Some Issues on Internet Resource Management

Ping Pan
IBM T. J. Watson Research Center
pan@watson.ibm.com

Issues:

- **Do we need resource reservation?**
- **How to set it up?**
  - RSVP and its problems
  - Alternative: YESSIR
  - Comparative Analysis
- **How to enforce it efficiently?**
  - Generic model and its problems
  - Buffer management
  - Results
- **Future works:**
  - inter-domain reservation

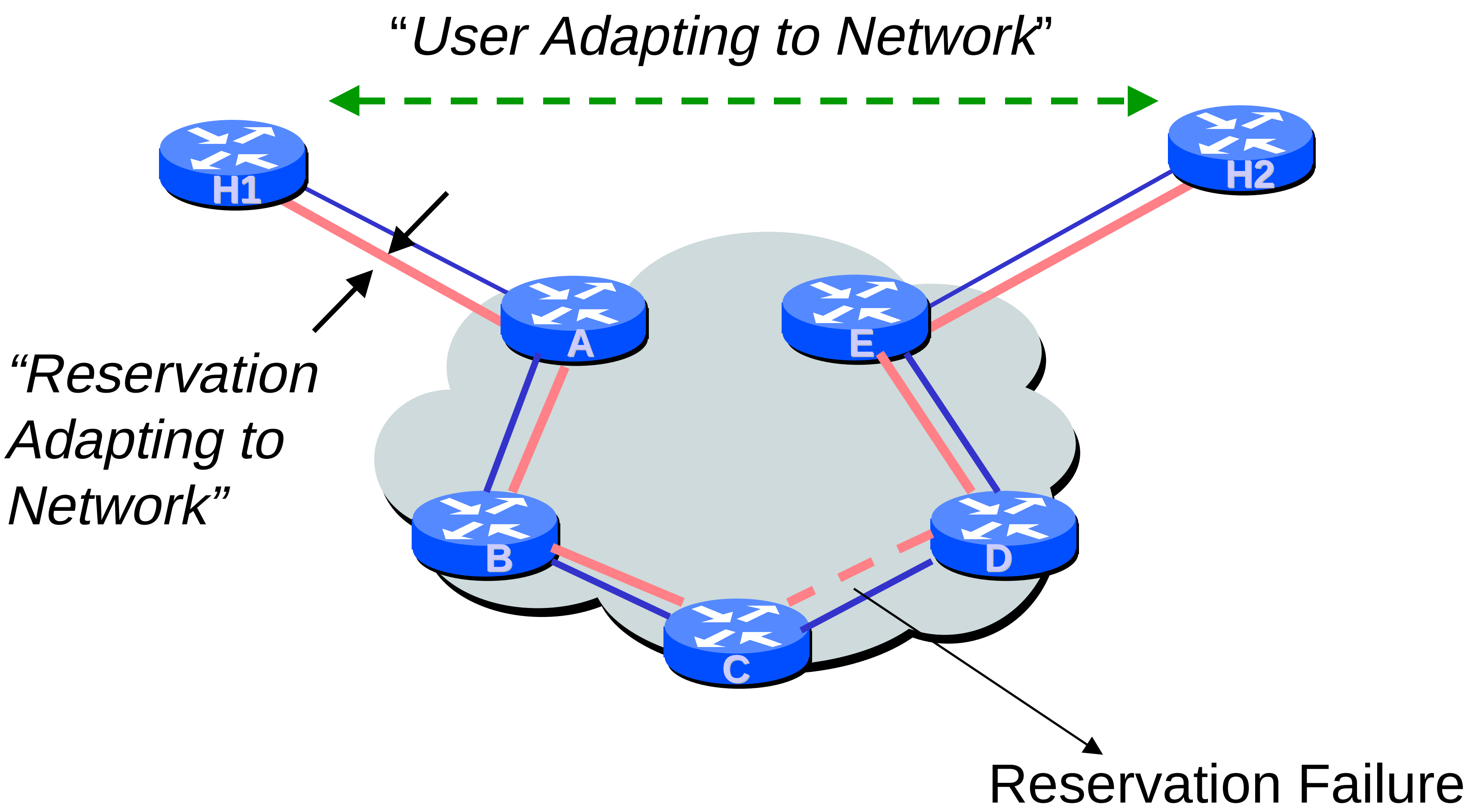
Do we need resource reservation?

No:

- Elastic traffic (ftp, email, stored a/v etc.).

Yes:

- If a network supports both real-time and elastic traffic, we ***must*** have reservation to provide traffic isolation.
  - Hard real-time (traditional voice traffic).
    - Strict (all-or-nothing) reservation.
  - Adaptive real-time (voice-over-IP).
    - Loose reservation, ...



***Claim*: For real-time application, having end-to-end reservation (even partial) is better than none!**

… What is partial reservation?

The entire path is not required to have reservation on every link. For links without reservation, traffic is treated as best-effort.

Immediate Advantages:

- low blocking rate;
- less added complexity to the network.

New Challenges:

- fragmentation effect;
- end-to-end reservation adaptation.

Issues:

- **Do we need resource reservation?**
- **How to set it up?**
  - RSVP and its problems
  - Alternative: YESSIR
  - Comparative Analysis
- **How to enforce it efficiently?**
  - Common model and its problems
  - Buffer management
  - Results
- **Future works:**
  - inter-domain reservation

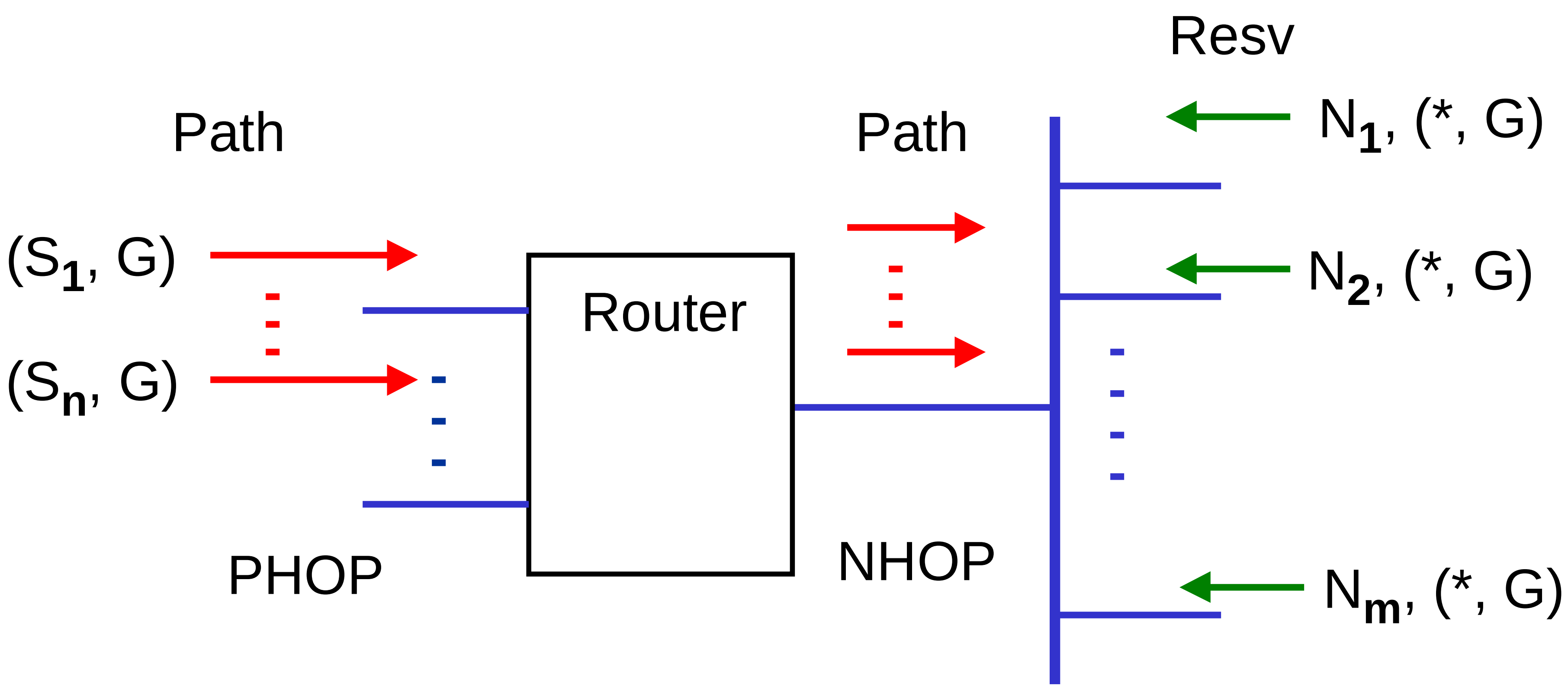
How to set up end-to-end reservation?

Background:

- RTP:
  - embedded in applications;
  - in-band signaling (RTCP for control).
- RSVP:
  - designed for end systems;
  - out-band signaling;
  - receiver initiation;
  - all-or-nothing reservation.
- IP Alert Option:
  - enable router to detect upper-layer information at IP level.

Problems with RSVP:

Implementation Complexity: *receiver-driven* (wildcard style, multicast, NBMA)



… Problems with RSVP

State scaling *(may not apply in newer routers)*

- Edge router memory: ~ 8-16 Mbytes
- Average route table size: ~ 0.5 Mbytes
- Average buffer space/port: ~ 0.5-1 Mbytes
- RSVP (1 sender,1 receiver): ~ 500 bytes
  - OC-3 (or 2400 56kb flows): 1.2 Mbytes !

… Problems with RSVP

CPU consumption for “soft-state”

- Router: generating message is costly
  - – memory access is expensive;
  - – locking interrupt to build packets is costly at forwarder;
  - – ~ **3-4** control messages/second/interface.
- RSVP refresh:
  - – generate packets periodically;
  - – OC-3: ~**150** messages/second/interface.

Let's solve the problem in two levels!

- End-to-End:
  – make it sender-initiated to simplify the process;
  – completely application-driven;
  – allow partial reservation;
  – allow fine granularity reservation;
  – make end systems to be adaptive to network resource condition
- Network-to-Network:
  – solve scaling problems by dealing with aggregated resource and routes.

