CEG 7450: Differentiated Services

- Reading
- **[B+98]** S. Blake et al, "An Architecture for Differentiated Services", RFC 2475, December 1998

Overview

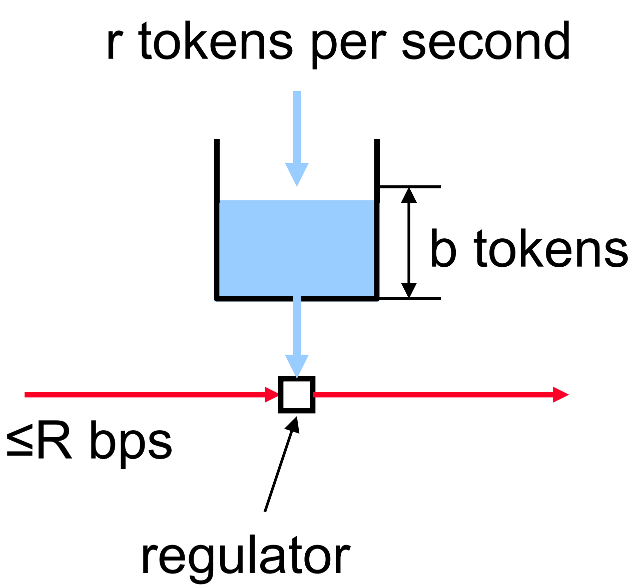
- Review of traffic and service characterization
- Differentiated services

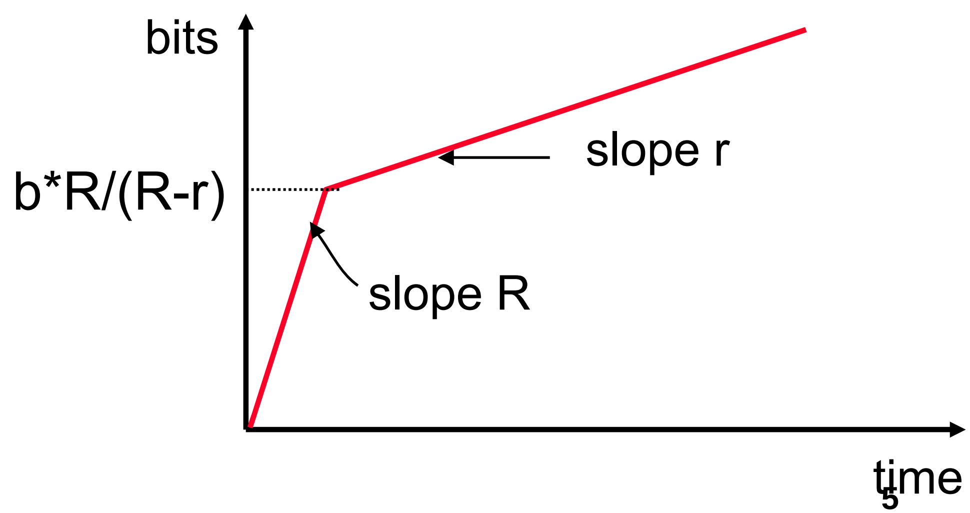
Traffic and Service Characterization

- To quantify a service one has to know
  - Flow's traffic arrival
  - Service provided by the router, i.e., resources reserved at each router
- Examples:
  - Traffic characterization: token bucket
  - Service provided by router: fix rate and fix buffer space

Token Bucket

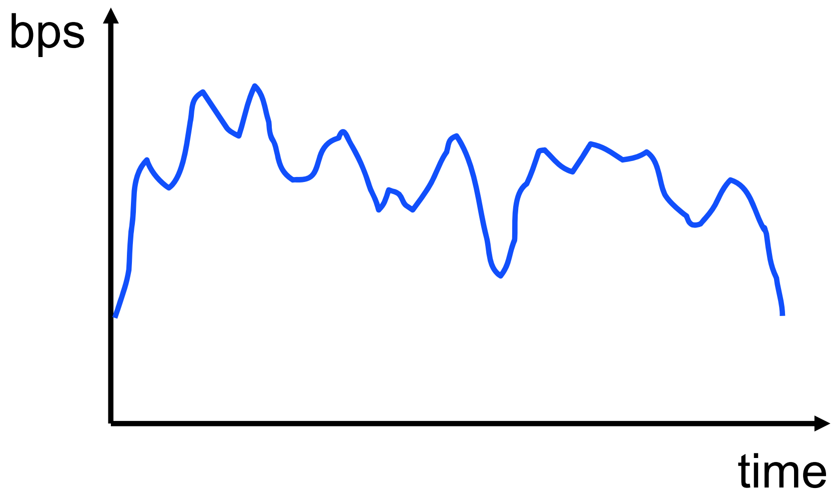
- Characterized by three parameters (b, r, R)
  - b – token depth
  - r – average arrival rate
  - R – maximum arrival rate (e.g., R link capacity)
- A bit is transmitted only when there is an available token
  - When a bit is transmitted exactly one token is consumed

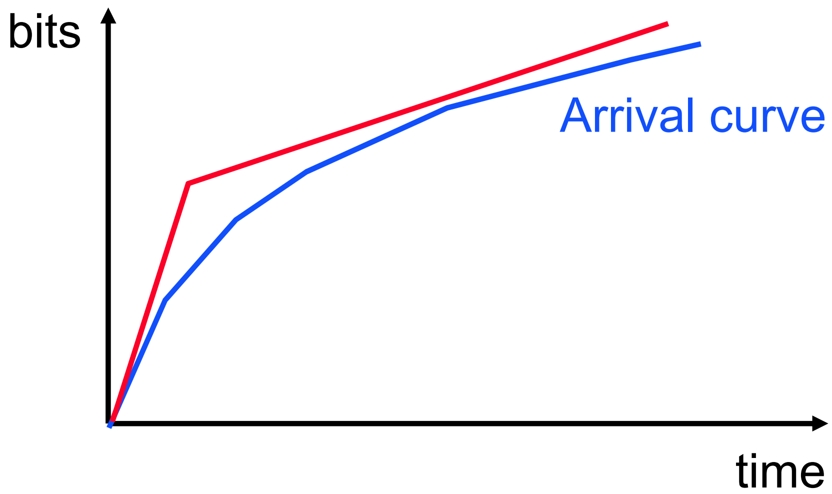




Characterizing a Source by Token Bucket

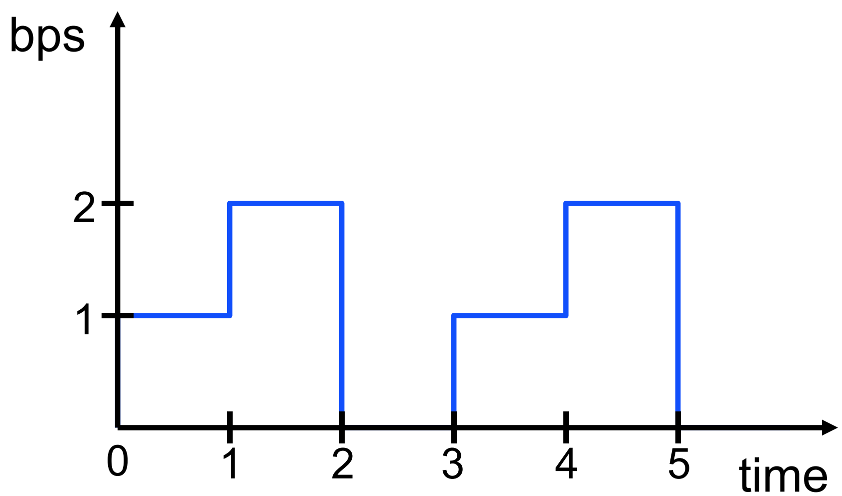
- Arrival curve – maximum amount of bits transmitted by time t
- Use token bucket to bound the arrival curve

