QoS Services with Dynamic Packet State

Ion Stoica

Carnegie Mellon University

(joint work with Hui Zhang and Scott Shenker)

Today's Internet

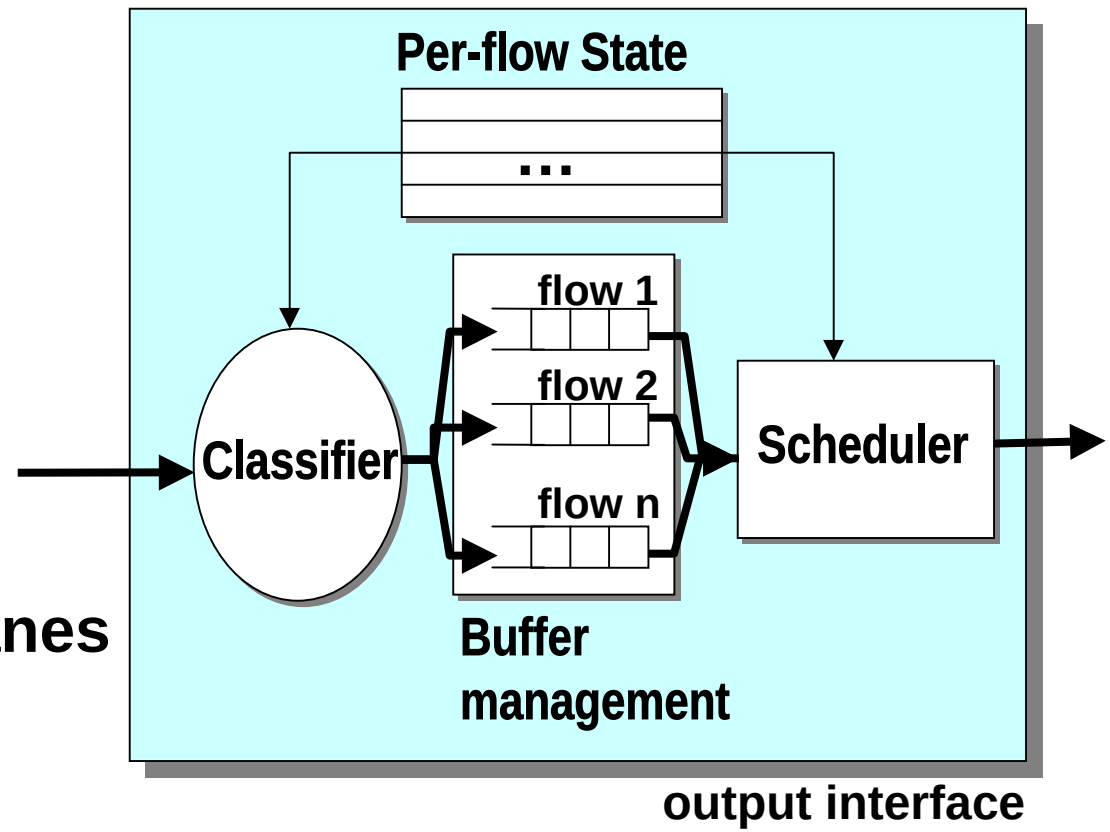
- **Service: best-effort datagram delivery**
- **Architecture: "stateless" routers**
  - **excepting routing state, routers do not maintain any fine grained state about traffic**
- **Properties**
  - **scalable**
  - **robust**

Trends

- **Deploy more sophisticated services, e.g., traffic management, Quality of Service (QoS)**
- **Two types 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**

Stateful Solutions: Router Complexity

- **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**



Stateless vs. Stateful

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

Question

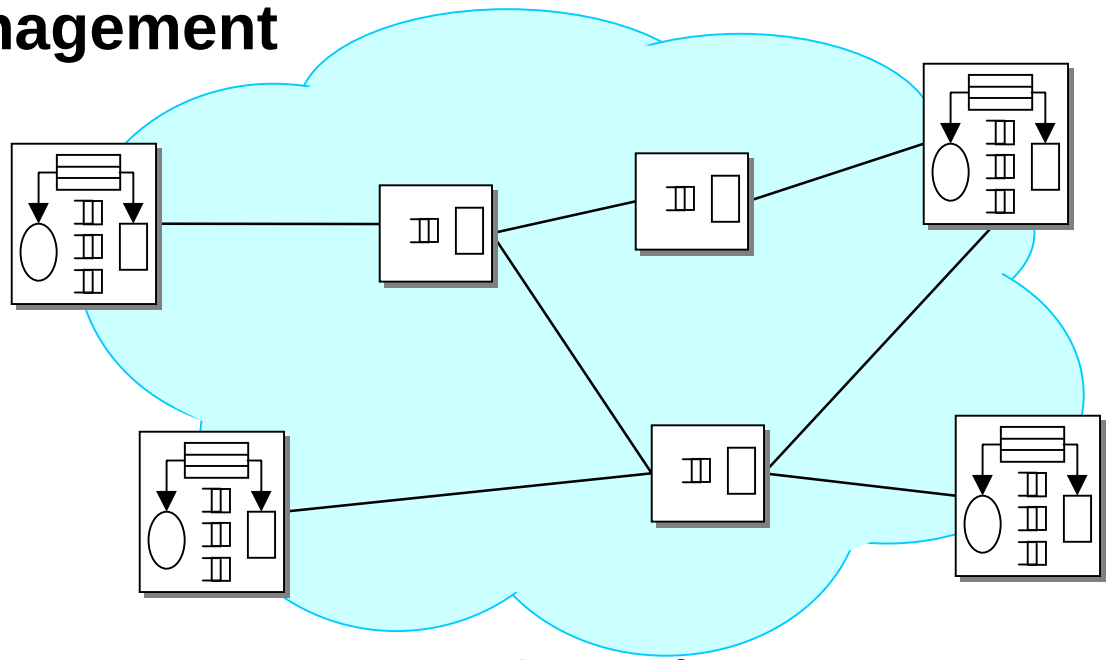
- **Can we achieve the best of two worlds, i.e., provide services implemented by stateful networks while maintaining advantages of stateless architectures?**

