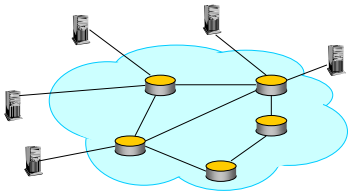
CS 194: Distributed Systems
Resource Allocation

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

Goals and Approach

- Goal: achieve predicable performances
- Three steps:
  1) Estimate application's resource needs (not in this lecture)
  2) Admission control
  3) Resource allocation



Type of Resources

- CPU
- Storage: memory, disk
- Bandwidth
- Devices (e.g., vide camera, speakers)
- Others:
  - File descriptors
  - Locks
  - …

Allocation Models

- **Shared**: multiple applications can share the resource
  - E.g., CPU, memory, bandwidth
- **Non-shared**: only one application can use the resource at a time
  - E.g., devices

Not in this Lecture…

- How application determine their resource needs
- How users "pay" for resources and how they negotiate resources
- Dynamic allocation, i.e., application allocates resources as it needs them

In this Lecture…

- Focus on bandwidth allocation
  - CPU similar
  - Storage allocation usually done in fixed chunks
- Assume application requests all resources at once

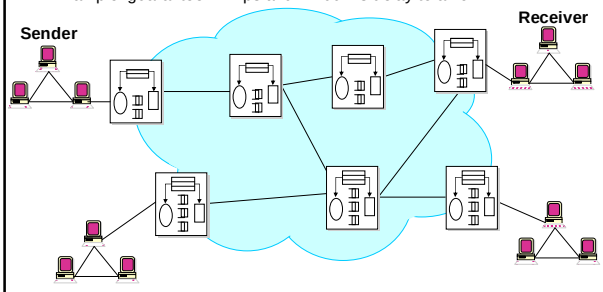
Two Models

- Integrated Services
  - Fine grained allocation; per-flow allocation
- Differentiated services
  - Coarse grained allocation (both in time and space)
- Flow: a stream of packets between two applications or endpoints

7

Integrated Services Example

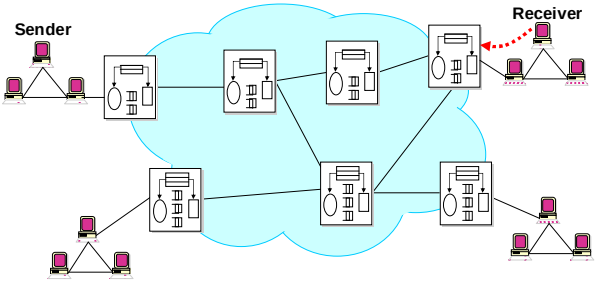
- Achieve per-flow bandwidth and delay guarantees
  - Example: guarantee 1MBps and < 100 ms delay to a flow



8

Integrated Services Example

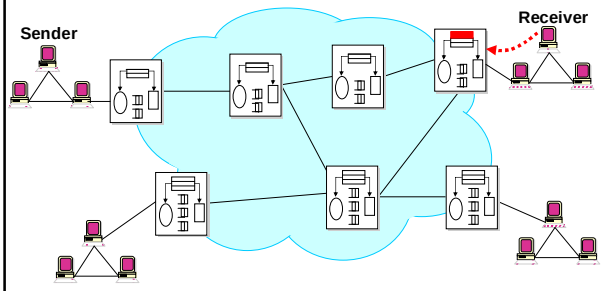
- Allocate resources - perform per-flow admission control



