15-441/641: Quality of Service

15-441 Fall 2019
Profs **Peter Steenkiste** & Justine Sherry

Fall 2019
https://computer-networks.github.io/fa19/

Carnegie Mellon University

Overview

- What is QoS?
- Queuing discipline and scheduling
- Traffic Enforcement
- Integrated services

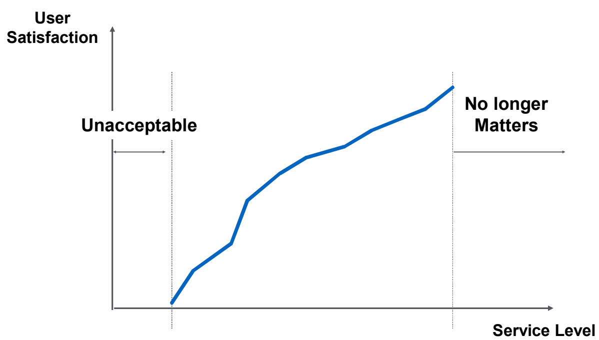
What is QoS?

- The Internet supports best effort packet delivery
  - Sufficient for most applications
  - But some applications require or can benefit from a “higher” level of service
- “Higher” quality of service can mean that bounds are provided for one or more performance parameters
  - Bandwidth: fast data transfers, video
  - Delay, jitter: telephony, interactive video
  - Packet loss: update services
- QoS can also mean that a user gets “better” treatment (than other users)
  - But no guarantees are given, e.g., the “10 items or less” line in the grocery store

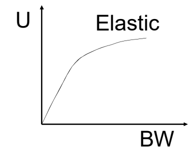
Why Should we Consider QoS?

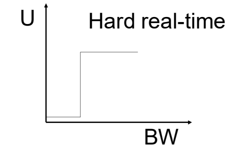
- What is the **basic objective** of network design?
  - Maximize total bandwidth? Minimize latency?
  - **Maximize user satisfaction** – the total **utility** given to users
  - Maximize profit?
- What does utility vs. bandwidth look like?
  - Utility: represents how satisfied a user is with the service
  - Shape depends on application
  - Must be non-decreasing function

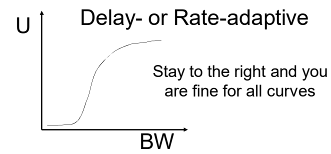
Performance versus Satisfaction



Utility Curve Shapes

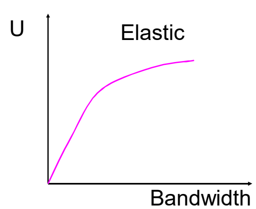






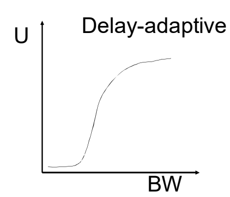
Does equal allocation of bandwidth maximize total utility?

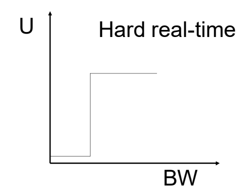
Utility curve – Elastic traffic



Does equal allocation of bandwidth maximize total utility?
Does adding users increase utility?

Utility Curves – Inelastic traffic





Does equal allocation of bandwidth maximize total utility?
Does adding users increase utility?

Inelastic Applications

- Continuous media applications
  - **Lower and upper limit** on acceptable performance.
  - BW below which video and audio are not intelligible
  - Internet telephones, teleconferencing with high delay (200 - 300ms) impair human interaction
  - Sometimes called "tolerant real-time" since they can adapt to the performance of the network
- Hard real-time applications
  - Require **hard limits on performance**
  - E.g. control applications