Answer

- **Yes, at least in some interesting cases:**
  - **Per-flow guaranteed services [SIGCOMM'99]**
  - **Fair Queueing approximation [SIGCOMM'98]**
  - **large spatial service granularity [NOSSDAV'98]**

Scalable Core (SCORE)

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

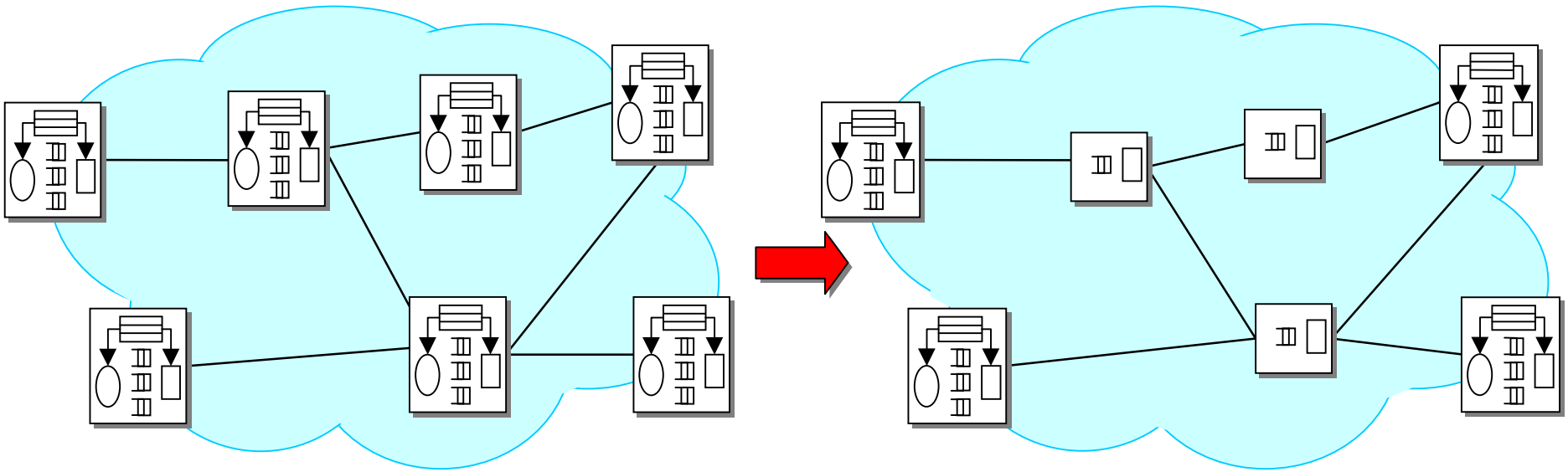


istoica@cs.cmu.edu

The Approach

- **Define a reference stateful network that implements the desired service**

2. **Emulate the functionality of the reference network in a SCORE network**



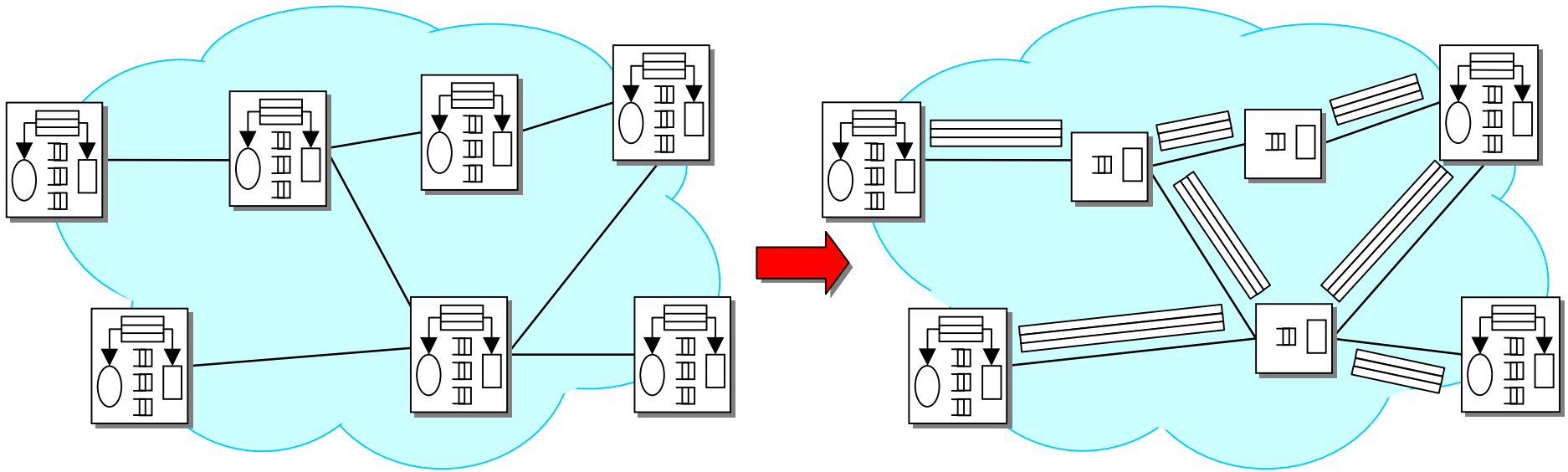
Reference Stateful Network

SCORE Network

istoica@cs.cmu.edu

The Idea

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



