CS 6323
Complex Networks and Systems:
Modeling and Inference

Lecture 11:
Performance Analysis:
Congestion Control

Prof. Gregory Provan
Department of Computer Science
University College Cork

Applications: Queues and Traffic shaping

- Big Ideas:
  - Traffic shaping:
    - Modify traffic at entrance points in the network
    - Can reason about states of the interior and link properties, loads. (E.g. Leaky, Token bucket)
    - Modify traffic in the routers
      - Enforce policies on “flows”
  - Queuing theory:
    - Analytic analysis of properties of queues as functions of the inputs and service types

DATA TRAFFIC

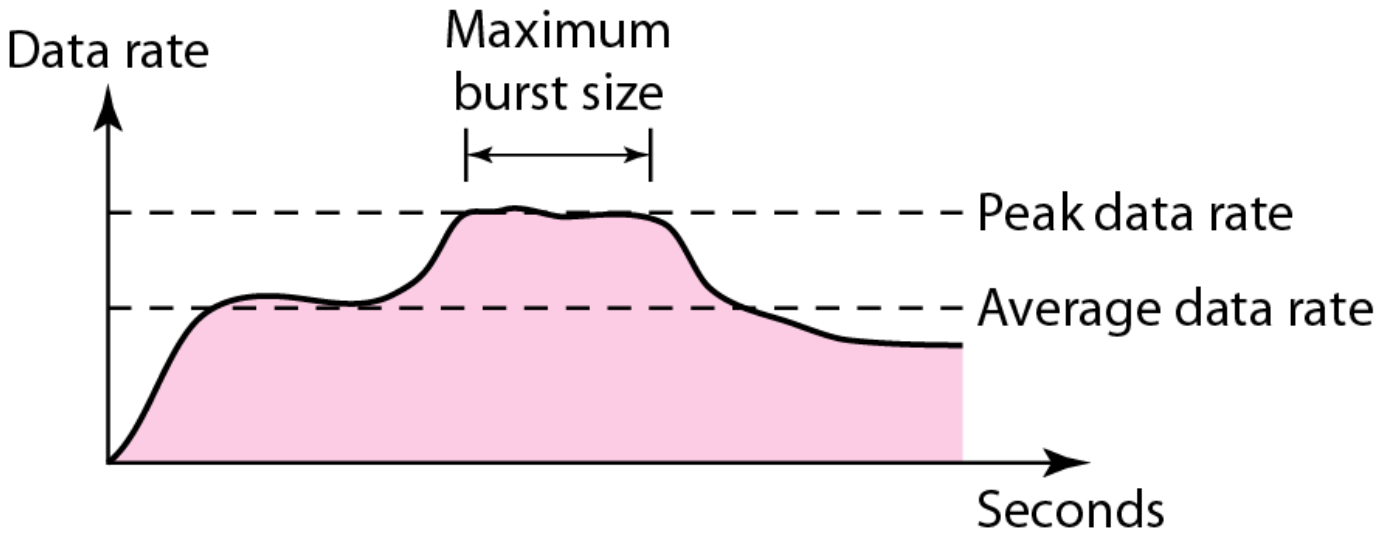
The main focus of congestion control and quality of service is data traffic. In congestion control we try to avoid traffic congestion. In quality of service, we try to create an appropriate environment for the traffic. So, before talking about congestion control and quality of service, we discuss the data traffic itself.

Topics discussed in this section:

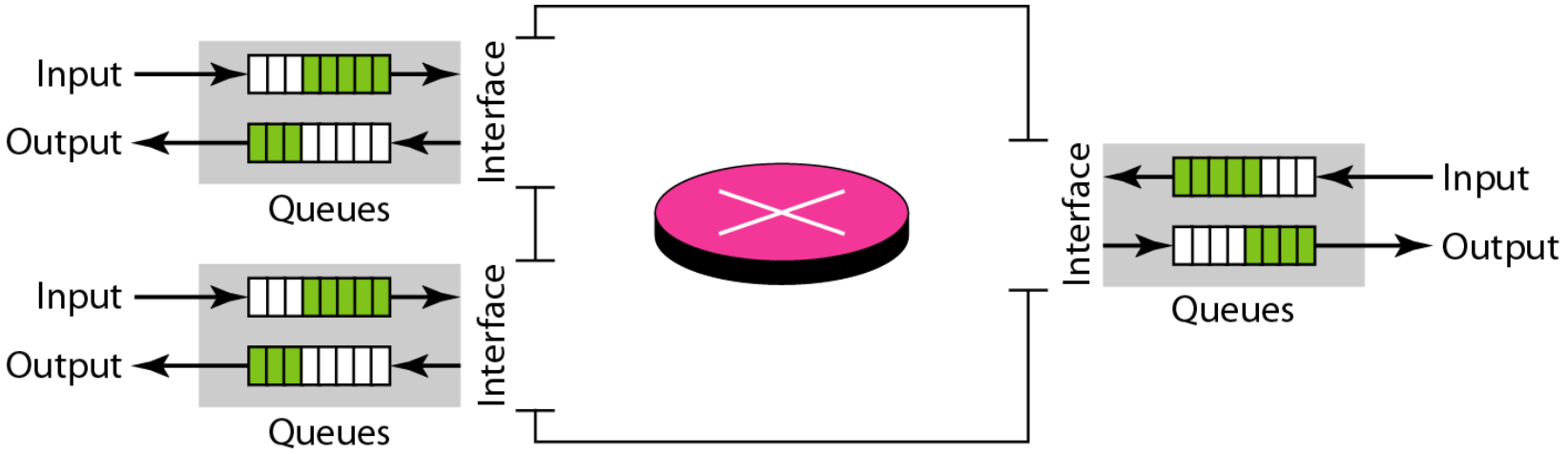
Traffic Descriptor

Traffic Profiles

Traffic descriptors

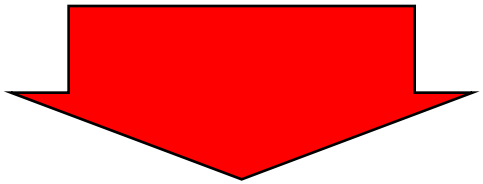


Queues in a router



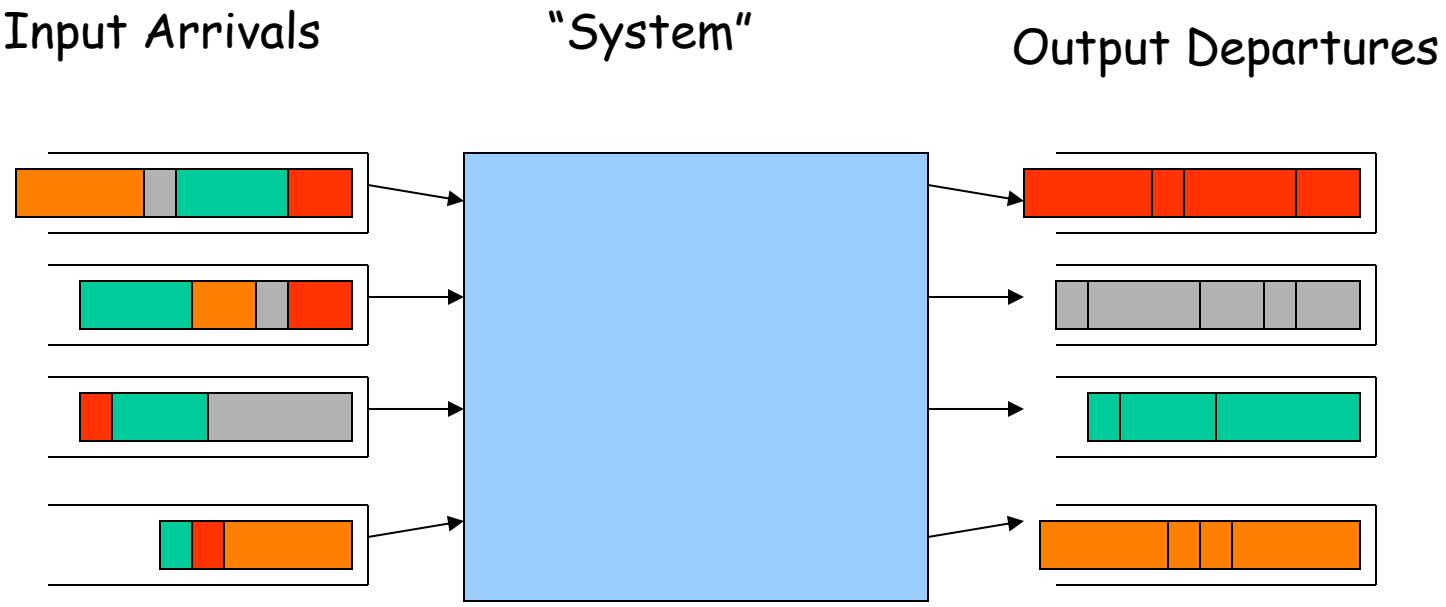
Congestion Control

Too many packets in some part of the system



Congestion

Simplified Network Model



Goal:

Move packets across the system from the inputs to output

System could be a single switch, or entire network

Problems with Congestion

- Performance
  - Low Throughput
  - Long Latency
  - High Variability (jitter)
  - Lost data
- Policy enforcement:
  - User X pays more than user Y, X should get more bandwidth than Y.
  - Fairness
    - One user/flow getting much more throughput or better latency than the other flows

Congestion Control

- Causes of congestion
- Types of congestion control schemes
- Solving congestion

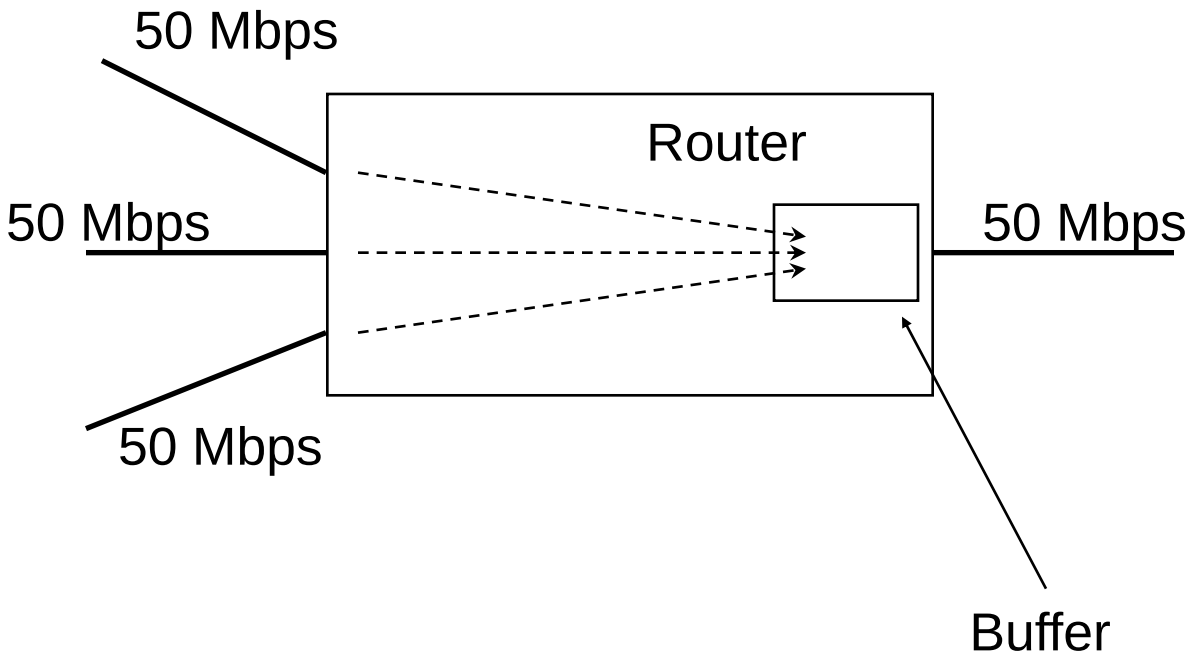
Causes of Congestion

- Many ultimate causes:

  New applications, new users, bugs, faults, viruses, spam, stochastic (time based) variations, unknown randomness.

- Congestion can be long-lived or transient
  - Timescale of the congestion is important
    - Microseconds vs seconds vs hours vs days
    - Different solutions to all the above!

Exhaustion of Buffer Space *(cont'd)*



Types of Congestion Control Strategies

- Terminate existing resources
  - Drop packets
  - Drop circuits
- Limit entry into the system
  - Packet level (layer 3)
    - Leaky Bucket, token bucket, WFQ
  - Flow/conversation level (layer 4)
    - Resource reservation
    - TCP backoff/reduce window
  - Application level (layer 7)
    - Limit types/kinds of applications

QUALITY OF SERVICE

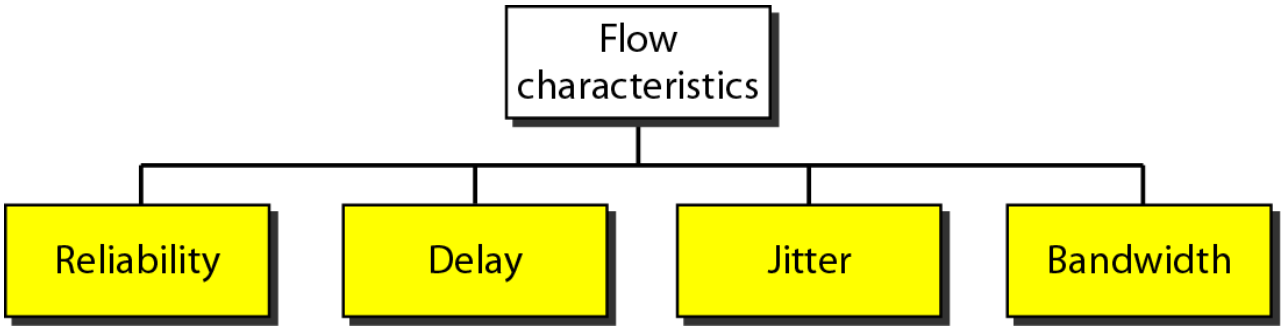
Quality of service (QoS) is an internetworking issue that has been discussed more than defined. We can informally define quality of service as something a flow seeks to attain.

Topics discussed in this section:

Flow Characteristics

Flow Classes

Flow characteristics



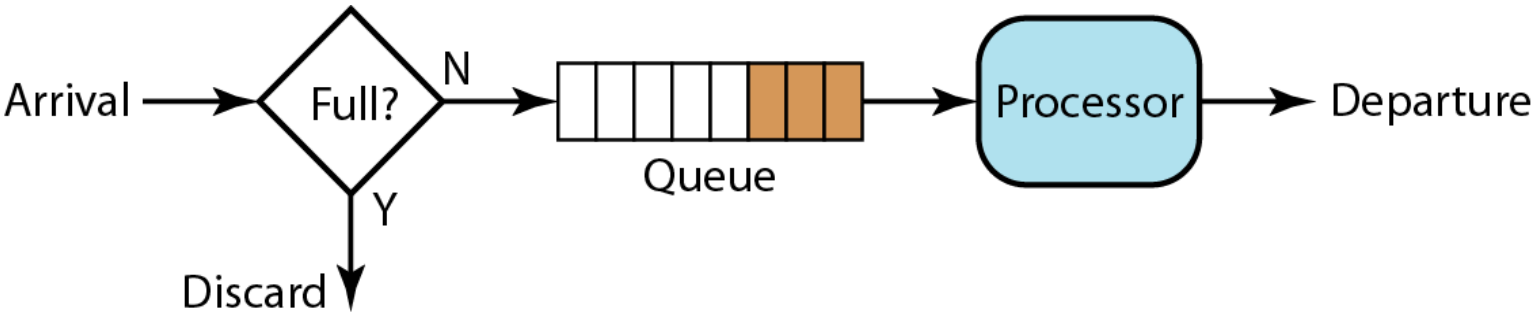
TECHNIQUES TO IMPROVE QoS

In this section, we discuss some techniques that can be used to improve the quality of service. We briefly discuss some common methods: scheduling, traffic shaping, and admission control.

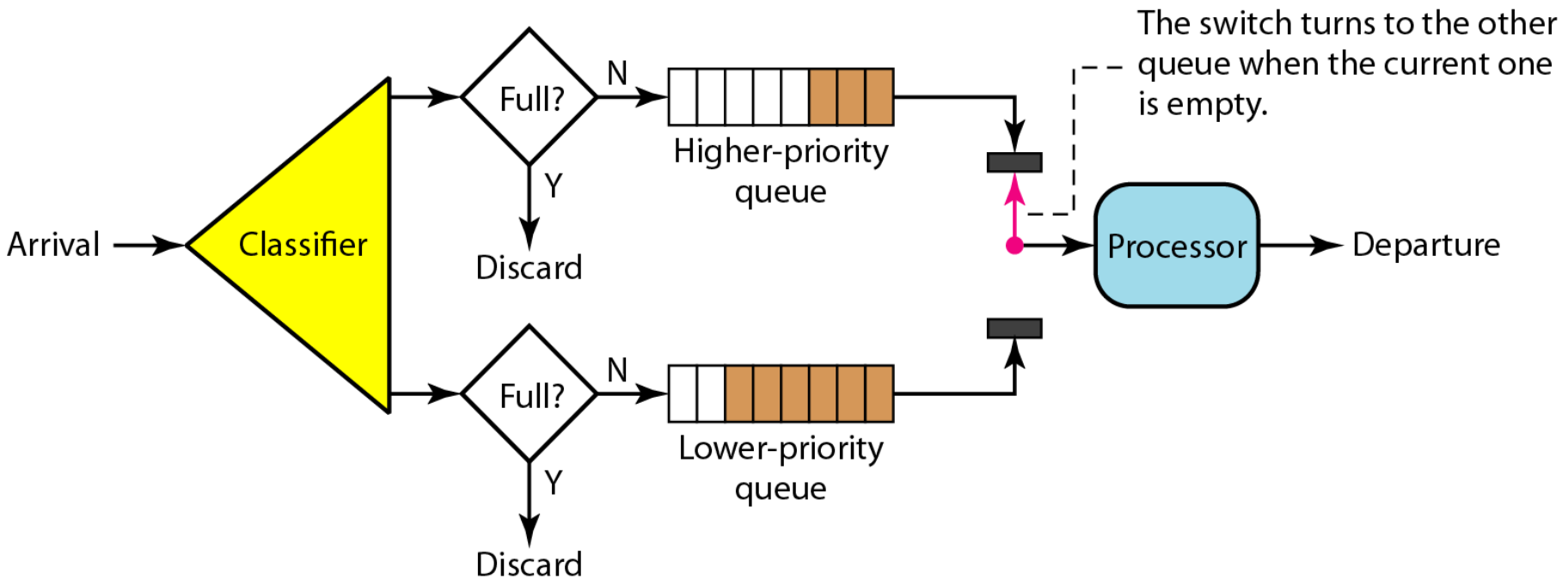
Topics discussed in this section:

Scheduling
Traffic Shaping
Resource Reservation
Admission Control

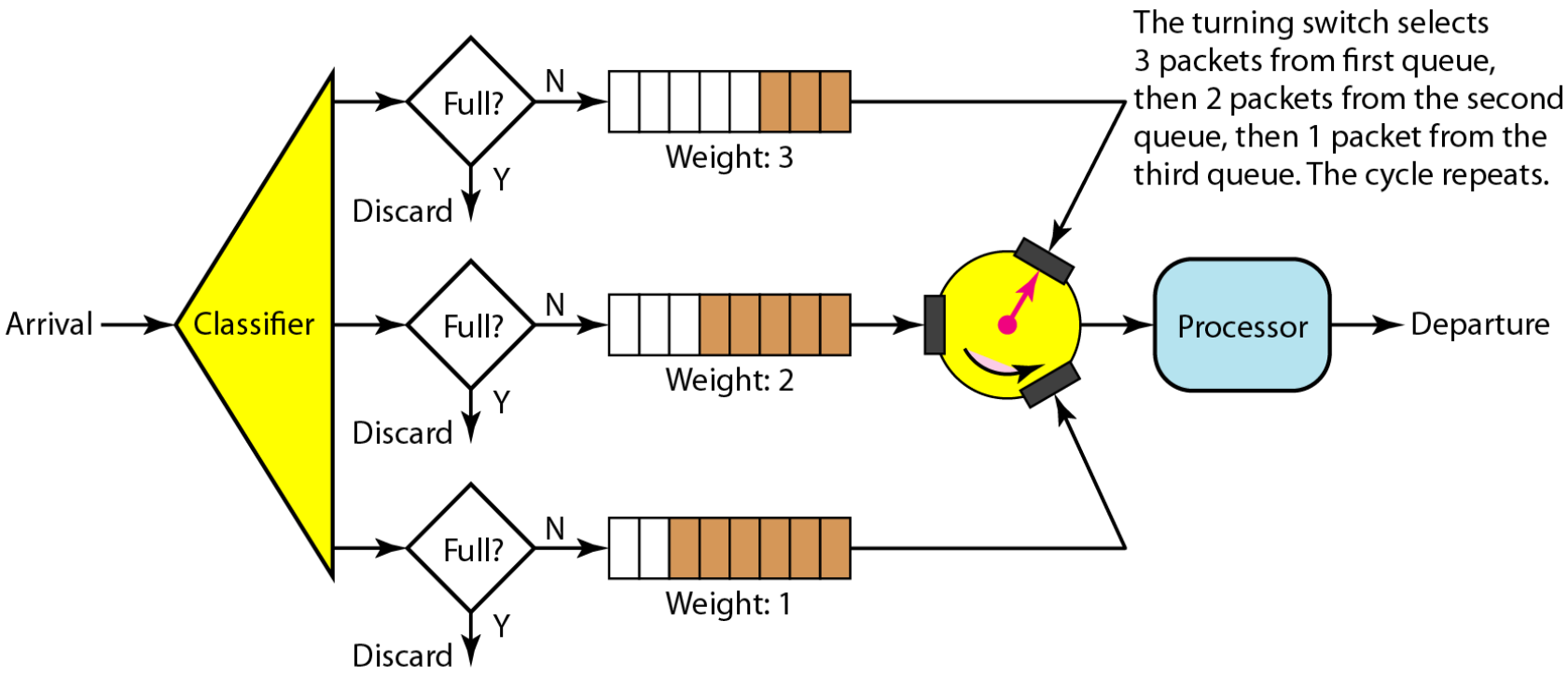
FIFO queue



Priority queuing



Weighted fair queuing



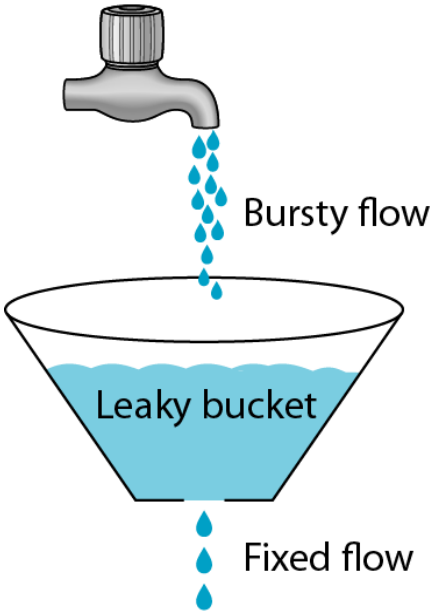
Traffic Regulators

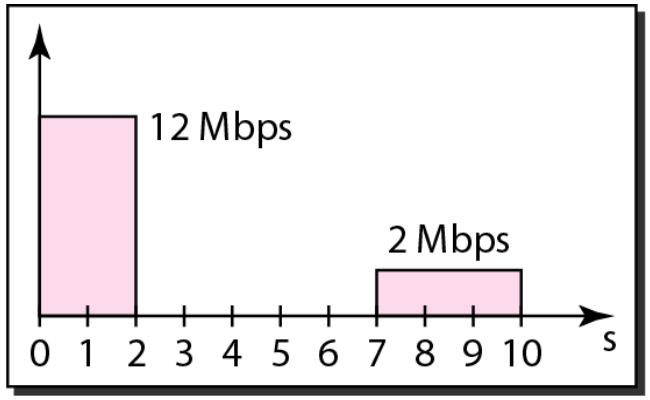
- Leaky bucket controllers
- Token bucket controllers

Leaky Bucket

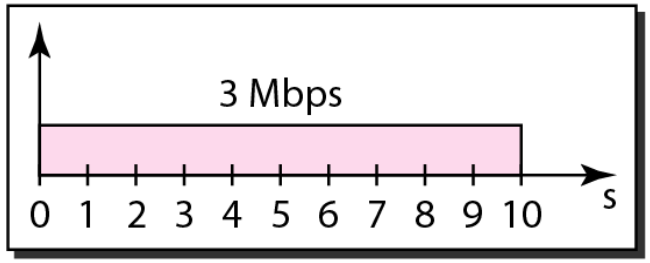
- Across a single link, only allow packets across at a constant rate
- Packets may be generated in a bursty manner, but after they pass through the leaky bucket, they enter the network evenly spaced
- If all inputs enforce a leaky bucket, its easy to reason about the total resource demand on the rest of the system

