Congestion Control (contd)

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Based in part upon slides of Prof. Raj Jain (OSU), Srini Seshan (CMU), J. Kurose (U Mass), I.Stoica (UCB)
Queue Management Schemes: RED, ARED, FRED, BLUE, REM
TCP Congestion Control (CC) Modeling, TCP Friendly CC
Accumulation-based Schemes: TCP Vegas, Monaco
Static Optimization Framework Model for Congestion Control
Explicit Rate Feedback Schemes (ATM ABR: ERICA)
Refs: Chap 13.21, 13.22 in Comer textbook
Floyd and Jacobson, "Random Early Detection gateways for Congestion Avoidance"
Ramakrishnan and Jain, A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer,
Padhye et al, "Modeling TCP Throughput: A Simple Model and its Empirical Validation"
Low, Lapsley: "Optimization Flow Control, I: Basic Algorithm and Convergence"
Kalyanaraman et al: "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks"
Harrison et al: "An Edge-based Framework for Flow Control"
Queuing Disciplines

- Each router must implement some queuing discipline
- Queuing allocates bandwidth and buffer space:
  - Bandwidth: which packet to serve next (scheduling)
  - Buffer space: which packet to drop next (buff mgmt)
- Queuing also affects latency

Class A
Class B
Class C

Traffic Sources
Traffic Classes

Scheduling

Buffer Management

Drop
Typical Internet Queuing

- **FIFO + drop-tail**
  - Simplest choice
  - Used widely in the Internet
- **FIFO (first-in-first-out)**
  - Implies single class of traffic
- **Drop-tail**
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - **FIFO**: scheduling discipline
  - **Drop-tail**: drop (buffer management) policy
FIFO + Drop-tail Problems

- **FIFO Issues:** In a FIFO discipline, the service seen by a flow is *convoluted* with the arrivals of packets from all other flows!
  - No isolation between flows: full burden on e2e control
  - No policing: send more packets → get more service

- **Drop-tail issues:**
  - Routers are forced to have have large queues to maintain high utilizations
  - Larger buffers => larger steady state queues/delays
  - Synchronization: end hosts react to same events because packets tend to be lost in bursts
  - Lock-out: a side effect of burstiness and synchronization is that a few flows can monopolize queue space
Design Objectives

- Keep throughput high and delay low (i.e. knee)
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes
Queue Management Ideas

- **Synchronization, lock-out:**
  - Random drop: drop a randomly chosen packet
  - Drop front: drop packet from head of queue

- **High steady-state queuing vs burstiness:**
  - Early drop: Drop packets before queue full
  - Do not drop packets “too early” because queue may reflect only burstiness and not true overload

- **Misbehaving vs Fragile flows:**
  - Drop packets proportional to queue occupancy of flow
  - Try to protect fragile flows from packet loss (e.g., color them or classify them on the fly)

- **Drop packets vs Mark packets:**
  - Dropping packets interacts w/ reliability mechanisms
  - Mark packets: need to trust end-systems to respond!
Packet Drop Dimensions

- Aggregation
  - Per-connection state
  - Single class

- Drop position
  - Head
  - Tail
  - Random location

- Class-based queuing

- Early drop
  - Overflow drop
Random Early Detection (RED)

Max thresh

Min thresh

Average Queue Length

P(drop)

1.0

max_p

min_{th} max_{th} Avg queue length
Random Early Detection (RED)

- Maintain running average of queue length
  - Low pass filtering
- If \( \text{avg } Q < \text{min}_\text{th} \) do nothing
  - Low queuing, send packets through
- If \( \text{avg } Q > \text{max}_\text{th} \), drop packet
  - Protection from misbehaving sources
- Else mark (or drop) packet in a manner proportional to queue length & bias to protect against synchronization
  - \( P_b = \text{max}_p(\text{avg } - \text{min}_\text{th}) / (\text{max}_\text{th} - \text{min}_\text{th}) \)
  - Further, bias \( P_b \) by history of unmarked packets
    - \( P_a = P_b / (1 - \text{count} \cdot P_b) \)
RED Issues

- Issues:
  - Breaks synchronization well
  - Extremely sensitive to parameter settings
  - Wild queue oscillations upon load changes
  - Fail to prevent buffer overflow as #sources increases
  - Does not help fragile flows (e.g., small window flows or retransmitted packets)
  - Does not adequately isolate cooperative flows from non-cooperative flows

- Isolation:
  - Fair queuing achieves isolation using per-flow state
  - RED penalty box: Monitor history for packet drops, identify flows that use disproportionate bandwidth
Variant: ARED (Feng, Kandlur, Saha, Shin 1999)

- Motivation: RED extremely sensitive to #sources and parameter settings
- Idea: adapt $\max_p$ to load
  - If avg. queue $< \min_{th}$, decrease $\max_p$
  - If avg. queue $> \max_{th}$, increase $\max_p$
- No per-flow information needed
Variant: FRED  (Ling & Morris 1997)

- Motivation: marking packets in proportion to flow rate is unfair (e.g., adaptive vs non-adaptive flows)