9

Integrated Services Example

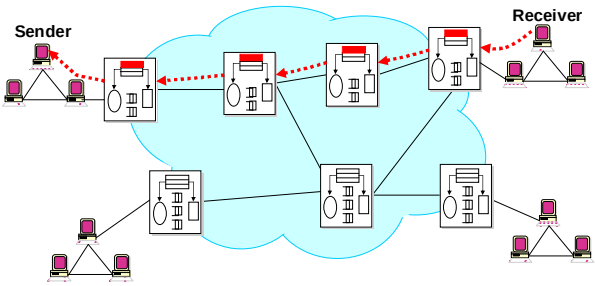
- Install per-flow state



10

Integrated Services Example

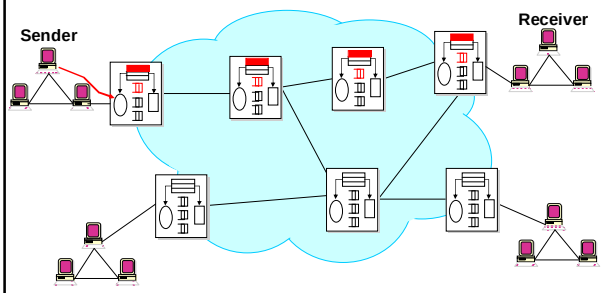
- Install per flow state



11

Integrated Services Example: Data Path

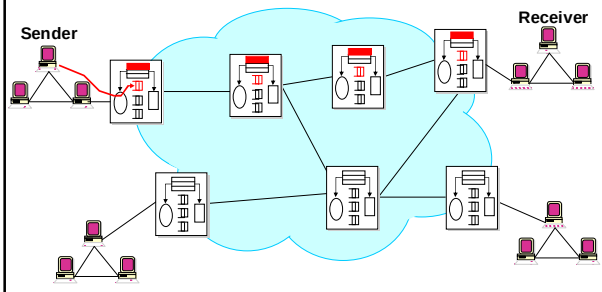
- Per-flow classification



12

Integrated Services Example: Data Path

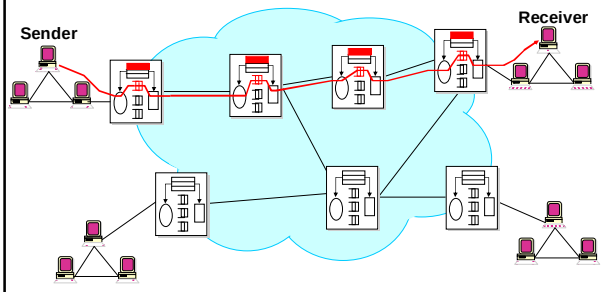
- Per-flow buffer management



13

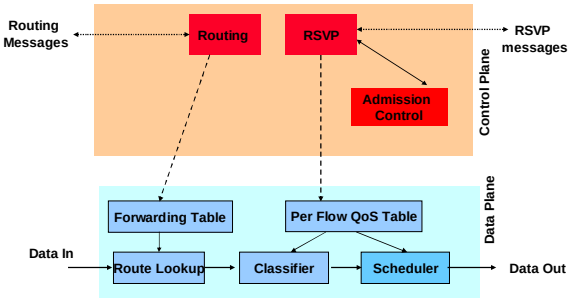
Integrated Services Example

- Per-flow scheduling



14

How Things Fit Together



15

Service Classes

- Multiple service classes
- Service: contract between network and communication client
  - End-to-end service
  - Other service scopes possible
- Three common services
  - Best-effort (“elastic” applications)
  - Hard real-time (“real-time” applications)
  - Soft real-time (“tolerant” applications)

16

Hard Real Time: Guaranteed Services

- Service contract
  - Network to client: guarantee a deterministic upper bound on delay for each packet in a session
  - Client to network: the session does not send more than it specifies
- Algorithm support
  - Admission control based on worst-case analysis
  - Per flow classification/scheduling at routers

17

Soft Real Time: Controlled Load Service

- Service contract:
  - Network to client: similar performance as an unloaded best-effort network
  - Client to network: the session does not send more than it specifies
- Algorithm Support
  - Admission control based on measurement of aggregates
  - Scheduling for aggregate possible