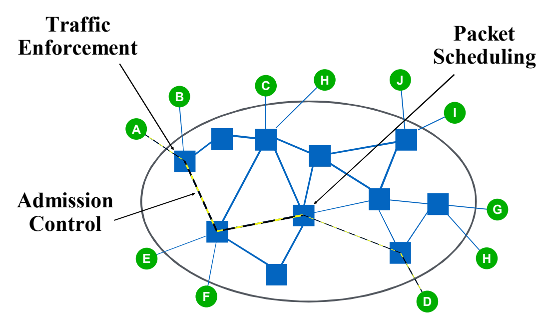
Quality of Service versus Fairness

- Traditional definition of fairness: treat all users equally.
  - E.g., share bandwidth on bottleneck link equally
- QoS: treat users differently.
  - For example, some users get a bandwidth guarantee, while others have to use best effort service
- The two are not in conflict
  - All else being equal, users are treated equally
  - Unequal treatment is based on policies, price:
    - Administrative policies: rank or position
    - Economics: extra payment for preferential treatment

QoS Analogy: Surface Mail

- The defaults is "first class mail".
  - Usually gets there within a few days
  - Sufficient for most letters
- Many "guaranteed" mail delivery services: next day, 2-day delivery, next day am, …..
  - Provide faster and more predictable service at a higher cost
  - Providers differentiate their services: target specific markets with specific requirements and budgets
- Why don't we do the same thing in networks?

QoS Framework



Observation: need full control over network to provide QoS
Internet versus single domain

How to Provide QoS?

- Admission control limits number of flows
  - You cannot provide guarantees if too many flows share resources (bandwidth)
  - For example, telephone networks - busy tone
  - This implies that your request for service can be rejected
- Traffic enforcement limits how much traffic flows can inject based on predefined limits.
  - Make sure user respects the traffic contract
  - Data outside of contract can be dropped or can be sent at a lower priority
- Scheduling support in the routers guarantee that users get their share of the bandwidth.
  - Again based on pre-negotiated bounds
- Analogy: service in a grocery store

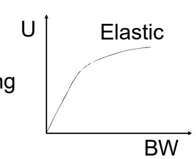
What is a flow?

- Defines the granularity of QoS and fairness
  - TCP flow
  - Traffic to or from a device, user, or network
  - Bigger aggregates for traffic engineering purposes
- Routers use a classifier to determine what flow a packet belongs to
  - Classifier uses a set of fields in the packet header to generate a flow ID
  - Example: (src IP, dest IP, src port, dest port, protocol)
  - Or: (src prefix, dest prfix), i.e., some fields are wildcards

Admission Control - Elastic

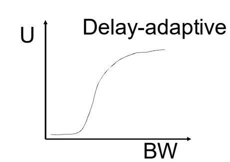
- If U(bandwidth) is concave
  → elastic applications
  - Incremental utility is decreasing with increasing bandwidth
  - It is always advantageous to have more flows with lower bandwidth
    - Increases total utility of flows served
    - No need of admission control

  This is why the Internet works!
- Not so for delay-adaptive and real-time applications



Admission Control - Guarantees

- If U is convex → inelastic applications
  - U(number of flows) is no longer monotonically increasing
  - Need admission control to maximize total utility
- **Admission control** → deciding when adding more people would reduce overall utility
  - E.g., bandwidth or latency guarantees
  - Basically avoids overload



Overview

- What is QoS?
- Queuing discipline and scheduling
- Traffic Enforcement
- Integrated services

Queuing Disciplines

- Each router must implement some queuing discipline
  - Since you have queues you will need a policy
- Queuing allocates both bandwidth and buffer space:
  - Bandwidth: which packet to serve (transmit) next
  - Buffer space: which packet to drop next (when required)
- Queuing discipline affects latency, bandwidth, ..

Network Queuing Disciplines

- First-in-first-out (FIFO) + drop-tail
  - Simplest choice - used widely in the Internet
  - FIFO means all packets treated equally
  - Drop-tail: new packets gets dropped when queue is
  - Important distinction:
    - FIFO: scheduling discipline
    - Drop-tail: drop policy
- Alternative is to do Active Queue Management
  - To improve congestion response
  - Support fairness in presence of non-TCP flows
  - To give flows different types of service – QoS

