

Providing Guaranteed Services without Per Flow Management

Ion Stoica and Hui Zhang
Carnegie Mellon University
{istoica,hzhang}@cs.cmu.edu

Today's Internet

- **Service: best-effort datagram delivery**
- **Architecture: “stateless” routers**
 - excepting routing state, routers do not maintain any fine grained state about traffic
- **Properties**
 - scalable
 - robust

istoica@cs.cmu.edu

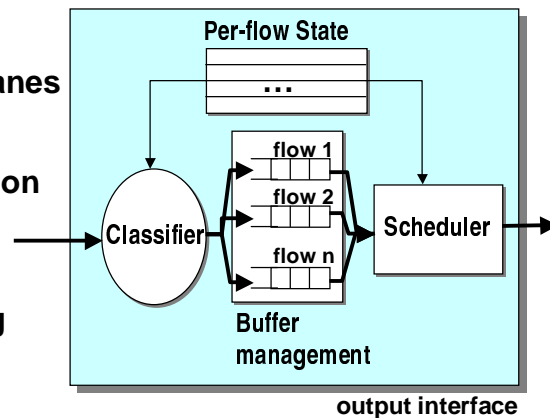
Trends

- Deploy more sophisticated services, e.g., traffic management, Quality of Service (QoS)
- Two type of solutions:
 - **Stateless:** preserve original Internet advantages
 - RED – support for congestion control
 - Differentiated services (Diffserv) – provide QoS
 - **Stateful:** routers perform per flow management
 - Fair Queueing - support for congestion control
 - Integrated services (Intserv) – provide QoS

istoica@cs.cmu.edu

Complexity of Stateful Solutions

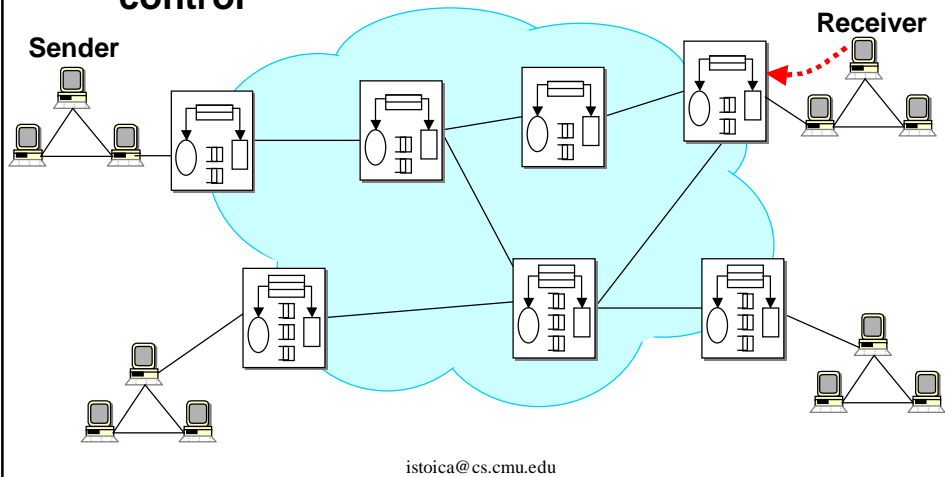
- Control path
 - install and maintain per-flow state for data and control planes
- Data path
 - Per-flow classification
 - Per-flow buffer management
 - Per-flow scheduling



istoica@cs.cmu.edu

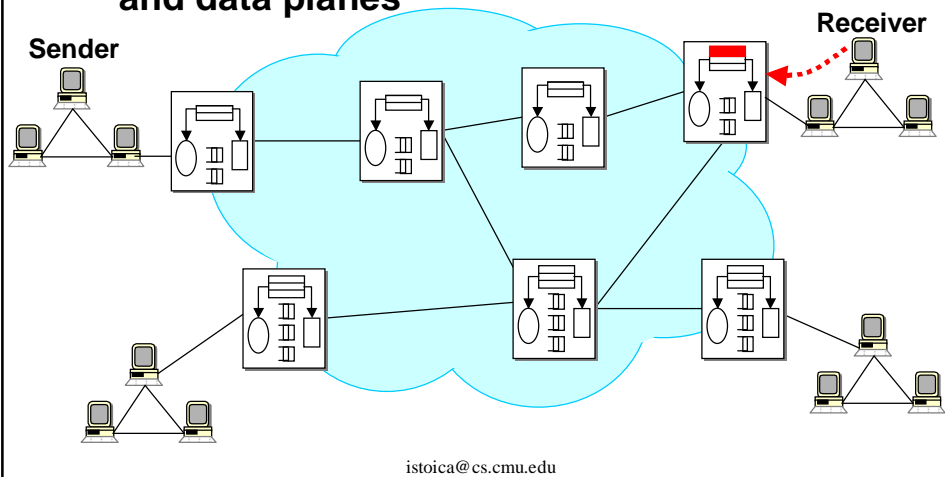
Example: Guaranteed Services (Intserv)