18

Role of RSVP in the Architecture

- Signaling protocol for establishing per flow state
- Carry resource requests from hosts to routers
- Collect needed information from routers to hosts
- At each hop
  - Consult admission control and policy module
  - Set up admission state or informs the requester of failure

19

RSVP Design Features

- IP Multicast centric design (not discussed here…)
- Receiver initiated reservation
- Different reservation styles
- Soft state inside network
- Decouple routing from reservation

20

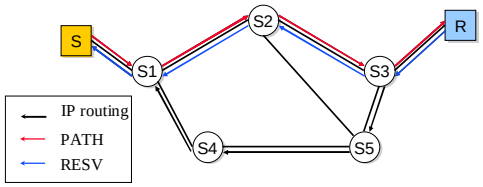
RSVP Basic Operations

- Sender: sends PATH message via the data delivery path
  - Set up the path state each router including the address of previous hop
- Receiver sends RESV message on the reverse path
  - Specifies the reservation style, QoS desired
  - Set up the reservation state at each router
- Things to notice
  - Receiver initiated reservation
  - Decouple routing from reservation
  - Two types of state: path and reservation

21

Route Pinning

- Problem: asymmetric routes
  - You may reserve resources on R→S3→S5→S4→S1→S, but data travels on S→S1→S2→S3→R !
- Solution: use PATH to remember direct path from S to R, i.e., perform route pinning



22

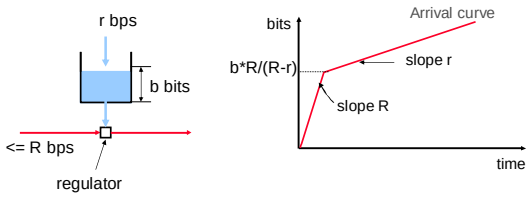
PATH and RESV messages

- PATH also specifies
  - Source traffic characteristics
    - Use token bucket
  - Reservation style – specify whether a RESV message will be forwarded to this server
- RESV specifies
  - Queueing delay and bandwidth requirements
  - Source traffic characteristics (from PATH)
  - Filter specification, i.e., what senders can use reservation
  - Based on these routers perform reservation

23

Token Bucket and Arrival Curve

- 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
- Arrival curve – maximum number of bits transmitted within an interval of time of size t



24

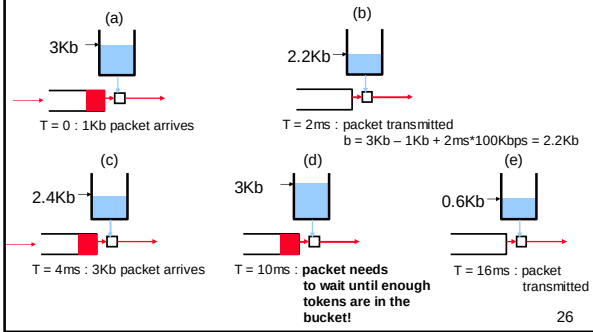
How Is the Token Bucket Used?

- Can be enforced by
  - End-hosts (e.g., cable modems)
  - Routers (e.g., ingress routers in a Diffserv domain)
- Can be used to characterize the traffic sent by an end-host

25

Traffic Enforcement: Example

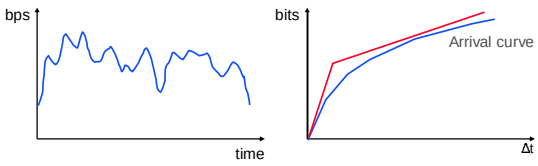
- r = 100 Kbps; b = 3 Kb; R = 500 Kbps



26

Source Traffic Characterization

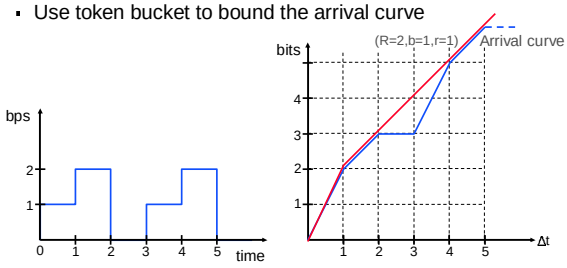
- Arrival curve – maximum amount of bits transmitted during an interval of time Δt
- Use token bucket to bound the arrival curve