- Idea
  - A flow can buffer up to $m_q$ packets w/o being marked
  - A flow that frequently buffers more than $\max_q$ packets gets penalized
  - All flows with backlogs in between are marked according to RED
  - No flow can buffer more than $\avg_{cq}$ packets persistently

- Need per-active-flow accounting
Variant: BLUE  (Feng, Kandlur, Saha, Shin 1999)

- Motivation: wild oscillation of RED leads to cyclic overflow & underutilization
- Algorithm
  - On buffer overflow, increment marking prob
  - On link idle, decrement marking prob
Variant: Stochastic Fair Blue

- Motivation: protection against non-adaptive flows
- Algorithm
  - $L$ hash functions map a packet to $L$ bins (out of $N \times L$)
  - Marking probability associated with each bin is
    - Incremented if bin occupancy exceeds threshold
    - Decremented if bin occupancy is 0
  - Packets marked with $\min \{p_1, \ldots, p_L\}$

![Diagram of hash function and bin occupancy](image)
SFB (contd)

- **Idea**
  - A non-adaptive flow drives marking prob to 1 at all $L$ bins it is mapped to
  - An adaptive flow may share *some* of its $L$ bins with non-adaptive flows
  - Non-adaptive flows can be identified and penalized with reasonable state overhead (not necessarily per-flow)
  - Large numbers of bad flows may cause false positives
Main ideas

- Decouple congestion & performance measure
- “Price” adjusted to match rate and clear buffer
- Marking probability exponential in `price’
Comparison of AQM Performance

**RED**
- $\min_{th} = 10$ pkts
- $\max_{th} = 40$ pkts
- $\max_p = 0.1$
- $\gamma = 0.05, \alpha = 0.4, \phi = 1.15$
- Queue = 1.5 pkts
- Utilization = 92%

**DropTail**
- Queue = 94%

**REM**
- Queue = 1.5 pkts
- Utilization = 92%
- $\gamma = 0.05, \alpha = 0.4, \phi = 1.15$
The DECbit Scheme

- Basic ideas:
  - Mark packets instead of dropping them
  - Special support at both routers and e2e

- Scheme:
  - On congestion, router sets *congestion indication* (CI) bit on packet
  - Receiver relays bit to sender
  - Sender adjusts sending rate

- Key design questions:
  - *When to set CI bit?*
  - *How does sender respond to CI?*
Setting CI Bit

AVG queue length = (previous busy+idle + current interval)/(averaging interval)
DECbit Routers

- Router tracks average queue length
  - **Regeneration cycle**: queue goes from empty to non-empty to empty
  - Average from start of previous cycle
  - If average > 1 → router sets bit for flows sending more than their share
  - If average > 2 → router sets bit in every packet
  - Threshold is a trade-off between queuing and delay
  - Optimizes power = (throughput / delay)
  - Compromise between sensitivity and stability
- Acks carry bit back to source
DECbit Source

- Source averages across acks in window
  - Congestion if > 50% of bits set
  - Will detect congestion earlier than TCP
- Additive increase, multiplicative decrease
  - Decrease factor = 0.875
  - Increase factor = 1 packet
- After change, ignore DECbit for packets in flight (vs. TCP ignore other drops in window)
- No slow start
Congestion Control Models

- **Loss-based**: TCP Reno etc
- **Accumulation-based** schemes: TCP Vegas, Monaco
  - Use per-flow queue contribution (backlog) as a congestion estimate instead of loss rate
- **Explicit rate-based** feedback
  - Controller at bottleneck assigns rates to each flow

- **Packet Pair** congestion control [Not covered]
  - WFQ at bottlenecks isolates flows, and gives fair rates
  - Packet-pair probing discovers this rate and sets source rate to that.
TCP Reno (Jacobson 1990)

SS: Slow Start
CA: Congestion Avoidance
Fast retransmission/fast recovery
TCP Vegas (Brakmo & Peterson 1994)

- Converges, no retransmission
- … provided buffer is large enough
Accumulation: Single Queue

- Flow \( i \) at router \( j \)
- Arrival curve \( A_{ij}(t) \)
- Service curve \( S_{ij}(t) \)
- Cumulative
- Continuous
- Non-decreasing
- If no loss, then

\[
\begin{align*}
\therefore q_{ij}(t) &= A_{ij}(t) - S_{ij}(t) \\
\therefore q_{ij}(t + \Delta t) &= A_{ij}(t + \Delta t) - S_{ij}(t + \Delta t) \\
\therefore \Delta q_{ij}(t, \Delta t) &= q_{ij}(t + \Delta t) - q_{ij}(t) \\
&= [A_{ij}(t + \Delta t) - A_{ij}(t)] - [S_{ij}(t + \Delta t) - S_{ij}(t)] \\
&= [\lambda_{ij}(t, \Delta t) - \mu_{ij}(t, \Delta t)] \times \Delta t \\
&= I_{ij}(t, \Delta t) - O_{ij}(t, \Delta t)
\end{align*}
\]
Accumulation: Series of Queues

