$$\min_{p \geq o} \quad D(p) := \max_{x} \sum_{s} U_s(x_s) - p^T(Rx - c)$$

$$= \max_{x} \sum_{s} (U_s(x_s) - x_s \sum_{c} R_{ls} p_l) + p^T c$$

  - Congestion control: given end-to-end price $q_s = \sum_l R_{ls} p_l$

$$x_s(t) = U_s'^{-1}(q_s(t))$$

  - Price update: given aggregate source rate $y_l = \sum_s R_{ls} x_s$

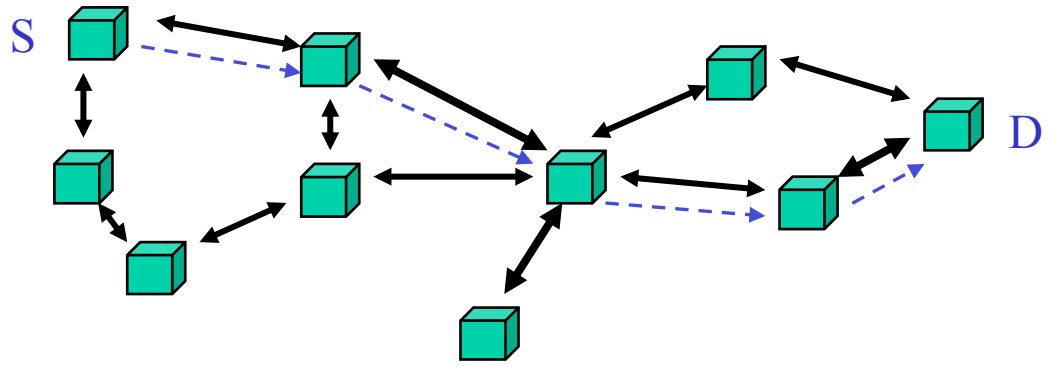
$$p_l(t+1) = [p_l(t) + \gamma(y_l(t) - c_l)]^+$$

  - Prices can be updated and fed back to sources implicitly

Agenda

- Utility maximization and dual decomposition
- An introduction to TCP congestion control
- A general dynamics model of TCP/AQM
- The duality model of TCP/AQM

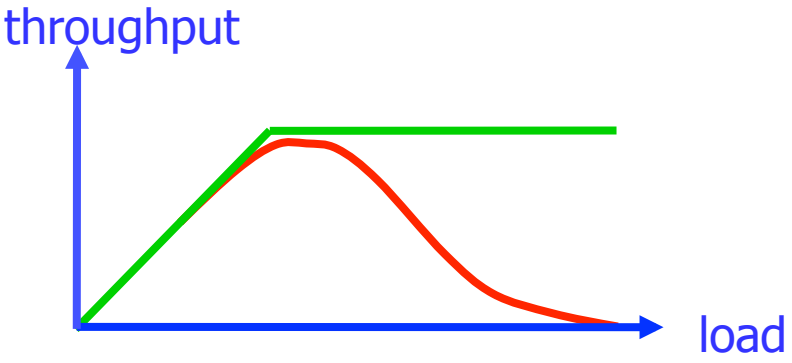
TCP/IP protocol stack



application	Does the part of task specific to the particular applications
transport	Provide reliable end-to-end transmission, congestion control
network	Provide end-to-end path between two end nodes (routing)
link	Provide reliable point-to-point transmission, channel access
physical	Provide a link for transmitting bits between two nodes