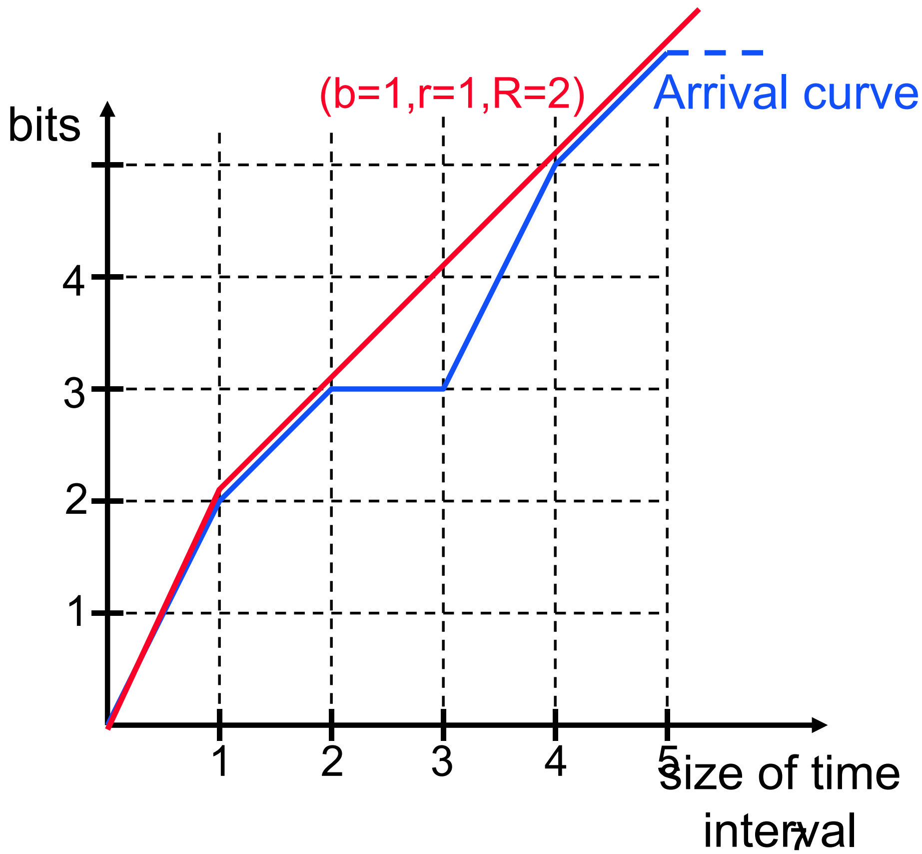




Example

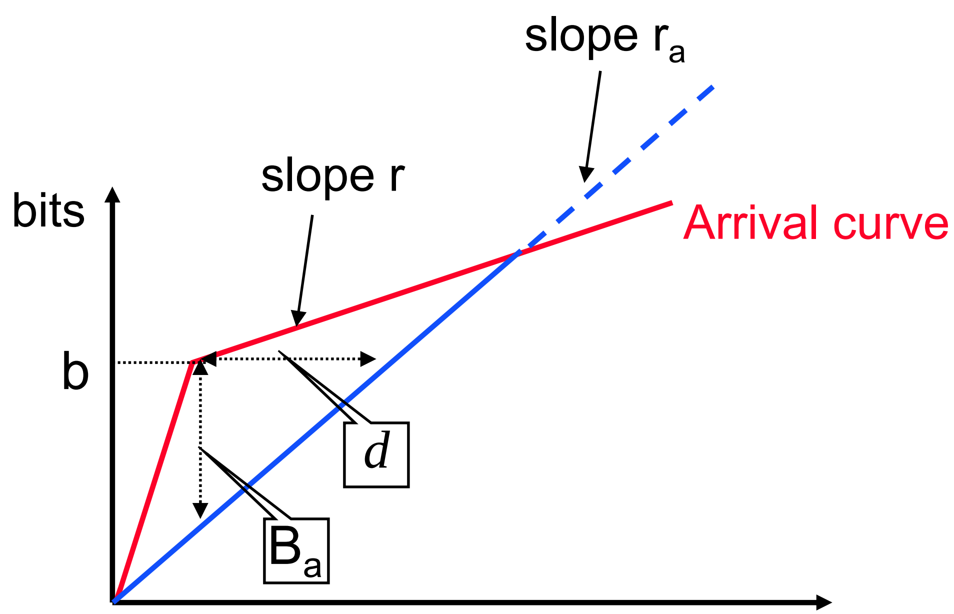
- Arrival curve – maximum amount of bits transmitted in an interval of size t
- Use token bucket to bound the arrival curve





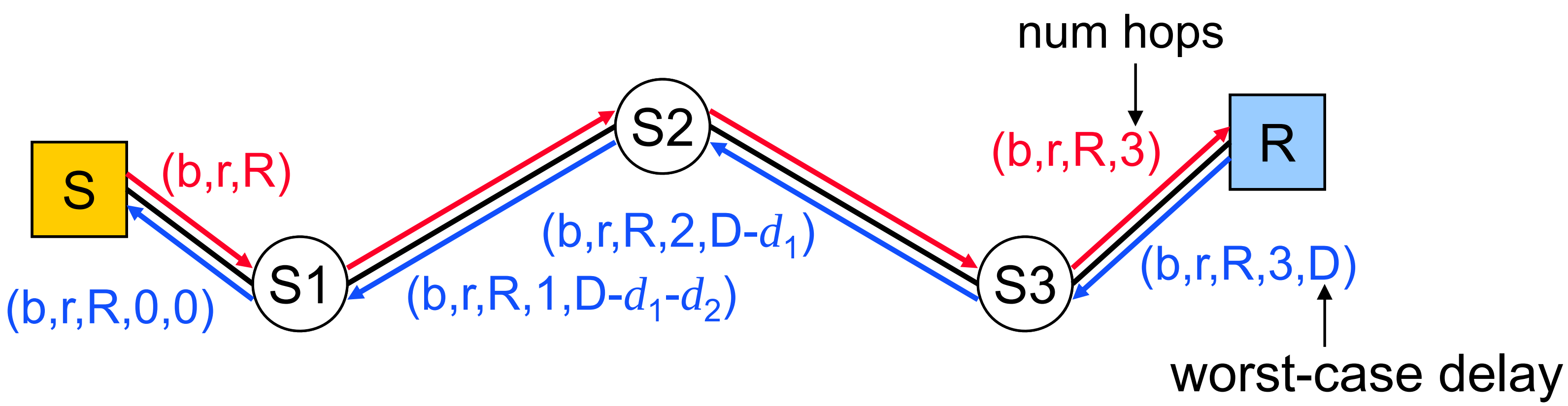
Per-hop Reservation

- Given b,r,R and per-hop delay d
- Allocate bandwidth r_a and buffer space B_a such that to guarantee d



End-to-End Reservation

- Source S sends a message containing traffic characteristics
  - r,b,R
  - This message is used to computes the number of hops
- Receiver R sends back this information + worst-case delay (D)
- Each router along path provide a per-hop delay guarantee and forwards the message
  - In simplest case routers split the delay D



Overview

- Review of traffic and service characterization
- Differentiated services

What is the Problem?