27

Source Traffic Characterization: Example

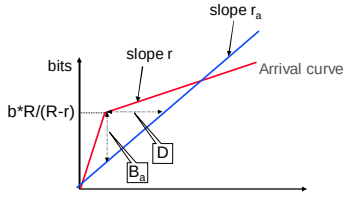
- Arrival curve – maximum amount of bits transmitted during an interval of time Δt
- Use token bucket to bound the arrival curve



28

QoS Guarantees: Per-hop Reservation

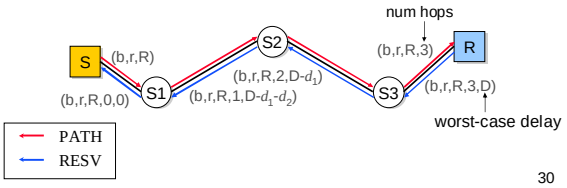
- End-host: specify
  - The arrival rate characterized by token-bucket with parameters (b,r,R)
  - The maximum maximum admissible delay D
- Router: allocate bandwidth r_a and buffer space B_a such that
  - No packet is dropped
  - No packet experiences a delay larger than D



29

End-to-End Reservation

- When R gets PATH message it knows
  - Traffic characteristics (tspec): (r,b,R)
  - Number of hops
- R sends back this information + worst-case delay in RESV
- Each router along path provide a per-hop delay guarantee and forward RESV with updated info
  - In simplest case routers split the delay



30

31

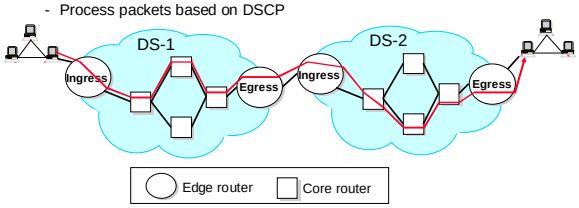
Differentiated Services (Diffserv)

- 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

32

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



33

Differentiated Services

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

34

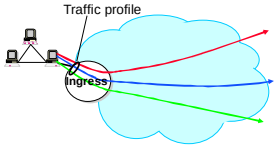
Assured Service [Clark & Wroclawski '97]

- 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

35

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) apriori
- This makes service very useful, but hard to provision (why ?)



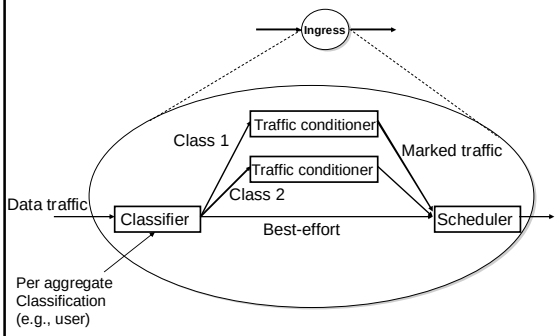
36

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

37

Edge Router



38

Assumptions

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

39

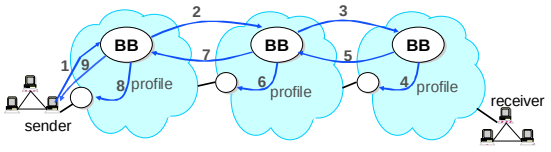
Control Path

- Each domain is assigned a Bandwidth Broker (BB)
  - Usually, used to perform ingress-egress bandwidth allocation
- BB is responsible to perform 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