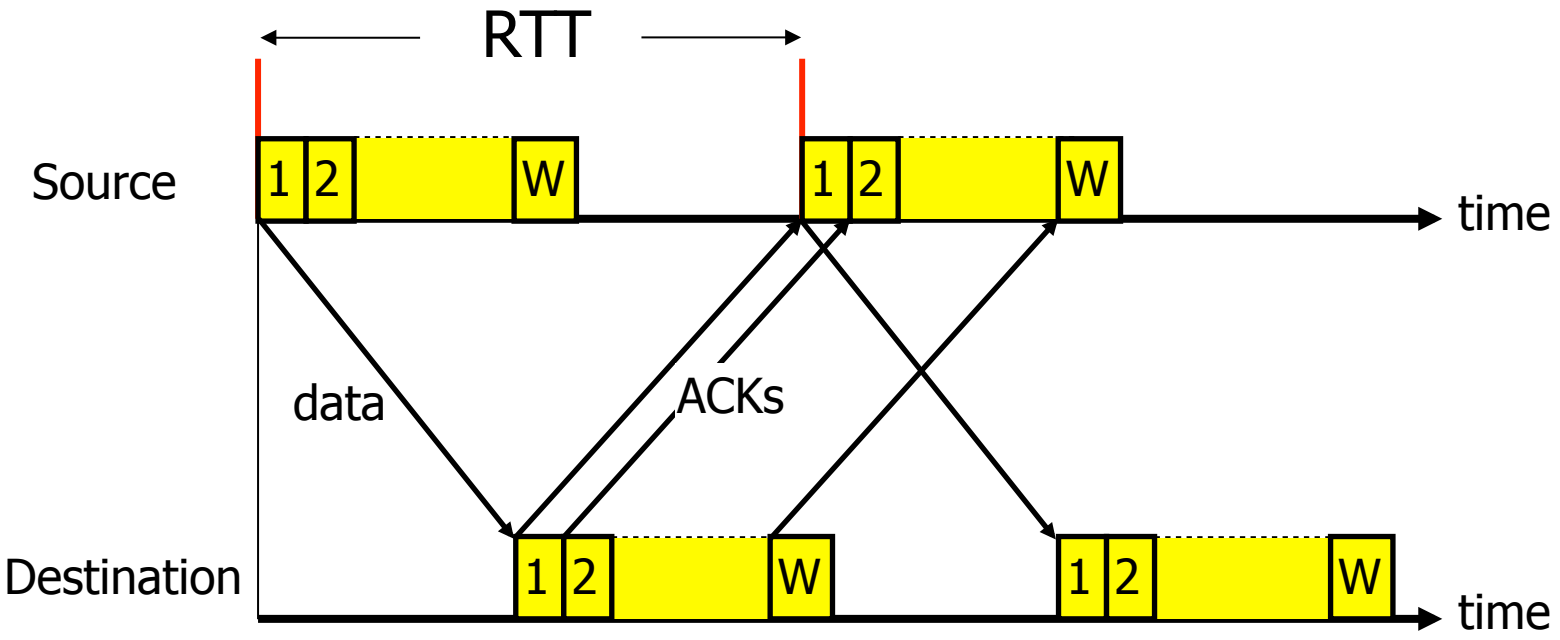
Congestion control

- Effect of congestion
  - Packet loss, retransmission, reduced throughput even congestion collapse
  - Internet has its first congestion collapse in Oct. 1986



- Congestion control
  - Achieve high utilization
  - Avoid congestion
  - Fair bandwidth sharing