- we have\[ \mu_{ij}(t - d_j) = \lambda_{i,j+1}(t) \quad \forall i, \forall 1 \leq j \leq J - 1 \]
- accumulation\[ a_i(t) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k) \quad d_i^f = \sum_{j=1}^{J-1} d_j \]
- then\[ \Delta a_i(t, \Delta t) = \sum_{j=1}^{J} \Delta q_{ij}(t - \sum_{k=j}^{J-1} d_k, \Delta t) \]
\[ = \sum_{j=1}^{J} [\lambda_{ij}(t - \sum_{k=j}^{J-1} d_k, \Delta t) - \mu_{ij}(t - \sum_{k=j}^{J-1} d_k, \Delta t)] \times \Delta t \]
\[ = [\lambda_i(t - d_i^f, \Delta t) - \mu_i(t, \Delta t)] \times \Delta t \]
\[ = I_i(t - d_i^f, \Delta t) - O_i(t, \Delta t) \]
Queue vs Accumulation Behavior

- queue \( q_{ij}(t) \) -- info of flow \( i \) queued in a fifo router \( j \)

\[
q_{ij}(t)
\]

\[
\Delta q_{ij}(t, \Delta t) = I_{ij}(t, \Delta t) - O_{ij}(t, \Delta t)
\]

- accumulation \( a_i(t) \) -- info of flow \( i \) queued in a set of fifo routers 1~J

\[
a_i(t) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k)
\]

\[
\Delta a_i(t, \Delta t) = I_i(t - d_i^f, \Delta t) - O_i(t, \Delta t)
\]

The collective queuing behavior of a set of fifo routers looks similar to that of one single fifo router.
Accumulation: Distributed, Time-shifted Sum

\[
q_{i1}(t - d_{i1}^f) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k)
\]

\[
I_i(t - d_{i1}^f, \Delta t) = \sum_{j=1}^{J} q_{ij}(t - \sum_{k=j}^{J-1} d_k)
\]

\[
a_i(t) = O_i(t, \Delta t) = a_i(t + \Delta t)
\]
Control Policy

- **control objective**: keep \( a_i(t) = \varepsilon_i > 0 \)
- if \( a_i(t) = 0 \), no way to probe increase of available bw;
- **control algorithm**:
  - if \( a_i(t) < \varepsilon_i \) then \( \lambda_i \uparrow \)
  - if \( a_i(t) > \varepsilon_i \) then \( \lambda_i \downarrow \)

\[
rec : \Delta a_i(t, \Delta t) = [\overline{\lambda}_i(t - d_i^f, \Delta t) - \overline{\mu}_i(t, \Delta t)] \times \Delta t
\]
Two Accumulation-Based Schemes

- Monaco
  - accumulation estimation: out-of-band / in-band
  - congestion response: additive inc / additive dec (aiad), etc.

- Vegas
  - accumulation estimation: in-band
  - congestion response: additive inc / additive dec (aiad)
Accumulation vs. Monaco Estimator

$q_{i1}(t - d_{i1}^f)$

$q_{ij}(t - \sum_{k=j}^{J-1} d_k)$

$t_q(i, j, t - \sum_{k=j}^{J-1} d_k)$

$q_{ij}(t)$

out-of-band

$q_m = a_i(t)$

in-band

ctrl pkt

Shivkumar Kalyanaraman
Accumulation vs. Monaco Estimator

\[ \Lambda_i, J_b, f_i \]

\[ j_f, d_{jf}, j_{f+1} \]

\[ \Lambda_{i,j+1}, \mu_i, J_f \]

\[ f_{i+1}, \text{out-of-bd ctrl}, \text{in-band ctrl, data pkt} \]

\[ \text{fifo} \]

\[ \text{data, ctrl} \]

\[ \text{ctrl} \]
- **congestion estimation:**
  - out-of-band and in-band control packets

- **congestion response:**
  - if $q_m < \alpha$, $cwnd(k+1) = cwnd(k) + 1$;
  - if $q_m > \beta$, $cwnd(k+1) = cwnd(k) - 1$; [$1 = \alpha < \beta = 3$]
TCP Vegas

- **congestion estimation:**
  - define $q_v = (\frac{cwnd}{rtt_p} - \frac{cwnd}{rtt}) \times rtt_p$;
  - where $rtt_p$ is round trip propagation delay (basertt)

- **congestion response:**
  - if $q_v < \alpha$, $cwnd(k+1) = cwnd(k) + 1$;
  - if $q_v > \beta$, $cwnd(k+1) = cwnd(k) - 1$; [$1 = \alpha < \beta = 3$]
Vegas Accumulation Estimator

- the physical meaning of $q_v$
  - $rtt = rtt_p + rtt_q$  \[ \text{[ rtt}_q \text{ is queuing time]} \]
  - $q_v = \left( \frac{cwnd}{rtt_p} - \frac{cwnd}{rtt} \right) \times rtt_p$
  - $= \left( \frac{cwnd}{rtt} \right) \times (rtt - rtt_p)$
  - $= \left( \frac{cwnd}{rtt} \right) \times rtt_q$  \[ \text{[ if rtt is typical]} \]
  - $= \text{sending rate} \times rtt_q$  \[ \text{[ little’s law]} \]
  - $= \text{packets backlogged}$  \[ \text{[ little’s law again]} \]