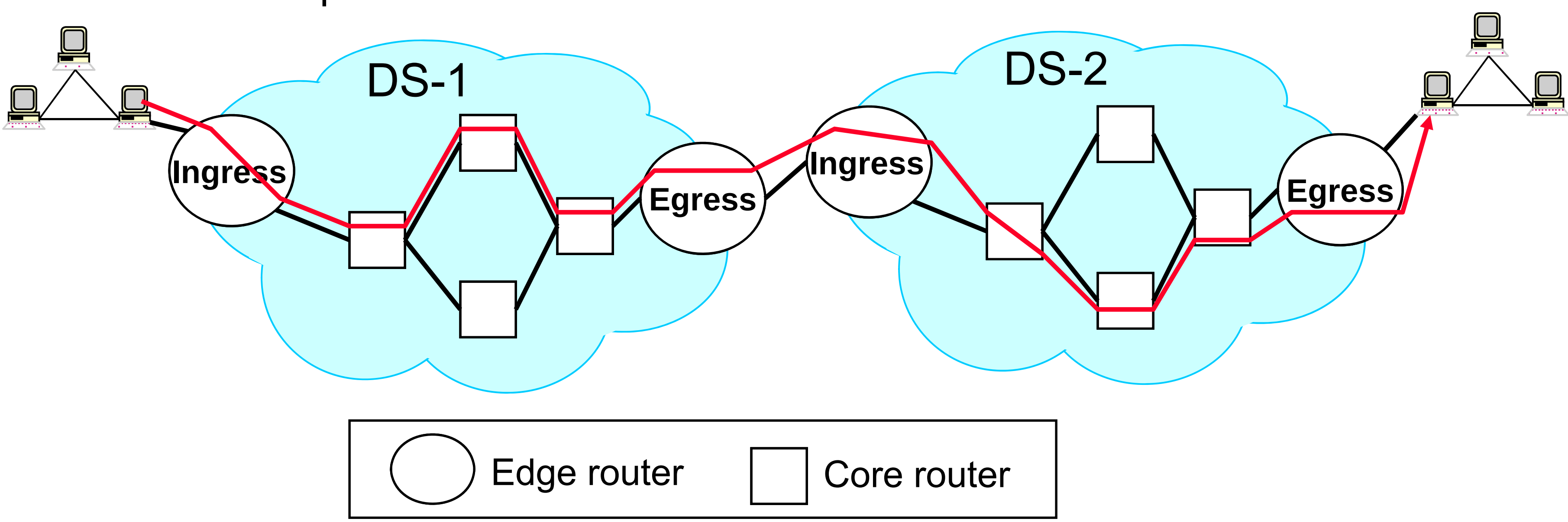
- Goal: provide support for wide variety of applications:
  - Interactive TV, IP telephony, on-line gamming (distributed simulations), VPNs, etc
- Problem:
  - Best-effort cannot do it (see previous lecture)
  - Intserv can support all these applications, but
    - Too complex
    - Not scalable

Differentiated Services (Diffserv)

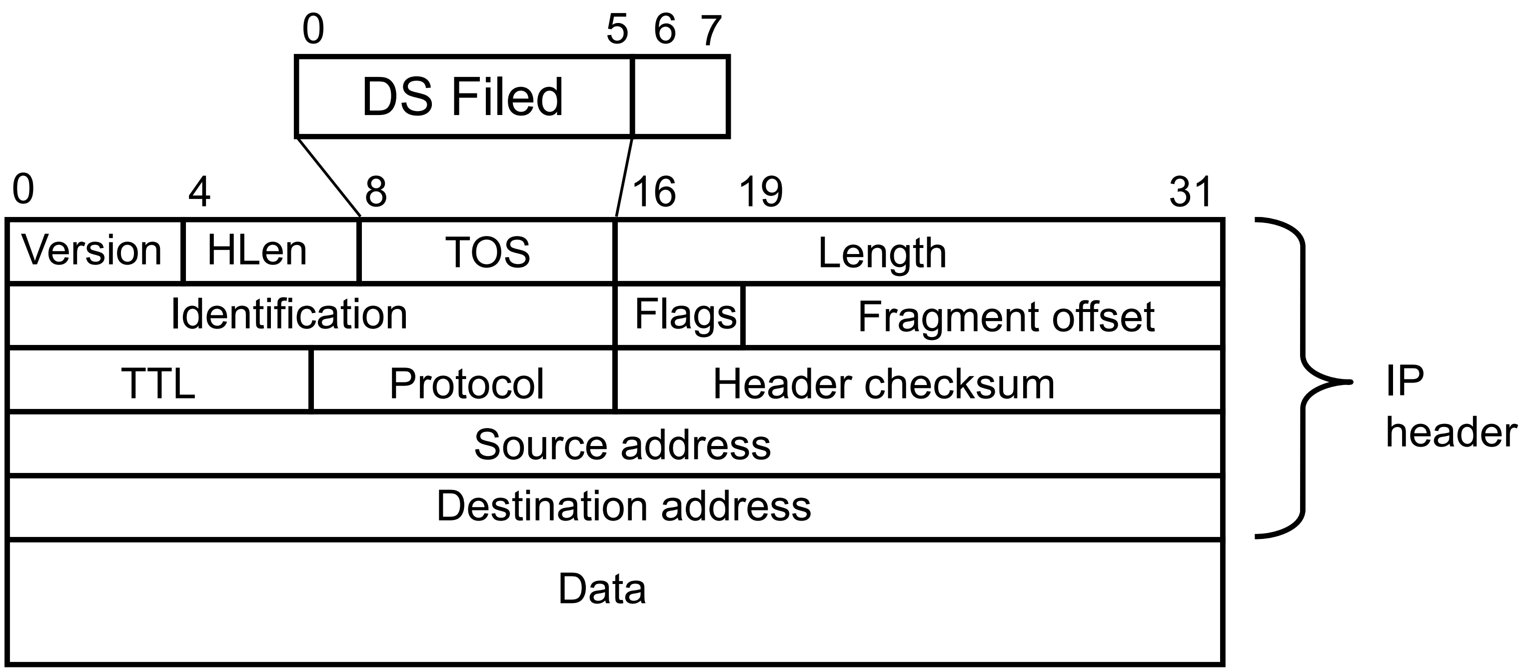
- Build around the concept of domain
- Domain – a contiguous region of network under the same administrative ownership
- Differentiate between edge and core routers
- Edge routers
  - Perform per aggregate shaping or policing
  - Mark packets with a small number of bits; each bit encoding represents a class (subclass)
- Core routers
  - Process packets based on packet marking
- Far more scalable than Intserv, but provides weaker services

Diffserv Architecture

- Ingress routers
  - Police/shape traffic
  - Set Differentiated Service Code Point (DSCP) in Diffserv (DS) field
- Core routers
  - Implement Per Hop Behavior (PHB) for each DSCP
  - Process packets based on DSCP



Differentiated Service (DS) Field



- DS field reuses the first 6 bits from the former Type of Service (TOS) byte
- The other two bits are proposed to be used by ECN

Differentiated Services

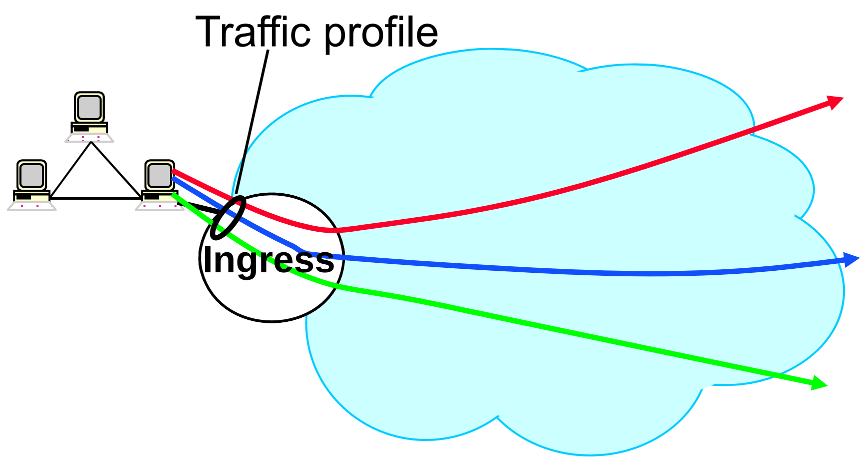
- Two types of service
  - Assured service
  - Premium service
- Plus, best-effort service

Assured Service [Clark & Wroclawski '97]

- Defined in terms of user profile, how much assured traffic is a user allowed to inject into the network
- Network: provides a lower loss rate than best-effort
  - In case of congestion best-effort packets are dropped first
- User: sends no more assured traffic than its profile
  - If it sends more, the excess traffic is converted to best-effort