- **Control Path: perform per-flow admission control**



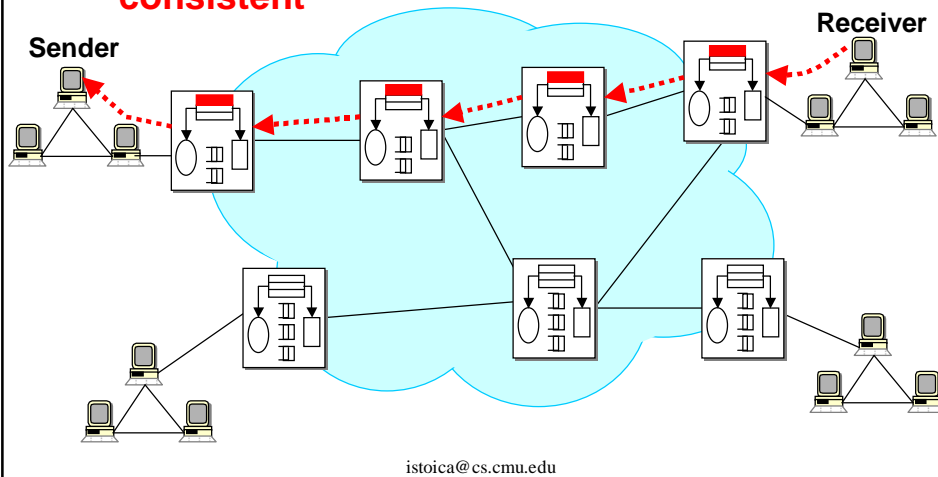
Example: Guaranteed Services (Intserv)

- **Control Path: install per-flow state for control and data planes**



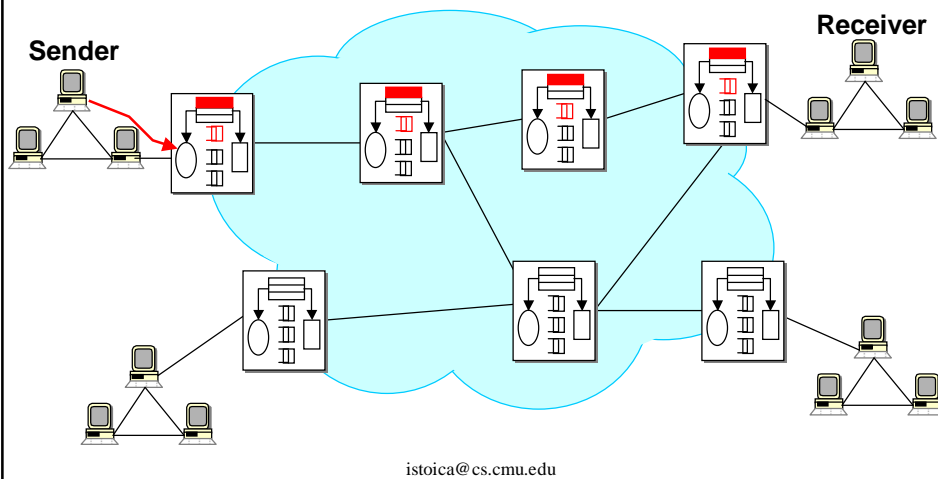
Example: Guaranteed Services (Intserv)

- Challenge: maintain per-flow state **consistent**



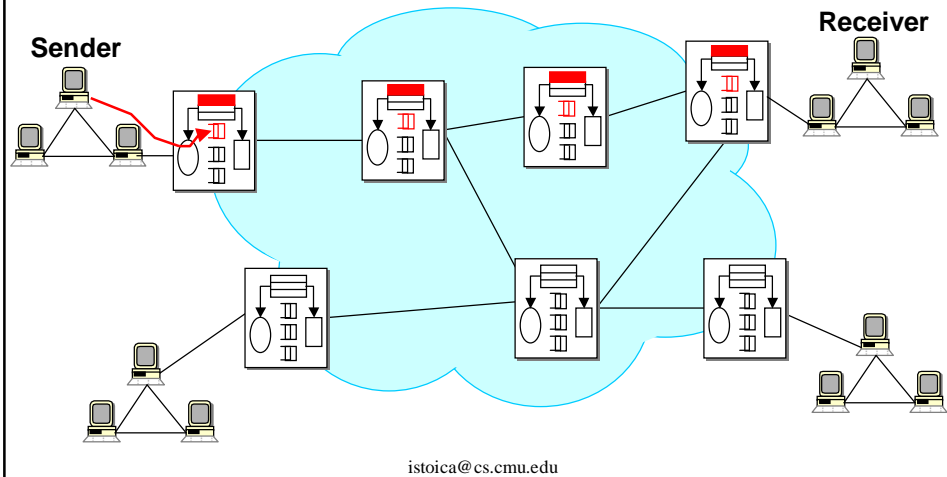
Example: Guaranteed Services (Intserv)

- Data Path: per-flow classification



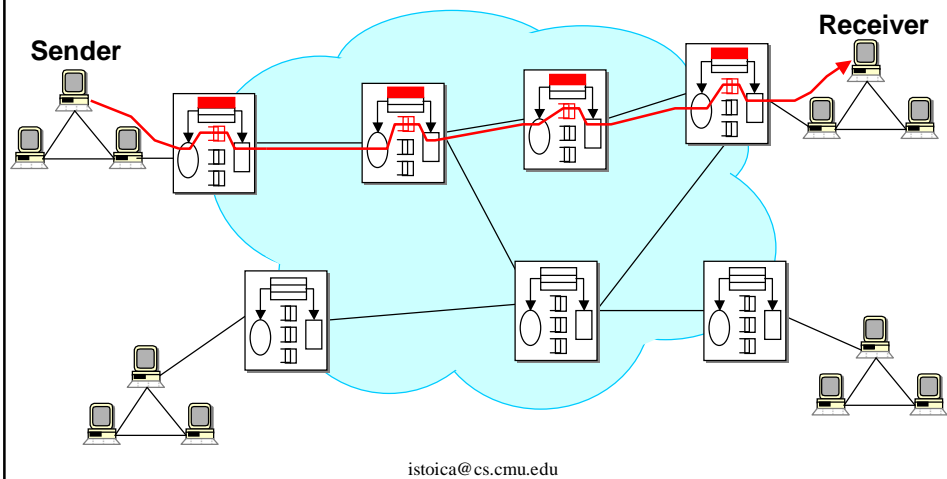
Example: Guaranteed Services (Intserv)