Leaky bucket



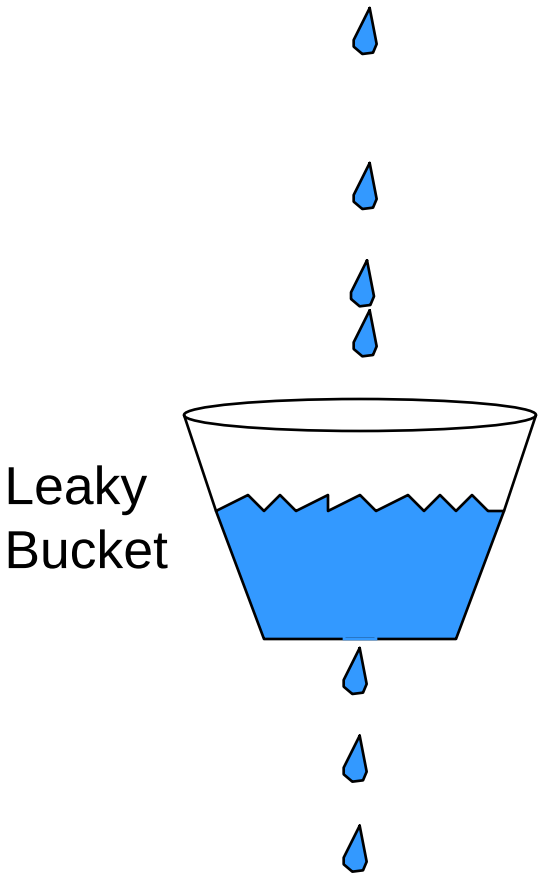


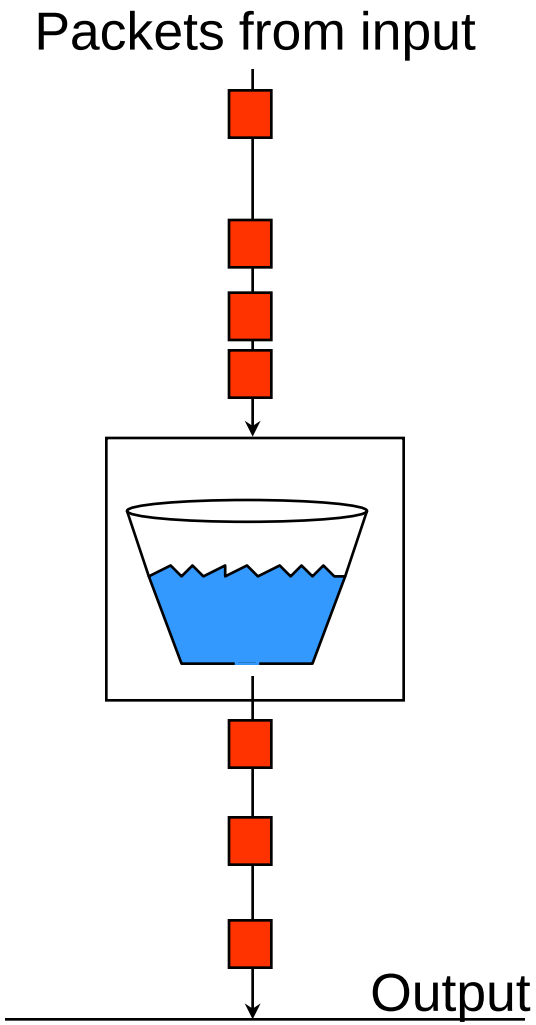
Bursty data



Fixed-rate data

Leaky Bucket: Analogy





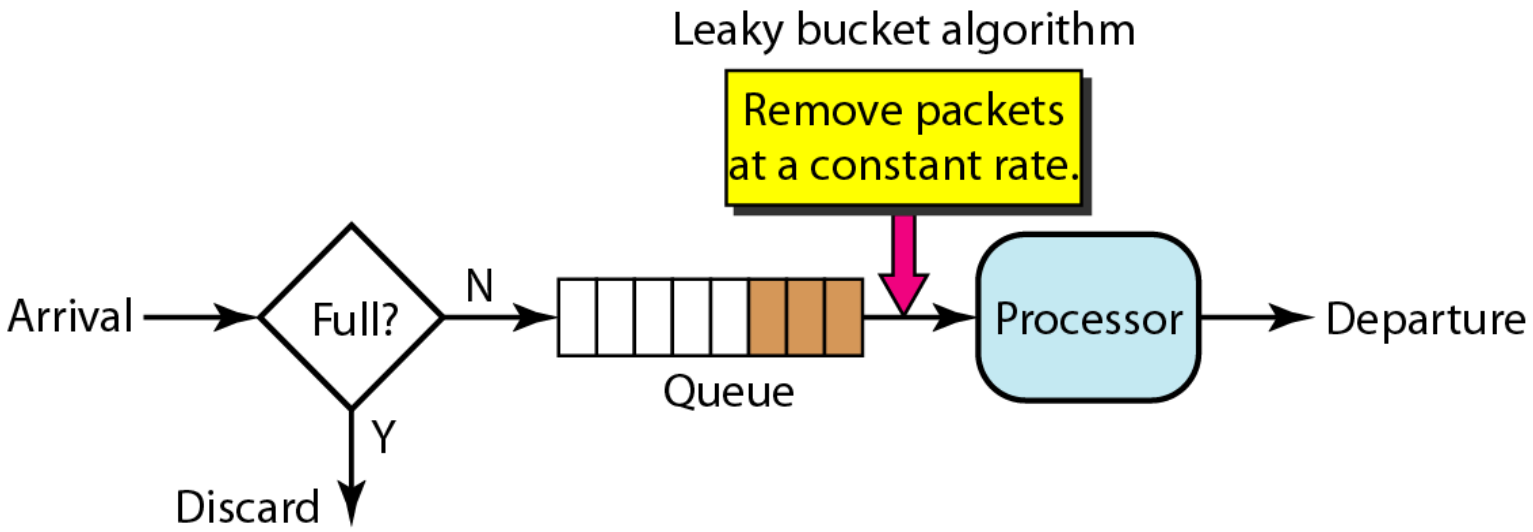
Leaky Bucket *(cont'd)*

- The leaky bucket is a "traffic shaper": It changes the characteristics of a packet stream
- Traffic shaping makes the network more manageable and predictable
- Usually the network tells the leaky bucket the rate at which it may send packets when a connection is established

Leaky Bucket: Doesn't allow bursty transmissions

- In some cases, we may want to allow short bursts of packets to enter the network without smoothing them out
- For this purpose we use a token bucket, which is a modified leaky bucket

Leaky bucket implementation



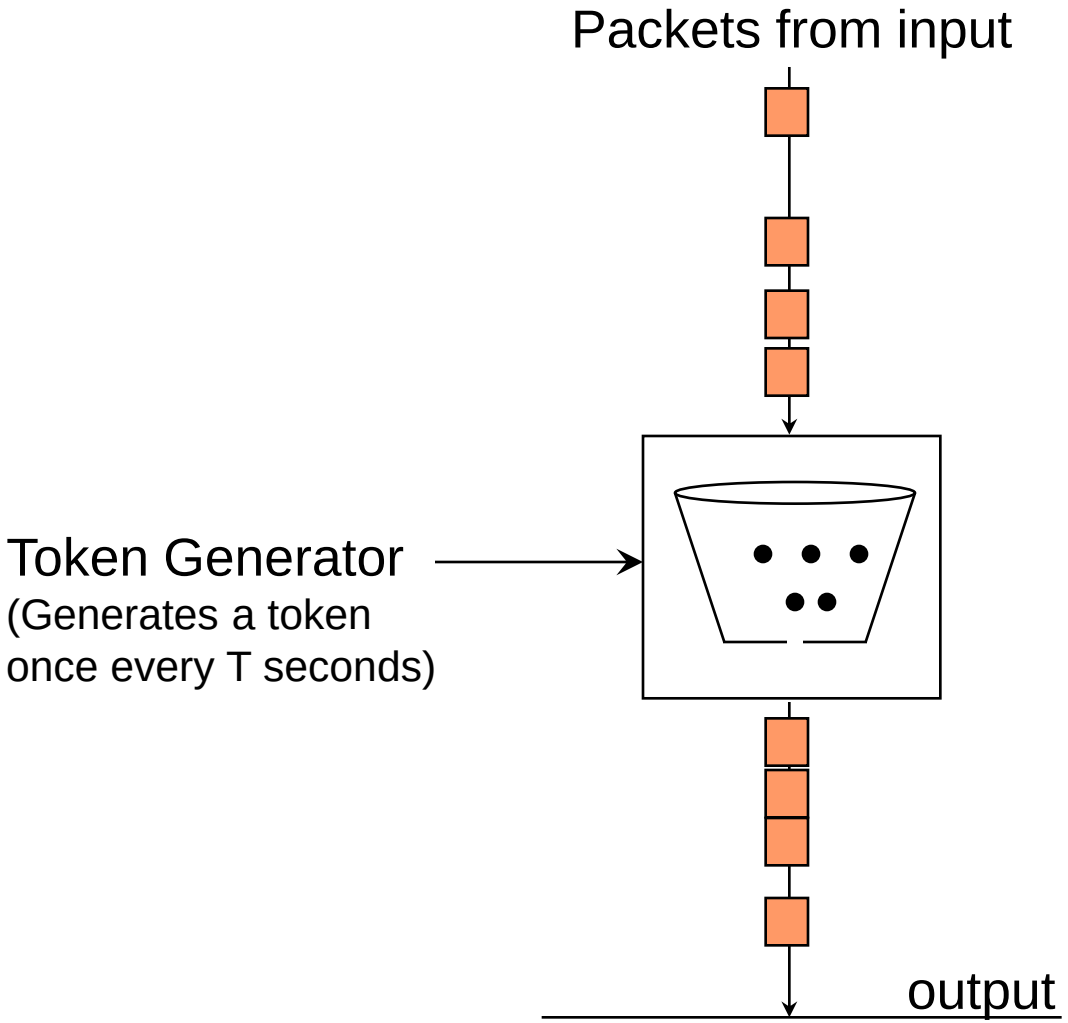
Note

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Token Bucket

- The bucket holds logical tokens instead of packets
- Tokens are generated and placed into the token bucket at a constant rate
- When a packet arrives at the token bucket, it is transmitted if there is a token available. Otherwise it is buffered until a token becomes available.
- The token bucket holds a fixed number of tokens, so when it becomes full, subsequently generated tokens are discarded
- Can still reason about total possible demand