Assured Service

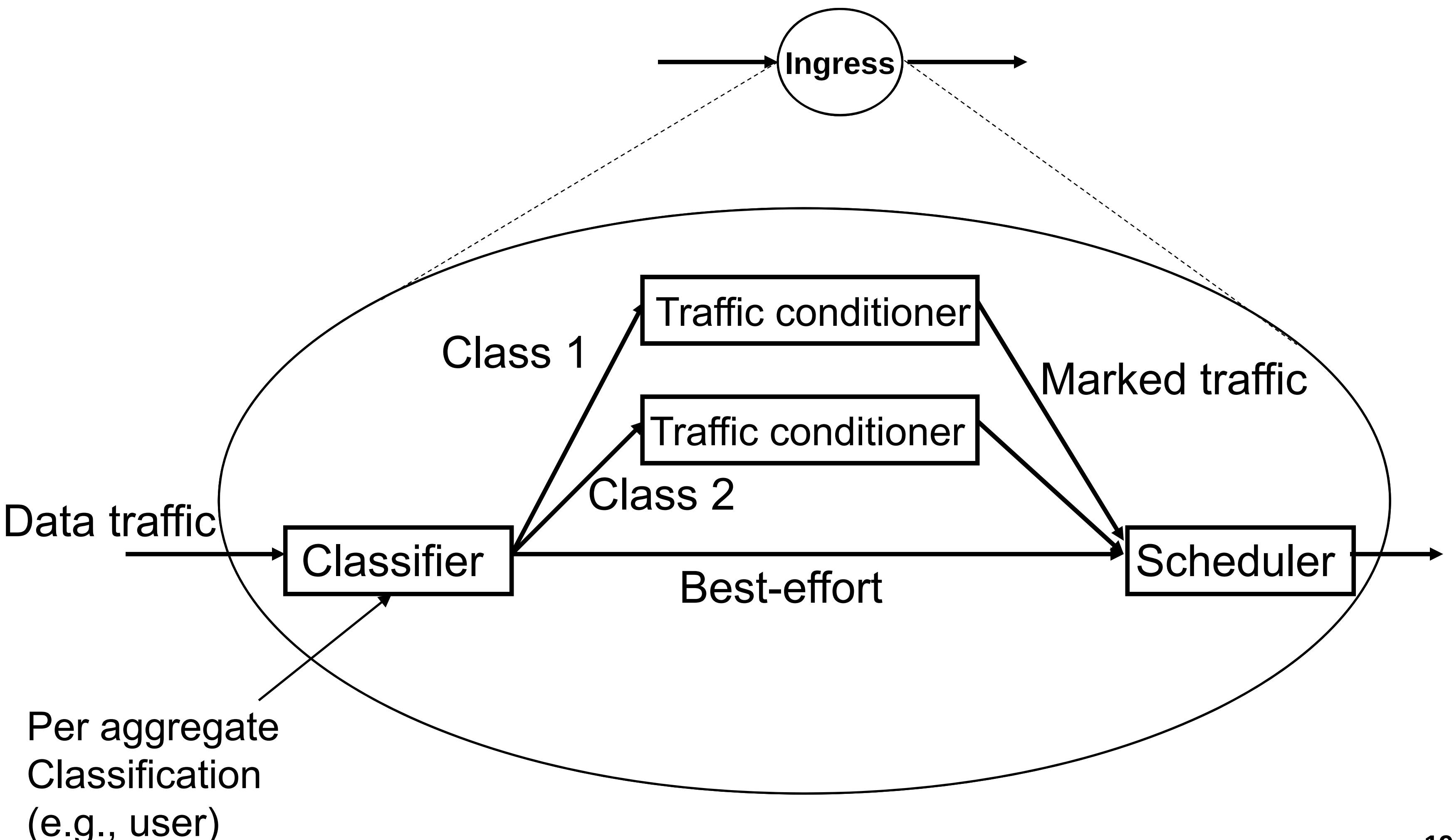
- Large spatial granularity service
- Theoretically, user profile is defined irrespective of destination
  - All other services we learnt are end-to-end, i.e., we know destination(s) a priori
- This makes service very useful, but hard to provision (why ?)



Premium Service [Jacobson '97]

- Provides the abstraction of a virtual pipe between an ingress and an egress router
- Network: guarantees that premium packets are not dropped and they experience low delay
- User: does not send more than the size of the pipe
  - If it sends more, excess traffic is delayed, and dropped when buffer overflows

Edge Router



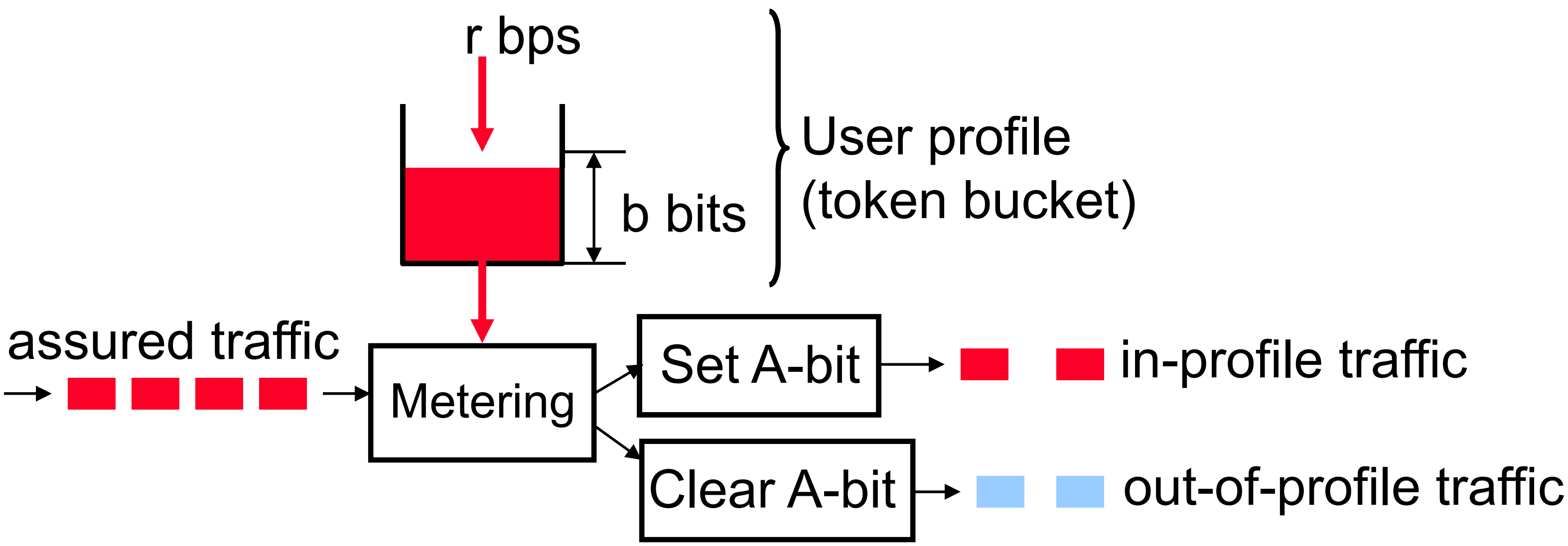
Ingress
Traffic conditioner
Class 1
Traffic conditioner
Marked traffic
Class 2
Data traffic
Classifier
Scheduler
Best-effort
Per aggregate
Classification
(e.g., user)

Assumptions

- Assume two bits
  - P-bit denotes premium traffic
  - A-bit denotes assured traffic
- Traffic conditioner (TC) implement
  - Metering
  - Marking
  - Shaping