Window-based flow control

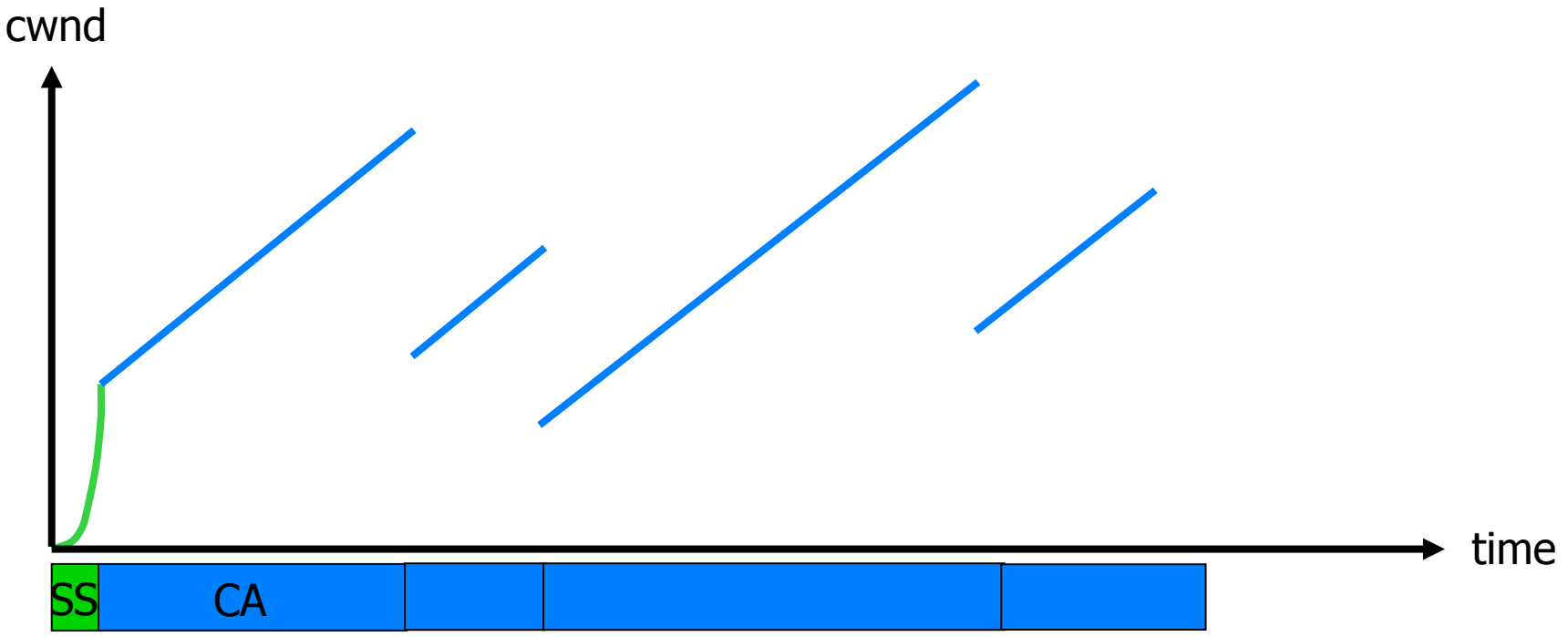


- ~ W packets per RTT (round trip time)
- Lost packet detected by missing ACK
- Source rate ~ W/RTT

TCP congestion control

- Source calculates cwnd from indication of network congestion
  - cwnd: congestion window size
- Congestion indications
  - Packet losses
  - Delay
  - Packet marks
- Algorithms to calculate cwnd
  - Tahoe, Reno, Vegas, …
  - DropTail, RED, REM, …

TCP Reno (Jacobson ‘90)