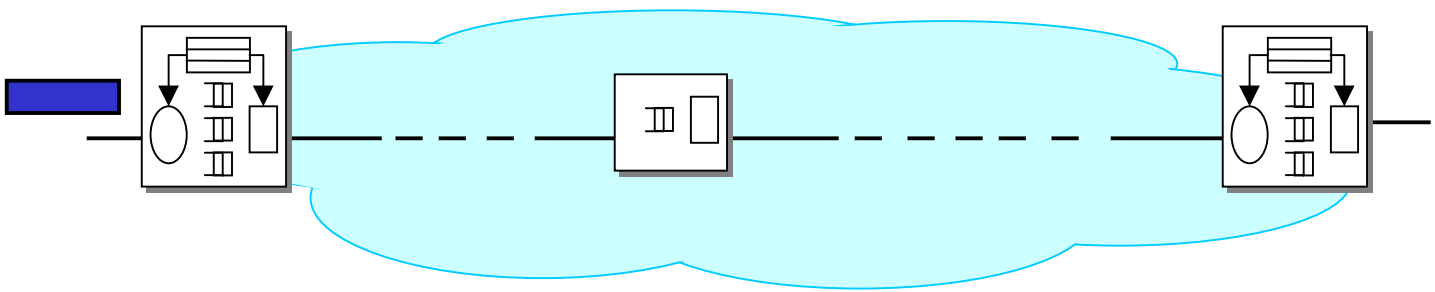
Reference Stateful Network

SCORE Network

istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

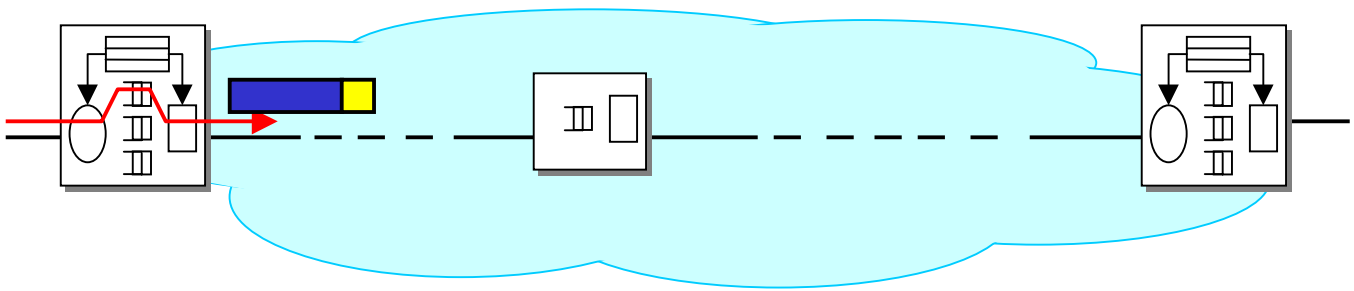
- **Ingress node: compute and insert flow state in packet's header**



istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

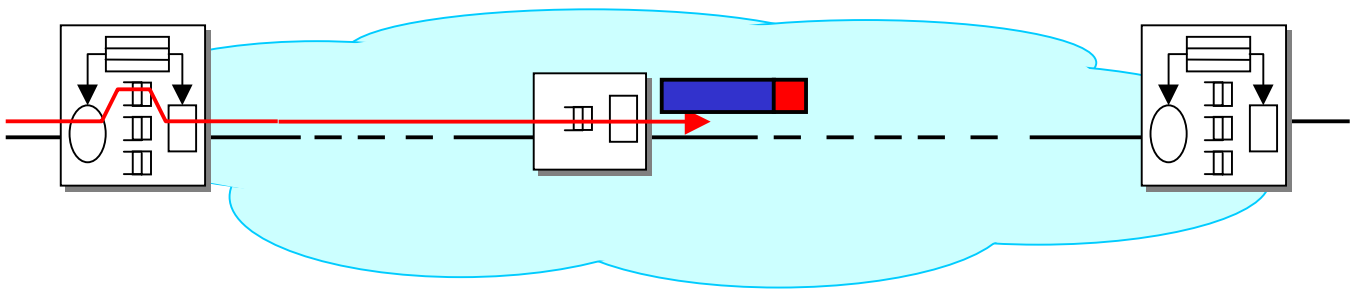
- **Ingress node: compute and insert flow state in packet's header**



istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

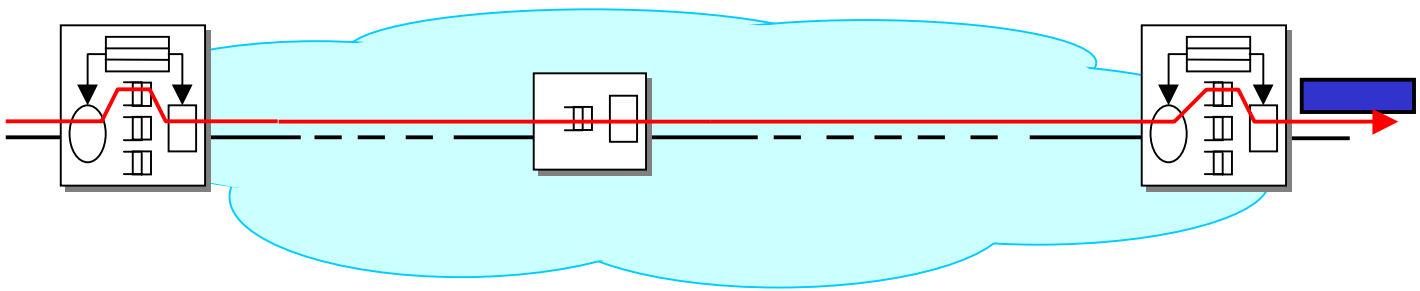
- **Core node:**
  - **process packet based on state it carries and node's state**
  - **update both packet and node's state**



istoica@cs.cmu.edu

The Technique: Dynamic Packet State (DPS)