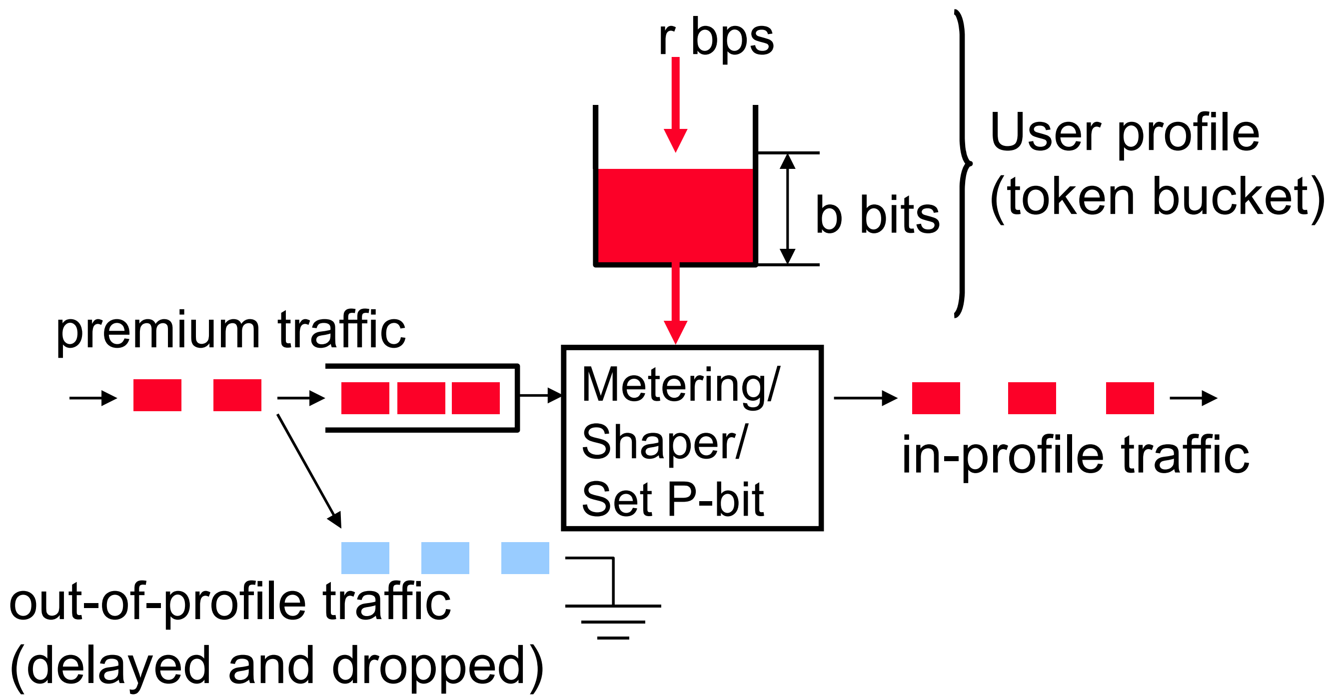
TC Performing Metering/Marking

- Used to implement Assured Service
- In-profile traffic is marked:
  - A-bit is set in every packet
- Out-of-profile (excess) traffic is unmarked
  - A-bit is cleared (if it was previously set) in every packet; this traffic treated as best-effort



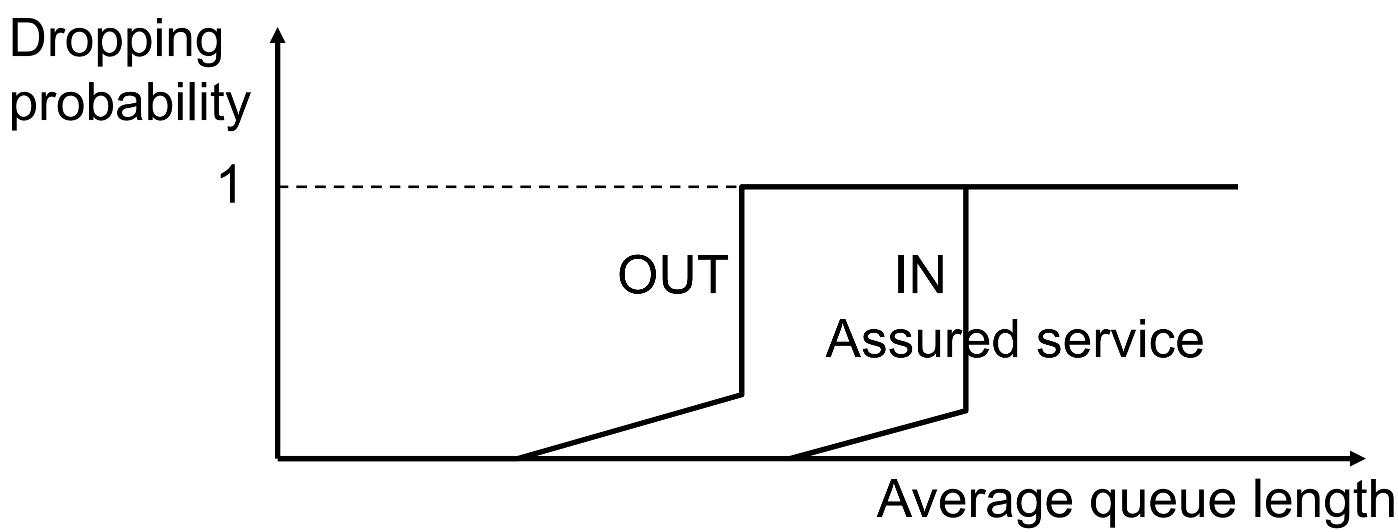
TC Performing Metering/Marking/Shaping

- Used to implement Premium Service
- In-profile traffic marked:
  - Set P-bit in each packet
- Out-of-profile traffic is delayed, and when buffer overflows it is dropped



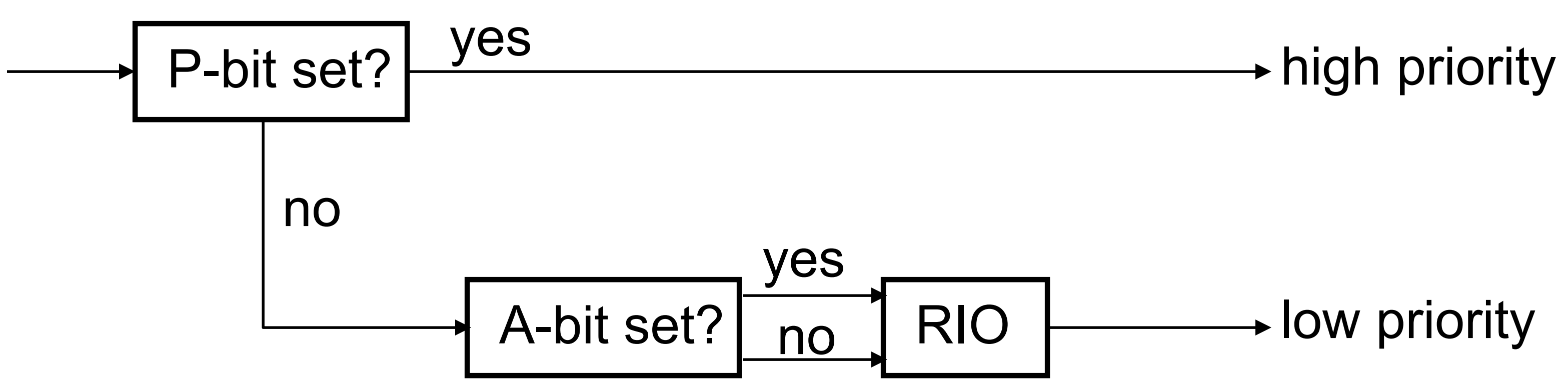
Scheduler

- Employed by both edge and core routers
- For premium service – use strict priority, or weighted fair queuing (WFQ)
- For assured service – use RIO (RED with In and Out)
  - Always drop OUT packets first
    - For OUT measure entire queue
    - For IN measure only in-profile queue



Scheduler Example

- Premium traffic sent at high priority
- Assured and best-effort traffic pass through RIO and then sent at low priority