SS: Slow Start
CA: Congestion Avoidance

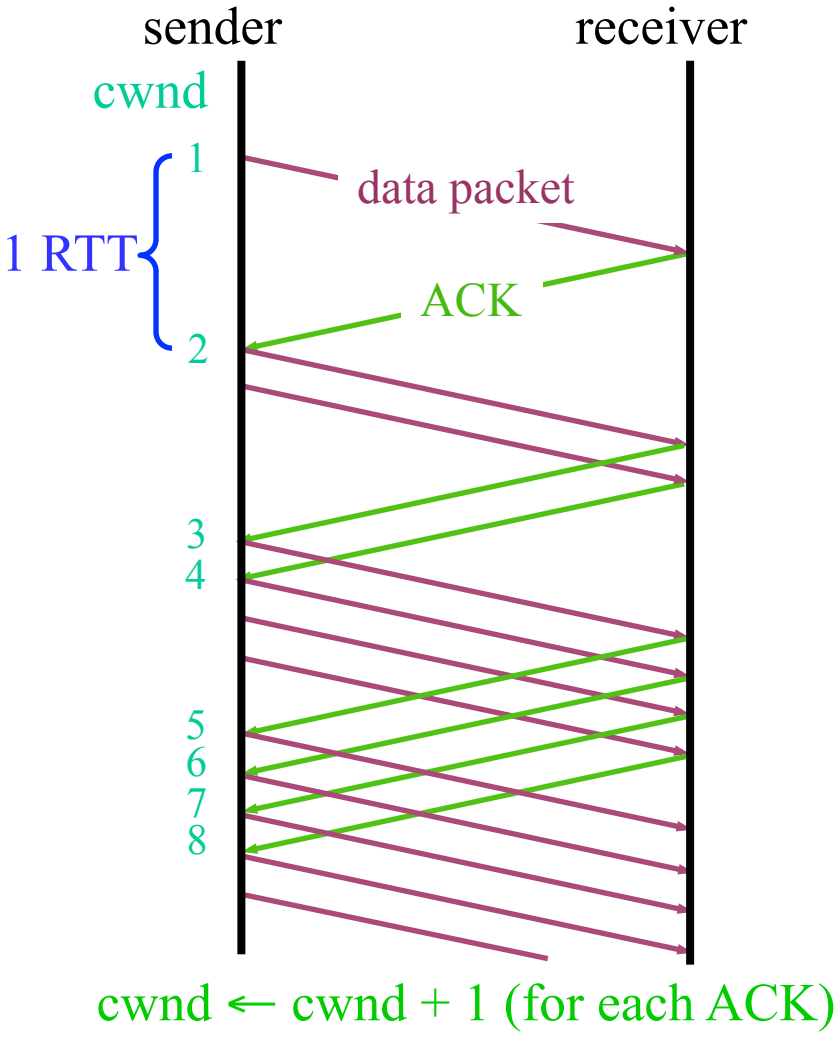
Slow start

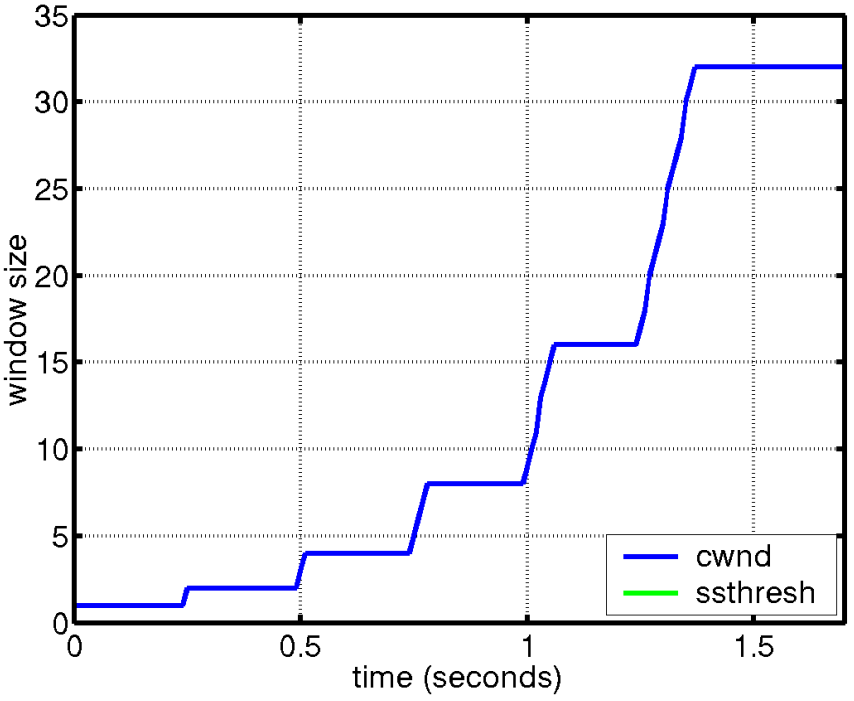
- Start with cwnd = 1 (slow start)
- On each successful ACK increment cwnd

  cwnd ← cnwd + 1
- Exponential growth of cwnd

  each RTT: cwnd ← 2 x cwnd
- Enter CA when cwnd >= ssthresh

Slow Start


sender
receiver
cwnd
1
data packet
1 RTT
ACK
2
3
4
5
6
7
8
cwnd ← cwnd + 1 (for each ACK)


