An Architecture for negotiating QoS in the Internet

Chandrasekar Ramachandran

M.S. Thesis defense

Committee:

Dr. Joseph B. Evans (chair)

Dr. David W. Petr

Dr. John Gauch

I T T C

Organization

- Introduction
- Related Work
- Architecture
- Evaluation
- Conclusions and future work

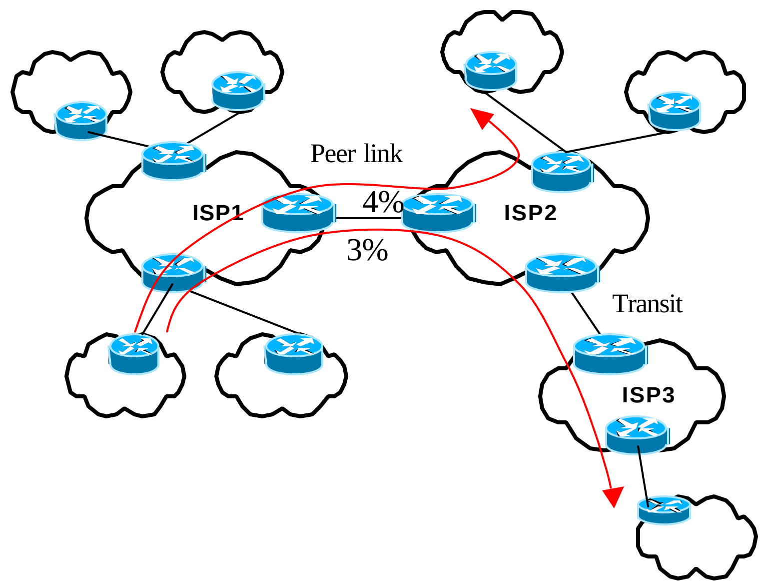
Introduction

- Over-provisioning primary method to satisfy growing demand
  - Internet Service Providers (ISPs) and enterprises provision capacity more than average utilization
  - lesser the utilization, greater the quality (delay, jitter, reliability)
- Not always true for
  - customer access links
  - ISP peering points
  - results in congestion
  - QoS needed primarily at these points

Introduction

QoS provisioning problem

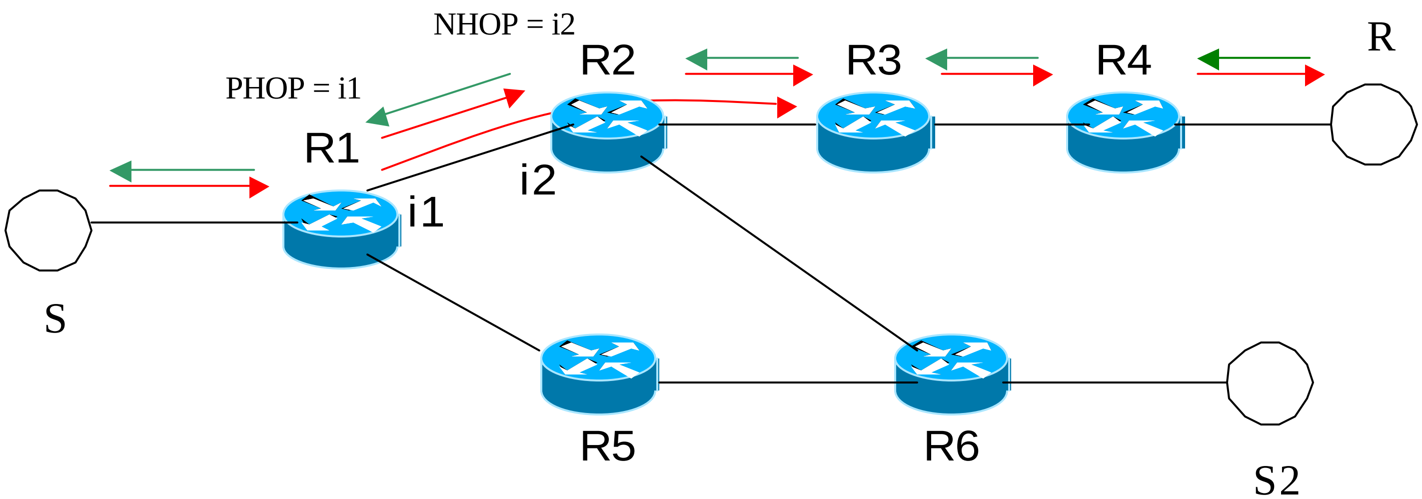
- Static
  - no signaling
  - ease of management
  - inefficient utilization
- dynamic
  - signaling required
  - added complexity
  - more efficient utilization
  - avoiding request rejects