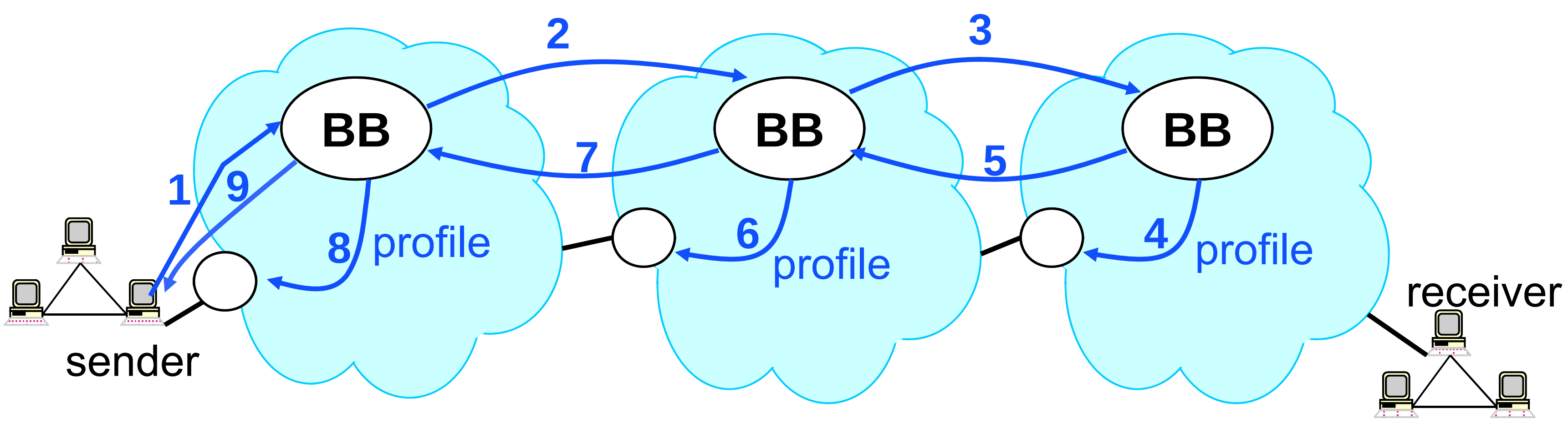


Control Path

- Each domain is assigned a Bandwidth Broker (BB)
  - Usually, used to perform ingress-egress bandwidth allocation
- BB is responsible for performing admission control in the entire domain
- BB not easy to implement
  - Require complete knowledge about domain
  - Single point of failure, may be performance bottleneck
  - Designing BB still a research problem

Example

- Achieve end-to-end bandwidth guarantee



Comparison to Best-Effort and Intserv

	Best-Effort	Diffserv	Intserv
Service	Connectivity No isolation No guarantees	Per aggregate isolation Per aggregate guarantee	Per flow isolation Per flow guarantee
Service scope	End-to-end	Domain	End-to-end
Complexity	No setup	Long term setup	Per flow steup
Scalability	Highly scalable (nodes maintain only routing state)	Scalable (edge routers maintains per aggregate state; core routers per class state)	Not scalable (each router maintains per flow state)

Summary

- Diffserv more scalable than Intserv
  - Edge routers maintain per aggregate state
  - Core routers maintain state only for a few traffic classes
- But, provides weaker services than Intserv, e.g.,
  - Per aggregate bandwidth guarantees (premium service) vs. per flow bandwidth and delay guarantees
- BB is not an entirely solved problem
  - Single point of failure
  - Handle only long term reservations (hours, days)

Random Early Detection Gateways for Congestion Avoidance

Outline

- Motivation
- RED overview
- The RED algorithm
- Tuning parameters
- Simulation results
- Discussions

Motivation

- Congestion avoidance
  - Current TCP protocol
  - Reducing window size before packets drop is desired
- Gateway is the right place to provide congestion avoidance
  - Detect the congestions effectively
  - Decide the duration and magnitude of transient congestion to be allowed

Drawbacks of Drop Tail

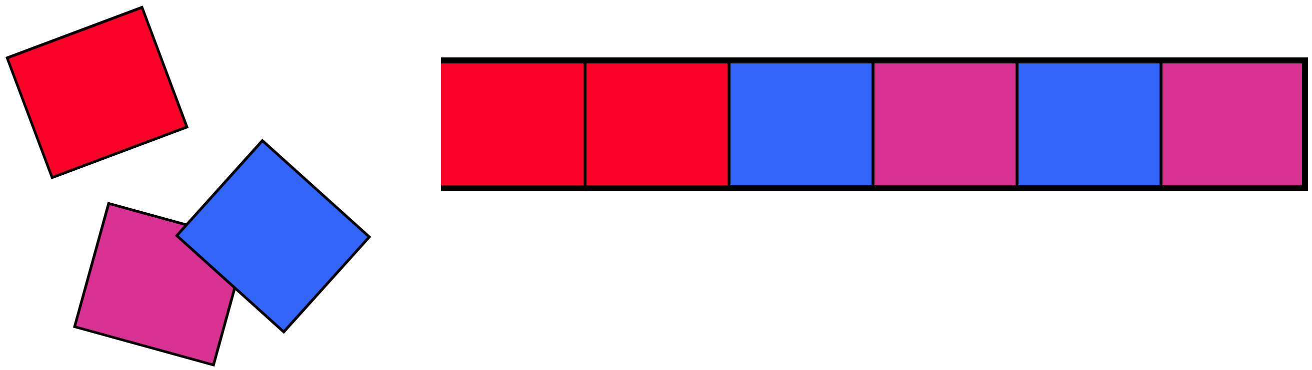
$$\text{Global Power} \propto \frac{\text{Throughput}}{\text{Delay}}$$

- Low global power:

  To accommodate transient congestion periods, queue must be large, thus causes delay

- Global synchronization
- Bias against bursty traffic

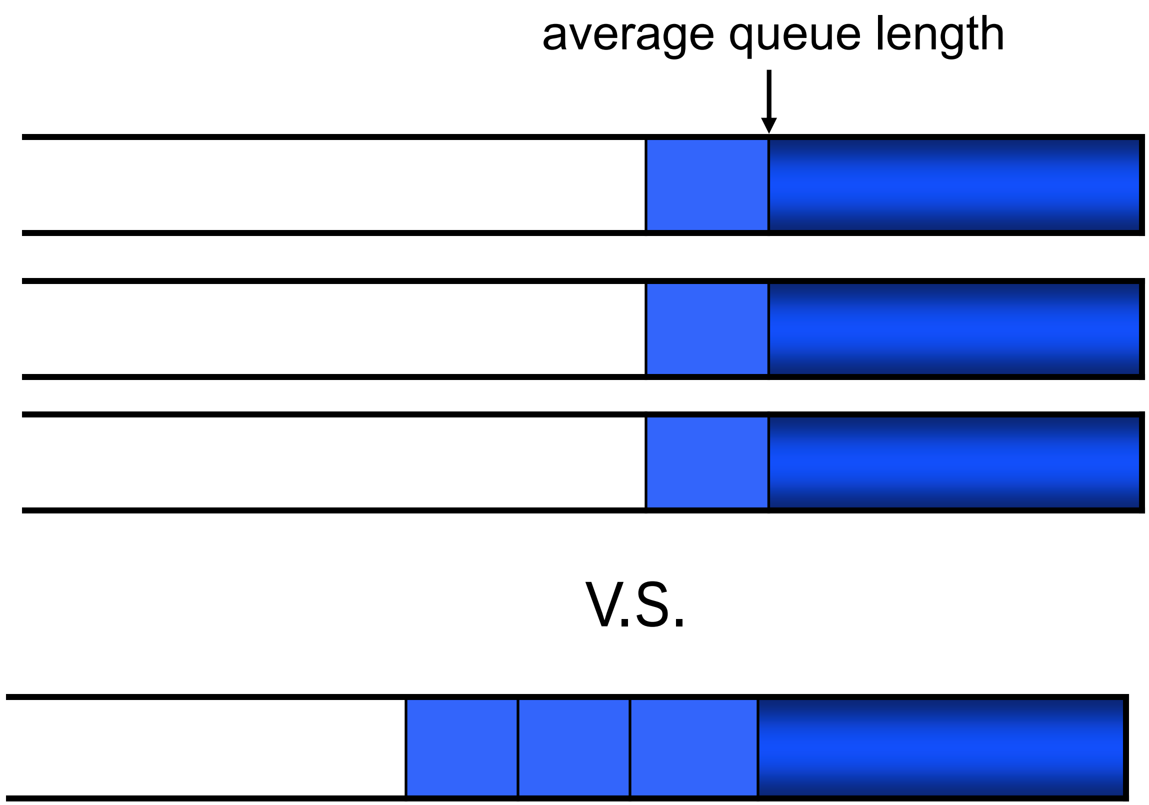
Global Synchronization



- When queue overflows, several connections decrease congestion windows simultaneously
- Key: one RTT is required for connection notified of congestion to decrease CW
- in the meantime all the others send and get packets dropped