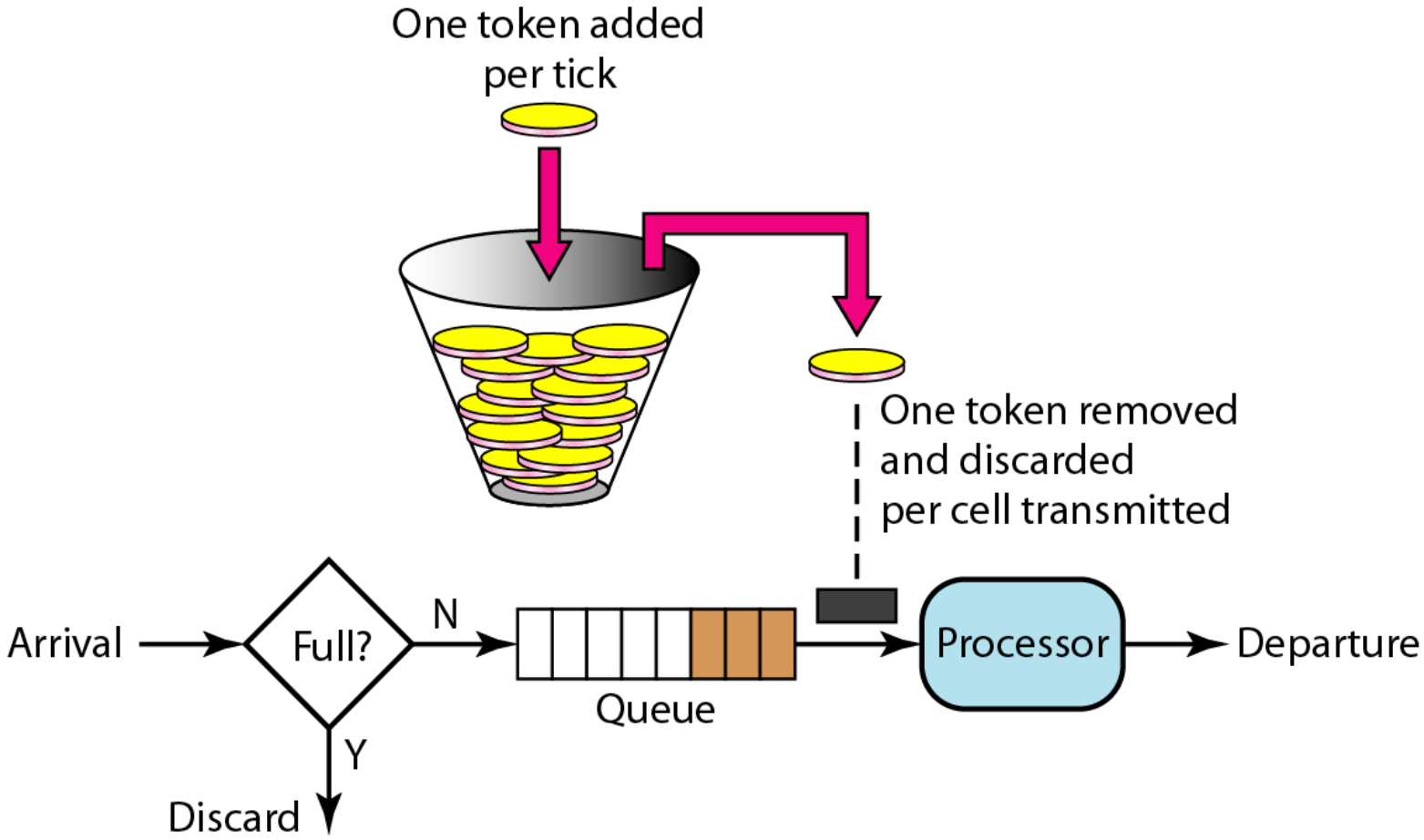
Token Bucket



Note

The token bucket allows bursty traffic at a regulated maximum rate.

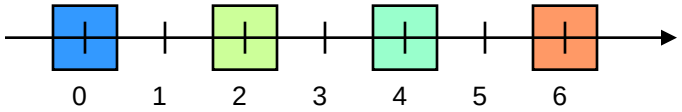
Token bucket



Token Bucket vs. Leaky Bucket

Case 1: Short burst arrivals

Arrival time at bucket

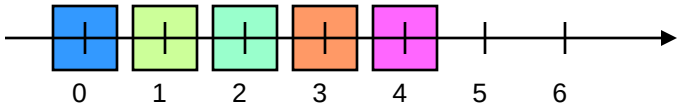


Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 4 packets

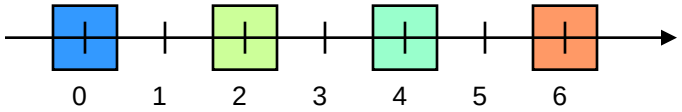
Departure time from a token bucket
Token bucket rate = 1 tokens / 2 time units
Token bucket size = 2 tokens

Token Bucket vs. Leaky Bucket

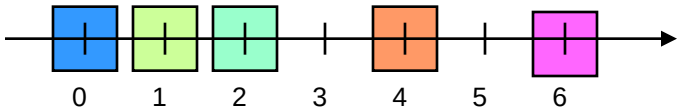
Case 2: Large burst arrivals



Arrival time at bucket



Departure time from a leaky bucket
Leaky bucket rate = 1 packet / 2 time units
Leaky bucket size = 2 packets

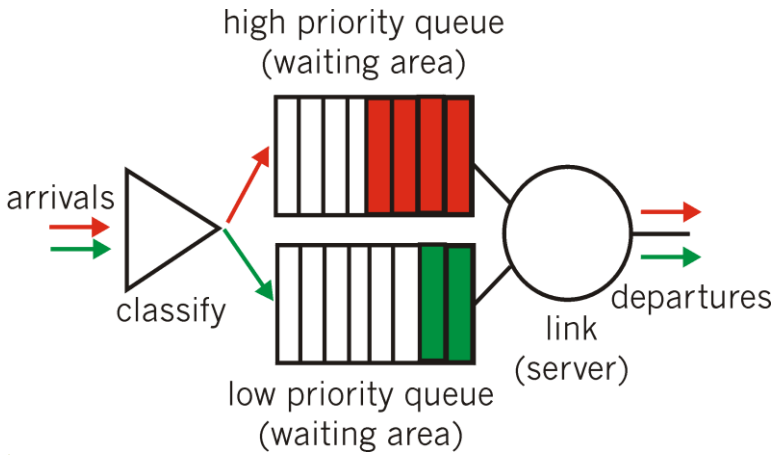


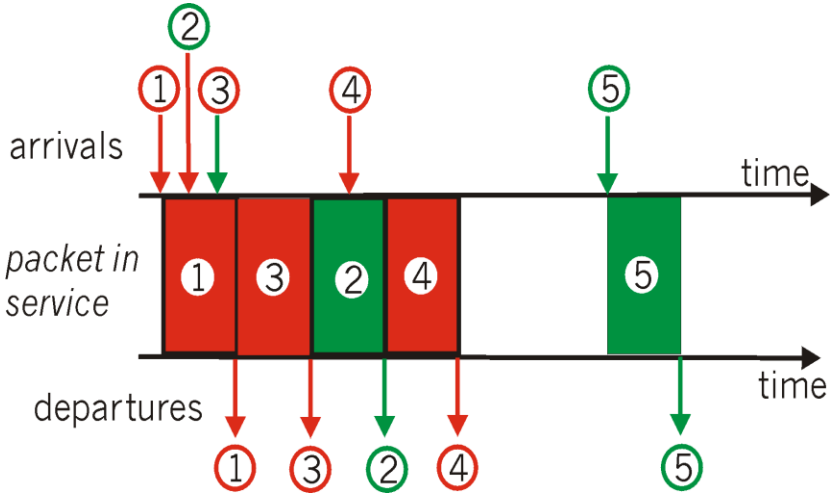
Departure time from a token bucket
Token bucket rate = 1 token / 2 time units
Token bucket size = 2 tokens

Priority Scheduling

transmit highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..