- Data Path: per-flow buffer management



Example: Guaranteed Services (Intserv)

- Data Path: per-flow scheduling



Stateless vs. Stateful

- **Stateless** solutions are more
 - scalable
 - robust
- **Stateful** solutions provide more powerful and flexible services
 - Fair Queueing vs. RED
 - Intserv vs. Diffserv

istoica@cs.cmu.edu

Intserv vs. Diffserv

- **Intserv:**
 - provide per flow bandwidth and delay guarantees, and achieve high resource utilization
 - support for fined grained and short-lived reservations
- **Diffserv (Premium Service)**
 - **cannot** provide low delay guarantees and high resource utilization simultaneously
 - even at low utilization (e.g., 10%) in a medium network (e.g., 15 hops) the worst case queueing delay > 200ms
 - centralized admission control (e.g., Bandwidth Broker) - **not** appropriate for short-lived reservations

istoica@cs.cmu.edu

Question

- Can we achieve the best of two worlds, i.e., provide services implemented by **stateful** networks while maintaining advantages of **stateless** architectures?

istoica@cs.cmu.edu

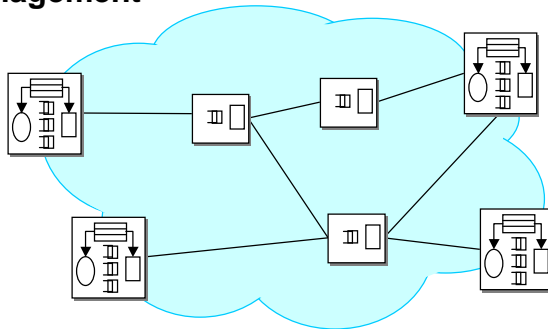
Answer

- Yes, at least in some interesting cases:
- Approximate Fair Queueing [SIGCOMM'98]
- **Provide guaranteed services**

istoica@cs.cmu.edu

Scalable Core (SCORE)

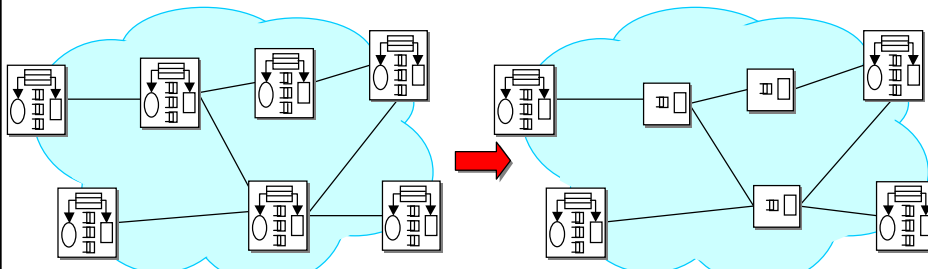
- A contiguous region of network in which
 - edge nodes – perform per flow management
 - core nodes – do **not** perform per flow management



istoica@cs.cmu.edu

The Approach

1. Define a reference **stateful** network that implements the desired service
2. **Emulate** the functionality of the reference network in a SCORE network



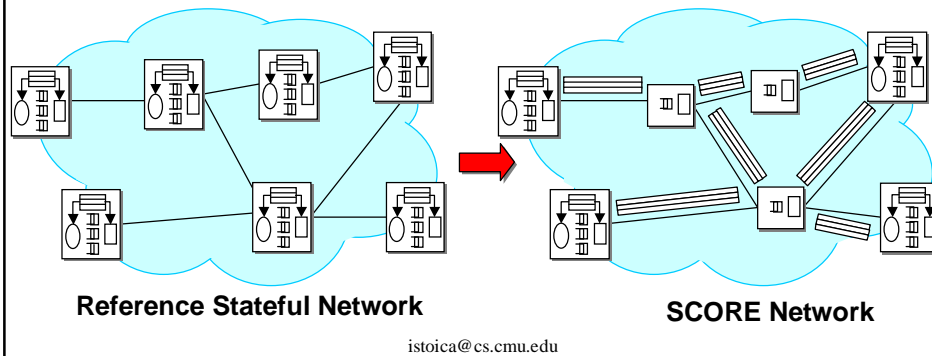
Reference Stateful Network

SCORE Network

istoica@cs.cmu.edu

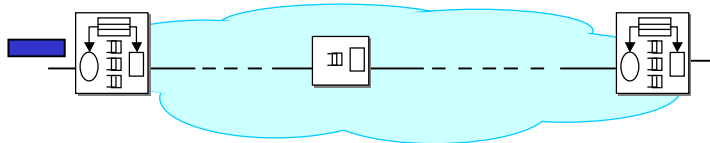
The Idea

- Instead of having core routers maintaining per-flow state **have packets carry per-flow state**