Related Work

- Integrated services or IntServ and RSVP
- Aggregating RSVP-based QoS requests
- Bandwidth Broker (BB) signaling

IntServ-RSVP signaling



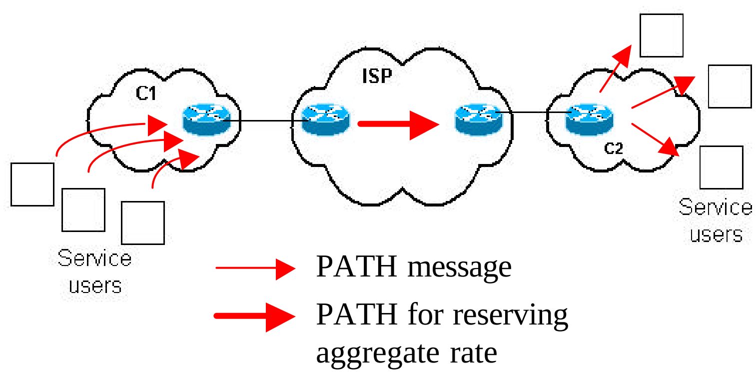
PATH (dest=R, src=S, Tspec[64Kbps,1000,64K,1500,64])

RESV (dest=R, src=S, Flowspec[64K,1000,64K,1500,64])

Related Work

Aggregating RSVP-based QoS requests

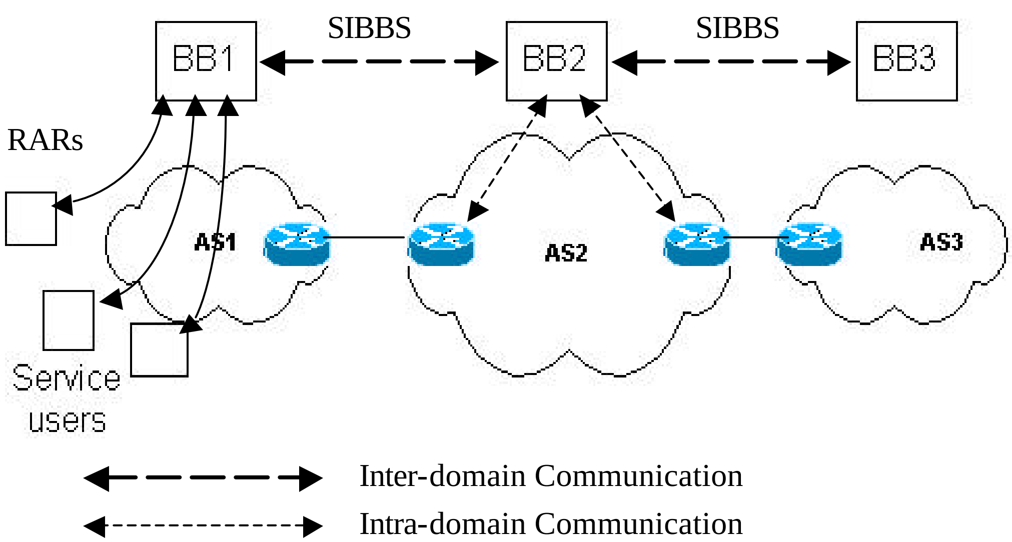
- forward individual PATH messages using a tunnel or new router alert option
- provider ingress reserves aggregate traffic volume in the core towards egress
- reduces state at ISP core, not at the edges
  - still a considerable overhead