YESSIR

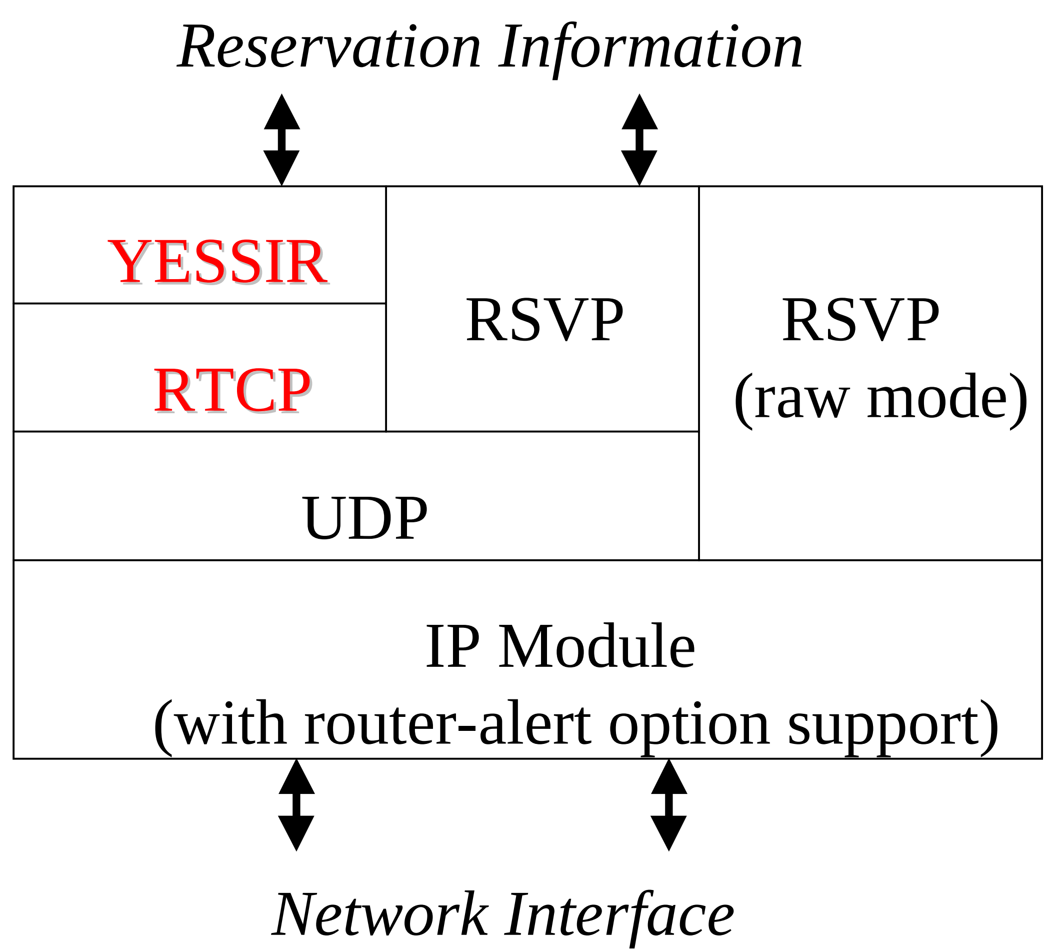
- A solution for **end-to-end** reservation.
- Embedded in applications (with **RTP**).
- Using **IP Alert Option** to notify routers to process the reservation requests.
- Use of **partial reservation** … network does not say “no” to a reservation request.

How does YESSIR work?

RTCP Sender-report messages contain a reservation request (YESSIR extension) that is initiated by the sender then processed at routers on their path.
If a router cannot grant the requested resource, the message is marked with a “failure” flag, and then forwarded to the next hop.

The receivers of a RTP session always communicate with the sender via RTCP Receiver-report messages periodically. So the sender can detect reservation conditions from RTCP RR, and dynamically adjust reservation accordingly.

Protocol relationship



IP Header with Router-Alert Option

UDP Header

RTCP message:

Sender Report:
- sender information
- detailed report for each source

YESSIR message:
- reservation command: active/passive
- reservation style, refresh interval
- reservation flow specification
- link resource collection
- reservation failure report

Profile-specific extensions

Measurement-based Reservation

YESSIR makes the use of the byte count and the timestamp fields in RTCP SR.

Initially, a router records the timestamp and the byte count of the first SR. After receiving the second SR, the router initiates a reservation request base on the difference in time stamps and byte counts.

The router updates reservation for each newly arrived SR message.

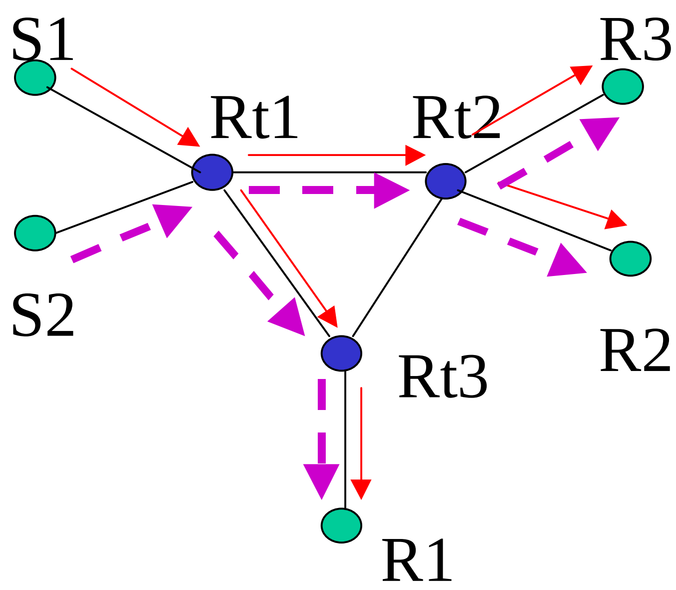
Partial Reservation

The entire path is not required to have reservation on every link. For links without reservation, traffic is treated as best-effort.

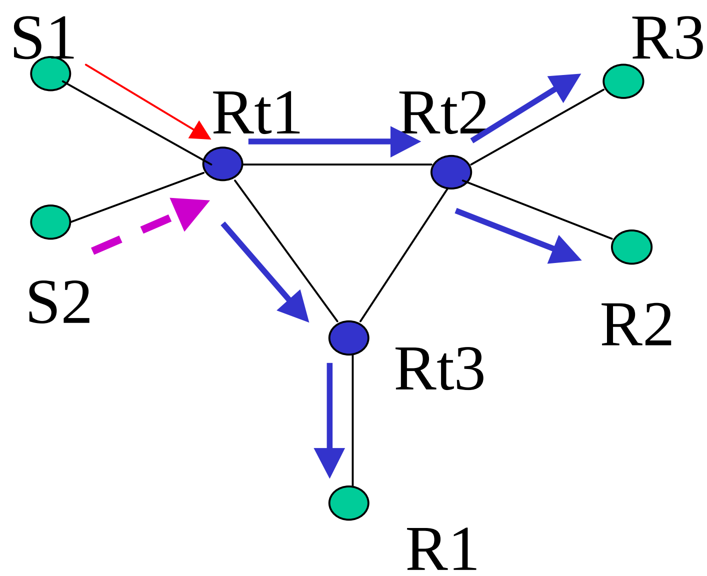
1. YESSIR is soft-state based. A flow will acquire a reservation at a link periodically anyway.

2. End users can always either put-up or quit from a session.

Reservation Styles

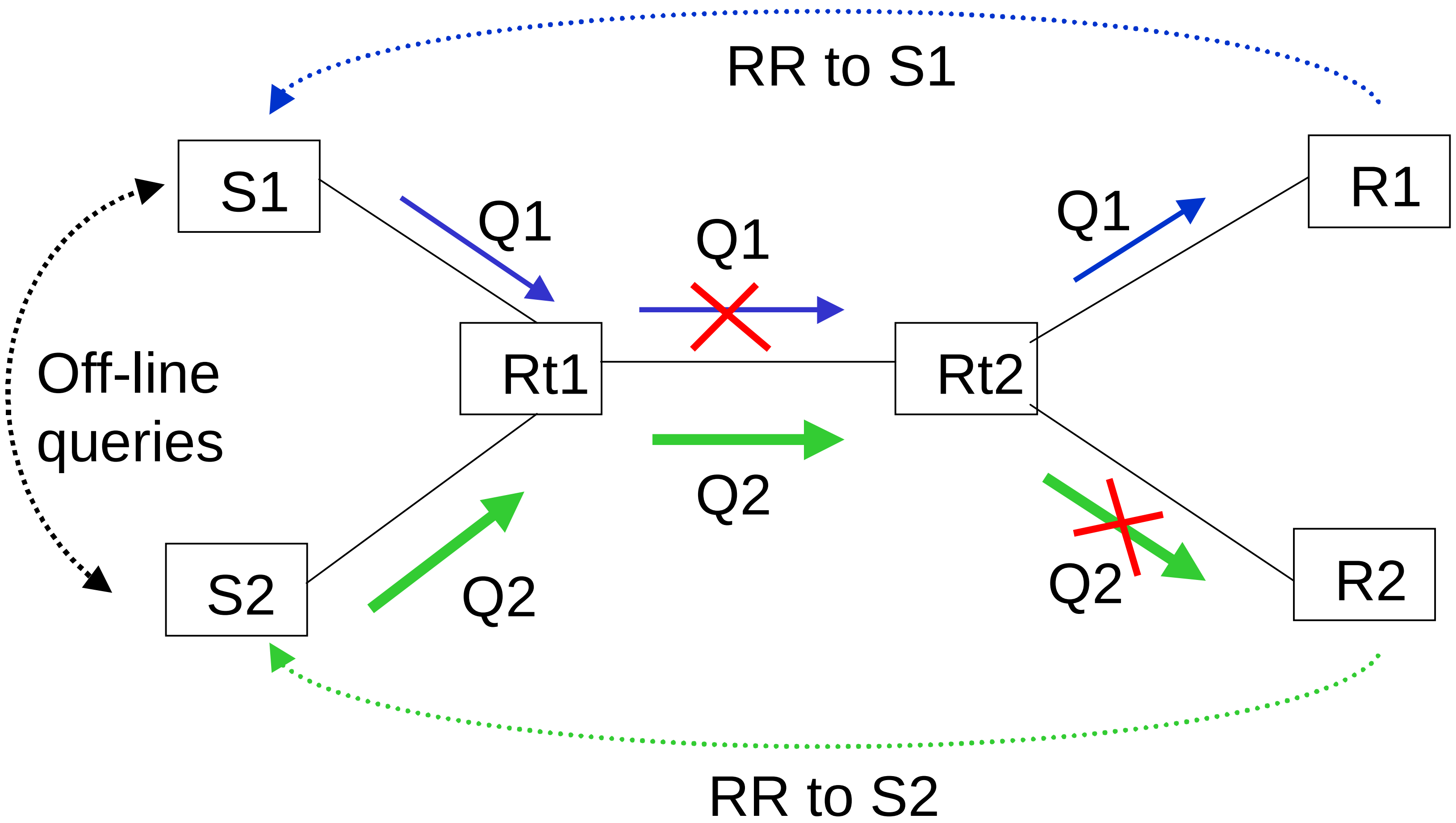


(a) *Distinct Reservation style*:
Reservations for S1are shown
as in solid line; S2, in dotted line.