Bias Against Bursty Traffic

- Bursty traffic more likely to be dropped



Congestion Avoidance

- Maintains low delay and high throughput
  - Average queue size is kept low
  - Actual queue size grows enough to handle:
    - Bursty traffic
    - Transient congestion
- Since the gateway knows most about the queue and sources contributing to congestion, avoidance is best done here

Goals and ways

- Predict congestions by monitoring the queue size
- Provide feedback before packets are dropped by marking the packets
- Reduce the delay by maintaining a reasonable queue size at each gateway
- Avoid global synchronization by randomly choosing packets to mark
- Avoid a bias against bursty traffic by uniformly marking the packets to be dropped
- Allow transient congestion by using low pass filter for computing the average queue size

RED solves the following problems…

- How to detect incipient congestion?
- How to decide which connections to notify of congestion?
- How to provide notification?

The RED Algorithm

- Computing the average queue size
  - Determines degree of burstiness allowed
  - Should still be responsive enough to detect incipient congestion
- Computing probability of packet drop
  - Determines average queue length
  - Should drop at evenly-space intervals (to avoid global-synch and bursty-traffic-bias)

Average Queue Size

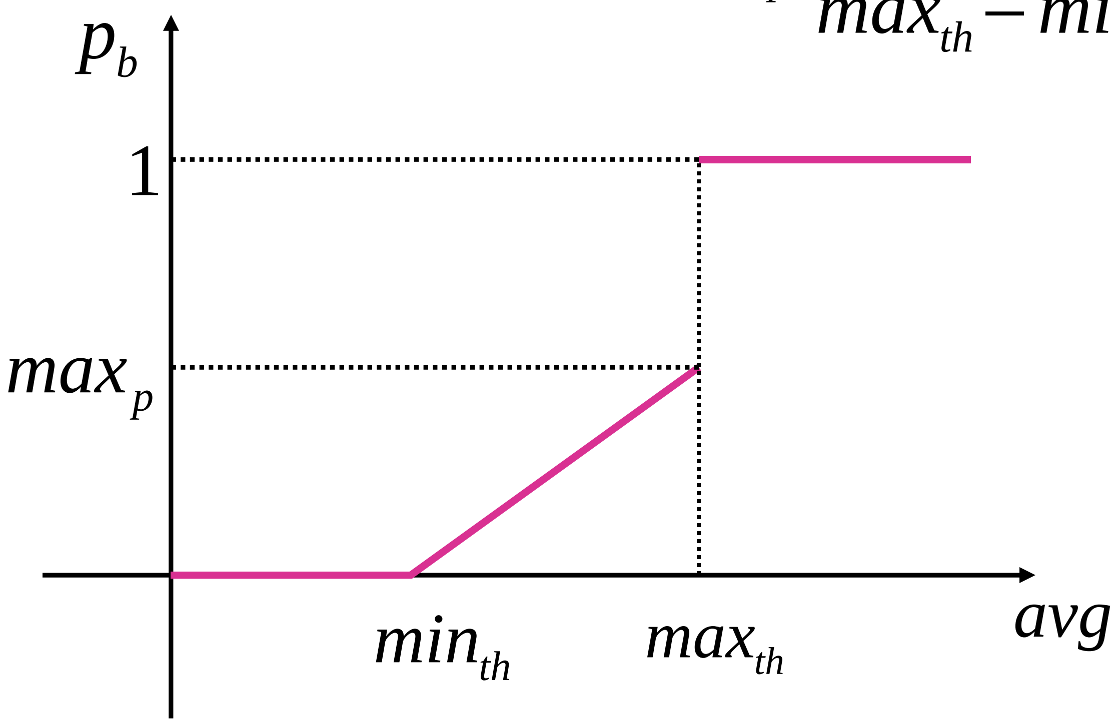
- Use low-pass filter (exponential weighted moving average)

$$avg \leftarrow (1 - w_q)avg + w_q q$$

- w_q should be small enough to filter out transient congestion, and large enough for the average to be responsive

Drop Probability

$$p_b = max_p \frac{avg - min_{th}}{max_{th} - min_{th}}$$



RED algorithm

- If $avg < min_{th}$
  - No marking
- If $Min_{th} < avg < max_{th}$
  - Randomly Mark the packets
- If $avg > Max_{th}$
  - Mark all the packets

Calculate the marking probability

- Geometric random variables:
  - Use P_b to mark the packets
  - so each packet is marked with probability

$$p_b \leftarrow max_p(avg - min_{th})/(max_{th} - min_{th})$$

$$Prob[X = n] = (1 - p_b)^{n-1} p_b$$

- Uniform random variables:
  - Use P_a instead of P_b to mark the packets

$$p_a \leftarrow p_b/(1 - count \cdot p_b)$$

where count is # of packets since last marking

Uniform Distribution of Drops

- If you just use p_b:

$$P[X = n] = (1 - p_b)^{n-1} p_b$$

- Using $p_b/(1 - count \text{ x } p_b)$:

$$P[X = n] = \frac{p_b}{1-(n-1)p_b} \prod_{i=0}^{n-2} \left(1 - \frac{p_b}{1 - i \cdot p_b} \right)$$

$$= \frac{p_b}{1-(n-1)p_b} \prod_{i=0}^{n-2} \left(\frac{1-(i+1)p_b}{1 - i \cdot p_b} \right)$$

$$= \frac{p_b}{1-(n-1)p_b} \frac{1-(n-1)p_b}{1} = p_b$$

Drop Probability Parameters

- To accommodate bursty traffic min_{th} must be large enough
- max_{th} depends in part on maximum average delay allowed
- (max_{th} - min_{th}) must be larger than typical increase in average queue length during an RTT

Decide $\min_{th}$ and $\max_{th}$

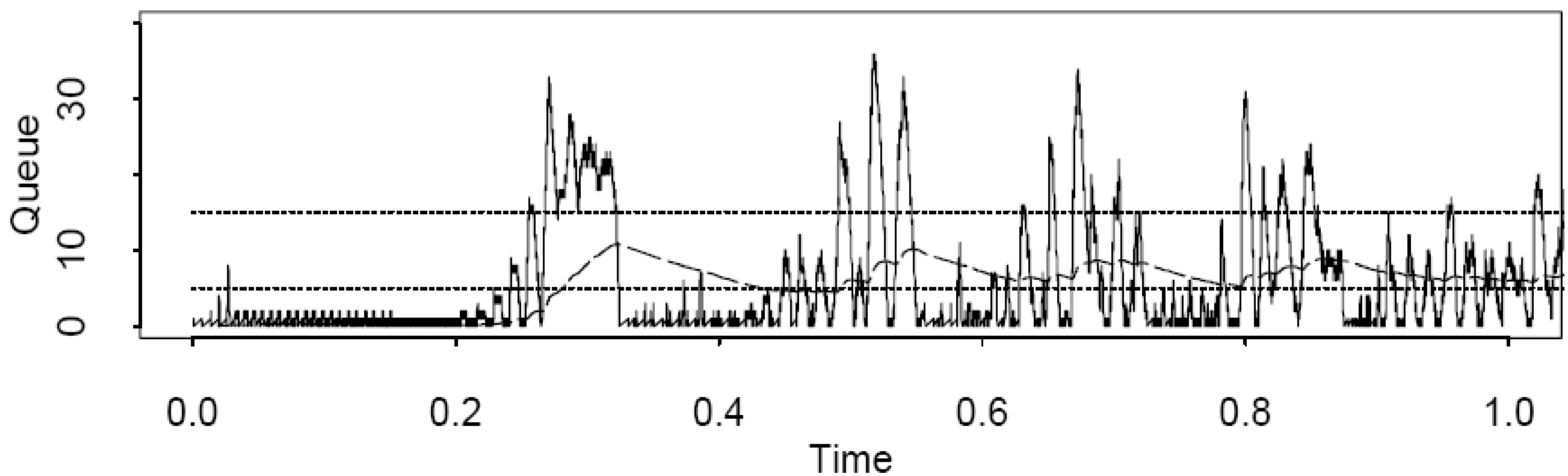
- Set min_{th} sufficient high to maximize network power

$$\text{Global Power} \propto \frac{\text{Throughput}}{\text{Delay}}$$

- Make max_{th} - min_{th} sufficient large to avoid global synchronization. It should be larger than the increase of queue size in one RTT
- Min_{th} depends on the magnitude of the transient congestions being allowed by the gateway
- Max_{th} depends on the maximum average delay of the network