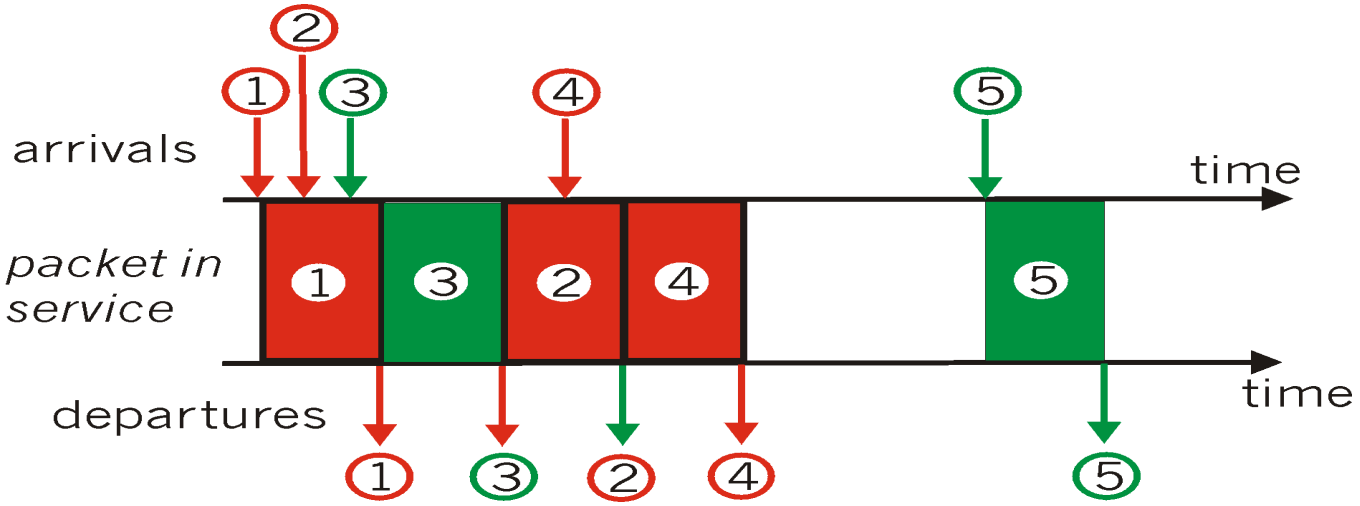


Priority Scheduling

- The scheduler serves a packet from priority level k only if there are no packets awaiting service in levels k+1, k+2, ..., n
- at least 3 levels of priority in an integrated services network?
- Starvation? Appropriate admission control and policing to restrict service rates from all but the lowest priority level
- Simple implementation

Round Robin Scheduling

- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- provides protection against misbehaving sources (also guarantees a minimum bandwidth to every connection)

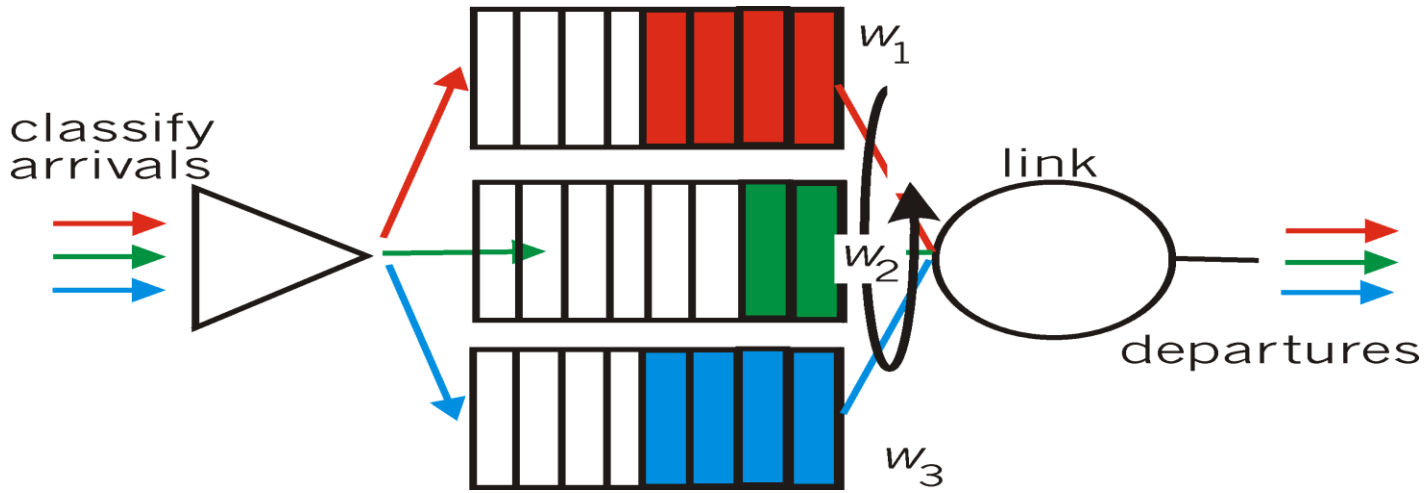


Max-Min Fair Share

- Fair Resource allocation to best-effort connections?
- Fair share allocates a user with a “small” demand what it wants, and evenly distributes unused resources to the “big” users.
- Maximize the minimum share of a source whose demand is not fully satisfied.
  - Resources are allocated in order of increasing demand
  - no source gets a resource share larger than its demand
  - sources with unsatisfied demand s get an equal share of resource
- A Generalized Processor Sharing (GPS) server will implement max-min fair share

Weighted Fair Queueing

- generalized Round Robin (offers differential service to each connection/class)
- each class gets weighted amount of service in each cycle



Policing Mechanisms

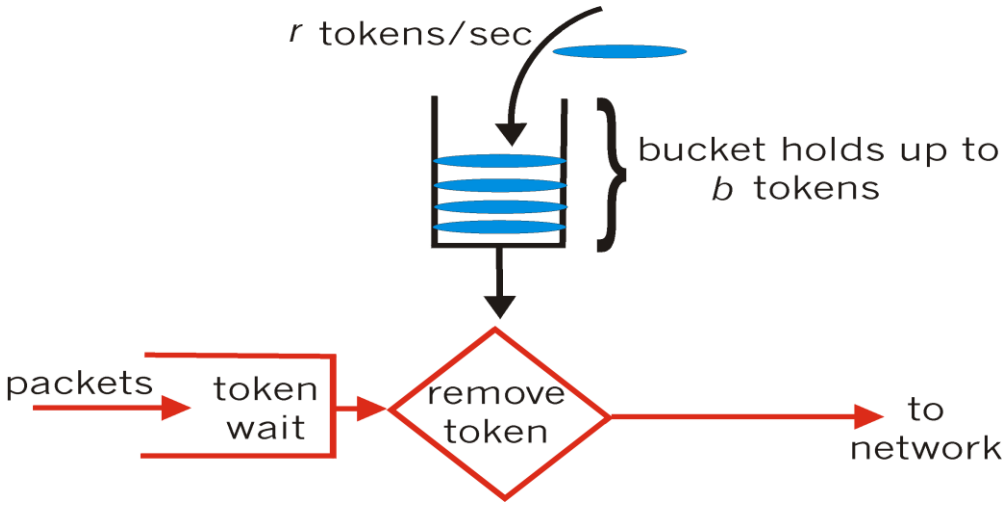
Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

- *(Long term) Average Rate:* how many pkts can be sent per unit time (in the long run)
  - crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!
- *Peak Rate:* e.g., 6000 pkts per min. (ppm) avg.; 1500 ppm peak rate
- *(Max.) Burst Size:* max. number of pkts sent consecutively (with no intervening idle)

Policing Mechanisms

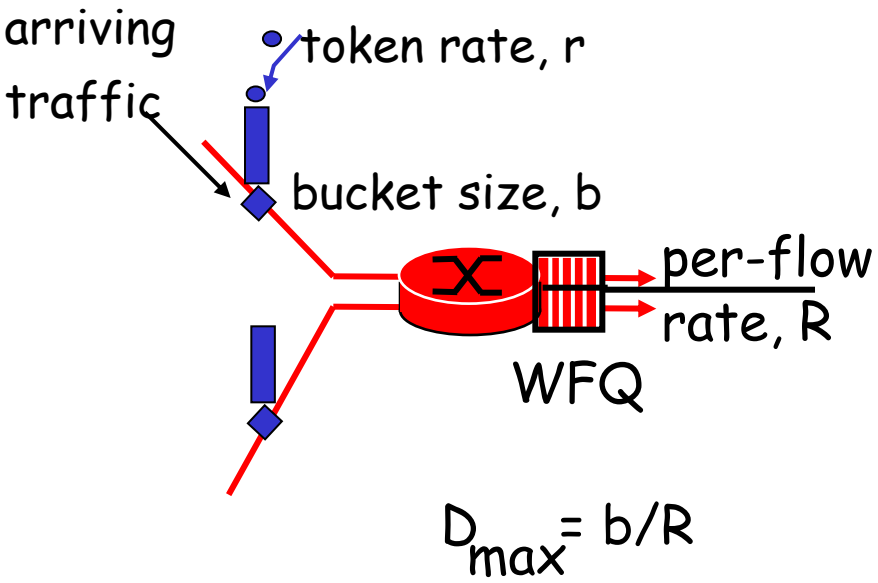
Token Bucket: limit input to specified Burst Size and Average Rate.