(b) *Shared Reservation style*:
At Rt1, after flow merging between reservation for S1 (solid line) and S2 (dotted line), a single reservation (thicker line) is made to Rt2 and Rt3.

Solving "*Killer*" Reservation: end-to-end negotiation



How to handling errors inside the network?

Answer: Don't!

Why?

1. simple to implement… Supporting all RSVP error handling, … just too costly.

2. In high-speed LAN, reservation is not necessary.

3. There is no clean solution to support reservation on shared-media.

Other Features

- **Flexible flow specification:**
  - IntServ compliant (carrying Tspec, Rspec)
  - DiffServ compliant (carrying DS)
  - native RTP (carrying RTP Pay-load)
- **Ease of updating packet classifier:**
  - Take the advantage of RTP data and RTCP port coupling feature.
- **Network Resource Advertisement:**
  - Use OPWA scheme to gather network resource.
  - RTCP RR feeds results back to senders.

Algorithm (at router)

- Locate the reservation flow from YESSIR message.
- If found nothing, create a new flow.
- Query routing table and find egress interface(s).
- Make a reservation on egress interface(s) according to YESSIR flow specification.
- Update reservation information to the local table.
- Relay the message downstream.

Experiments

- Platform: IBM 2210 router (Motorola 68040 processor at bus speed of 32 MHz).

- Processing speed: read-out from timer register with 31.25 nsec resolution

- RSVP flows: Controlled load, fixed-filter

- YESSIR flows: Use RTP pay-load as flowspec

Results (RSVP Trigger messages)

	Time (usec)	% of Total
<u>PATH Processing:</u>		
PATH entry creation	410.51 +/- 7.51	37.12%
Router Query	40.61 +/- 1.84	3.67%
Send PATH downstream	283.16 +/- 3.85	25.61%
<u>RESV Processing:</u>		
RESV entry look-up	11.03 +/- 0.43	1.00%
Update reservation info	126.35 +/- 1.10	11.43%
Flow merging and forward	234.14 +/- 3.27	21.17%
<u>Single setup overhead</u>	**1,105.80 +/- 9.47**	**100%**

Results (YESSIR Trigger messages)

	Time (usec)	% of Total
YESSIR entry creation	41.40 +/- 1.24	11.61%
Router Query	38.43 +/- 2.00	10.77%
Update reservation info	23.33 +/- 0.38	6.54%
Forward YESSIR message	253.53 +/- 0.74	71.08%
Single setup overhead	**256.68 +/- 2.84**	**100%**

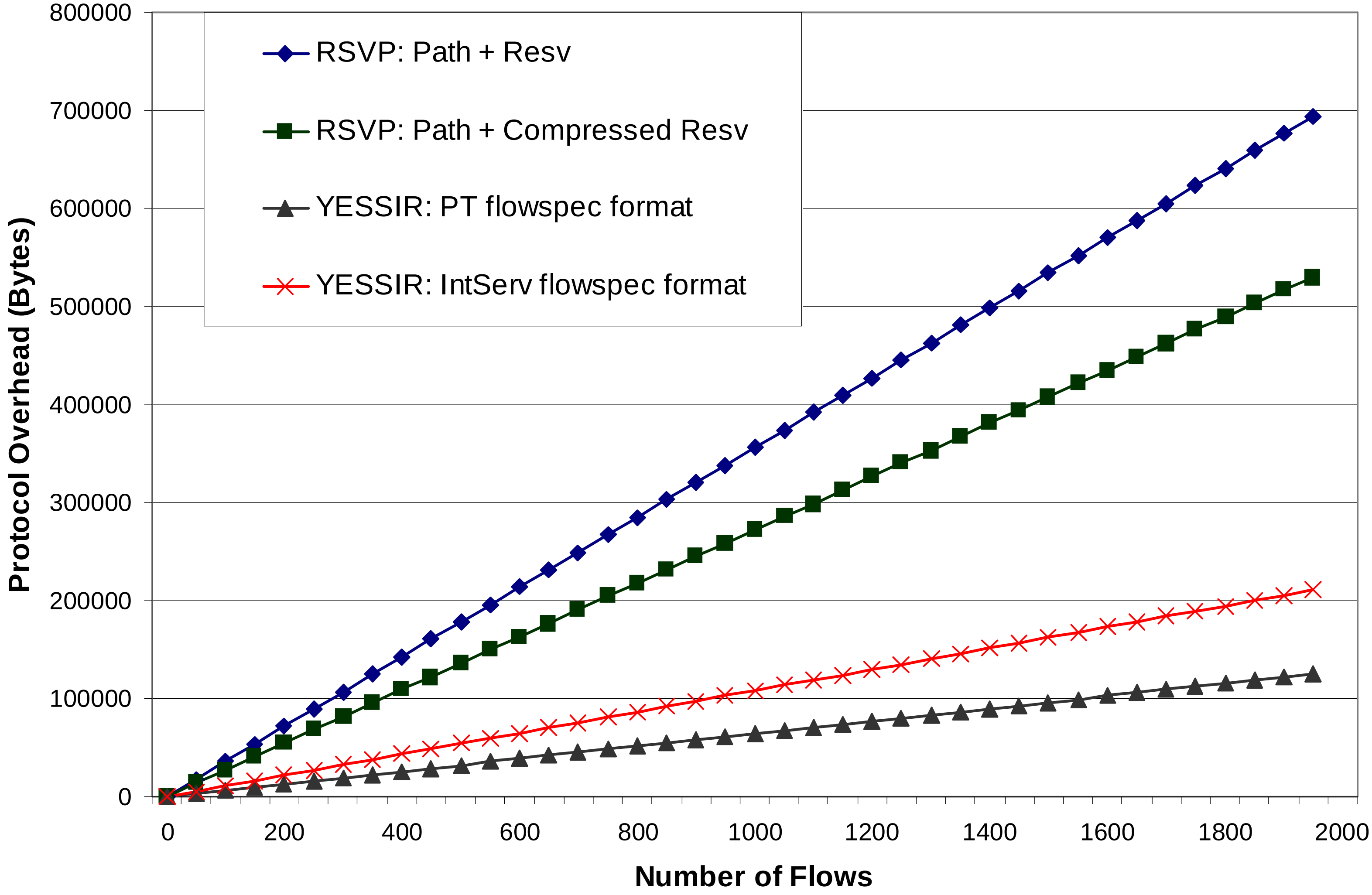
Results (RSVP refresh messages)

	Time (usec)
On Receive:	
PATH entry creation	30.39 +/- 0.76
Router Query	37.94 +/- 1.99
RESV entry look-up	11.01 +/- 0.35
Update reservation info	44.05 +/- 1.39
Timer routine:	
Send PATH Refresh	262.02 +/- 10.20
Send RESV Refresh	239.06 +/- 3.44
Single refresh overhead	**624.46 +/- 12.26**

Results (YESSIR refresh messages)

	Time (usec)
<u>On Receive:</u>	
YESSIR entry creation	19.31 +/- 0.69
Router Query	39.93 +/- 0.38
Update reservation info	24.49 +/- 0.31
Forward YESSIR downstream	252.56 +/- 2.00
<u>Timer routine:</u>	
YESSIR flow checking	8.03 +/- 0.73
<u>Single refresh overhead</u>	**344.32 +/- 1.88**