window size
time (seconds)
cwnd
ssthresh
0
5
10
15
20
25
30
35
0
0.5
1
1.5

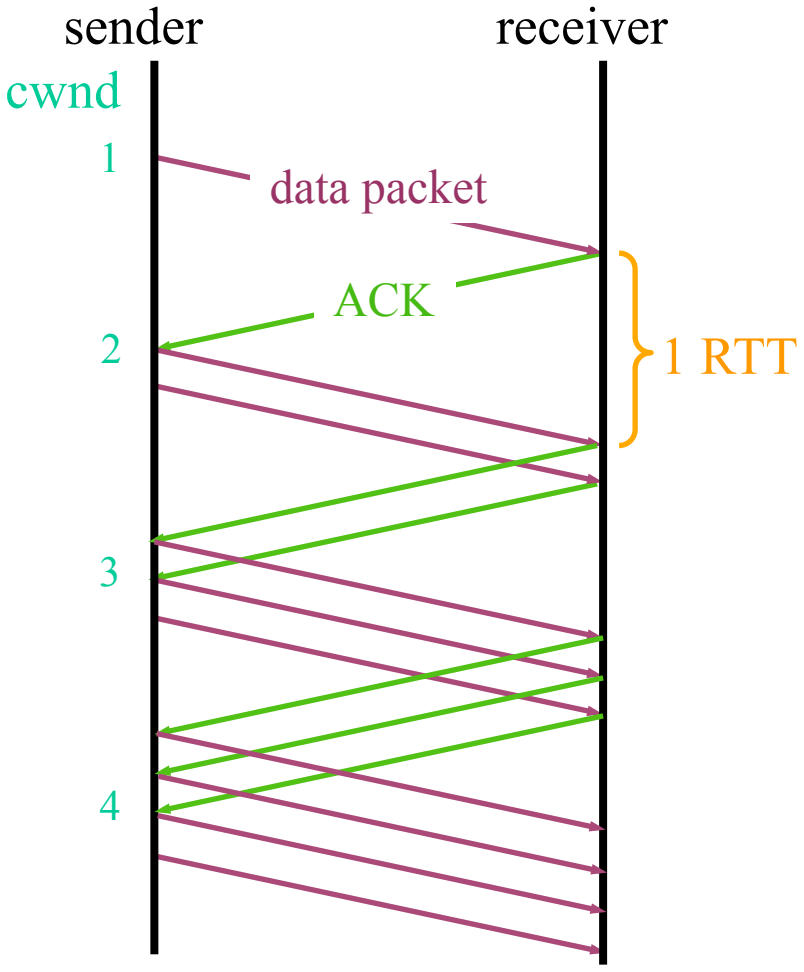
Congestion Avoidance

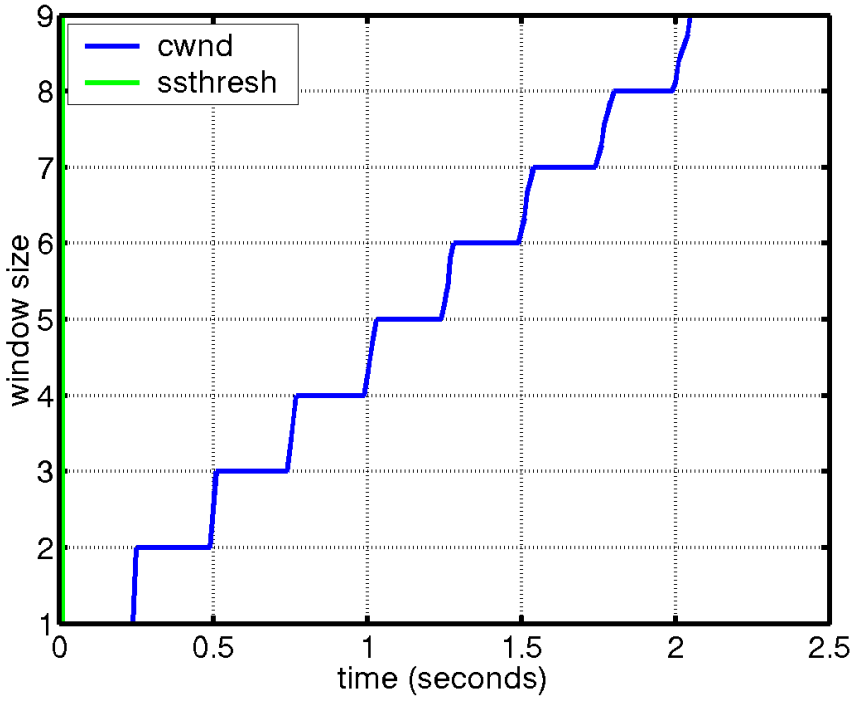
- Starts when cwnd ≥ ssthresh
- On each successful ACK:

  cwnd ← cwnd + 1/cwnd
- Linear growth of cwnd

  each RTT: cwnd ← cwnd + 1

Congestion Avoidance

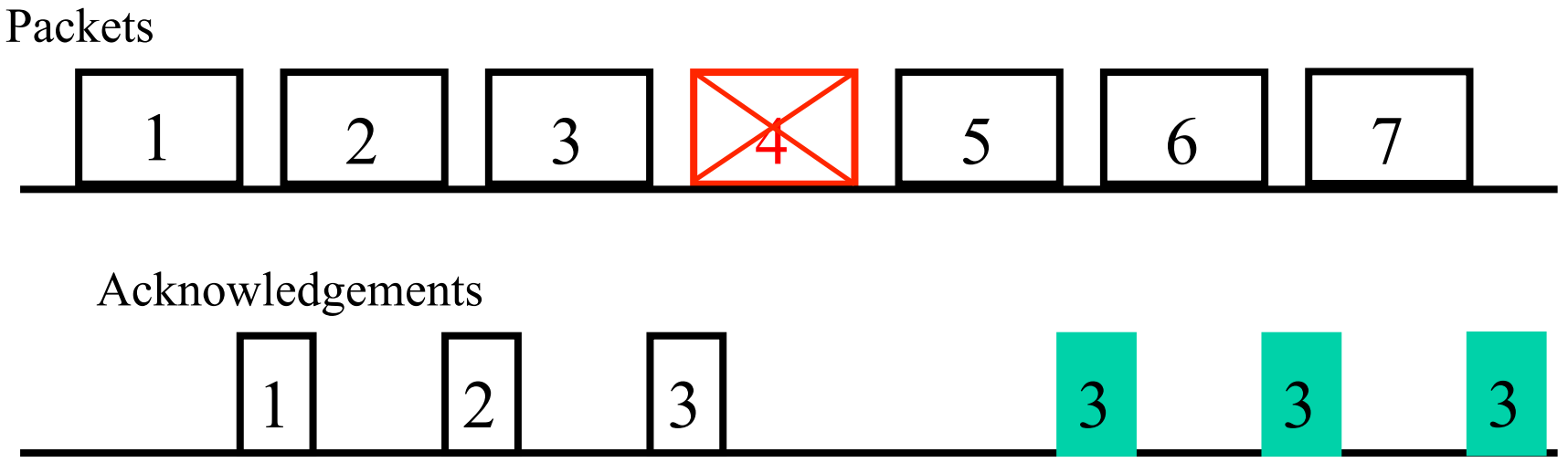




cwnd ← cwnd + 1 (for each cwnd ACKS)