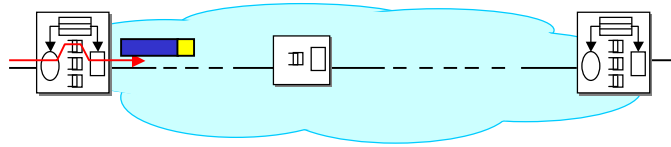
The Technique: Dynamic Packet State (DPS)

- Ingress node: compute and insert flow state in packet's header



The Technique: Dynamic Packet State (DPS)

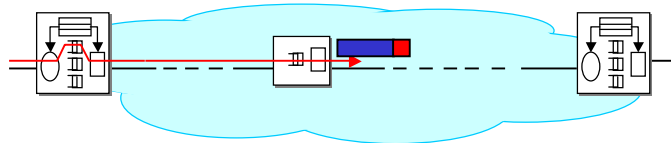
- **Ingress node: compute and insert flow state in packet's header**



istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

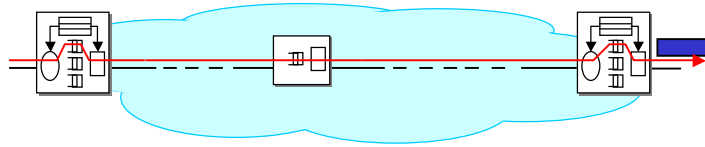
- **Core node:**
 - process packet based on state it carries and node's state
 - update both packet and node's state



istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

- Egress node: remove state from packet's header



istoica@cs.cmu.edu

Outline

- Problem formulation
- Eliminate **per-flow** state on the data path
- Eliminate **per-flow** state on the control path
- Experimental results

istoica@cs.cmu.edu

Problem

- **Provide:**
 - Unicast Intserv guaranteed service semantic
 - Diffserv like scalability

istoica@cs.cmu.edu

Reference Network

- **Data path: each router implements Jitter-Virtual Clock discipline**
- **Control path: each router performs per flow admission control**

istoica@cs.cmu.edu

Outline

- Problem formulation
- Eliminate **per-flow** state on the **data** path
- Eliminate **per-flow** state on the **control** path
- Experimental results

istoica@cs.cmu.edu

Data Path

- Use DPS to approximate Jitter-Virtual Clock (Jitter-VC) with an algorithm, **Core-Jitter Virtual Clock (CJVC)**, that does not require core routers to maintain per flow state

istoica@cs.cmu.edu

Jitter Virtual Clock (Jitter-VC)

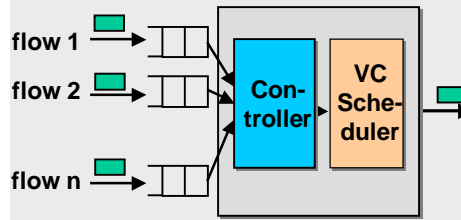
- **Components:**

- Delay-jitter rate-controller
- Virtual Clock scheduler

- **Algorithm:**

- Assign each packet a **eligible time** and **deadline**
- Keep packet in rate-controller until becomes eligible
- Schedule packets in increasing order of their deadline