Bandwidth Broker (BB) Signaling

- ISPs negotiate only traffic aggregates requiring specific service quality
- Simple Inter-domain Bandwidth Broker Signaling (SIBBS)
  - is lightweight since no multicast is considered
  - granularity in address blocks (CIDR prefixes) rather than individual addresses
  - request need not necessarily travel end-to-end

BB Signaling

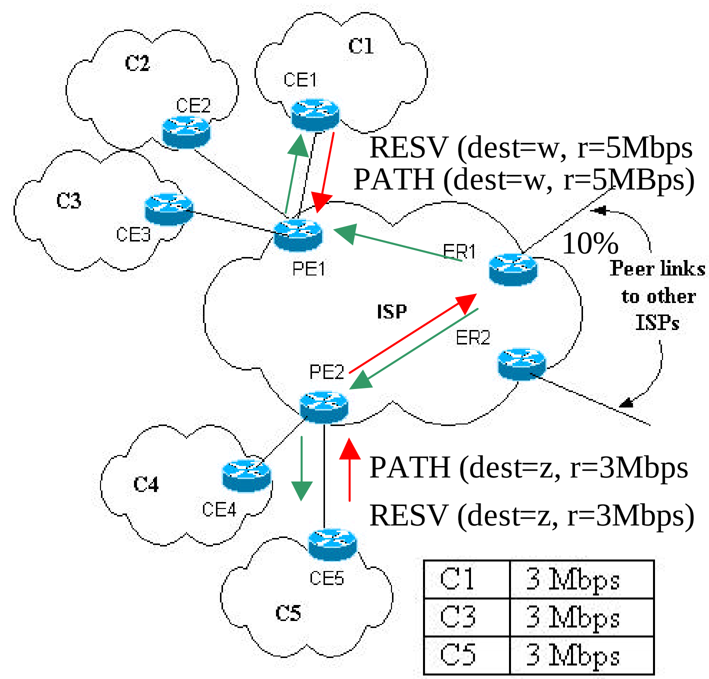


Architecture

- RSVP widely available in commercial routers
- adapts automatically to routing changes
  - knowledge of routing table not necessary
- RSVP receiver proxy controlled by policy
- Creating classifiers based on source or destination address prefixes

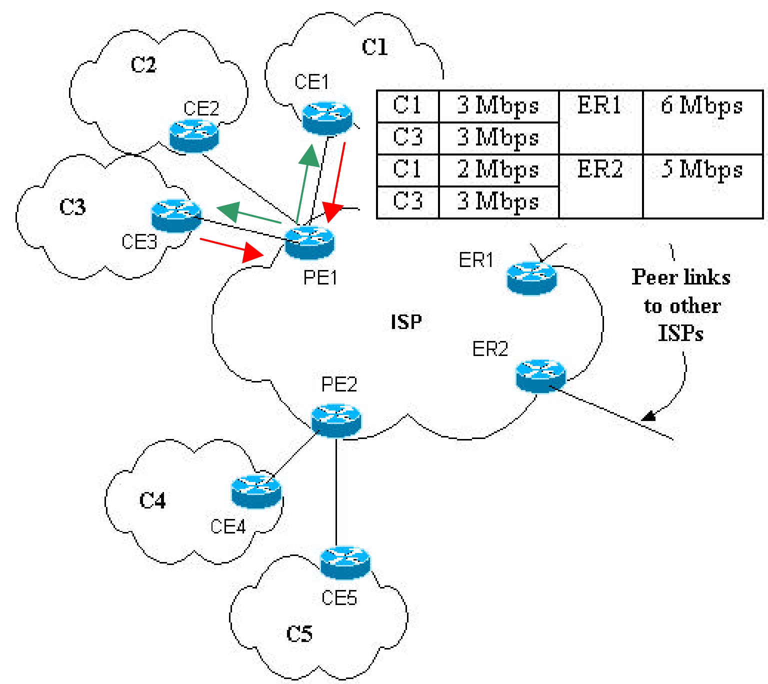
Architecture - Case #1

- Provider egress router:
  - sends RESV message depending on availability
  - contains access list to fairly allocate traffic rate during high utilization periods
- ingress routers only mark DSCP before forwarding packets to the core