- **Egress node: remove state from packet's header**



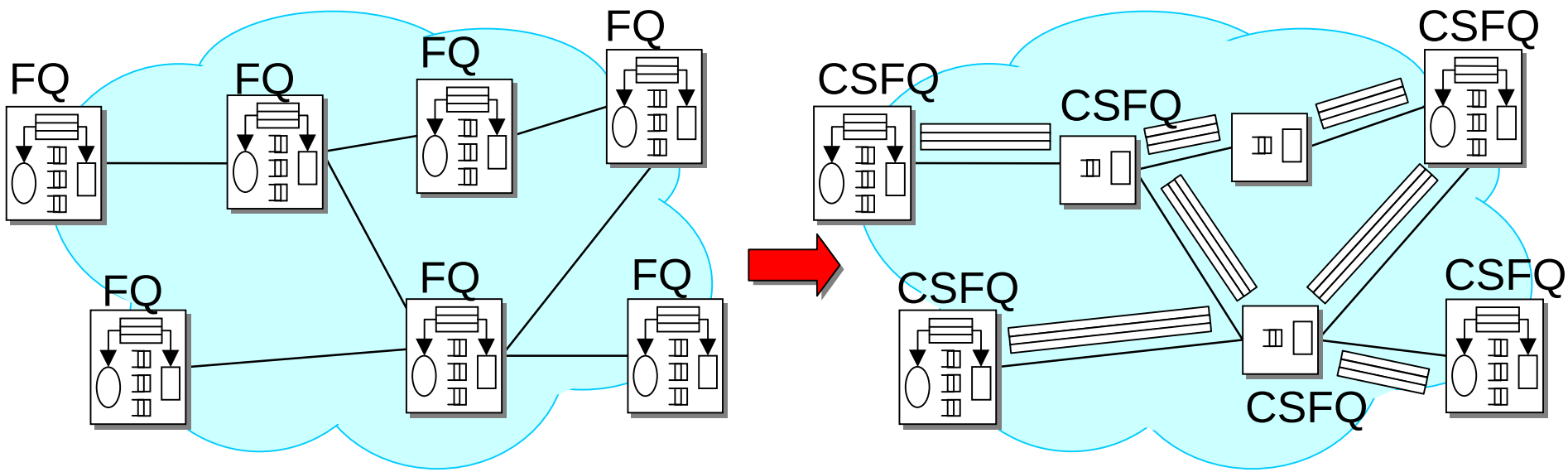
istoica@cs.cmu.edu

Examples

- **Support for congestion control**
- **Per flow guaranteed services**

Core-*Stateless* Fair Queueing (CSFQ)

- **Approximate functionality of a network in which every node performs Fair Queueing (FQ)**



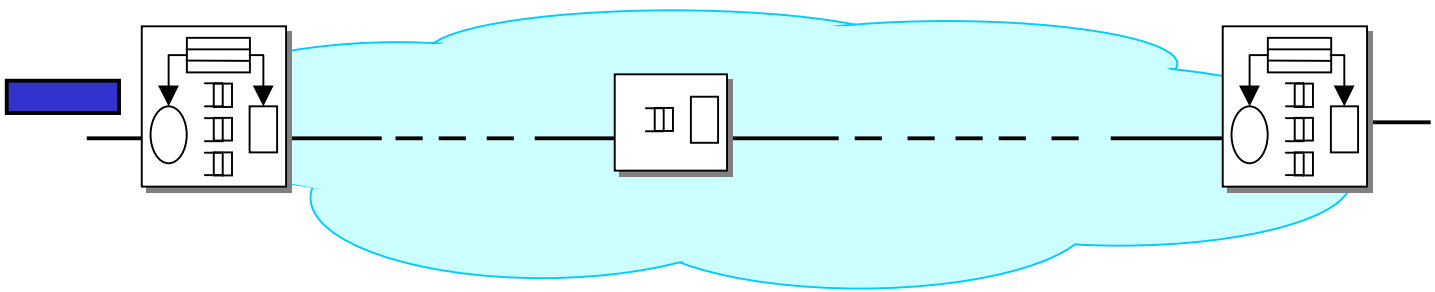
Reference Stateful Network

SCORE Network

istoica@cs.cmu.edu

Algorithm Outline

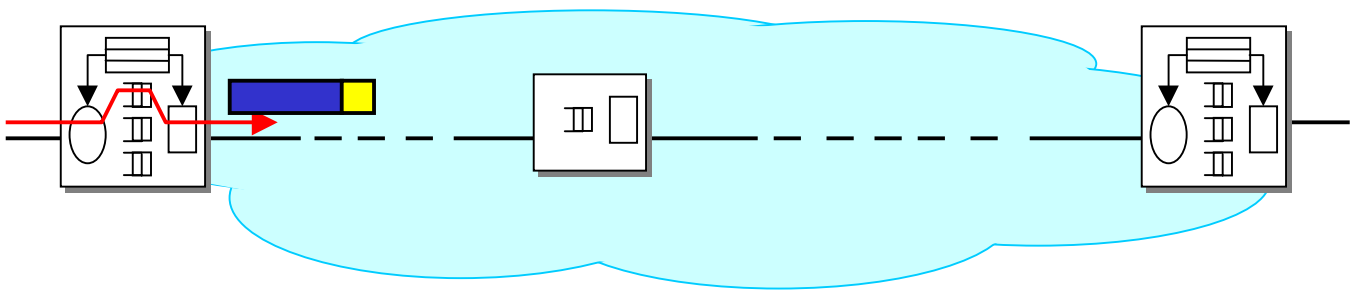
- **Ingress nodes: estimate rate *r* for each flow and insert it in the packets' headers**