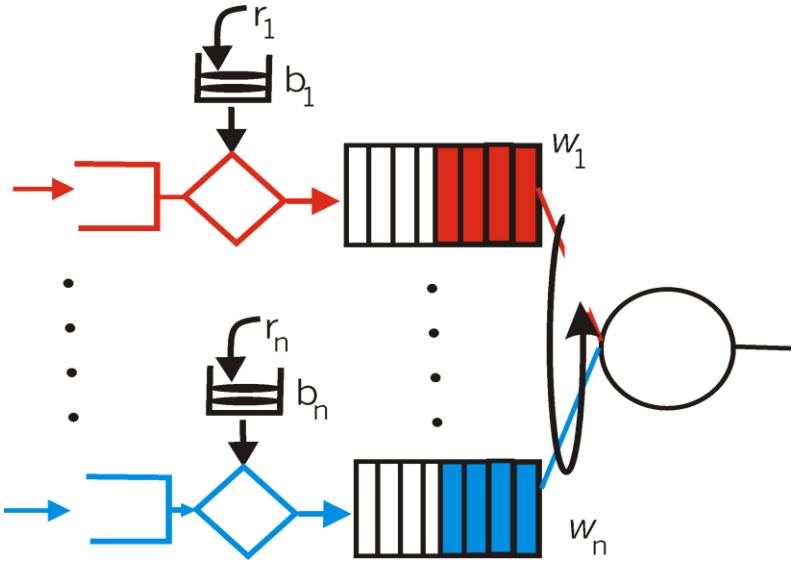


- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b).*

Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., *QoS guarantee*!





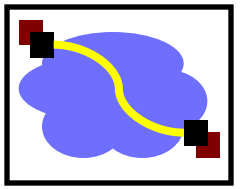
Multi-link congestion management

- Token bucket and leaky bucket manage traffic across a single link.
- But what if we do not trust the incoming traffic to behave?
- Must manage across multiple links
  - Round Robin
  - Fair Queuing

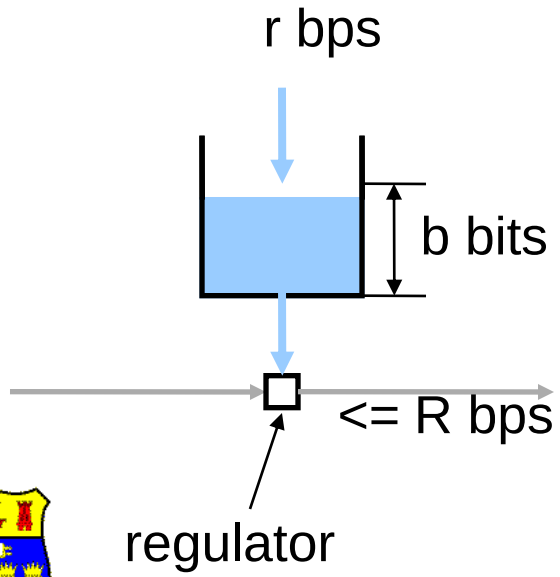
Multi-queue management

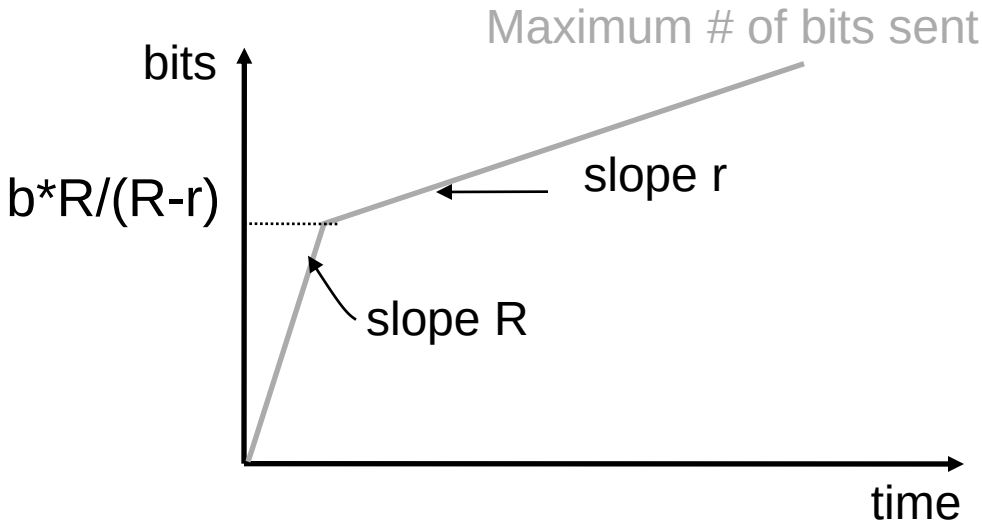
- If one source is sending too many packets, can we allow other sources to continue and just drop the "bad" source?
- First cut: round-robin
  - Service input queues in round-robin order
- What if one flow/link has all large packets, another all small packets?
  - Smaller packets get more link bandwidth

Token Bucket Example

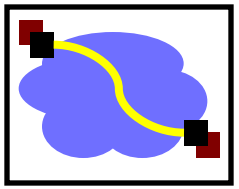


- Parameters
  - r – average rate, i.e., rate at which tokens fill the bucket
  - b – bucket depth
  - R – maximum link capacity or peak rate (optional parameter)
- A bit is transmitted only when there is an available token



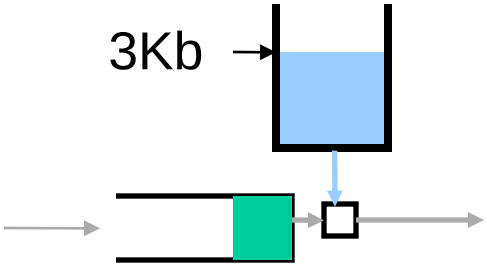


Traffic Enforcement: Example



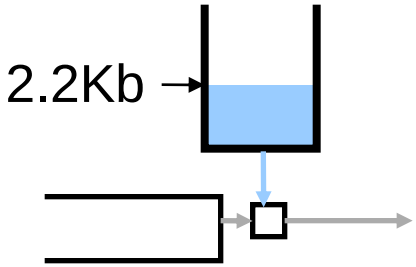
- r = 100 Kbps; b = 3 Kb; R = 500 Kbps

(a)



T = 0 : 1Kb packet arrives

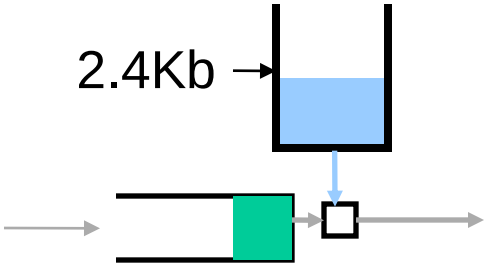
(b)



T = 2ms : packet transmitted

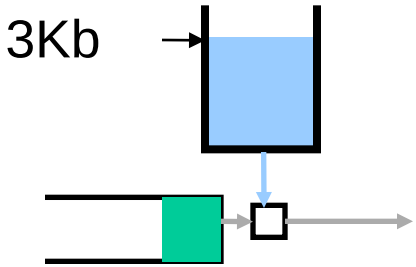
b = 3Kb – 1Kb + 2ms*100Kbps = 2.2Kb

(c)



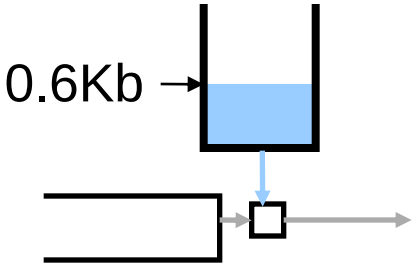
T = 4ms : 3Kb packet arrives

(d)



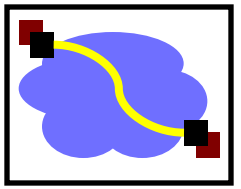
T = 10ms :

(e)