40

Example

- Achieve end-to-end bandwidth guarantee



41

Comparison to Best-Effort and Intserv

	Diffserv	Intserv
Service	Per aggregate isolation Per aggregate guarantee	Per flow isolation Per flow guarantee
Service scope	Domain	End-to-end
Complexity	Long term setup	Per flow steup
Scalability	Scalable (edge routers maintains per aggregate state; core routers per class state)	Not scalable (each router maintains per flow state)

42

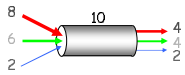
Weighted Fair Queueing (WFQ)

- The scheduler of choice to implement bandwidth and CPU sharing
- 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

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

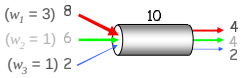


f = 4:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

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 >= w_i*C/(Σ_k w_k)

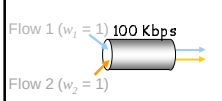
⇩

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

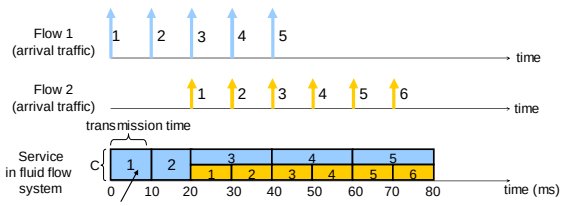
Fluid Flow System

- Flows can be served one bit at a time
- WFQ can be implemented using bit-by-bit weighted round robin
  - During each round from each flow that has data to send, send a number of bits equal to the flow's weight

Fluid Flow System: Example 1

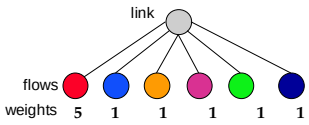


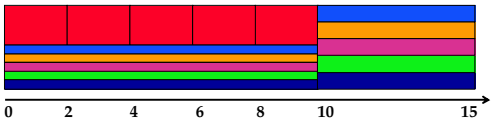
	Packet Size (bits)	Packet inter-arrival time (ms)	Rate (C) (Kbps)
Flow 1	1000	10	100
Flow 2	500	10	50



Fluid Flow System: Example 2

- Red flow has sends packets between time 0 and 10
  - Backlogged flow → flow's queue not empty
- Other flows send packets continuously
- All packets have the same size



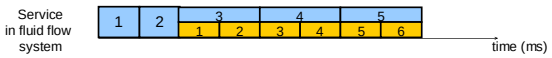


Implementation In Packet System

- Packet (Real) system: packet transmission cannot be preempted.
- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system

49

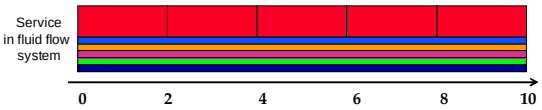
Packet System: Example 1



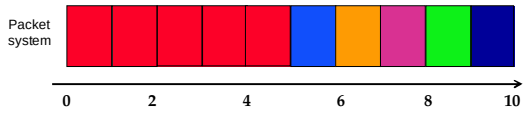
- Select the first packet that finishes in the fluid flow system

50

Packet System: Example 2



- Select the first packet that finishes in the fluid flow system



51

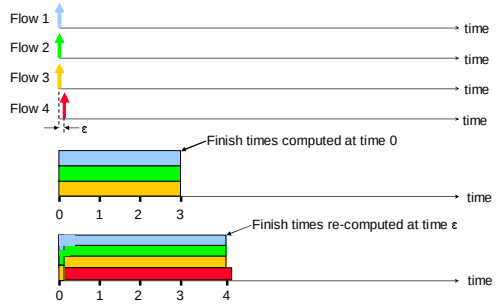
Implementation Challenge

- Need to compute the finish time of a packet in the fluid flow system…
- … but the finish time may change as new packets arrive!
- Need to update the finish times of all packets that are in service in the fluid flow system when a new packet arrives
  - But this is very expensive; a high speed router may need to handle hundred of thousands of flows!