- so vegas maintains $\alpha \sim \beta$ number of packets queued inside the network

- it adjusts sending rate \textbf{additively} to achieve this
Accumulation vs. Vegas estimator

- Backlog, \( q_{v,i}(t) = \overline{\lambda}_i(t) \times (rtt_{q}^f + rtt_{q}^b) \)

\[
\approx \sum_{j_f=1}^{J_f} q_{i,j_f} (t - d_i^b - \sum_{m=j_f}^{J_f} d_m^f ) \\
+ \sum_{j_b=1}^{J_b} q_{i,j_b} (t - \sum_{n=j_b}^{J_b} d_n^b ) \\
= a_i^f (t - d_i^b ) + a_i^b (t )
\]
Vegas vs. Monaco estimators

- Vegas accumulation estimator
  - ingress-based
  - round trip (forward data path and backward ack path)
  - sensitive to ack path queuing delay
  - sensitive to round trip propagation delay
  - measurement error

- Monaco accumulation estimator
  - egress-based
  - one way (only forward data path)
  - insensitive to ack path queuing delay
  - no need to explicitly know one way propagation delay
Queue, Utilization w/ Basertt Errors

(a1) Vegas Queue Length
(b1) Vegas-k Queue Length
(c1) Monaco-AIAD Queue Length

(a2) Vegas Utilization
(b2) Vegas-k Utilization
(c2) Monaco-AIAD Utilization
TCP Modeling

- Given the congestion behavior of TCP can we predict what type of performance we should get?
- What are the important factors
  - **Loss rate**
    - Affects how often window is reduced
  - **RTT**
    - Affects increase rate and relates BW to window
  - **RTO**
    - Affects performance during loss recovery
  - **MSS**
    - Affects increase rate
Overall TCP Behavior

• Let’s focus on steady state (congestion avoidance) with no slow starts, no timeouts and perfect loss recovery

☐ Some additional assumptions
  ☐ Fixed RTT
  ☐ No delayed ACKs
Each cycle delivers $2w^2/3$ packets

Assume: each cycle delivers $1/p$ packets = $2w^2/3$

Delivers $1/p$ packets followed by a drop

=> Loss probability = $p/(1+p) \sim p$ if $p$ is small.

Hence $w = \sqrt{3/2p}$

Shivkumar Kalyanaraman
Alternate Derivation

- **Assume**: loss is a Bernoulli process with probability $p$
- **Assume**: $p$ is small
- $w_n$ is the window size after $n$th RTT

\[
\begin{align*}
  w_{n+1} &= \begin{cases} 
    w_n / 2, & \text{if a packet is lost (prob. } p w_n) \\
    w_n + 1, & \text{if no packet is lost (prob. } (1 - p w_n)) 
  \end{cases} \\
  \bar{w} &= \frac{\bar{w}}{2} p \bar{w} + (\bar{w} + 1)(1 - p \bar{w}) \\
  \bar{w}^2 &\approx 2 / p \\
  \bar{w} &\approx \sqrt{2 / p}
\end{align*}
\]
Equilibrium window size

\[ w_s = \frac{a}{\sqrt{p}} \]

Equilibrium rate

\[ x_s = \frac{a}{D_s \sqrt{p}} \]

Empirically constant \( a \sim 1 \)

Verified extensively through simulations and on Internet

References

Implications

- **Applicability**
  - Additive increase multiplicative decrease (Reno)
  - Congestion avoidance dominates
  - No timeouts, e.g., SACK+RH
  - **Small** losses
  - Persistent, greedy sources
  - Receiver not bottleneck

- **Implications**
  - Reno equalizes window
  - Reno discriminates against long connections
  - Halving throughput => quadrupling loss rate!
Refinement (Padhye, Firoin, Towsley & Kurose 1998)

- Renewal model including
  - FR/FR with Delayed ACKs (b packets per ACK)
  - Timeouts
  - Receiver wnd limitation
- Source rate

When $p$ is small and $W_r$ is large, reduces to

$$x_s = \frac{a}{D_s \sqrt{p}}$$
TCP Friendliness

- What does it mean to be TCP friendly?
  - TCP is not going away
  - Any new congestion control must compete with TCP flows
    - Should not clobber TCP flows and grab bulk of link
    - Should also be able to hold its own, i.e. grab its fair share, or it will never become popular
Binomial Congestion Control

- In AIMD
  - Increase: $W_{n+1} = W_n + \alpha$
  - Decrease: $W_{n+1} = (1- \beta) W_n$

- In Binomial
  - **Increase**: $W_{n+1} = W_n + \frac{\alpha}{W_n^k}$
  - **Decrease**: $W_{n+1} = W_n - \beta W_n^l$
  - $k=0$ & $l=1 \rightarrow$ AIMD
  - $l < 1$ results in less than multiplicative decrease
    - Good for multimedia applications
Binomial Congestion Control