Alternative Drop Policies

- Avoid lockout and full queue problems
- Random drop and drop front policies
  - Drop random packet or packet at the head of the queue if the queue is full and a new packet arrives
  - Solve the lock-out problem but not the full-queues problem
  - May trigger congestion response faster
- Random Early Discard (RED) and Explicit Congestion Notification (ECN) slow down receivers before queues are full
  - RED: drop some packets before queue is full
  - ECN: mark a bit in the headers to notify receiver (who notifies the sender) of congestion onset without dropping a packet

Problems in Achieving fairness

- In the Internet, fairness is only achieved if all flows play by the same rules
  - But it is complicated: fairness is poorly defined for short flows, many versions of TCP co-exist, etc.
- In practice: <u>most</u> sources must use TCP or be "TCP friendly"
  - Most sources are cooperative
  - Most sources implement homogeneous/compatible control law
    - Compatible does not mean identical
    - Typically means less aggressive than TCP
- What if sources do not play by the rule?
  - E.g., TCP versus UDP without congestion control

Fairness Goals In Practice

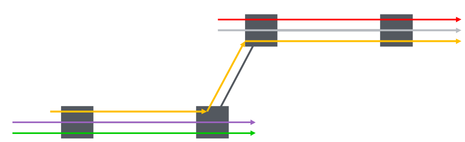
- Allocate resources fairly
  - Partially achieved by using similar congestion control rules
- Isolate ill-behaved users
  - This is challenging
  - How about users who start with a large initial congestion window
  - How about UDP flows (good news: uncommon)
  - How about users who modify TCP (good news: very hard)
- Still achieve statistical multiplexing
  - One flow can fill entire pipe if no contenders
  - Work conserving → scheduler never idles link if it has a packet

What is Fairness?

- At what granularity?
  - Flows, connections, domains?
- What if users have different RTTs/links/etc.
  - Should it share a link fairly or be TCP fair?
- Maximize fairness index?
  - Fairness = $(\Sigma x_i)^2/n(\Sigma x_i^2)$ $0<\text{fairness}<1$
- Basically a tough question to answer!
- Good to separate the design of the mechanisms from definition of a policy
  - User = arbitrary granularity
- One example: max-min fairness

Max-min Fairness

- Give users with "small" demand what they want, evenly divide unused resources to "big" users
- Formally:
  - Resources allocated in terms of increasing demand
  - No source gets resource share larger than its demand
  - Sources with unsatisfied demands get equal share of resource

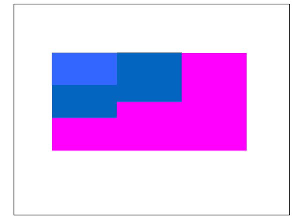


Implementing Max-min Fairness

- Generalized processor sharing
  - Fluid fairness
  - Bitwise round robin among all queues
- Why not simple round robin?
  - Variable packet length → can get more service by sending bigger packets
  - Unfair instantaneous service rate
    - What if packets arrive just before/after packet departs?
- We will use bit-bit round robin as an example
  - Many other algorithms exist

Bit-by-bit RR Illustration

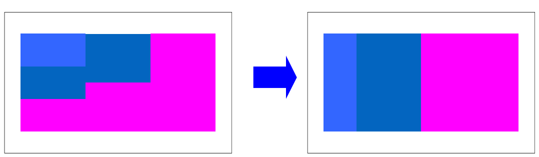
- Send one bit for every flow that has data queued – perfect!
- … but not feasible to interleave bits on real networks
  - FQ simulates bit-by-bit RR



FYI Only

Fair Queuing

- Mapping bit-by-bit schedule onto packet transmission schedule
- Transmit packet sequentially but in bit RR order
  - How do you compute this packet order?
  - Must be efficient and work for any order