istoica@cs.cmu.edu

Algorithm Outline

- **Ingress nodes: estimate rate *r* for each flow and insert it in the packets' headers**



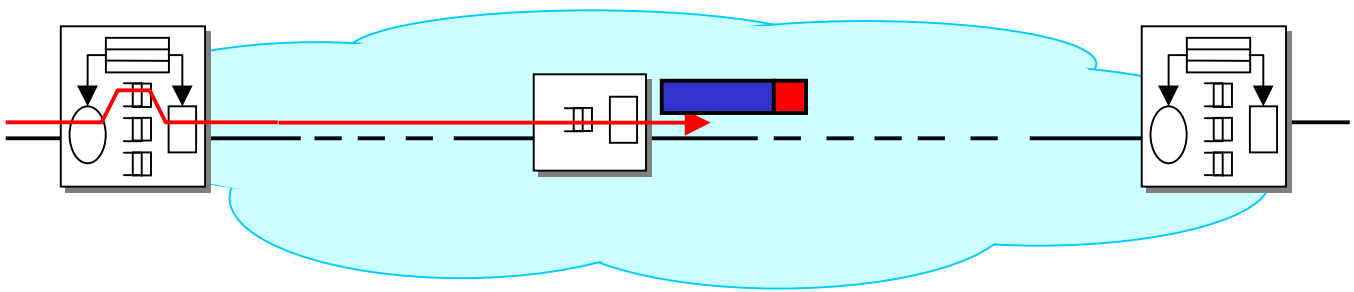
istoica@cs.cmu.edu

Algorithm Outline

- **Core node:**
  - **Compute fair rate *f* on the output link**
  - **Enqueue packet with probability**

$$P = \min(1, f / r)$$

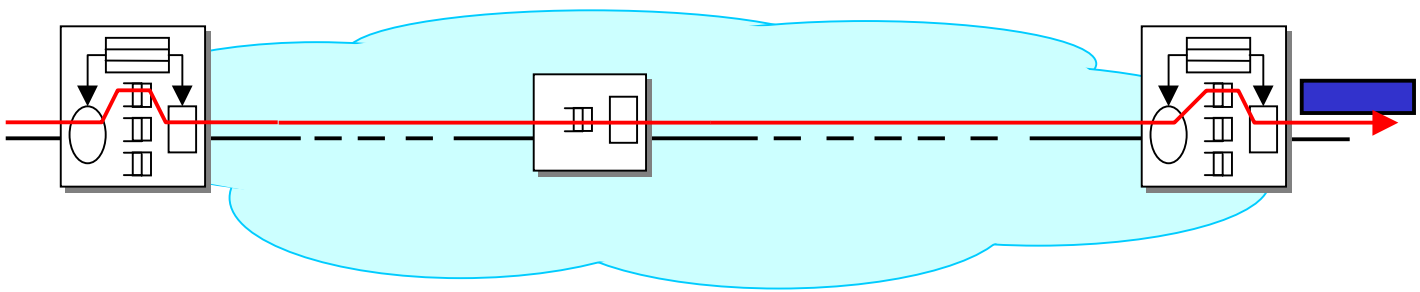
  - **Update packet label to $r = \min(r, f)$**