Packet Loss

- Assumption: loss indicates congestion
- Packet loss detected by
  - Retransmission TimeOuts (RTO timer)
  - Duplicate ACKs (at least 3)

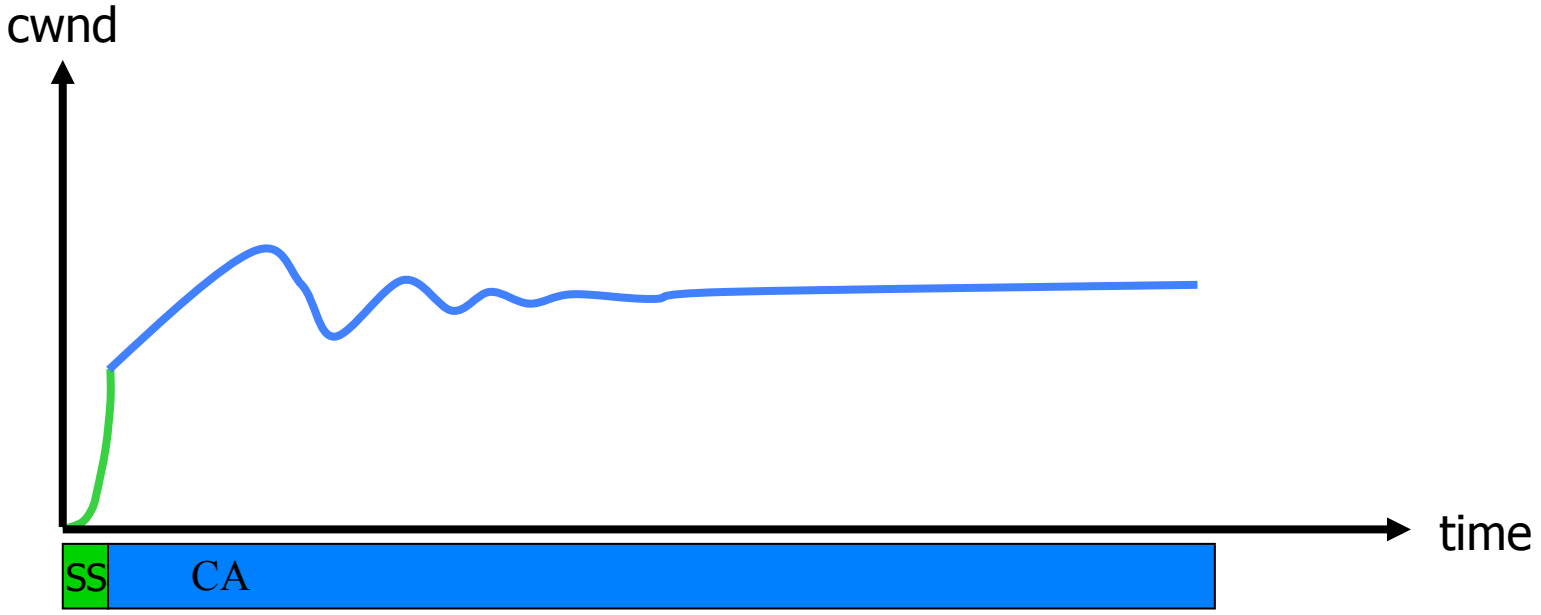


Summary: Reno

- Basic ideas
  - Gently probe network for spare capacity
  - Drastically reduce rate on congestion

```
for every ACK {
    if (W < ssthresh) then W++     (SS)
    else  W += 1/W                 (CA)
}
for every loss {
    ssthresh = W/2
          W  = ssthresh
}
```

TCP Vegas (Brakmo & Peterson '94)



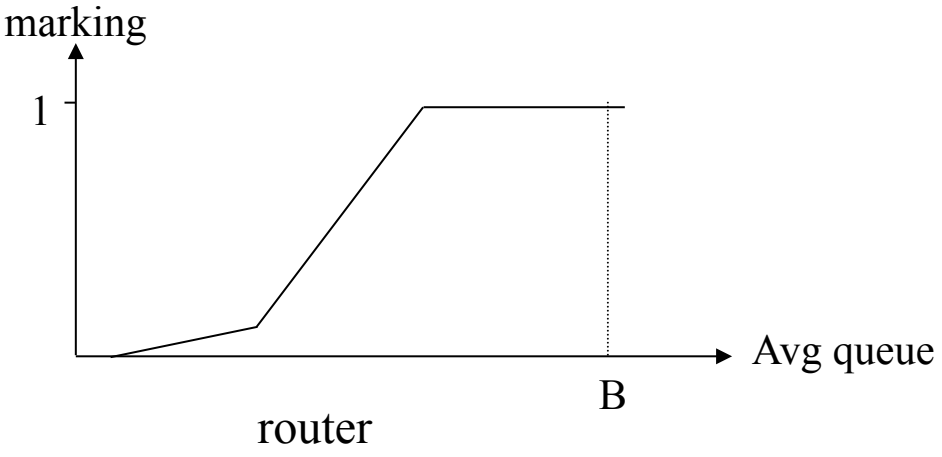

```
for every RTT
{    if W/RTT_min - W/RTT < α then W ++
     if W/RTT_min - W/RTT > α then W --    }
for every loss
        W := W/2
```

queue size

Congestion measure: end-to-end queueing delay

Link algorithms (AQM)