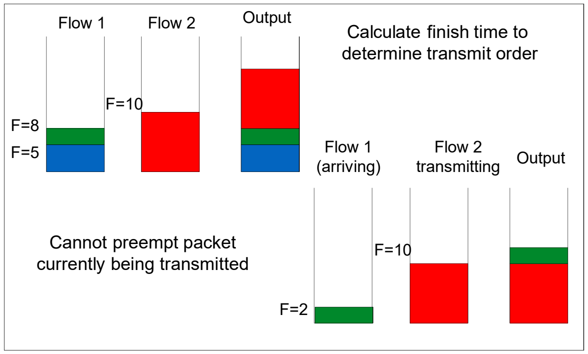
FYI Only

Approximating Bit-by-bit RR

- Single flow: clock ticks when a bit is transmitted. For packet i:
  - A_i = arrival time, S_i = transmit start time, P_i = transmission time, F_i = finish transmit time
  - $F_i = S_i+P_i = \max(F_{i-1}, A_i) + P_i$
- Multiple flows: clock ticks when a bit from all active flows is transmitted → round number
  - Models the fact that you would transmit one bit from each flow in bit RR
  - Can now calculate F_i for each packet if number of flows is know at all times – determines packet order
  - Need to know flow count to calculate clock tick time

FYI Only

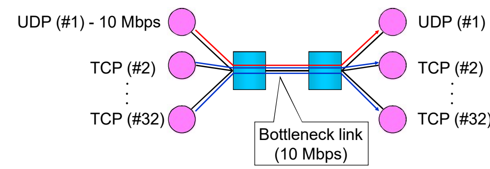
Bit-by-bit RR Example



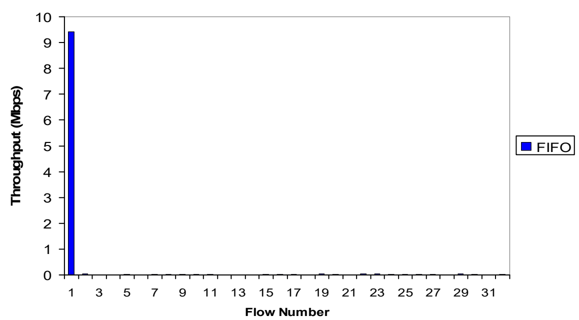
FYI Only

An Example: TCP versus UDP

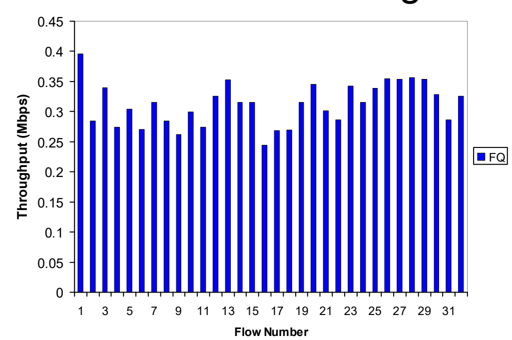
1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps line



Throughput of UDP and TCP Flows With FIFO



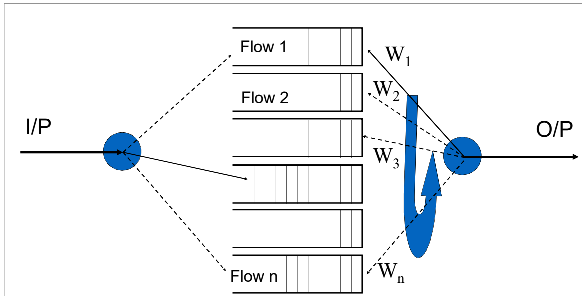
Example: Throughput of TCP and UDP Flows With Fair Queueing Router



Fair Queuing Tradeoffs

- Complex computation
  - Overhead of classification and scheduling
  - Must keep queues sorted by finish times
  - Computation changes whenever the flow count changes
- Complex state – must keep queue per flow
  - Hard in routers with many flows (e.g., backbone routers)
  - Flow aggregation is a possibility (e.g. do fairness per domain)
- FQ can control congestion by monitoring flows
- Weighted fair queuing can give flows a different fraction of the bandwidth - controlled by a weight W_i
  - Bandwidth of flow i is $W_i / \sum W_j$

WFQ Illustration

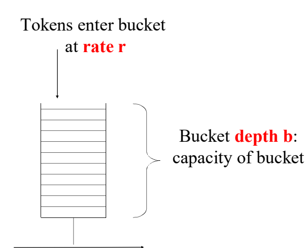


Variation: Weighted Fair Queuing (WFQ)