istoica@cs.cmu.edu

Algorithm Outline

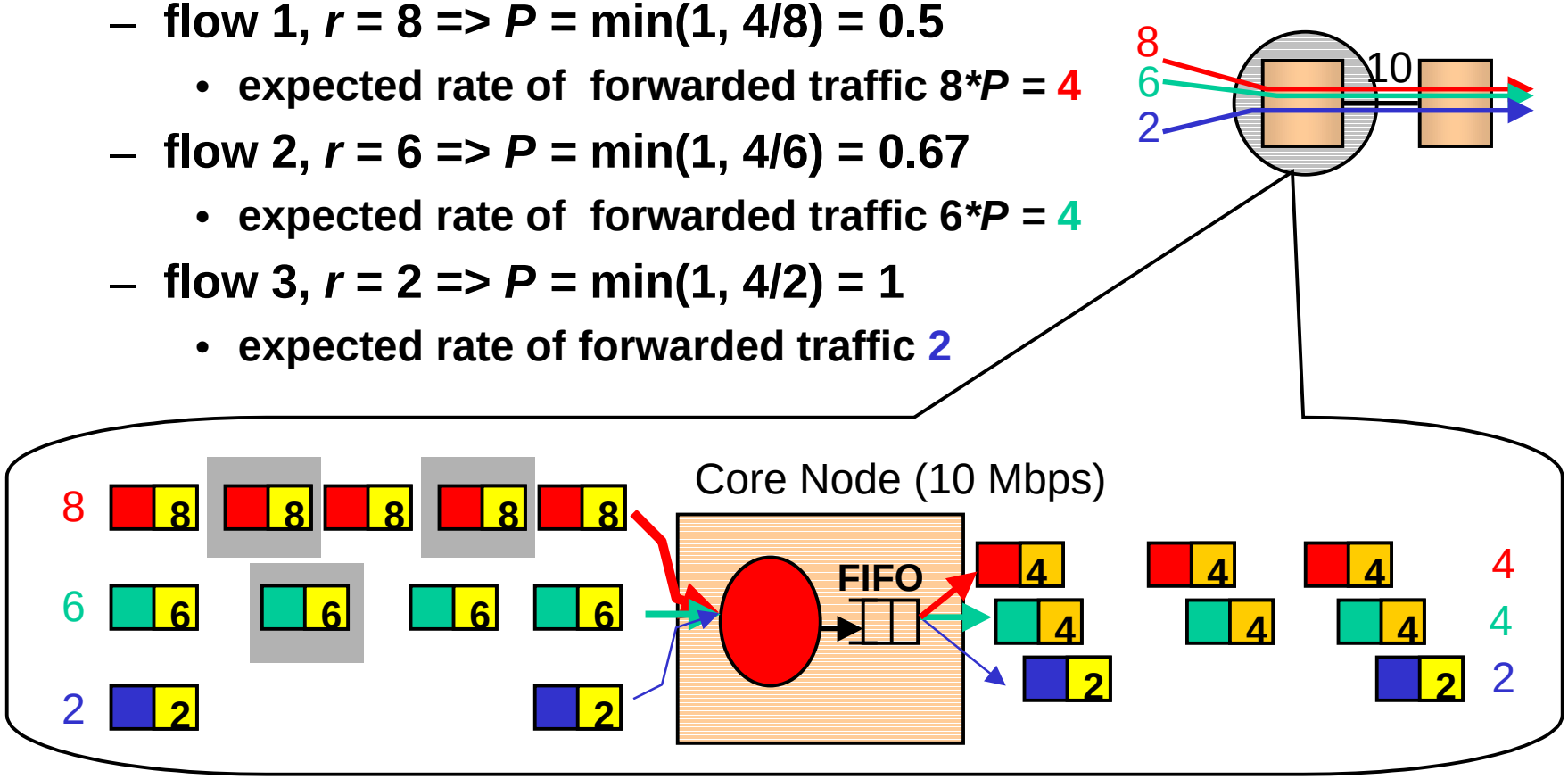
- **Egress node: remove state from packet's header**



istoica@cs.cmu.edu

Example: CSFQ Core Core

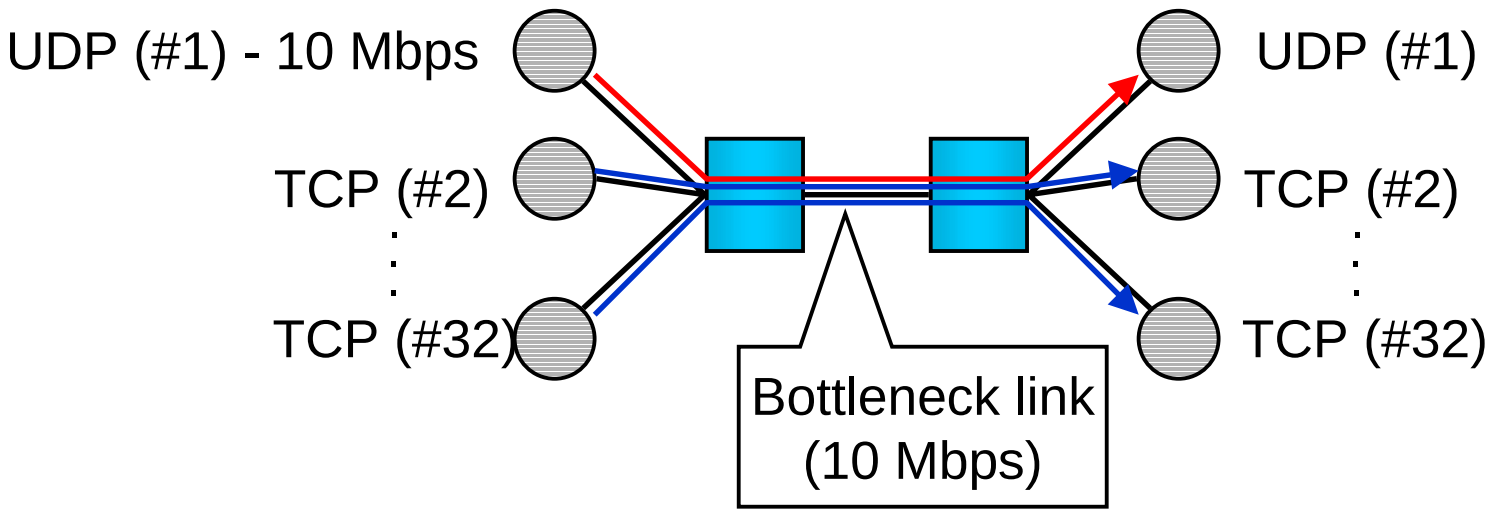
- **Assume estimated fair rate $f = 4$**
  - **flow 1, $r = 8 \Rightarrow P = \min(1, 4/8) = 0.5$**
    - **expected rate of forwarded traffic $8*P = 4$**
  - **flow 2, $r = 6 \Rightarrow P = \min(1, 4/6) = 0.67$**
    - **expected rate of forwarded traffic $6*P = 4$**
  - **flow 3, $r = 2 \Rightarrow P = \min(1, 4/2) = 1$**
    - **expected rate of forwarded traffic 2**