- Rate $\sim \frac{1}{(\text{loss rate})^{1/(k+l+1)}}$
- If $k+l=1 \rightarrow \text{rate} \sim \frac{1}{p^{0.5}}$
  - TCP friendly
- AIMD ($k=0$, $l=1$) is the most aggressive of this class
- SQRT ($k=1/2$, $l=1/2$) and IIAD ($k=1$, $l=0$)
- Good for applications that want to probe quickly and can use any available bandwidth
Static Optimization Framework

Duality theory $\rightarrow$ equilibrium
- Source rates $x_i(t)$ are \textit{primal} variables
- Congestion measures $p_i(t)$ are \textit{dual} variables
- Congestion control is optimization process over Internet

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
Overview: equilibrium

- Interaction of source rates $x_s(t)$ and congestion measures $p_j(t)$
- Duality theory
  - They are primal and dual variables
  - Flow control is optimization process
- Example congestion measure
  - Loss (Reno)
  - Queueing delay (Vegas)
Overview: equilibrium

- Congestion control problem

\[
\max_{x_s \geq 0} \sum_s U_s(x_s)
\]
subject to \( x^l \leq c_l, \quad \forall l \in L \)

- Primal-dual algorithm

\[
x(t+1) = F(p(t), x(t))
\]
\[
p(t+1) = G(p(t), x(t))
\]

- TCP/AQM protocols \((F, G)\)
  - Maximize aggregate source utility
  - With different utility functions \(U_s(x_s)\)

Reno, Vegas
DropTail, RED, REM

Shivkumar Kalyanaraman
Model

- **Sources** $s$
  - $L(s)$ - links used by source $s$
  - $U_s(x_s)$ - utility if source rate = $x_s$
- **Network**
  - Links $l$ of capacities $c_l$
**Primal problem**

\[
\begin{align*}
\max_{x_s \geq 0} & \quad \sum_s U_s(x_s) \\
\text{subject to} & \quad x^l \leq c_l, \quad \forall l \in L
\end{align*}
\]

- **Assumptions**
  - Strictly concave increasing \( U_s \)
  - Unique optimal rates \( x_s \) exist
  - Direct solution impractical
Duality Approach

Primal: \[ \max_{x_s \geq 0} \sum_s U_s(x_s) \quad \text{subject to } x^l \leq c_l, \quad \forall l \in L \]

Dual: \[ \min_{p \geq 0} D(p) = \left( \max_{x_s \geq 0} \sum_s U_s(x_s) + \sum_l p_l(c_l - x^l) \right) \]

Primal-dual algorithm:

\[ x(t+1) = F(p(t), x(t)) \]
\[ p(t+1) = G(p(t), x(t)) \]
Gradient algorithm

source: \( x_i(t+1) = U_i^{-1}(q_i(t)) \)

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

**Theorem** (Low, Lapsley, 1999)

Converges to optimal rates in an asynchronous environment
Example

\[
\begin{align*}
\text{max} & \quad \sum_s \log x_s \\
\text{subject to} & \quad x_1 + x_2 \leq 1 \\
& \quad x_1 + x_3 \leq 1
\end{align*}
\]

Lagrange multiplier: \( p_1 = p_2 = 3/2 \)

Optimal: \( x_1 = 1/3 \), \( x_2 = x_3 = 2/3 \)
Example

- $x_s$ : proportionally fair \textit{(Vegas)}
- $p_i$ : Lagrange multiplier, (shadow) price, congestion measure
- How to compute $(x, p)$?
  - \textit{Gradient algorithms, Newton algorithm, Primal-dual algorithms}...
- Relevance to TCP/AQM ??
  - \textit{TCP/AQM protocols implement primal-dual algorithms over Internet}
Example

- \( x_s \): proportionally fair \((Vegas)\)
- \( p_t \): Lagrange multiplier, (shadow) price, congestion measure
- How to compute \((x, p)\)?
  - Gradient algorithms, Newton algorithm, Primal-dual algorithms…
- Relevance to TCP/AQM ??
  - TCP/AQM protocols implement primal-dual algorithms over Internet

\[
p_1(t + 1) = \left[ p_1(t) + \gamma \left( x_1(t) + x_2(t) - 1 \right) \right]^+
\]

\[
p_2(t + 1) = \left[ p_2(t) + \gamma \left( x_1(t) + x_3(t) - 1 \right) \right]^+
\]

\[
x_1(t + 1) = \frac{1}{p_1(t) + p_2(t)};
\]

\[
x_2(t + 1) = \frac{1}{p_1(t)}; \quad x_3(t + 1) = \frac{1}{p_2(t)};
\]

Aggregate rate
Active queue management

- Idea: provide congestion information by probabilistically marking packets

- Issues
  - How to measure congestion ($p$ and $G$)?
  - How to embed congestion measure?
  - How to feed back congestion info?

$$x(t+1) = F(p(t), x(t))$$

$$p(t+1) = G(p(t), x(t))$$

Reno, Vegas

DropTail, RED, REM
RED (Floyd & Jacobson 1993)

- Congestion measure: average queue length
  \[ p_l(t+1) = [p_l(t) + x^l(t) - c_l]^+ \]

- Embedding: p-linear probability function

![Graph showing marking against average queue length]
REM (Athuraliya & Low 2000)