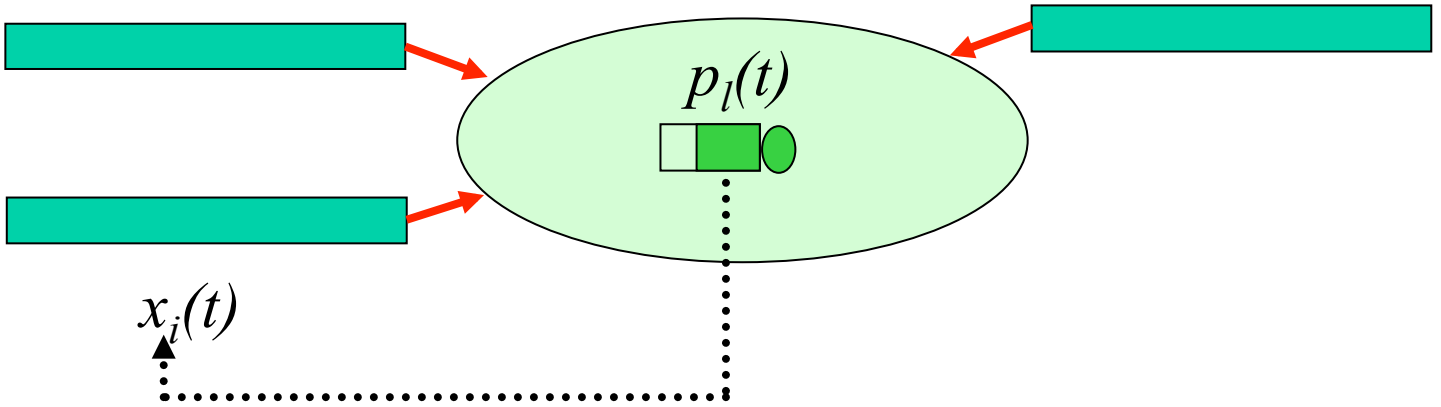
- DropTail: drop coming packets when buffer is full
- RED (random early detection): warn sources of incipient congestion by probabilistically marking/ dropping packets
  - Probabilistically drop packets
  - Probabilistically mark packets



Agenda

- Utility maximization and dual decomposition
- An introduction to TCP congestion control
- A general dynamical model of TCP/AQM
- The duality model of TCP/AQM

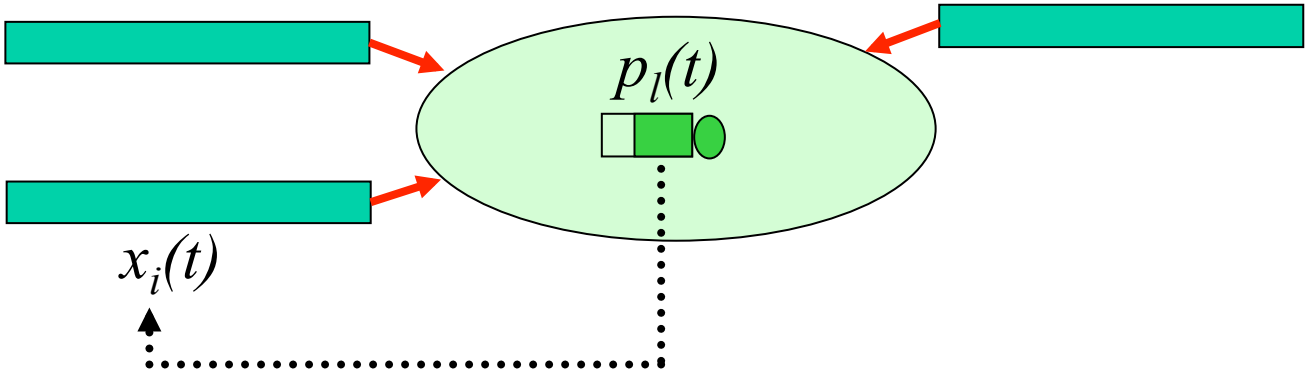
TCP & AQM



- Two components
  - A source algorithm: adjust the sending rate based on congestion
    - Implemented in TCP (**T**ransmission **C**ontrol **P**rotocol)
  - A link algorithm: update a congestion measure and send it back to the sources
    - Congestion measure: loss probability and delay
    - In form of loss/mark or delay
    - Carried out by AQM (**A**ctive **Q**ueuing **M**anagement)