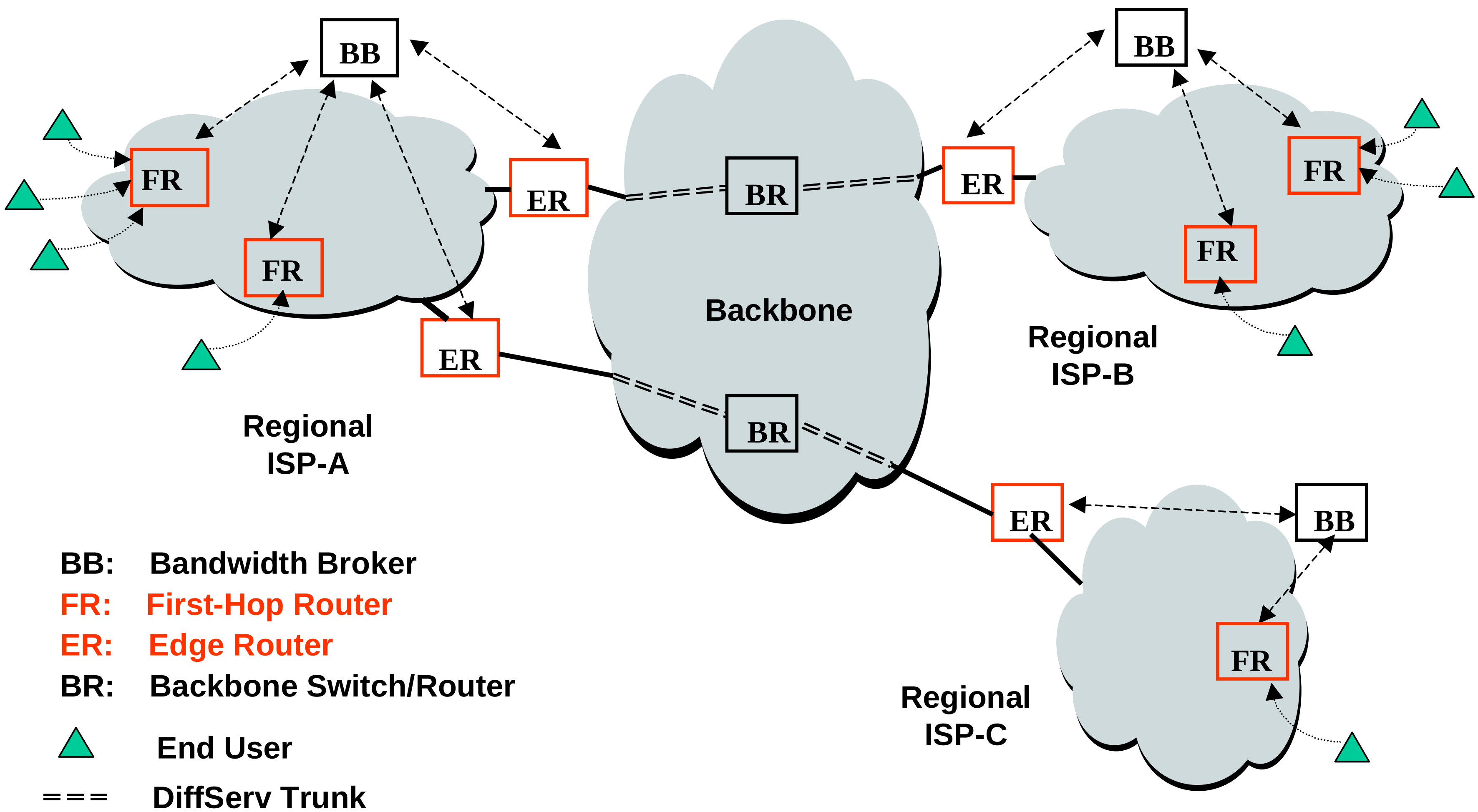
Results (Storage Saving)



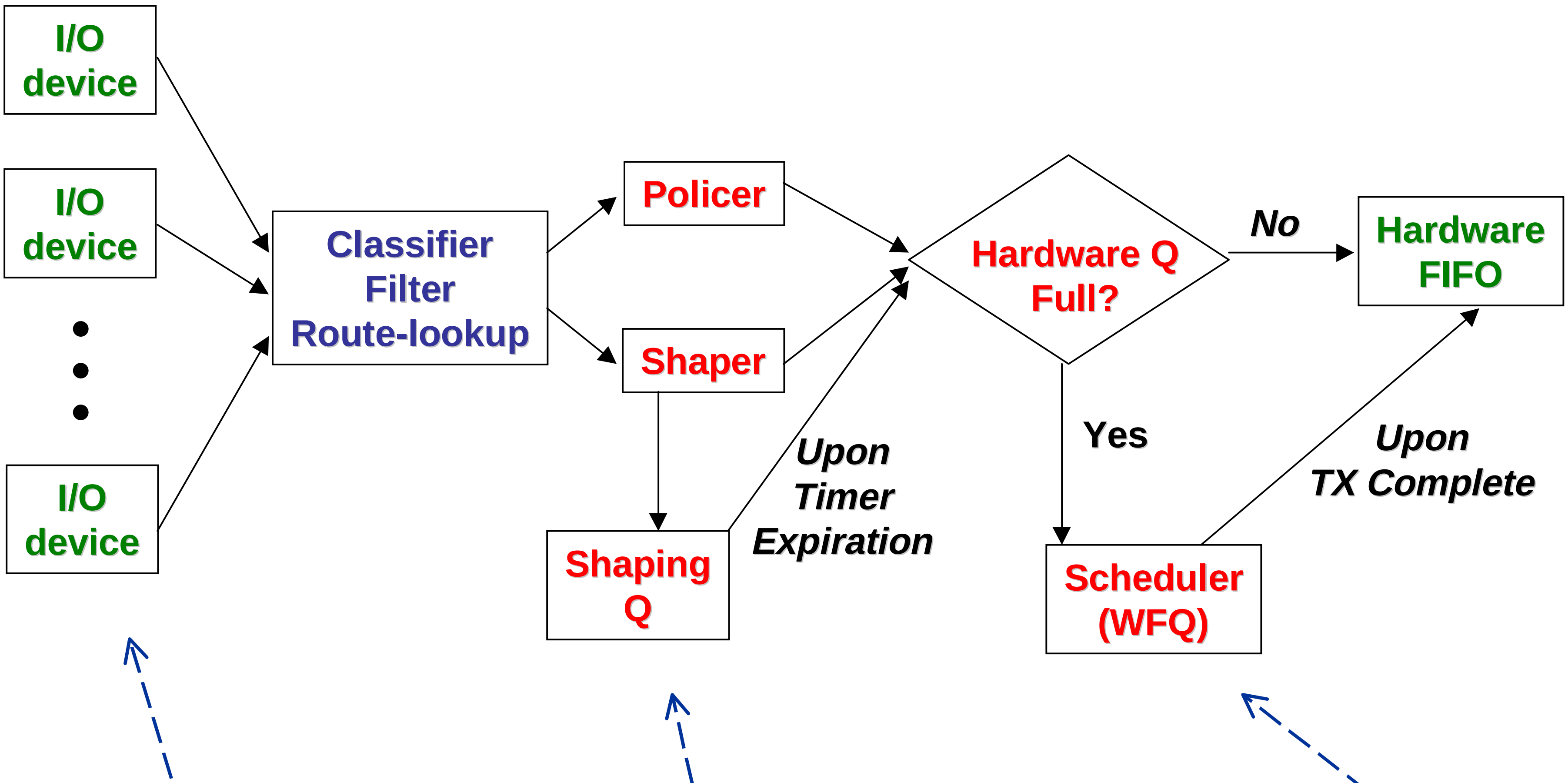
Issues:

- **Do we need resource reservation?**
- **How to set it up?**
  - RSVP and its problems
  - Alternative: YESSIR
  - Comparative Analysis
- **How to enforce it efficiently?**
  - Generic model and its problems
  - Buffer management
  - Results
- **Future works:**
  - inter-domain reservation

How to support QoS efficiently?



A Generic Data-path Model

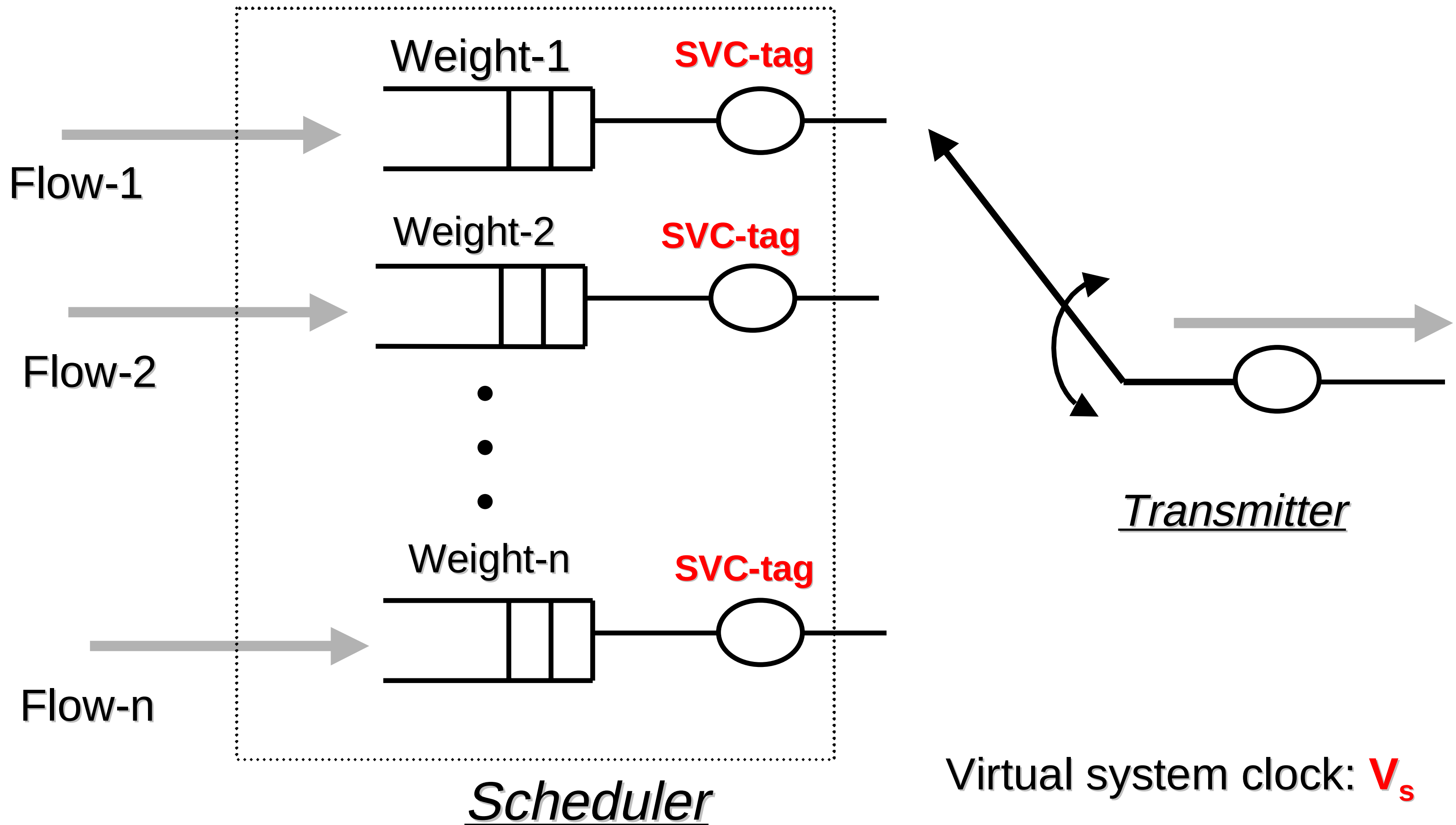


What if receive over-run?