- Congestion measure: price
  \[ p_{l}(t+1) = p_{l}(t) + \gamma(\alpha_{l} b_{l}(t) + x^{l}(t) - c_{l}) \]

- Embedding: exponential probability function
Key features

- Clear buffer and match rate

\[ p_l(t+1) = \left[ p_l(t) + \gamma \left( \alpha_l b_l(t) + \hat{x}'(t) - c_l \right) \right]^+ \]

- Sum prices

\[ 1 - \phi^{-p_l(t)} \Rightarrow 1 - \phi^{-p^s(t)} \]

**Theorem** (Paganini 2000)

Global asymptotic stability for general utility function (in the absence of delay)
## AQM Summary

<table>
<thead>
<tr>
<th></th>
<th>$p_l(t)$</th>
<th>$G(p(t), x(t))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropTail</td>
<td>loss</td>
<td>$[1 - c_l/x^l(t)]^+$ (?)</td>
</tr>
<tr>
<td>RED</td>
<td>queue</td>
<td>$[p_l(t) + x^l(t) - c_l]^+$</td>
</tr>
<tr>
<td>Vegas</td>
<td>delay</td>
<td>$[p_l(t) + x^l(t)/c_l - 1]^+$</td>
</tr>
<tr>
<td>REM</td>
<td>price</td>
<td>$[p_l(t) + \gamma(\alpha_l b_l(t) + x^l(t) - c_l)]^+$</td>
</tr>
</tbody>
</table>

$x(t+1) = F(p(t), x(t))$  
$p(t+1) = G(p(t), x(t))$  

Reno, Vegas: $F(p(t), x(t))$  
DropTail, RED, REM: $G(p(t), x(t))$
Reno: \( F \)

\[
\Delta w_s(t) = \frac{x_s(t)(1 - p(t))}{w_s} - \frac{w_s(t)}{2} x_s(t) p(t)
\]

\[
F_s(p(t), x(t)) = x_s(t) + \frac{(1 - p(t))}{D_s^2} - \frac{x_s^2(t)}{2} p(t)
\]

Primal-dual algorithm:

\[
x(t+1) = F(p(t), x(t))
\]

\[
p(t+1) = G(p(t), x(t))
\]

Reno, Vegas

DropTail, RED, REM
Reno Implications

- Equilibrium characterization
  \[ \frac{1 - q_i}{\tau_i^2} = \frac{x_i^2}{2} q_i \]

- Duality
  \[ U_{s}^{reno} (x_s) = \frac{\sqrt{2}}{\tau_i} \tan^{-1}\left(\frac{x_i \tau_i}{2}\right) \]

- Congestion measure \( p = \text{loss} \)
- Implications
  - Reno equalizes window \( w_i = \tau_i x_i \)
  - inversely proportional to delay \( \tau_i \)
  - \( 1/\sqrt{p} \) dependence for small \( p \)
  - DropTail fills queue, regardless of queue capacity
Reno & gradient algorithm

- **Gradient algorithm**
  
  source: \[ x_i(t+1) = U_i^{-1}(q_i(t)) \]
  
  link: \[ p_l(t+1) = [p_l(t) + \gamma(y_l(t) - c_l)]^+ \]

- **TCP approximate version of gradient algorithm**
  
  \[
  F_i(q_i(t), x_i(t)) = x_i(t) + \frac{(1 - q_i(t))}{\tau_i^2} - \frac{x_i^2(t)}{2}q_i(t)
  \]
Reno & gradient algorithm

Gradient algorithm

source: \[ x_i(t+1) = U_i^{-1}(q_i(t)) \]

link: \[ p_i(t+1) = [p_i(t) + \gamma (y_i(t) - c_i)]^+ \]

TCP approximate version of gradient algorithm

\[ x_i(t+1) = \left[ x_i(t) + \frac{q_i(t)}{2} (\bar{x}_i^2(t) - x_i^2(t)) \right]^+ \]
Vegas

\[
\text{for every RTT} \\
\{ \begin{align*}
\text{if } & W_{\text{RTT}_{\text{min}}} - W_{\text{RTT}} < \alpha \text{ then } W++ \\
\text{if } & W_{\text{RTT}_{\text{min}}} - W_{\text{RTT}} > \alpha \text{ then } W-- \end{align*} \}
\]

\text{for every loss} \quad W := W/2

\[
F: \quad x_s(t+1) = \begin{cases} 
   x_s(t) + \frac{1}{D^2_s} & \text{if } w_s(t) - d_s x_s(t) < \alpha_s d_s \\
   x_s(t) - \frac{1}{D^2_s} & \text{if } w_s(t) - d_s x_s(t) > \alpha_s d_s \\
   x_s(t) & \text{else}
\end{cases}
\]

\[
G: \quad p_l(t+1) = [p_l(t) + x^l(t)/c_l - 1]^+
\]
Advanced Topics In Congestion Control
TCP over Highly Lossy Networks

LT-TCP: TCP over Lossy Networks

Rensselaer Polytechnic Institute
Shivkumar Kalyanaraman
Wireless PHY: Performance Variability