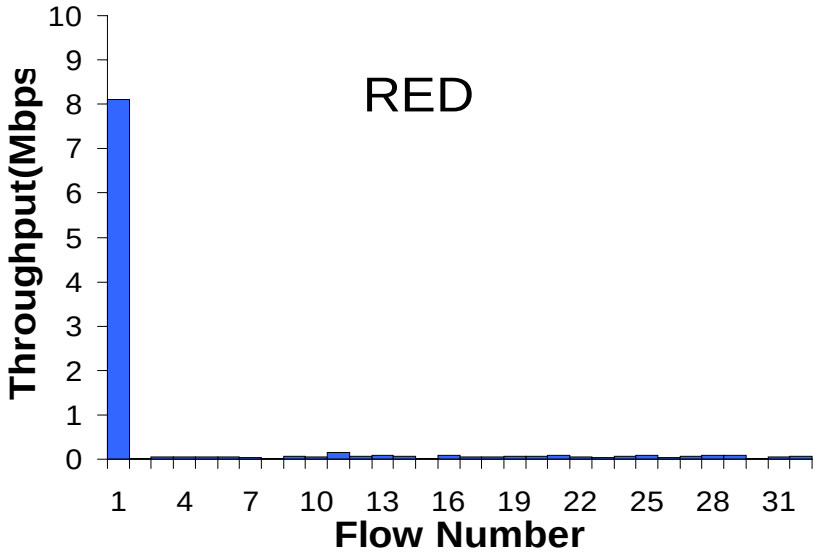
istoica@cs.cmu.edu

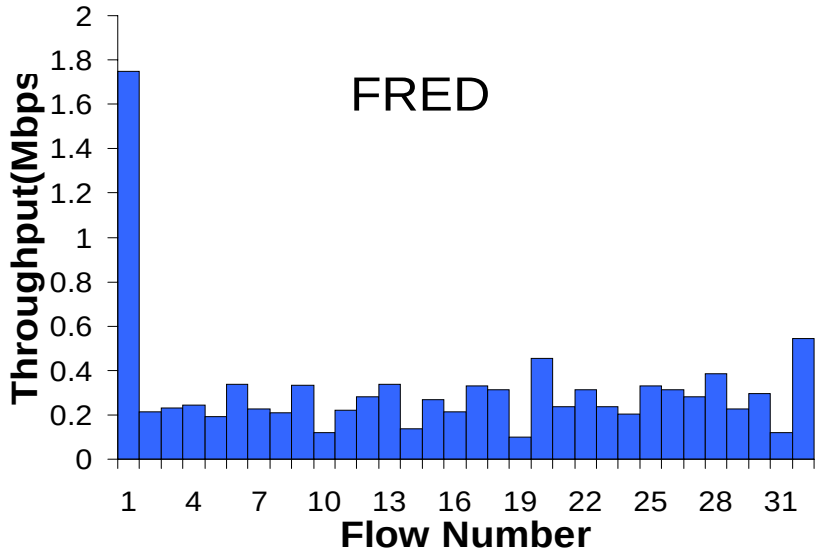
Simulation Results

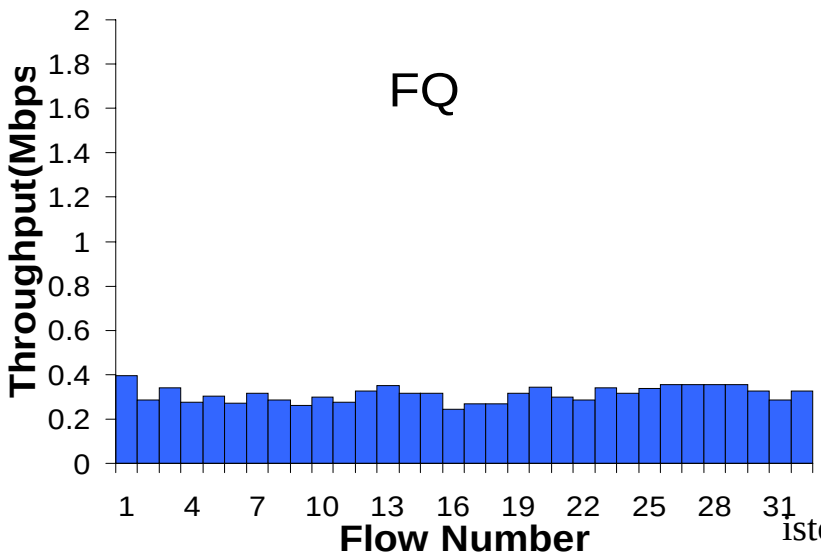
- **1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps link**
  - **fair rate 0.31 Mbps**