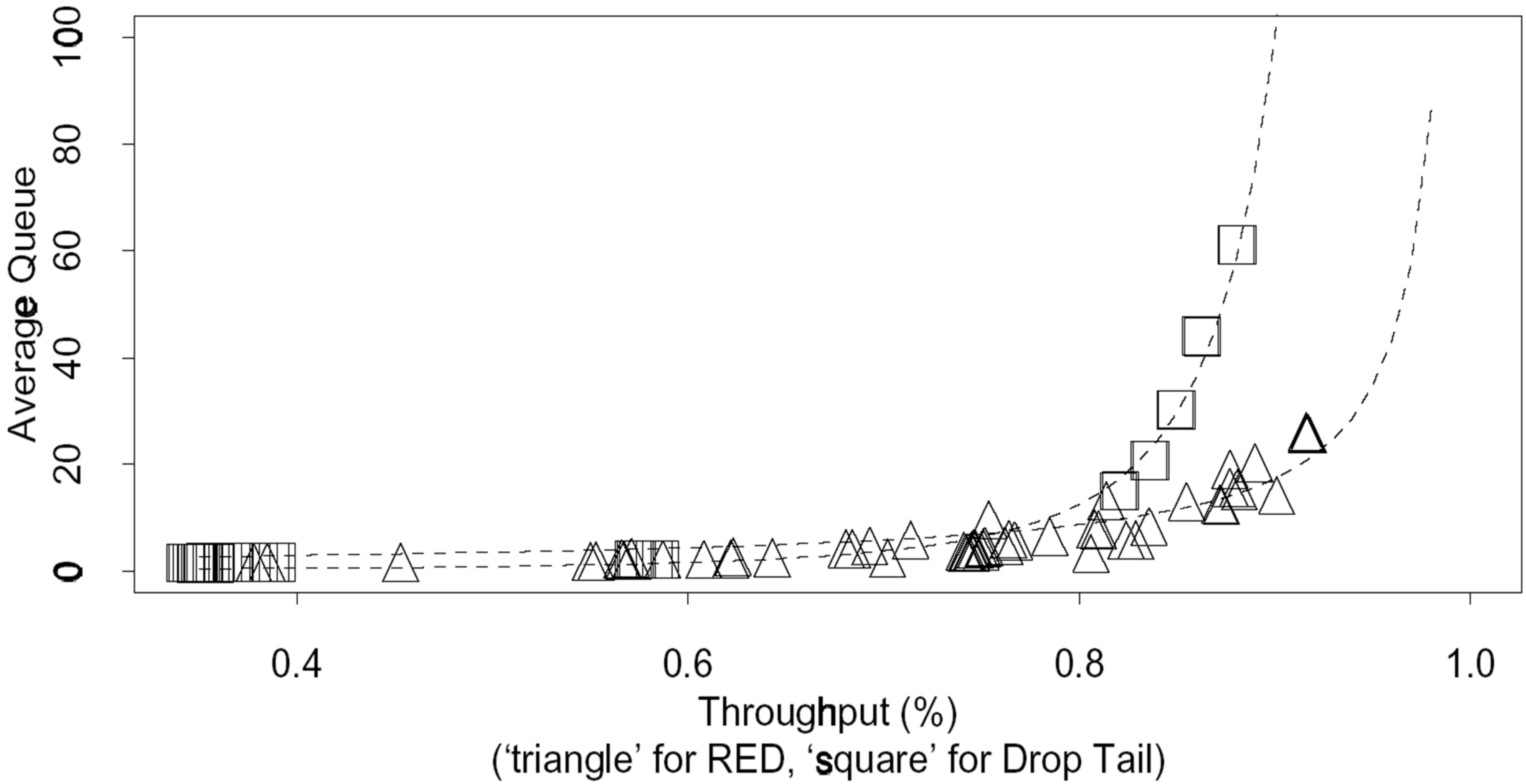
Average Queue Size

- Low-pass filter:
- Example: $avg \leftarrow (1 - w_q) avg + w_q \, q$



Queue size (solid line) and average queue size (dashed line).

Simulation Results

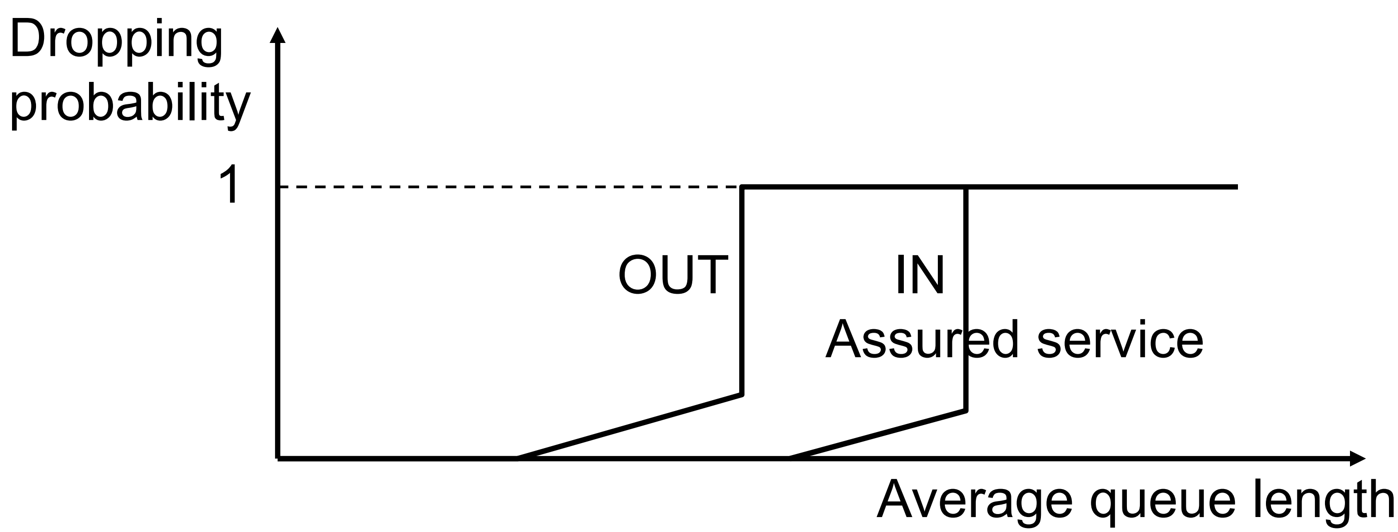


The Advantages of RED

- No bias against bursty traffic
- No global synchronization
- Packet marking probability proportional to connection's share of bandwidth
- Gradual integration possible
  - Free choice of transport protocol
    - If uncooperative, drop packets
  - No assumption about other gateways
- Scalable: no per-connection state

RIO for DiffServ

- For assured service – use RIO (RED with In and Out)
  - Always drop OUT packets first
    - For OUT measure entire queue (in-profile+out-of-profile)
    - For IN measure only in-profile queue



Evaluation of RED

- Congestion avoidance
- Appropriate time scales
- No global synchronization
- Simplicity
- Maximizing global network power
- Appropriate for a wide range of environment
- ? Fairness