- Herculean challenges: Path loss, shadowing, multipath, doppler
- TCP sees a variable performance channel w/ bursty residual losses

Capacity in Nakagami Fading ($m = 2$).
Wireless Mesh Networks

- Well provisioned and well managed wireless links will have low average erasure rates
  - But burst losses and temporarily lousy channels can still lead to:
    - variable capacity (eg: multi-rate) and residual packet erasures
  - PHY and MAC layers have limits on error resilience support
  - Non-standard => different links have different ARQ/FEC support

- Wireless Mesh Infrastructures:
  - More opportunity for residual erasures and capacity variation as seen by TCP (end-to-end)
  - MIT’s GRID project reported significant loss rates 40-60% due to various PHY phenomena
Problem Motivation

- **Dynamic Range:**
  - Can we extend the dynamic range of TCP into high loss regimes?
  - Can TCP perform close to the residual capacity available under high loss rates?

- **Congestion Response:**
  - How should TCP respond to notifications due to congestion?
  - … but not respond to packet erasures that do not signal congestion?

- **Mix of Reliability Mechanisms:**
  - What mechanisms should be used to extend the operating point of TCP into loss rates from 0% - 30% - 50% packet loss rate?
  - How can Forward Error Correction (FEC) help?
  - How should the FEC be split between sending it proactively (insuring the data in anticipation of loss) and reactively (sending FEC in response to a loss)?

- **Timeout Avoidance:**
  - Timeouts: Useful as a fall-back mechanism but wasteful otherwise especially under high loss rates.
  - How can we add mechanisms to minimize timeouts?
Transport/Link Layer: Reliability Model

- Sequence Numbers
- CRC or Checksum
- Proactive FEC

- ACKs
- NAKs, dupacks
- SACKs
- Bitmaps

- Rexmitted Packets
- Reactive FEC
Building Blocks...

- **ECN-Environment:**
  - We infer congestion solely based on ECN markings.
  - Window is cut in response to
    - ECN signals: hosts/routers have to be ECN-capable.
    - Timeouts: The response to a timeout is the same as with standard TCP.

- **Window Granulation and Adaptive MSS:**
  - We ensure that the window always has at least G segments (allows for dupacks to help recover from loss at small windows)
  - Avoids timeouts
    - Window size in bytes initially is the same as normal SACK TCP.
  - Initial segment size is small to accommodate G segments.
  - Packet size is continually adjusted so that we have at least G segments. Once we have G segments, packet size increases with window size.

- **Loss Estimation:**
  - The receiver continually tracks the loss rate and provides a running estimate of perceived loss back to the TCP sender through ACKs.
  - We use an EWMA smoothed estimate of packet erasure rate.
Building Blocks: Adaptive MSS

- Smaller MSS when higher loss rate anticipated

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
**BBlocks: Reed-Solomon FEC: RS(N,K)**

- **RS(N,K)**
- **FEC (N-K)**
- **Block Size (N)**
- **Data = K**
- **Lossy Network**
- **>= K of N received**
- **Recover K data packets!**
Building Blocks ...

- **Proactive FEC:**
  - TCP sender sends data in blocks where the block contains $K$ data segments and $R$ FEC packets. The amount of FEC protection ($K$) is determined by the current loss estimate.
  - Proactive FEC based upon estimate of per-window loss rate (Adaptive)

- **Reactive FEC:**
  - Upon receipt of *dupacks* Reactive FEC packets are sent based on the following criteria.
    - Number of Proactive FEC packets already sent.
    - Cumulative hole size seen in the decoding block at the receiver.
    - Loss rate currently estimated.
    - Reactive FEC to complement retransmissions
  - both used to reconstruct packets at receiver
Shortened Reed Solomon FEC (per-Window)

RS(N,K)

Zeros (z)

Reactive FEC (R)

Proactive FEC (F)

Data = d

Window (W)

RS(N,K)

Block Size (N)

K = d + z

Data = d

Window (W)
Putting it Together….

Application Data

MSS Adaptation

Granulated Window Size

Parameter Estimation

(n,k)

FEC Computation

Loss Estimate

Window

P-FEC Data
LT-TCP vs SACK Performance

Performance of SACK and LT-TCP (Single Flow)

Performance of SACK and LT-TCP (Multiple Flows)

Comparison of LT-TCP Performance to Theoretical Optimum

2.75 Mbps out of 5 Mbps MAX at 50% PER!
SACK vs. LTTTCP as function of RTT

- TCP Goodput drops more drastically with larger RTTs as error rate increases.
- LTTTCP on the other hand has a more linear fall comparatively even at 50% end-end loss-rate.
LT-TCP and SACK w/ Bursty Losses: Gilbert Error Model

- Comparative results with Bursty Errors
  - Gilbert ON-OFF (50% duty cycle) model with loss probability $2p$ during the ON periods
Summary: LT-TCP