"Wake-up" timer management is tough.

WFQ: Scheduling at per-flow level is costly.

Scheduling: SCFQ



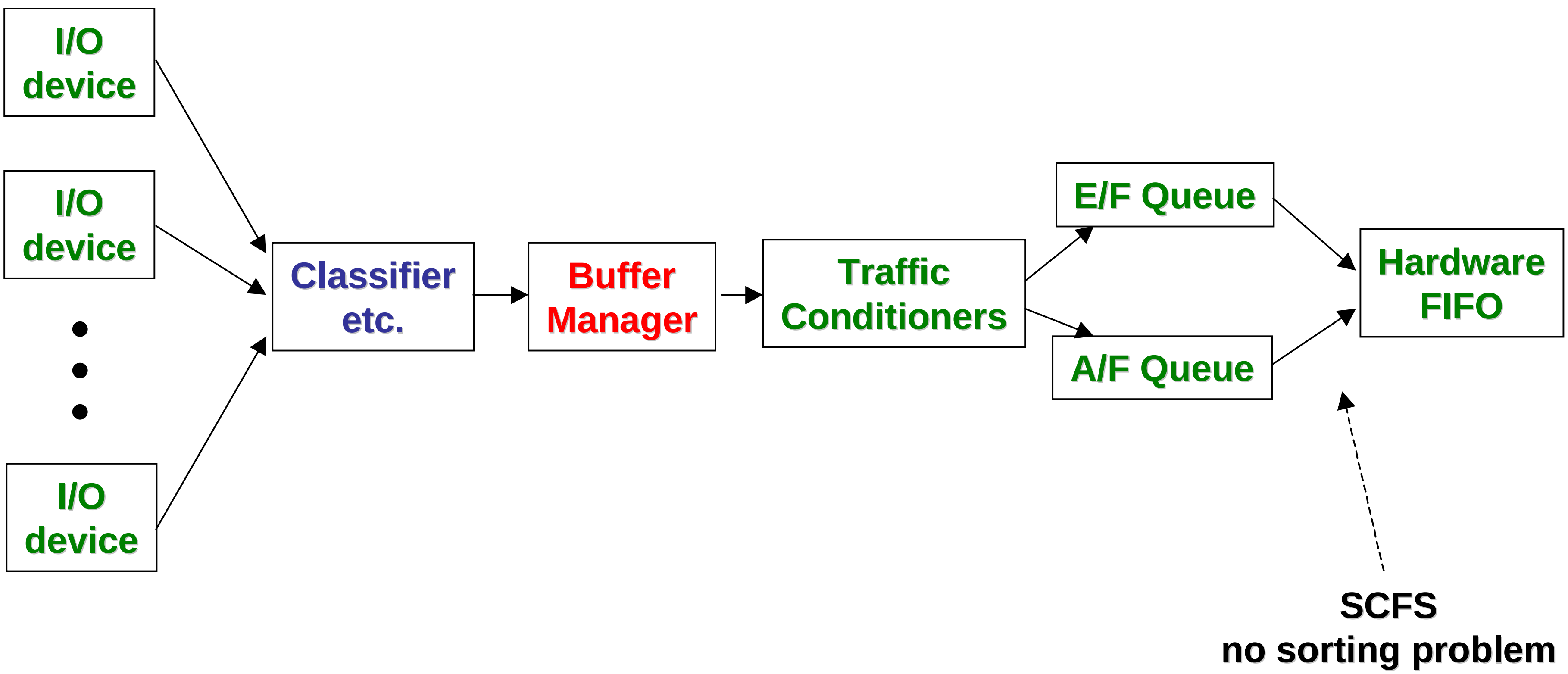
Scheduling Difficulty (SCFQ example):

```
While (less than tx-queue threshold) {

        If ( all queues are empty) {
                virtual-time = 0;
                break;
        }

        find a non-empty queue, k, with the smallest svc-tag;
        transmit the first packet from k;
        virtual-time = k.svc_tag;

        if (queue k is non-empty)
                k.svc_tag = packet-length / k.weight + virtual-time;
        else
                k.svc_tag = INFINITY;
}
```

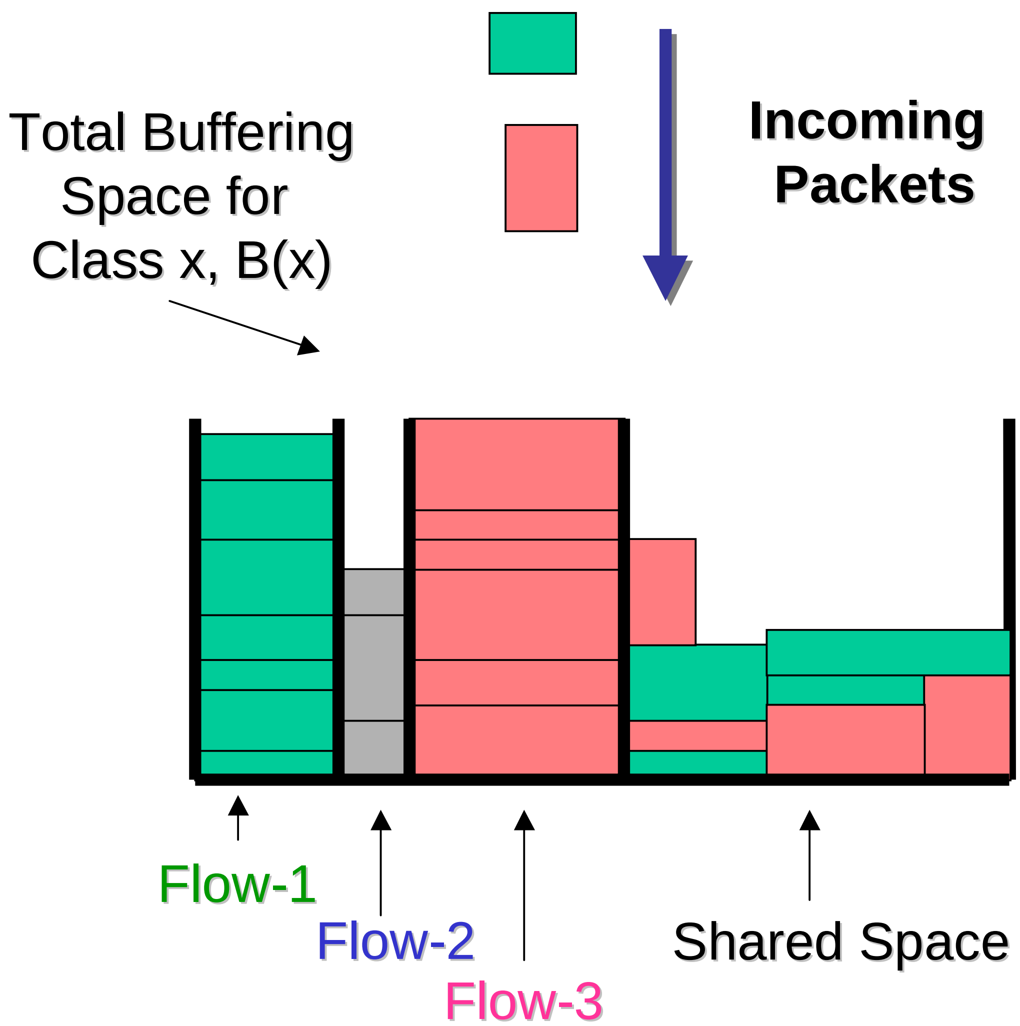
Conditioner Problem (GCRA example):

```
now = read_timer();
token = (flow->token + flow->LCT) - now;
length = MAX(length, flow->police_unit);
current_token = length / flow->token_rate;

if (token > 0) {
        if (token > (flow->max_token - current_token))
                if (flow->type == EF)        /* shaping */
                        add_to_shaping_queue(&packet, token);
                else        /* AF type, policing */
                        mark_congestion(&packet);
        else {
                flow->token = current_token + token;
                flow->LCT = now;
        }
}
else {
        flow->token = current_token;
        flow->LCT = now;
}
```