52

Example

- Four flows, each with weight 1



53

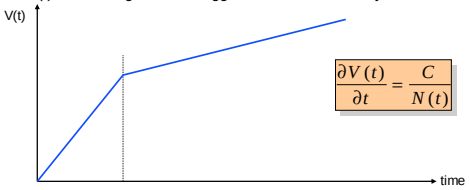
Solution: Virtual Time

- Key Observation: while the finish times of packets may change when a new packet arrives, the order in which packets finish doesn't!
  - Only the order is important for scheduling
- Solution: instead of the packet finish time maintain the number of rounds needed to send the remaining bits of the packet (virtual finishing time)
  - Virtual finishing time doesn't change when the packet arrives
- System virtual time – index of the round in the bit-by-bit round robin scheme

54

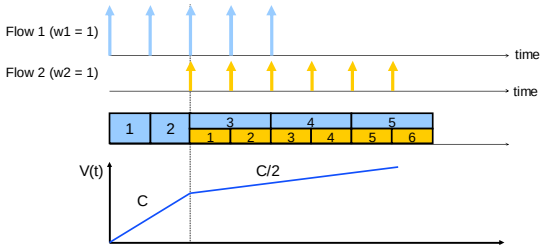
System Virtual Time: *V*(t)

- Measure service, instead of time
- V(t) slope – normalized rate at which every backlogged flow receives service in the fluid flow system
  - *C* – link capacity
  - *N*(t) – total weight of backlogged flows in fluid flow system at time t

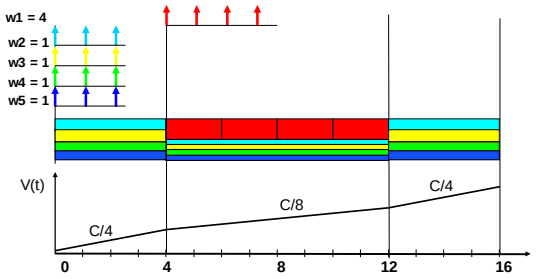


System Virtual Time (V(t)): Example 1

- V(t) increases inversely proportionally to the sum of the weights of the backlogged flows



System Virtual Time: Example



Fair Queueing Implementation

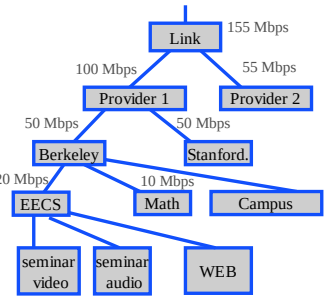
- Define
  - F_i^k - virtual finishing time of packet k of flow i
  - a_i^k - arrival time of packet k of flow i
  - L_i^k - length of packet k of flow i
  - w_i – weight of flow i
- The finishing time of packet $k+1$ of flow i is

$$F_i^{k+1} = \max(V(a_i^{k+1}), F_i^k) + L_i^{k+1}/w_i$$

Properties of WFQ

- Guarantee that any packet is transmitted within *packet_lengt/link_capacity* of its transmission time in the fluid flow system
  - Can be used to provide guaranteed services
- Achieve max-min fair allocation
  - Can be used to protect well-behaved flows against malicious flows

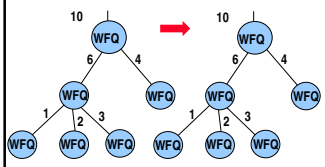
Hierarchical Link Sharing



- Resource contention/sharing at different levels
- Resource management policies should be set at different levels, by different entities
  - Resource owner
  - Service providers
  - Organizations
  - Applications

Packet Approximation of H-WFQ

Fluid Flow H-WFQ ➡ Packetized H-WFQ



- Idea 1
  - Select packet finishing first in H-WFQ assuming there are no future arrivals
  - Problem:
    - Finish order in system dependent on future arrivals
    - Virtual time implementation won't work
- Idea 2
  - Use a hierarchy of WFQ to approximate H-WFQ

61