- **Property: Jitter-VC guarantees that no packet misses its deadline**



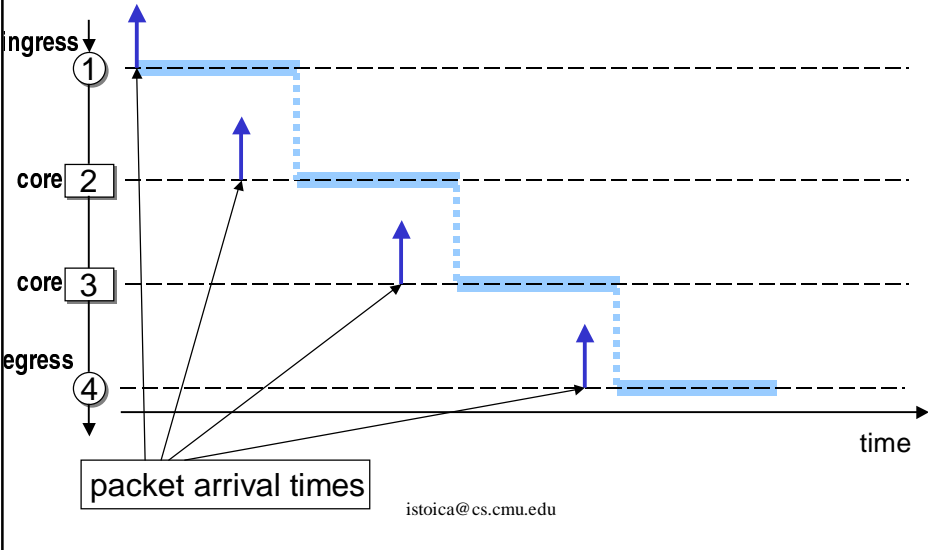
istoica@cs.cmu.edu

Jitter-VC: Eligible Time and Deadline

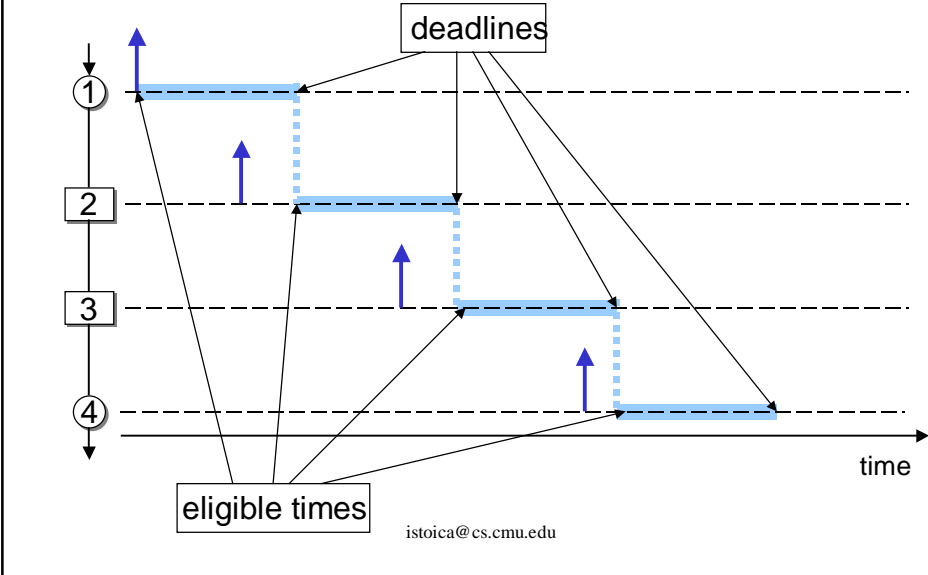
- **Intuition: eligible time and deadline of a packet belonging to a flow with reservation r \Leftrightarrow start and finish time of transmitting the packet in an **ideal** network in which flow has **dedicated** links of capacity r**
- **Eligible time = max of**
 - arrival time
 - packet's deadline at previous node + prop. delay
 - previous packet's deadline at current node
- **Deadline =**
eligible time + (packet length) / (flow reserved rate)

istoica@cs.cmu.edu

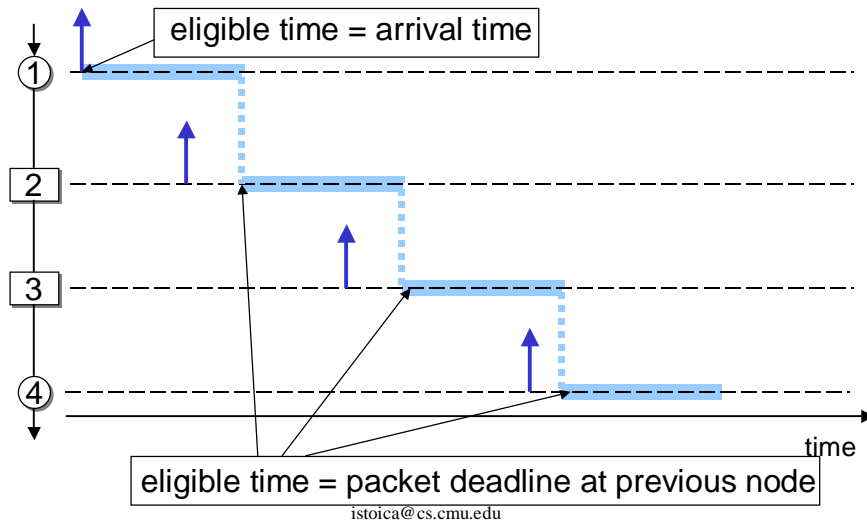
Jitter-VC: Example



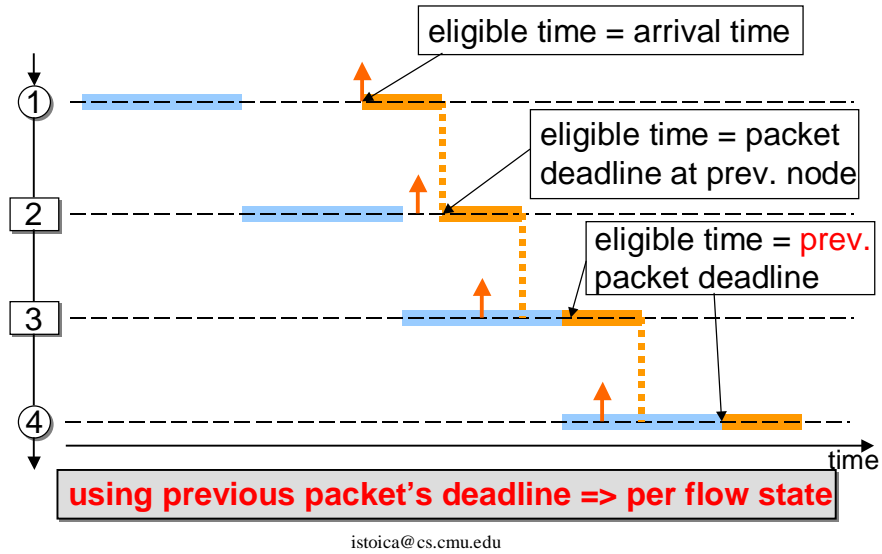
Jitter-VC: Example



Jitter-VC: Example



Jitter-VC: Example



Core Jitter Virtual Clock (CJVC)