A Modified Model for DiffServ



Use of Buffer Management: *(An Accounting Mechanism)*



$B(i, x) = B(x) * r(i, x) / R(x)$

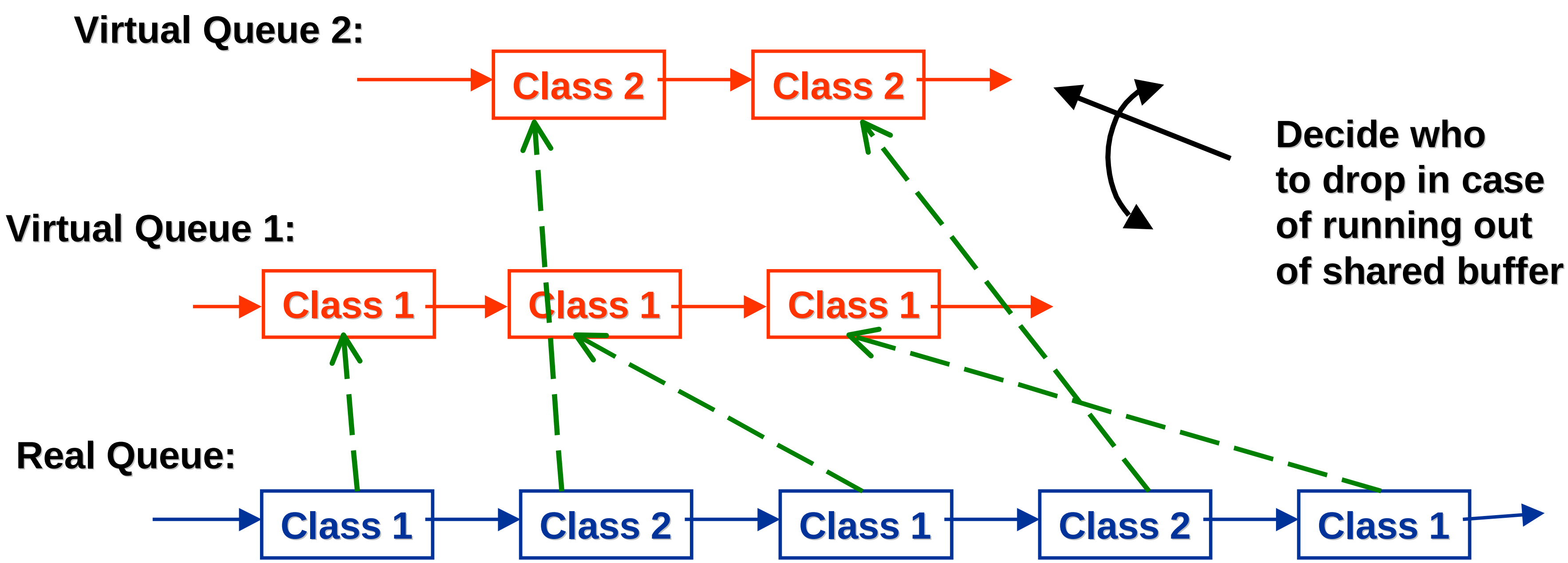
where

- $B(i, x)$: space for flow i;
- $R(x)$: Rate for class x;
- $r(i, x)$: rate for flow i.

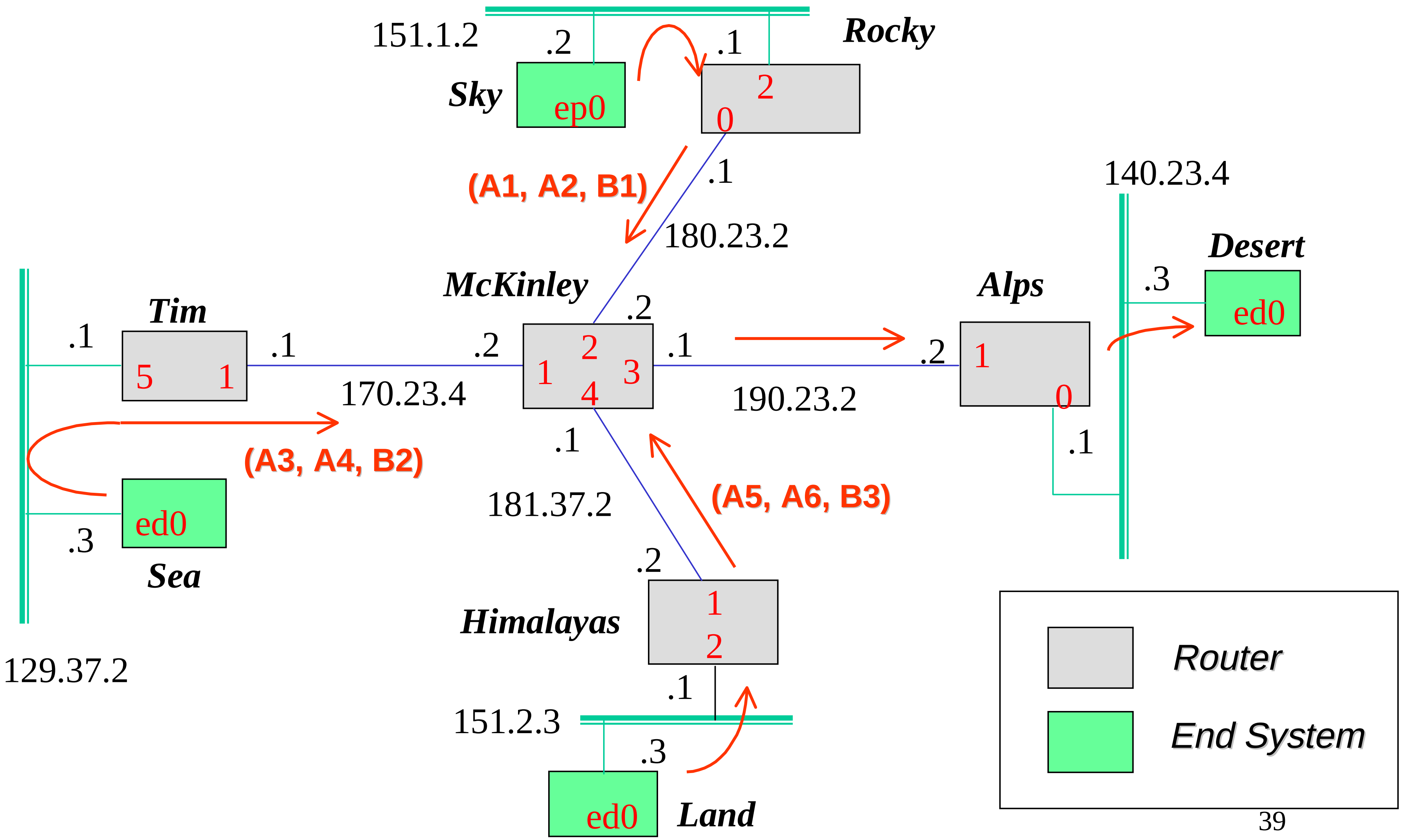
… Buffer Management

- Rate guarantee at flow level;
- use shared buffering space for excess traffic;
- low processing overhead:
  - ~ 20 extra C-code in critical data path;
  - a couple thousand lines in the background.
- mapping to DiffServ model (2 classes):
  - E/F : guarantee the rate;
  - A/F : provide in-profile rate;
  - A/F : apply a simple priority scheme among classes in shared buffering space....

… Buffer Management: support multiple AF classes



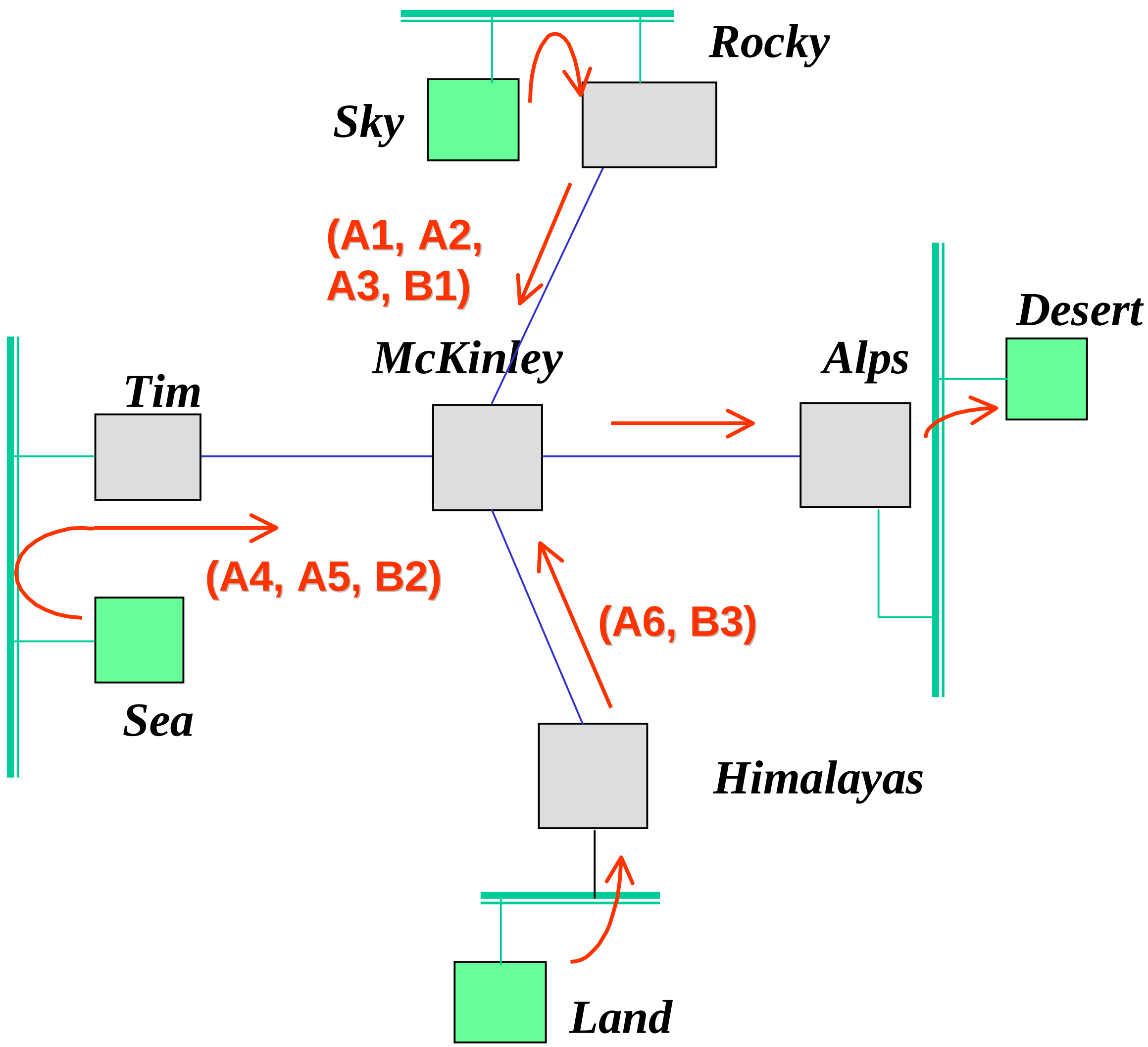
Experiment: *Testing Case 1, uniform periodic traffic*