Overview

- What is QoS?
- Queuing discipline and scheduling
- Traffic Enforcement
- Integrated services

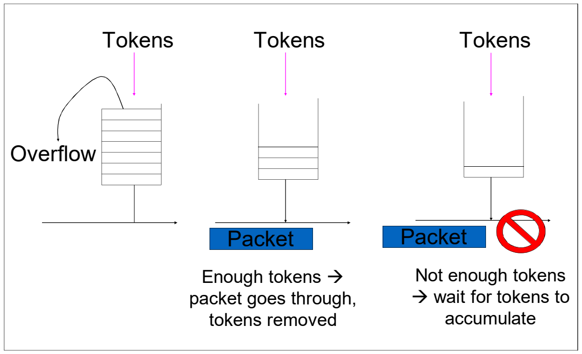
Traffic Enforcement: Token Bucket Filter



Operation:

- If bucket fills, tokens are discarded
- Sending a packet of size P uses P tokens
- If bucket has P tokens, packet sent at max rate, else must wait for tokens to accumulate

Token Bucket Operation

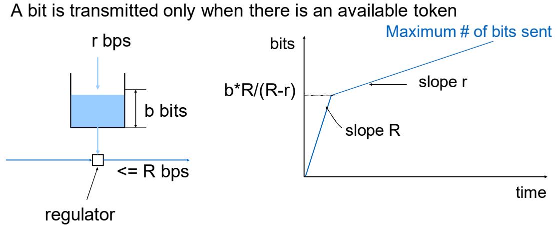


Token Bucket Characteristics

- Can characterize flow using a token bucket: smallest parameters for which no packets will be delayed
- On the long run, rate is limited to r
- On the short run, a burst of size b can be sent
- Maximum amount of traffic that can enter the network in time interval T is bounded by:
  - Simple case: Traffic = b + r*T
- Information useful to admission algorithm

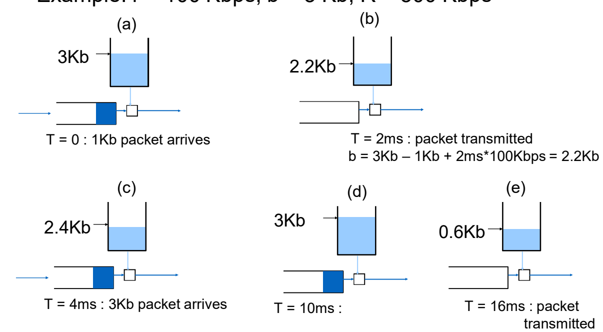
Token Bucket

- 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

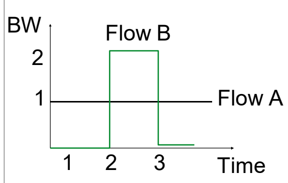


Traffic Enforcement: Example

Example: r = 100 Kbps; b = 3 Kb; R = 500 Kbps



Token Bucket Specs - Example



Flow A: r = 1 MBps, b=1 byte

Flow B: r = 1 MBps, b=1MB

Overview

- What is QoS?
- Queuing discipline and scheduling
- Traffic Enforcement
- Integrated services

Integrated Services Traffic Classes

- IETF RFC 1633 (1994)
- **Guaranteed** service
  - For **hard real-time** applications
  - Fixed guarantee rate, assuming clients send at agreed-upon rate
- **Predicted** service
  - For **delay-adaptive** applications
  - Two components
    - If conditions do not change, commit to current service
    - If conditions change, take steps to deliver consistent performance (help apps minimize playback delay)
    - Implicit assumption – network does not change much over time
- **Datagram/best effort service**
- Also includes Resource reSerVation Protocol (RSVP) for establishing paths; may also need routing support

Lessons

- What type of applications are there? → Elastic, adaptive real-time , and hard real-time.
- Why do we need admission control → to maximize utility
- How do token buckets + WFQ provide QoS guarantees?