Architecture - Case #2

- In previous scheme, egress routers need to store reservation state
- Each provider ingress allocated certain portion of peer link bandwidth
- suitable when
  - signaling is not end-to-end
  - ISP has good idea of traffic patterns



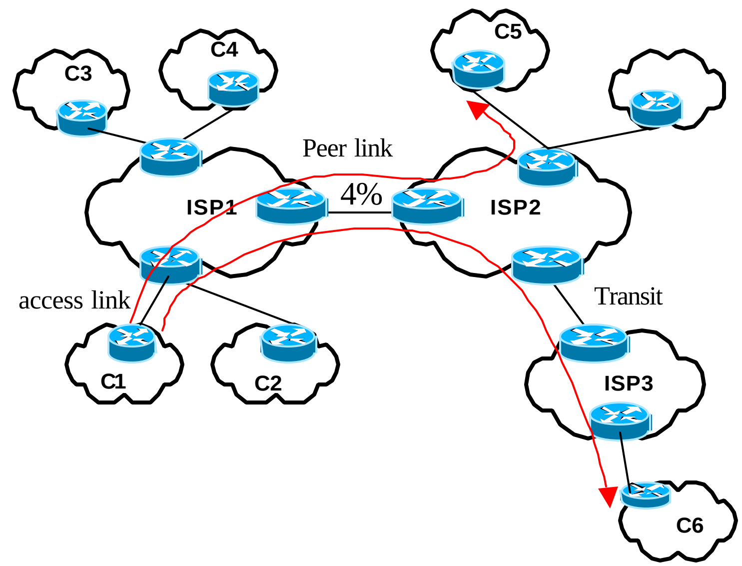
C1	3 Mbps	ER1	6 Mbps
C3	3 Mbps		
C1	2 Mbps	ER2	5 Mbps
C3	3 Mbps		

Architecture

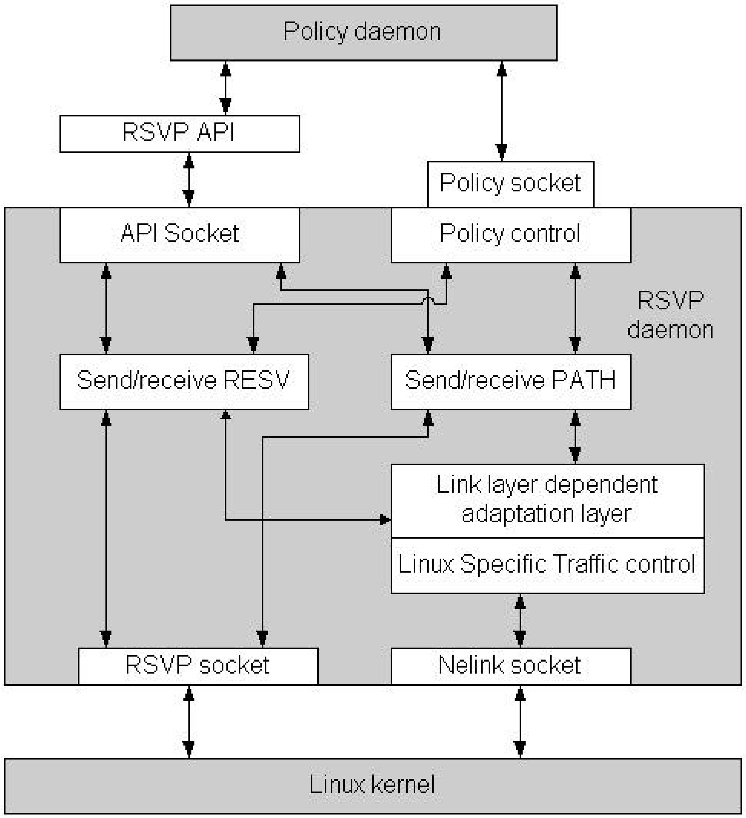
C1 wants ISP to reserve 4% of peer link to C5
- dynamic signaling

how to reserve on access link to C5?