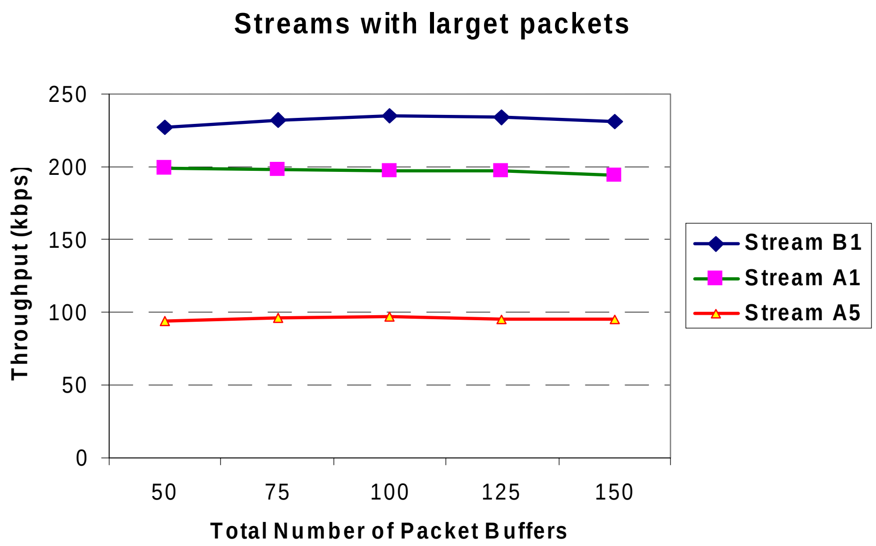
… Results for Case 1:

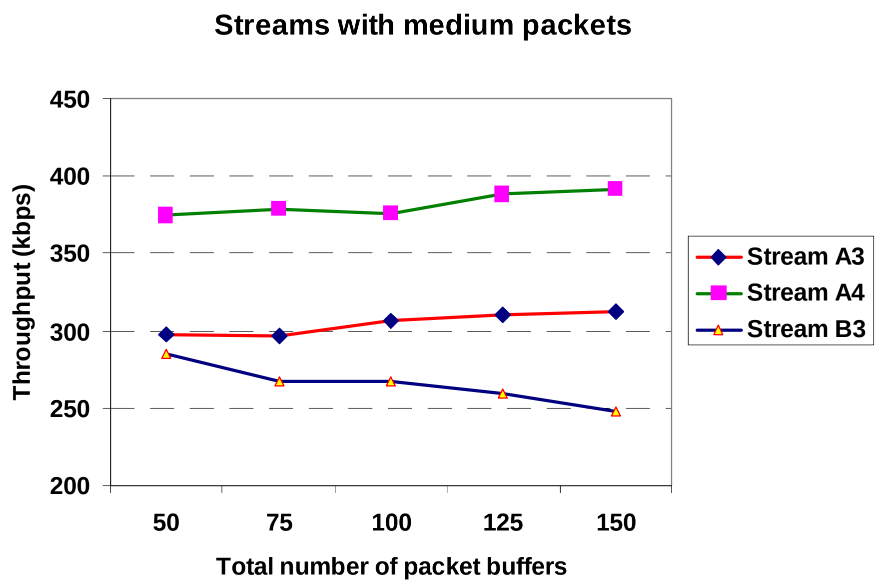
Flows	Snt (pps)	Snt (kbps)	Resv (kbps)	Rcv (kbps)	Rcv+QoS (kbps)
B1	300.0	240	----	222.1	186.8
A1	200.0	160	160	123.2	157.7
A2	200.0	160	160	122.2	159.7
B2	300.0	240	---	202.4	180.7
A3	200.0	160	160	118.4	159.5
A4	200.0	160	160	128.3	159.9
B3	300.0	240	---	234.3	141.2
A5	200.0	160	160	155.3	159.5
A6	200.0	160	160	155.1	156.1
Total	2100.0	1680.0		1461.7	1460.9

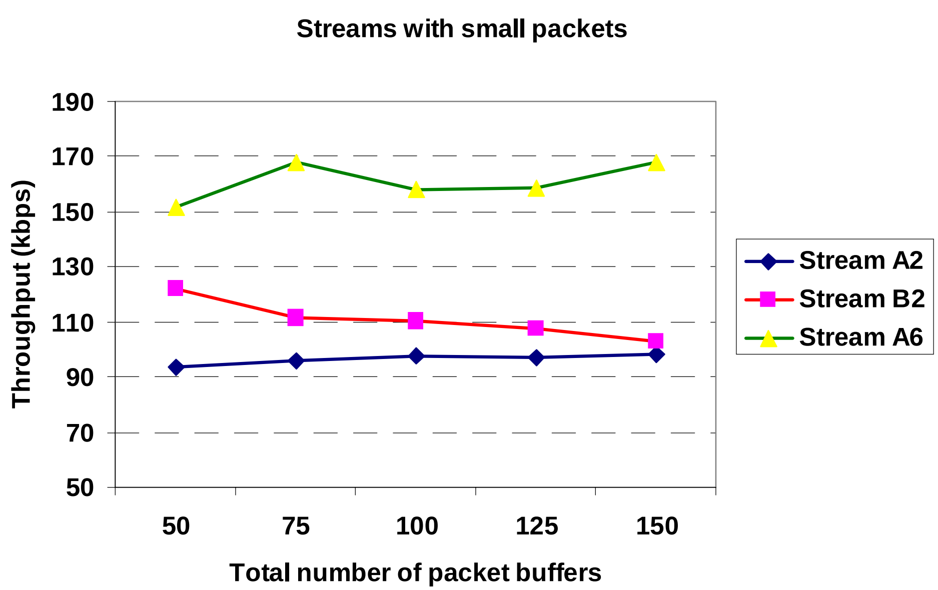
Experiment: *Testing Case 2, varying traffic*



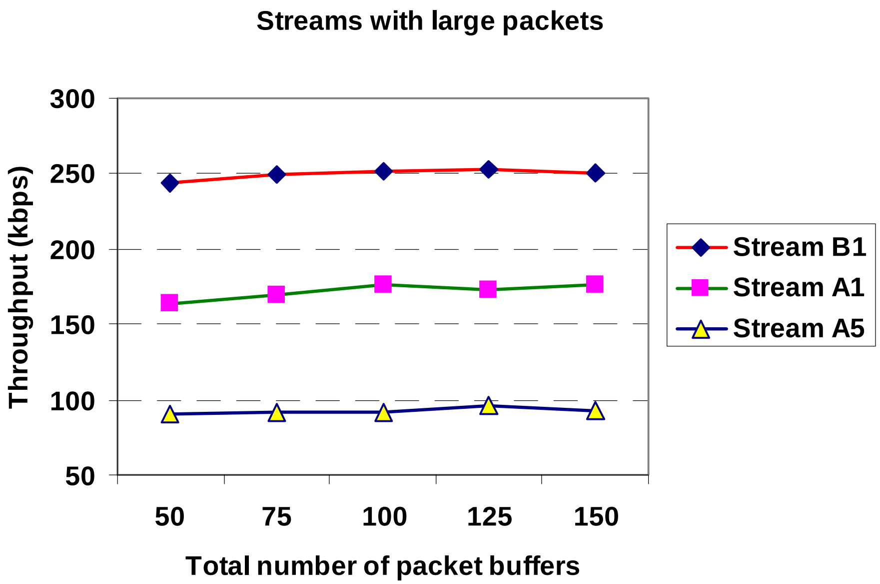
	Size	Resv	Actual
A1:	1000B	10%	10%
A2:	200B	5%	5%
A3:	500B	5%	20%
B1:	1000B	-	50%
A4:	500B	10%	20%
A5:	1000B	5%	5%
B2:	200B	-	20%
A6:	200B	10%	10%
B3:	500B	-	40%