Dynamic model of TCP/AQM

- Notation
  - $x_s(t)$: source rate at time t
  - $y_l(t) = \sum_s R_{ls} x_s(t)$: aggregate source rate at link l
  - $p_l(t)$: link congestion measure
  - $q_s(t) = \sum_l R_{ls} p_l(t)$: end-to-end congestion measure of source s
- Source s can observe only $x_s(t)$ and $q_s(t)$
- Link l can observe only $p_l(t)$ and $y_l(t)$



- Dynamical model

$$x_s(t+1) = F_s(x_s(t), q_s(t))$$
$$p_l(t+1) = G_l(p_l(t), y_l(t))$$

- The exact forms of F_s and G_l are determined by the specific TCP/AQM protocol

Example: Vegas

$$D_s = d_s + q_s$$

```
for every RTT
{   if W/RTT_min - W/RTT < α then W ++
    if W/RTT_min - W/RTT > α then W --   }
for every loss
        W := W/2
```

queue size

$$F: \quad x_s(t+1) = \begin{cases} x_s(t) + \dfrac{1}{D_s^2} & \text{if } w_s(t) - d_s x_s(t) < \alpha_s d_s \\ x_s(t) - \dfrac{1}{D_s^2} & \text{if } w_s(t) - d_s x_s(t) > \alpha_s d_s \\ x_s(t) & \text{else} \end{cases}$$

$$G: \quad p_l(t+1) = [p_l(t) + y_l(t)/c_l - 1]^+$$

Agenda

- Utility maximization and dual decomposition
- An introduction to TCP congestion control
- A general dynamics model of TCP/AQM
- The duality model of TCP/AQM

Duality model of TCP/AQM

- Denote by (x^*, p^*) the equilibrium of the system

$$x_s(t+1) = F_s(x_s(t), q_s(t))$$
$$p_l(t+1) = G_l(p_l(t), y_l(t))$$

- The fixed point equation $x_s^* = F_s(x_s^*, q_s^*)$ implicitly define a relation $q_s^* = f_s(x_s^*) > 0$
  - F_s is continuously differentiable, and $\partial F_s / \partial q_s \neq 0$
- Define a utility function for each source

$$U_s(x_s) = \int f_s(x_s) dx_s, \; x_s > 0$$

  - Usually continuous, increasing and strictly concave
  - Only determined by tcp algorithms

Example: Vegas

$$F: \quad x_s(t+1) = \begin{cases} x_s(t) + \dfrac{1}{D_s^2} & \text{if} \quad w_s(t) - d_s x_s(t) < \alpha_s d_s \\ x_s(t) - \dfrac{1}{D_s^2} & \text{if} \quad w_s(t) - d_s x_s(t) > \alpha_s d_s \\ x_s(t) & \text{else} \end{cases}$$

- At equilibrium

$$w_s^* - d_s x_s^* = \alpha_s d_s \Rightarrow x_s^*(D_s^* - d_s) = \alpha_s d_s$$

$$\Rightarrow x_s^* q_s^* = \alpha_s d_s \Rightarrow q_s^* = \alpha_s d_s / x_s^*$$

- Utility function $U_s(x_s) = \alpha_s d_s \log x_s$

- Define utility maximization

$$\max_{x} \sum_{s} U_s(x_s)$$

$$s.t. \quad Rx \le c$$

- Dual problem

$$\min_{p \ge o} \quad D = \max_{x} \sum_{s} (U_s(x_s) - x_s \sum_{c} R_{ls} p_l) + p^T c$$

- Interpret source rate x as primal variable and the congestion price p as dual variable

- The equilibrium (x^*, p^*) solves the primal and dual, if it satisfies KKT condition

$$y_l^* \le c_l$$
$$p_l^*(y_l^* - c_l) = 0$$
$$p_l^* \ge 0$$
$$U_s'(x_s^*) - q_s^* = 0$$

  - The complementary slackness condition is satisfied by any AQM that stabilizes the queues
  - AQM should match input rate to capacity to maximize utilization at every bottleneck link

Remarks

- Reverse engineering: the network as an optimization solver, and different TCP/AQM protocols as distributed primal-dual algorithms to solve the utility maximization and its dual
- Forward engineering: guide new congestion control design
  - By carefully choosing utility function
  - By proposing better convergent algorithm

- Can extend to provide a mathematical theory for network architecture and a general approach to cross-layer design