- Improvement in TCP performance over lossy links with residual erasure rates 0-50% (short- or long-term).
- End-to-End FEC and Adaptive MSS:
  - Granulation ensures better flow of ACKs especially in \textit{small window regime}.
  - Adaptive FEC (proactive and reactive) can protect critical packets appropriately
  - Adaptive => No overhead when there is no loss.
- ECN to distinguish congestion from loss
- Ongoing Work:
  - Study of interaction between LT-TCP and link-layer schemes in 802.11b networks.
  - Optimal division of reliability functions between PHY, MAC, Transport layer
ATM ABR Explicit Rate Feedback

- Sources regulate transmission using a "rate" parameter
- **Feedback scheme:**
  - Every $(n+1)$th cell is an RM (control) cell containing current cell rate, allowed cell rate, etc
  - Switches adjust the rate using rich information about congestion to calculate explicit, multi-bit feedback
  - Destination returns the RM cell to the source
- **Control policy:** Sources adjust to the new rate
ERICA: Design Goals

- Allows utilization to be 100% (better tracking)
- Allows operation at any point between the knee and the cliff
  - The queue length can be set to any desired value (tracking).
- Max-min fairness (fairness)
Efficiency vs Fairness: OSU Scheme

- Efficiency = high utilization
- Fairness = Equal allocations for contending sources
- Worry about fairness after utilization close to 100% utilization. Target Utilization (U) and Target Utilization Band (TUB).

<table>
<thead>
<tr>
<th>Total Load</th>
<th>Overload Region</th>
<th>Underload Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>U=95%</td>
<td>91%</td>
</tr>
</tbody>
</table>

worry about fairness here
ERICA Switch Algorithm

- **Overload** = Input rate/Target rate
- **Fair Share** = Target rate/# of active VCs
- **This VC’s Share** = VC’s rate / Overload
- **ER** = Max(Fair Share, This VC’s Share)
- **ER in Cell** = Min(ER in Cell, ER)

- This is the basic algorithm.
- Has more steps for improved fairness, queue management, transient spike suppression, averaging of metrics.
TCP Rate Control

- **Step 1:** Explicit control of window:
  - Congestion window (CWND)
  - Actual Window = Min(Cwnd, Wr)

- **Step 2:** Control rate of acks (*ack-bucket*): Tradeoff ack queues in reverse path for fewer packets in forward path
Can we use fewer bits than explicit rate feedback?
Why? #1: TCP does not scale for Large-BD Products

TCP + RED / ECN (abbrev. TCP)

TCP
Motivation #1: Why TCP does not scale?

- Both loss and one-bit Explicit Congestion Notification (ECN) are binary congestion signals.
- Additive increase is too slow when bandwidth is large.
Motivation #2: XCP (or Explicit Rate Control) scales

- It decouples efficiency control and fairness control
- But, it needs multiple bits (128 bits in its current IETF draft) to carry the congestion-related information from/to network
Variable-structure congestion Control Protocol (VCP)

- Routers signal roughly the level of congestion
- End-hosts adapt the control algorithm accordingly

<table>
<thead>
<tr>
<th>control</th>
<th>load</th>
<th>region</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplicative Decrease (MD)</td>
<td>overload</td>
<td>(11)</td>
<td></td>
</tr>
<tr>
<td>Additive Increase (AI)</td>
<td>high-load</td>
<td>(10)</td>
<td></td>
</tr>
<tr>
<td>Multiplicative Increase (MI)</td>
<td>low-load</td>
<td>(01)</td>
<td></td>
</tr>
</tbody>
</table>

(sender) — traffic rate — router — link capacity — receiver

- Multiplicative Decrease (MD)
- Additive Increase (AI)
- Multiplicative Increase (MI)
An illustration example

- MI tracks available bandwidth exponentially fast
- After high utilization is attained, AIMD provides fairness
VCP key ideas and properties

- Using network link load factor as the congestion signal

- Decoupling efficiency/fairness control in different load regions
  - It approaches XCP’s efficiency, loss, and queue behavior
  - Its fairness model and convergence are similar to TCP
  - Its fairness convergence is much slower than XCP
Three key design issues

- At the router
  - How to measure and encode the load factor?

- At the end-host
  - What MI/AI/MD parameters to use? When to switch from MI to AI?
  - How to handle heterogeneous RTTs?
VCP achieves high efficiency

bottleneck utilization

![Graph showing bottleneck utilization vs bandwidth (Mbps)]

bottleneck utilization

![Graph showing bottleneck utilization vs round-trip delay (ms)]
VCP minimizes packet loss rate

packet loss rate

TCP

XCP  VCP

bandwidth (Mbps)

packet loss rate

TCP

XCP  VCP

round-trip delay (ms)
VCP keeps small bottleneck queue

- **Queue length in % buffer size**
- **Bandwidth (Mbps)**
  - TCP
  - XCP
  - VCP

- **Round-trip delay (ms)**
  - VCP
  - XCP
  - TCP
Summary

- Active Queue Management (AQM): RED, REM etc
- Alternative models:
  - Accumulation-based schemes: Monaco, Vegas
- TCP stochastic modeling:
- Static (Duality) Optimization Framework
- Advanced Topics:
  - Loss-Tolerant TCP for highly lossy networks
  - Explicit Rate-based Schemes, 2-bit Feedback Scheme