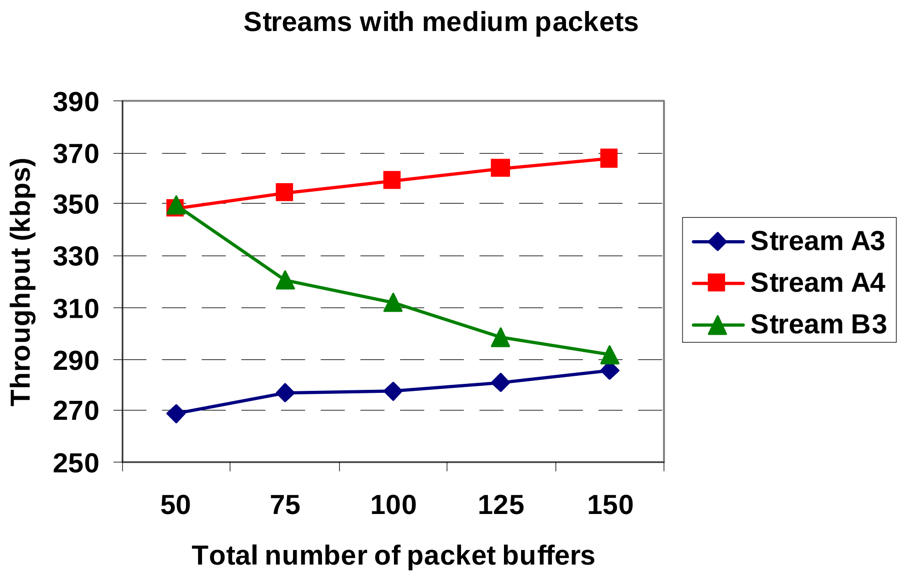


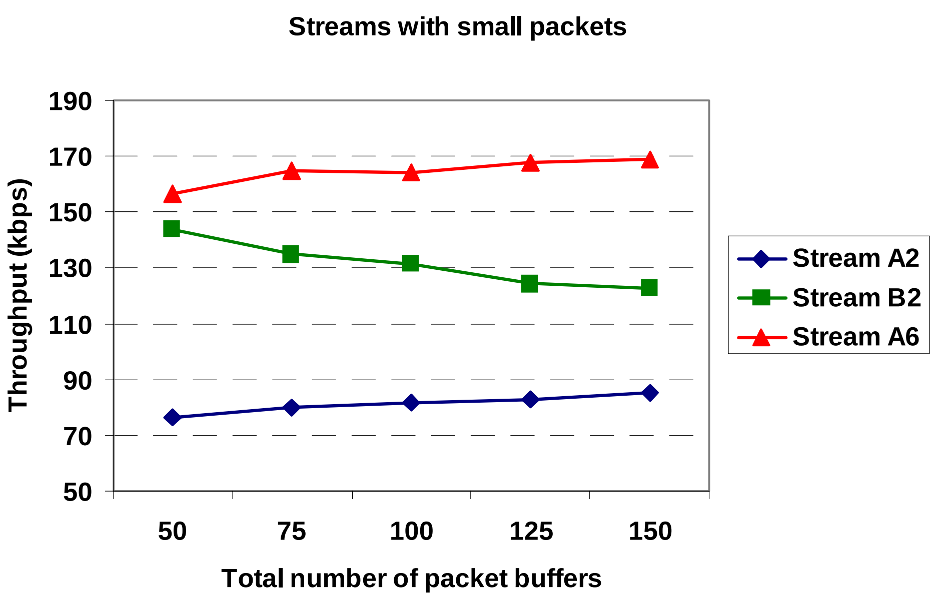




*** Sensitivity of Throughput and Rate Guarantee to Number of Packet Buffers (Periodic traffic)**







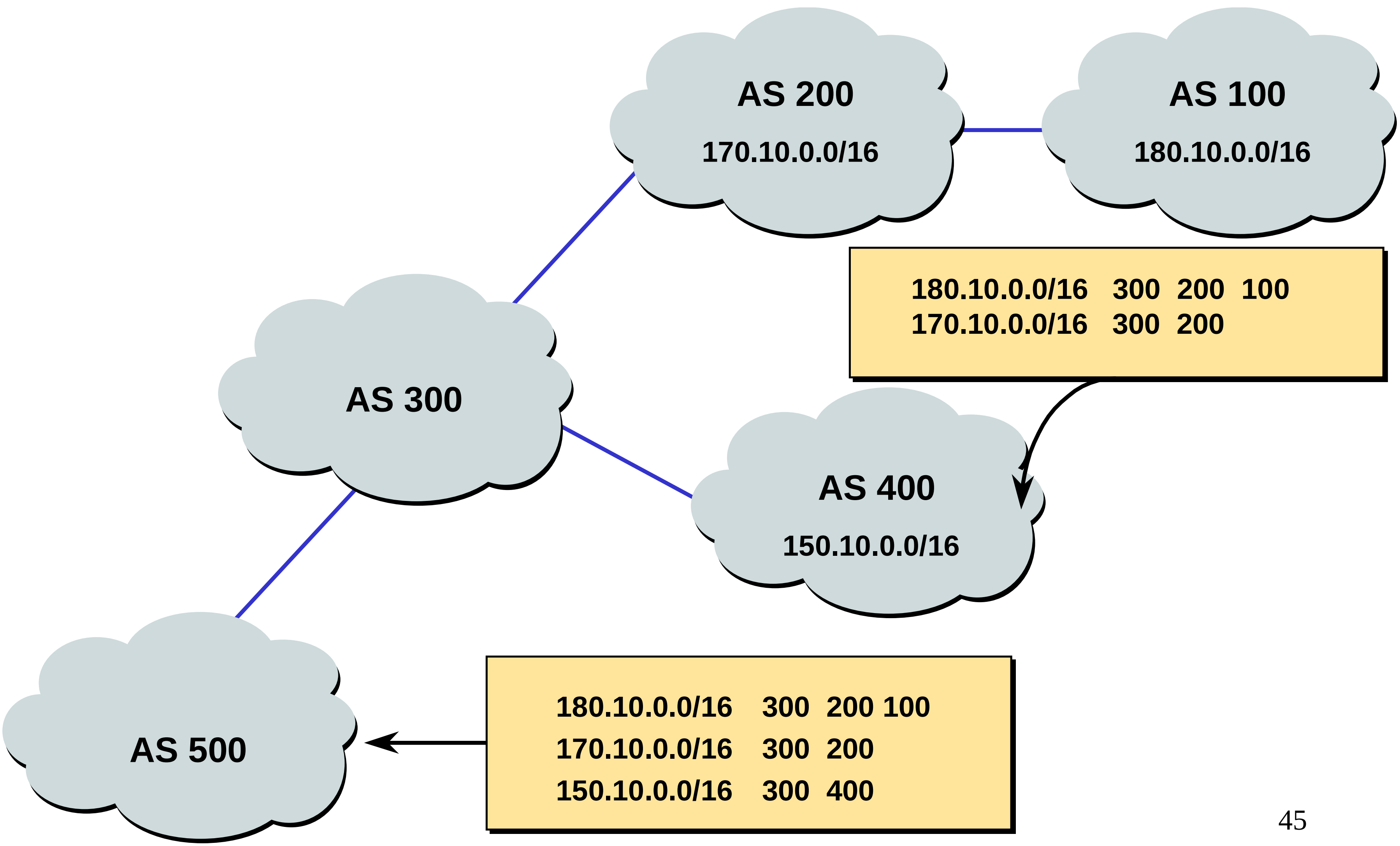
*** Sensitivity of Throughput and Rate Guarantee to Number of Packet Buffers (Poisson traffic)**

Issues:

- **Do we need resource reservation?**
- **How to set it up?**
  - RSVP and its problems
  - Alternative: YESSIR
  - Comparative Analysis
- **How to enforce it efficiently?**
  - Common model and its problems
  - Buffer management
  - Results
- **Future works:**
  - inter-domain reservation

Inter-domain Routing: BGP4

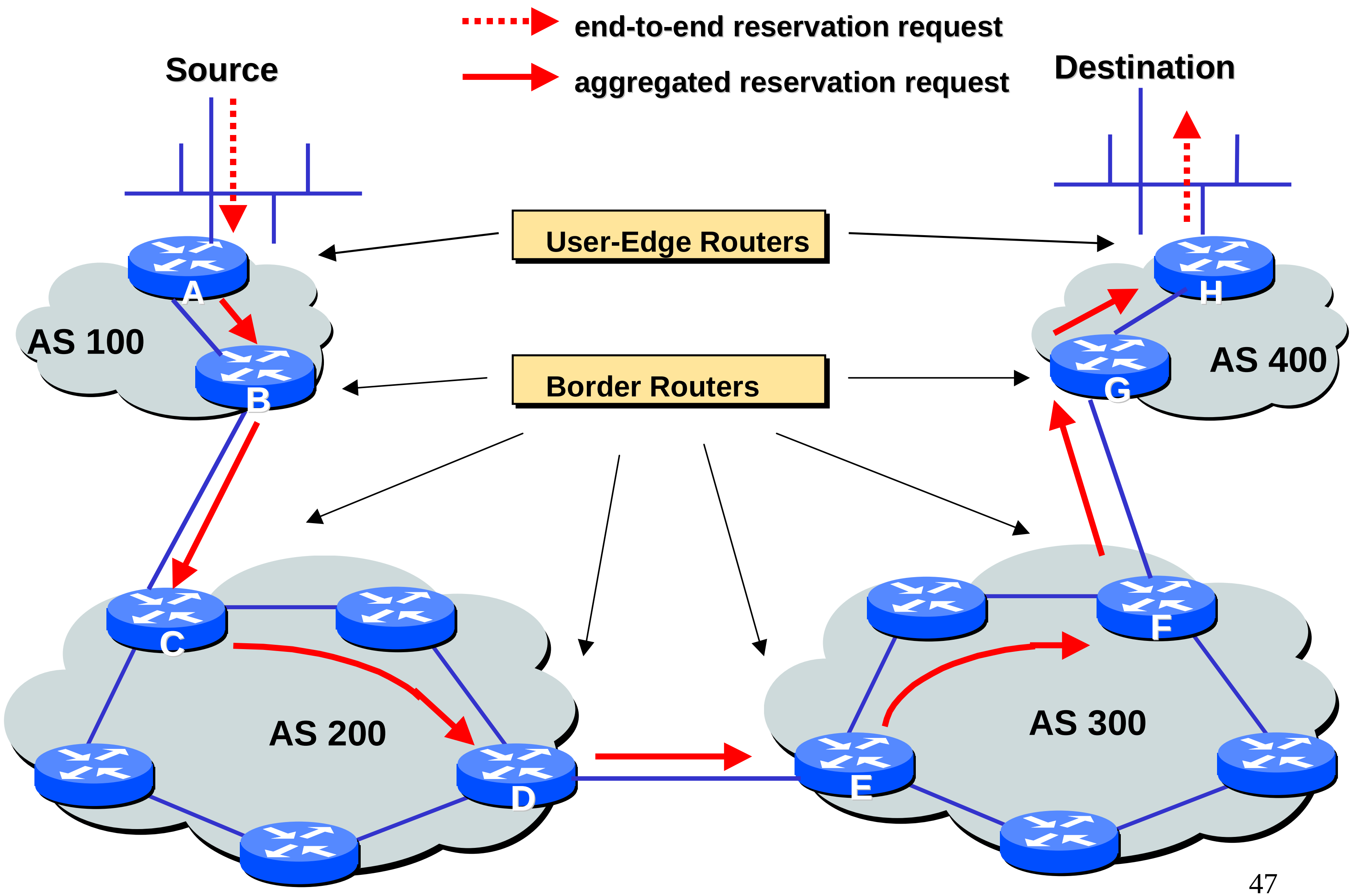
CIDR, domain, path-vector, ...



Inter-domain Reservation: (or aggregation)

- Advertise resource per-domain level;
- interface with BGP routing database at border;
- interface with intra-domain protocols to set up local domain “pipes”;
- “soft-state” with reliability support;
- policy-driven at edge/border.

Example


end-to-end reservation request
aggregated reservation request
Source
Destination
User-Edge Routers
Border Routers
AS 100
AS 200
AS 300
AS 400
A
B
C
D
E
F
G
H

Other On-going Research Activities

- Resource Reservation:
  - partial reservation analysis (experimental):
    - user behavior (automatic back-off, retry).
  - measurement-based reservation:
    - network utilization, blocking probability.
- QoS Support:
  - Traffic Shaper:
    - scalable;
    - low overhead.

Thanks!