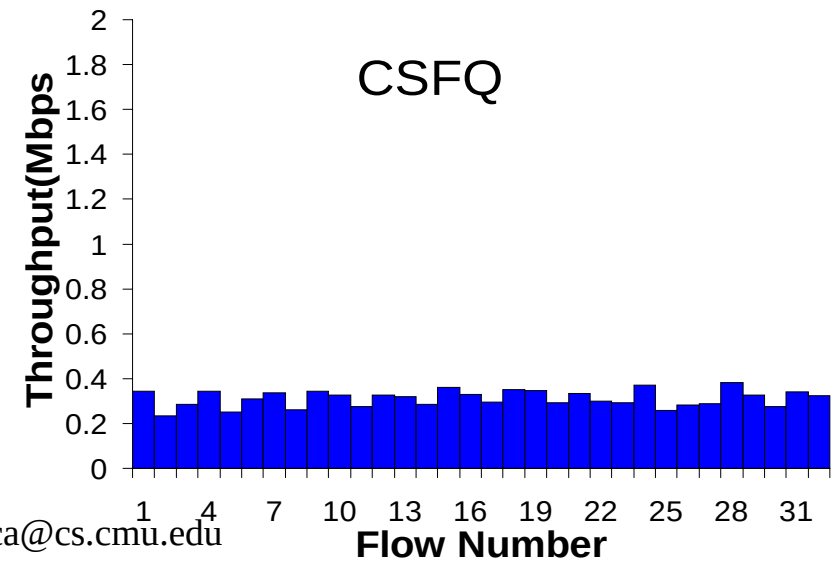


Throughput of TCP and UDP Flows with RED, FRED, FQ, CSFQ









istoica@cs.cmu.edu

Results

- **Complexity**
  - **n – number of (active) flows**

	FIFO/RED	FRED	FQ	CSFQ
State	O(1)	O(n)	O(n)	O(n) - edge O(1) - core
Time	O(1)	O(1)	O(log n)	O(1)

- **Accuracy**
  - **the extra service that a flow can receive in CSFQ as compared to FQ is bounded**

istoica@cs.cmu.edu

Examples

- **Support for congestion control**
- **Per flow guaranteed services**

Guaranteed Services

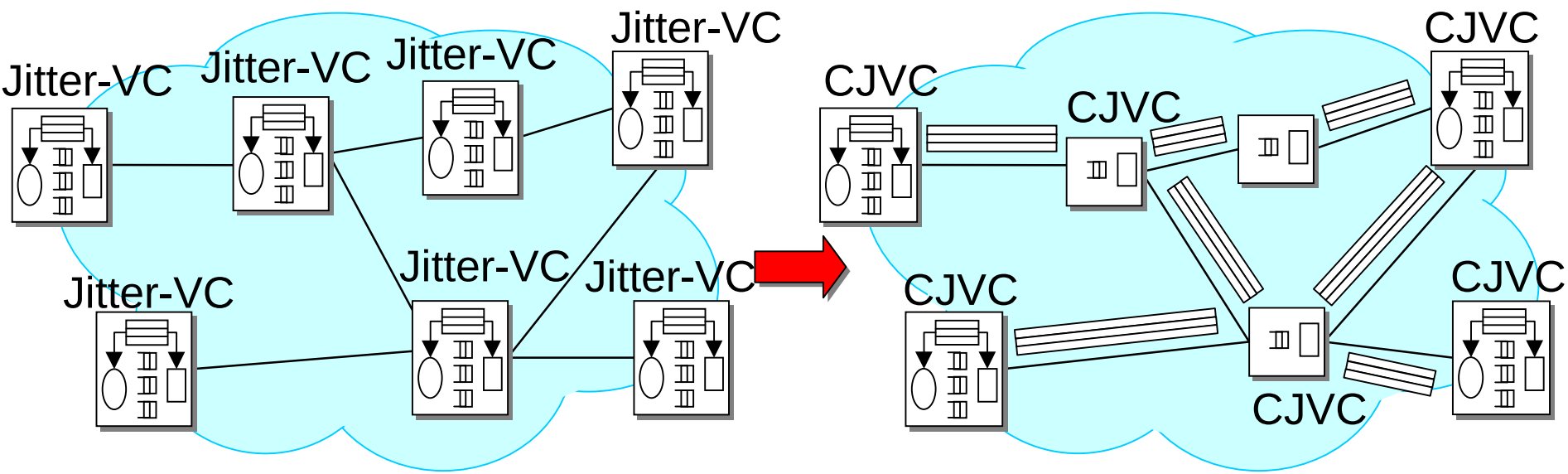
- **Intserv:**
  - **provide per flow bandwidth and delay guarantees, and achieve high resource utilization**
  - **support for fined grained and short-lived reservations**
  - **not scalable**
- **Diffserv (Premium Service):**
  - **Scalable (on data path)**
  - **cannot provide low delay guarantees 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**

Goal

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

Solution

- **Data path: approximate Jitter-Virtual Clock (Jitter-VC) with Core-Jitter Virtual Clock (CJVC)**
- **Control path: approximate distributed admission control**



Reference Stateful Network

SCORE Network

istoica@cs.cmu.edu

Theoretical Results

- **CJVC provides same end-to-end delay guarantees as Jitter-VC (and Weighted Fair Queueing)**
- **Admission control: provides semantic of a hard state protocol, but…**
  - **typically achieves only 80 % link utilization**

Implementation

- **Problem: Where to insert the state ?**
- **Possible solutions:**
  - **between link layer and network layer headers (e.g., MPLS)**
  - **as an IP option**
  - **find room in IP header**
- **Current implementation (FreeBSD 2.2.6): use 17 bits in IP header**
  - **4 bits in DS field (former TOS)**
  - **13 bits by reusing fragment offset**

Status

- **Working prototype in FreeBSD 2.2.6 that implements:**
  - **Core-Stateless Fair Queueing**
  - **Guaranteed services**
    - **data path – Core Jitter Virtual Clock**
    - **control path – distributed admission control**

Conclusions

- **Diffserv has serious limitations:**
  - **no flow protection**
  - **cannot provide guaranteed services and high resource utilization simultaneously**
  - **no scalable admission control architecture (e.g. Bandwidth Broker)**
- **DPS compatible with Diffserv: can greatly enhances the functionality while requiring minimal changes**
- **Let's do it in Qbone !**

istoica@cs.cmu.edu

More Information

http://www.cs.cmu.edu/~istoica/DPS

istoica@cs.cmu.edu