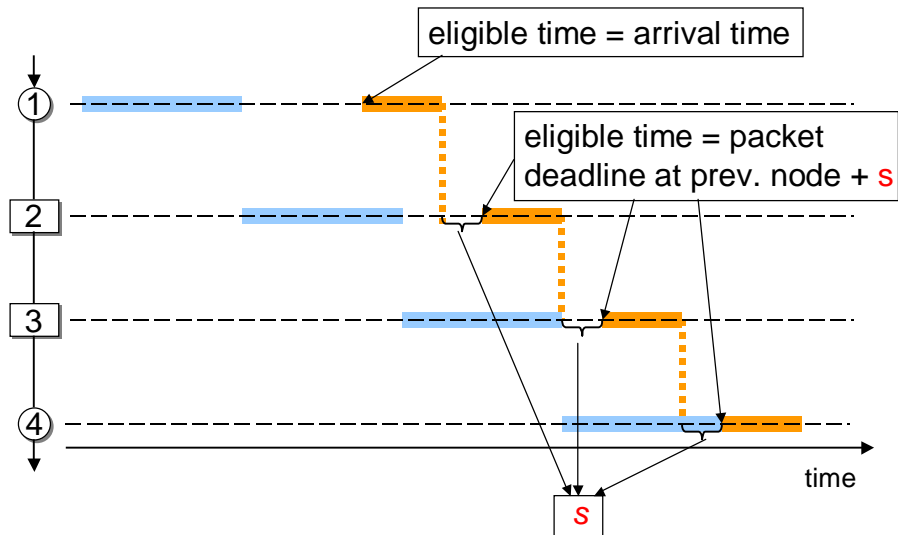
- **Goal: eliminate per flow state at core routers**
 - eliminate dependence on previous packet deadline
- **Solution: introduce a slack variable s , such that at **each** core node**

$$(\text{packet deadline at prev. node} + \text{prop. delay} + s) \geq (\text{deadline of previous packet})$$

- **Then compute eligible time as**
(packet deadline at prev. node + prop. delay + s)

istoica@cs.cmu.edu

CJVC



istoica@cs.cmu.edu

Properties of Slack Variable: s

- The eligible times and deadlines at the **last** hop are the **same** in both CJVC and Jitter-VC => CJVC and Jitter-VC provide the **same** end-to-end delay bounds
- Can be computed at ingress: depends on
 - lengths of previous and current packet
 - slack variable associated to previous packet
 - number of hops – computed at admission time

istoica@cs.cmu.edu

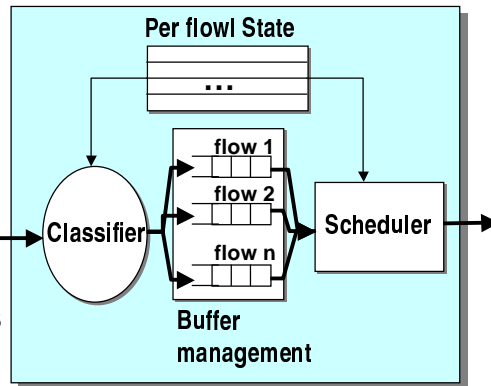
CJVC Algorithm

- Each packet carries in its header three variable
 - slack variable s (inserted by ingress)
 - flow's reserved rate (inserted by ingress)
 - ahead of schedule (inserted by previous node)
- Eligible time = (arrival time + **ahead of schedule** + s)
- Deadline = eligible time + (pkt. length)/(**flow rate**)
- NOTE:
 - ahead of schedule = deadline – departure time
 - (arrival time + ahead of schedule at **previous** node) = (deadline at **previous** node + propagation delay)

istoica@cs.cmu.edu

Jitter-VC: Core Router

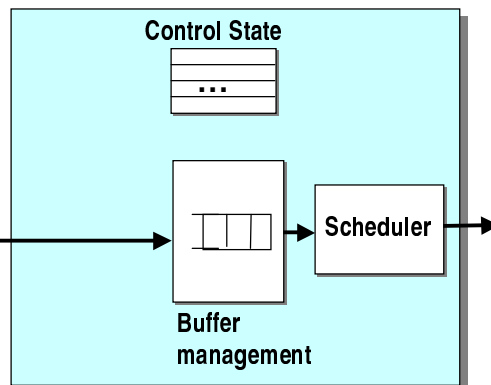
- Data path
 - Per-flow classification
 - Per-flow buffer management
 - Per-flow scheduling
- Control path
 - install and maintain per-flow state for data and control planes



istoica@cs.cmu.edu

CJVC: Core Router

- Data path
 - ~~Per-flow~~ classification
 - ~~Per-flow~~ buffer management
 - Per-**packet** scheduling
- Control path
 - Install and maintain per flow state for ~~data and~~ control plane



istoica@cs.cmu.edu

Outline

- Problem formulation
- Eliminate **per-flow** state on the **data** path
- Eliminate **per-flow** state on the **control** path
- Experimental results

istoica@cs.cmu.edu

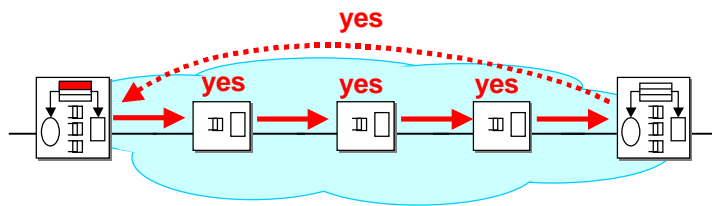
Control Path: Current Approaches for Admission Control