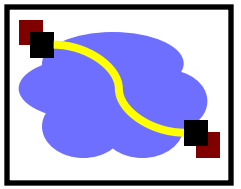
T = 16ms : packet transmitted

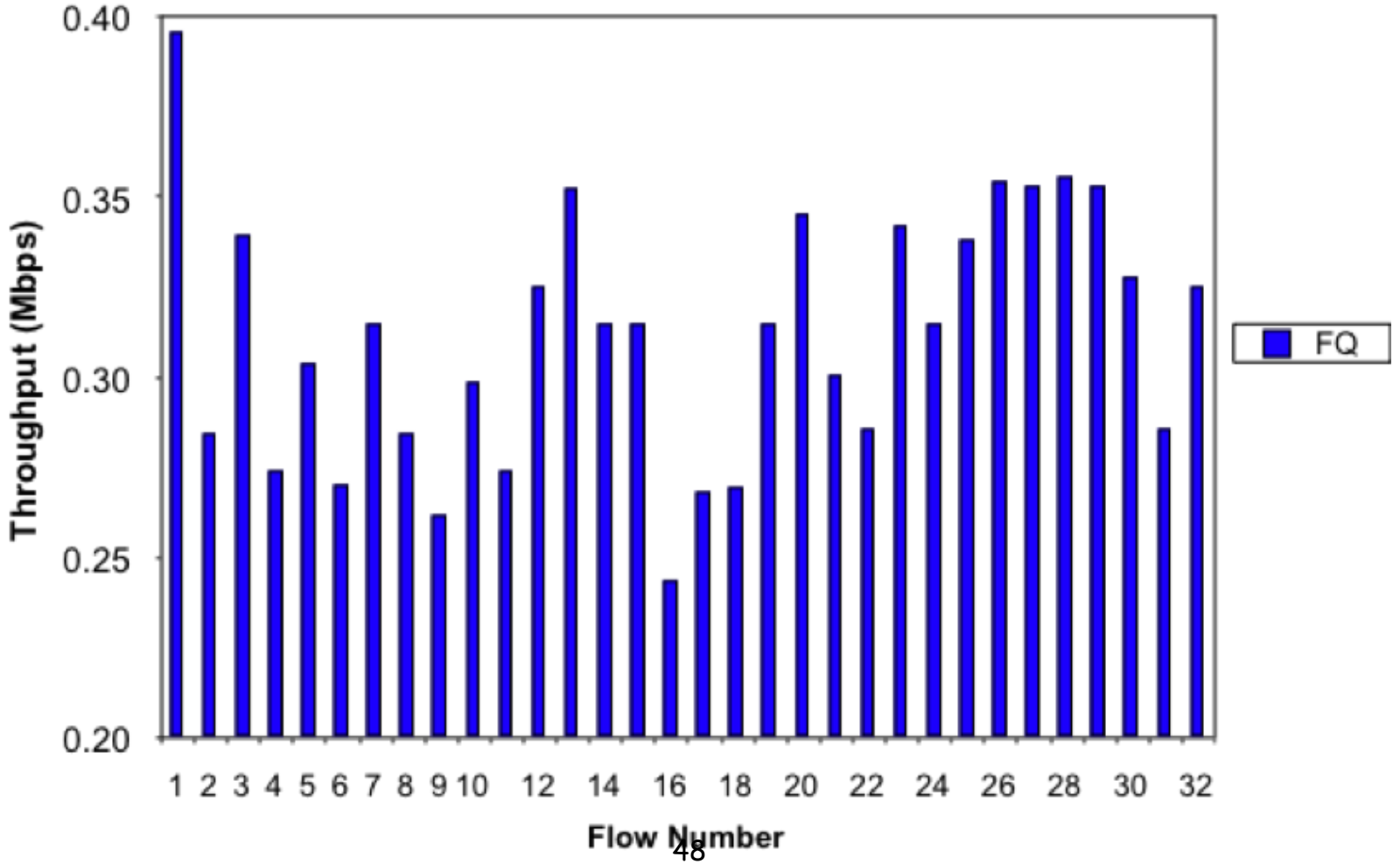
Rate-Limiting and Scheduling



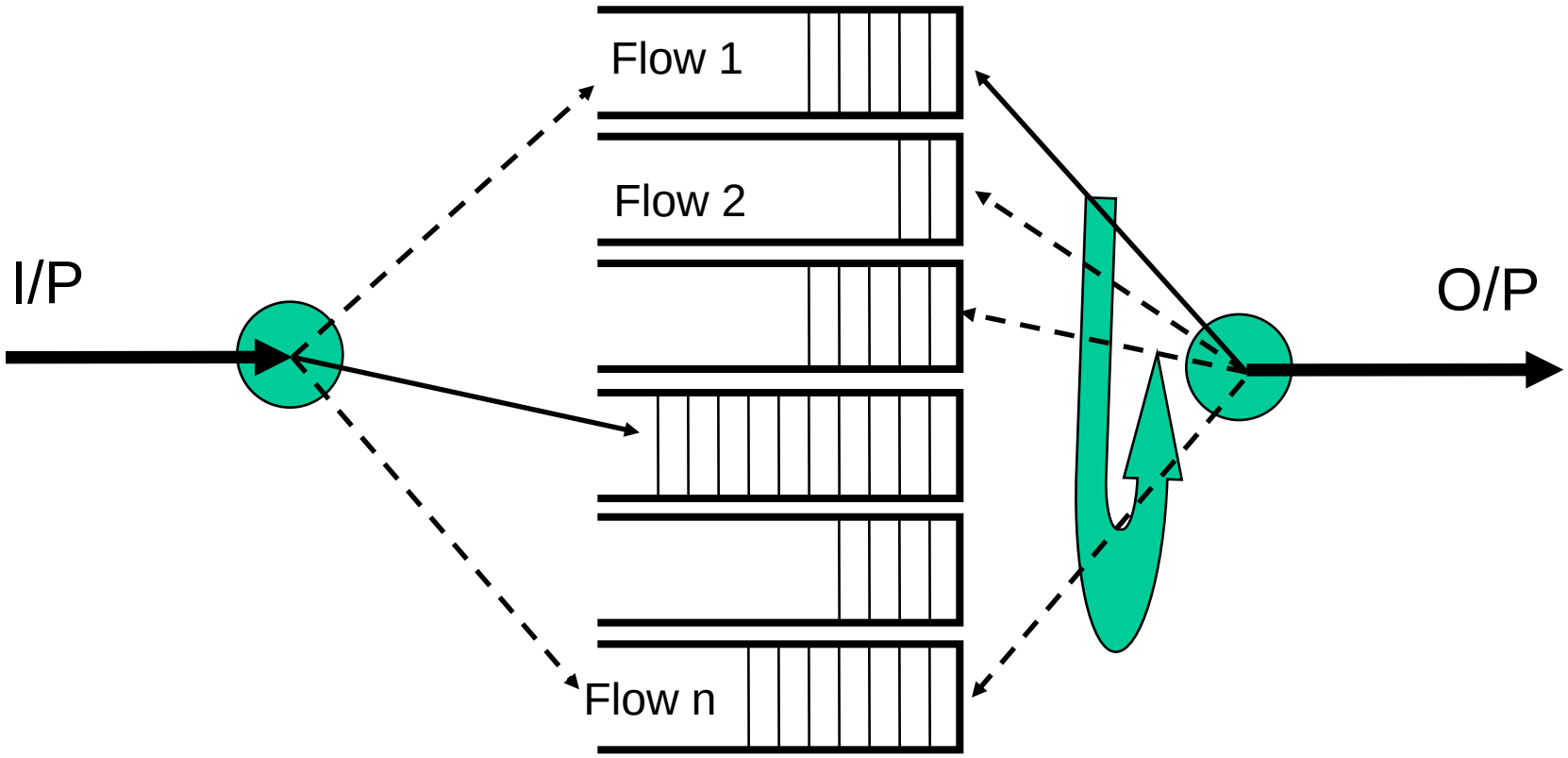
- Rate-limiting: limit the rate of one flow regardless the load in the network
- Scheduling: dynamically allocates resources when multiple flows competing

Example Outcome: Throughput of TCP and UDP Flows With Fair Queueing Router



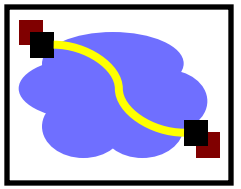


Fair Queueing



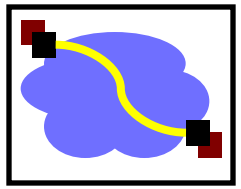
Variation: Weighted Fair Queuing (WFQ)

Fair Queueing



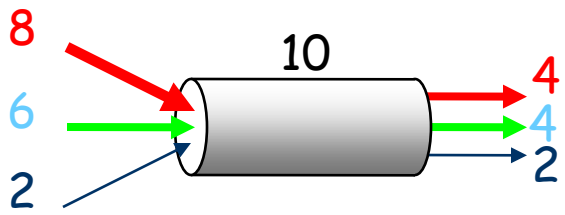
- Maintain a queue for each flow
  - What is a flow?
- Implements max-min fairness: each flow receives $min(r_i, f)$, where
  - r_i – flow arrival rate
  - f – link fair rate (see next slide)
- Weighted Fair Queueing (WFQ) – associate a weight with each flow

Fair Rate Computation: Example 1



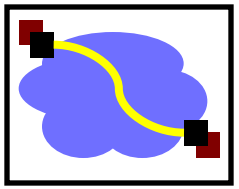
- If link congested, compute f such that

$$\sum_i \min(r_i, f) = C$$



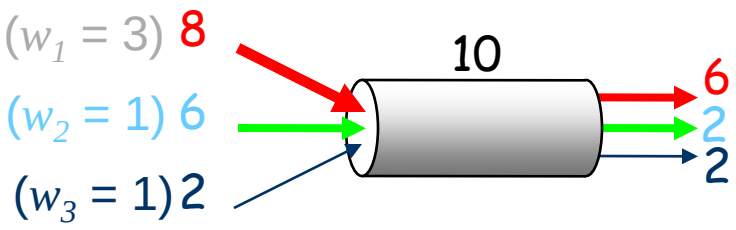
$f = 4$:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Fair Rate Computation: Example 2



- Associate a weight w_i with each flow i
- If link congested, compute f such that

$$\sum_i \min(r_i, f \times w_i) = C$$



f = 2:
min(8, 2*3) = 6
min(6, 2*1) = 2
min(2, 2*1) = 2

Flow i is guaranteed to be allocated a rate >= wi*C/(Σ_k w_k)

⇩

If Σ_k w_k <= C, flow i is guaranteed to be allocated a rate >= w_i