- **Distributed (e.g., RSVP, ATM UNI)**
 - ↑ Support for fine grained and short lived reservations
 - ↓ Not scalable
 - RSVP aggregation still requires $O(N^2)$ state, where N is number of edge nodes
 - ↓ Robustness hard to achieve
 - hard-state solutions – complex, less robust
 - soft-state solutions – higher overhead, weaker semantic
- **Centralized (e.g., Bandwidth Broker)**
 - ↑ Simple to implement
 - ↓ Not scalable - appropriate only for long lived flows

istoica@cs.cmu.edu

Admission Control with No Per Flow State

- Distributed approach
- Light-weight signaling protocol that does **not** require core nodes to maintain per-flow state



istoica@cs.cmu.edu

Per-hop Admission Control

- A node admits a reservation r , if $r \leq C - R$
 - C – output link capacity
 - R – aggregate reservation: $R = \sum_i r_i$
- Approach: maintain aggregate reservation R
- Problem: it requires **per flow** state to handle partial reservation failures and message loss

istoica@cs.cmu.edu

Solution

1. Estimate aggregate reservation R_{est}
2. Account for approximations and compute an upper bound R_{bound} , i.e., $R_{bound} \geq R$
3. Use R_{bound} , instead of R , to perform admission control, i.e., admit a reservation r if $r \leq C - R_{bound}$

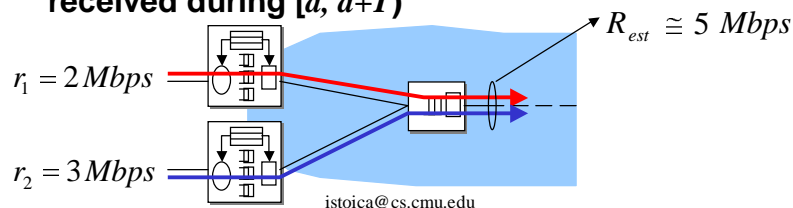
istoica@cs.cmu.edu

Estimating Aggregate Reservation (R_{est})

- **Observation:** If all flows were **sending** at their **reserved** rates, computing R_{est} is trivial:
 - just measure the traffic throughput, e.g.,

$$R_{est} = \frac{\sum_{i \in S(a, a+T)} length(i)}{T}$$

where $S(a, a+T)$ contains **all** packets of **all** flows received during $[a, a+T)$



Virtual Length

- **Problem:** What if flows do **not** send at their reserved rates ?

istoica@cs.cmu.edu

Virtual Length

- **Problem:** What if flows do **not** send at their reserved rates ?
- **Solution:** Associate to each packet a **virtual length**

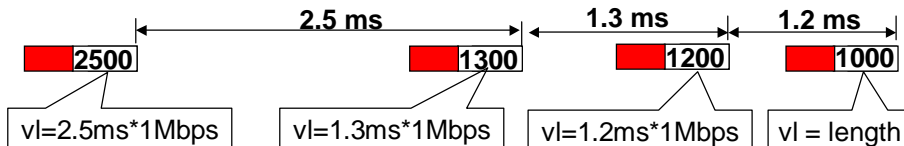
$$\text{virtualLength} = r \times (\text{crt_time} - \text{prev_time})$$

- r – flow reserved rate
- crt_time – transmission time of current packet
- prev_time – transmission time of previous packet
- Then, use **virtual** lengths instead of **actual** packet lengths in computing R_{est}

istoica@cs.cmu.edu

Example

- Assume a flow with reservation $r = 1$ Mbps sending 1000 bits packets



- Note:** If lengths of all packets of a flow were equal to their virtual lengths, the flow sends at its reserved rate!

istoica@cs.cmu.edu

Estimating Aggregate Reservation (R_{est})

- Use Dynamic Packet State (DPS)
- Ingress node: upon each packet departure computes the **virtual length** and inserts it in the packet header
- Core node: Estimate R_{est} on each output link as

$$R_{est} = \frac{\sum_{i \in S(a, a+T)} \text{virtualLength}(i)}{T}$$

- where $S(a, a+T)$ contains of **all** packets of **all** flows received during $[a, a+T)$

istoica@cs.cmu.edu

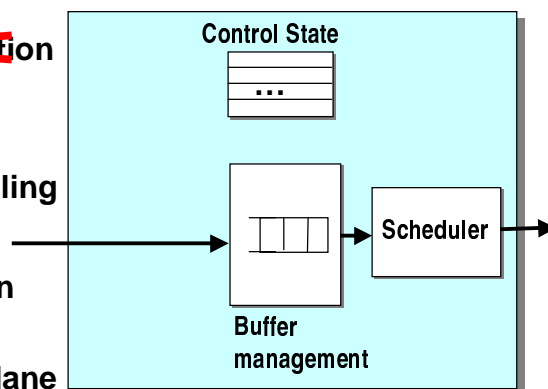
Aggregate Reservation Estimation: Discussion

- The estimation algorithm is robust in presence of control message loss and duplication
 - their effect is “forgotten” after one time interval
- Overestimation error typically around 10%-20%
- If no packet of a flow departs during a predefined interval (i.e., maximum inter-departure time), ingress node generates a dummy packet

istoica@cs.cmu.edu

Core Router

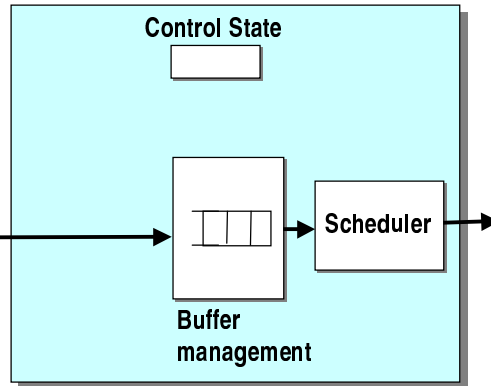
- Data path
 - ~~Per-flow classification~~
 - ~~Per-flow~~ buffer management
 - Per-**packet** scheduling
- Control path
 - Install and maintain per flow state for ~~data and~~ control plane



istoica@cs.cmu.edu

Core Router

- Data path
 - ~~Per-flow classification~~
 - ~~Per-flow~~ buffer management
 - Per-**packet** scheduling
- Control path
 - ~~Install and maintain per-flow state for data and control plane~~



no need to maintain consistency of per-flow state

istoica@cs.cmu.edu

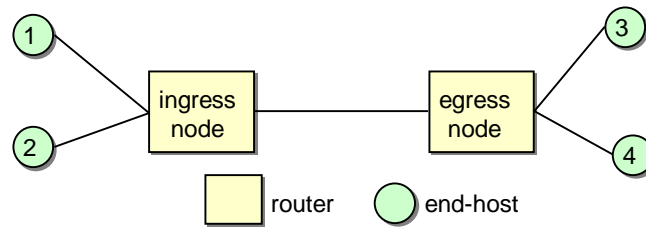
Implementation

- Problem: Where to insert the state ?
- Possible solutions:
 - between link layer and network layer headers
 - as an IP option
 - find room in IP header

istoica@cs.cmu.edu

Experiments

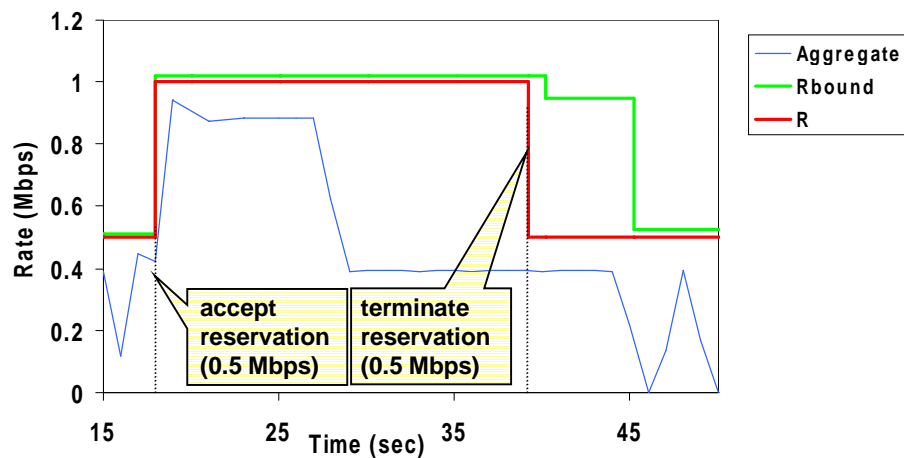
- **Darwin testbed**
 - FreeBSD 2.2.6
 - use 17 bits in IP header to encode state
 - 266 and 300 MHz Pentium II
 - 100 Mbps Ethernet



istoica@cs.cmu.edu

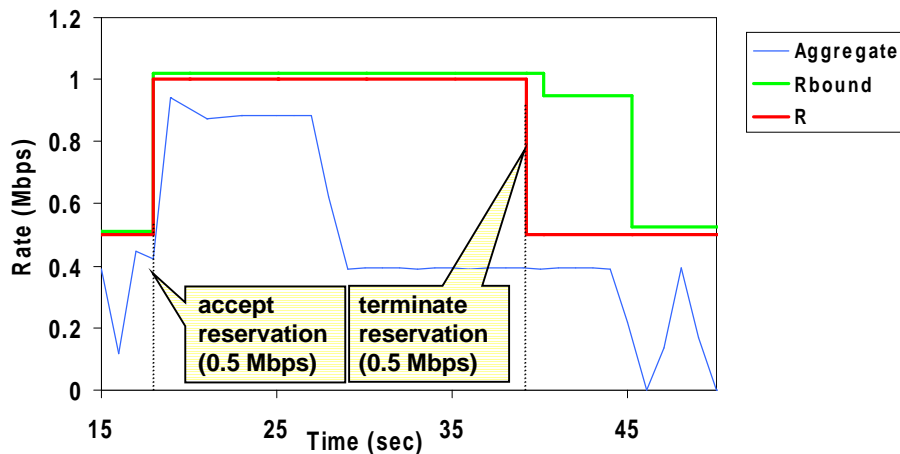
Aggregate Reservation Computation

- 0.5 Mbps reservation active during entire interval
- 0.5 Mbps reservation starting at 18 sec; ending at 39 sec



Aggregate Reservation Computation

- 0.5 Mbps reservation active during entire interval
- 0.5 Mbps reservation starting at 18 sec; ending at 39 sec



Conclusions

- Provide Intserv guaranteed service semantic without core nodes maintaining per-flow state
- Dynamic Packet State - a powerful technique that can be used to design algorithms that emulate/approximate the behavior of a broad class of “stateful” networks
 - other examples: Core-Stateless Fair Queueing
- Using DPS in context of Diffserv significantly increases flexibility and capabilities of services that can be implemented in this architecture

istoica@cs.cmu.edu