- dynamic
- Static
  - security?

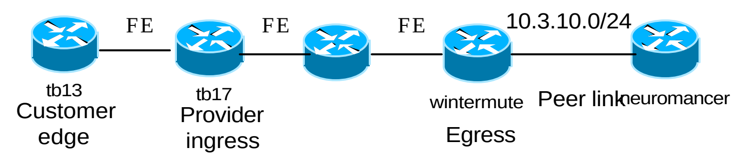


Implementation



Evaluation

- testbed13 is the customer edge
  - sends PATH to request bandwidth (4MBps) to 10.3.10.2
- egress (wintermute) sends RESV
- testbed17 is the provider ingress



RSVP dump at CE

17:01:37.718| Snd Raw PATH 10.3.10.2/0[17] 0=>eth0

PATH: Sess: 10.3.10.2/0[17] R: 30000 PHOP: <testbed13.ittc.ku.edu LIH=0>

testbed13.ittc.ku.edu/0 T=[4M(15K) 4MB/s 64 1.5K]

Adspec(1 hop 1.25MBW 0us 1500B, G={br!}, CL={br!})

17:01:55.259| Rcv Raw RESV 10.3.10.2/0[17] eth0<=0

RESV: Sess: 10.3.10.2/0[17] R: 30000 NHOP: <testbed17.ittc.ku.edu LIH=0>

FF testbed13.ittc.ku.edu/0 [CL T=[4M(15K) 4MB/s 64 1.5K]]

17:01:55.290 >>>>>>>>>> Internal STATE: <<<<<<< 66184 <<<<<<

FF Resv: Iface 0=>eth0 Nhop <testbed17.ittc.ku.edu LIH=0> TTD 223684

Filter testbed13.ittc.ku.edu/0 Flowspec [CL T=[4M(15K) 4MB/s 64 1.5K]]

Kernel reservation: Iface 0 (testbed13.ittc.ku.edu) Rhandle 0

Filter testbed13.ittc.ku.edu/0 Flowspec [CL T=[4M(15K) 4MB/s 64 1.5K]]

RSVP dump at egress

17:01:37.719| Rcv Raw PATH 10.3.10.2/0[17] eth0<=0
PATH: Sess: 10.3.10.2/0[17] R: 30000 PHOP: <testbed17.ittc.ku.edu/0 LIH=0>

FF Resv: Iface 5=>eth2 <10.3.10.2 LIH=5> TTD 219739
Filter testbed13.ittc.ku.edu/0 Flowspec [CL T=[4M(15K) 4MB/s 64 1.5K]]
Kernel reservation: Iface 5 (10.3.10.1) Rhandle 0
Filter testbed13.ittc.ku.edu/0 Flowspec [CL T=[4M(15K) 4MB/s 64 1.5K]]

17:01:54.745| Snd Raw RESV 10.3.10.2/0[17] 0=>eth0
RESV: Sess: 10.3.10.2/0[17] R: 30000 NHOP: <wintermute.ittc.ku.edu LIH=0>
FF testbed13.ittc.ku.edu/0 [CL T=[4M(15K) 4MB/s 64 1.5K]]

Observations

- Time taken to complete reservation : 17s 541ms
- Time taken for router to process PATH and send RESV ~ 17s
  – almost all the time taken at the router that sends RESV
- Path State Block (PSB) requires 200 bytes
- Reservation State Block (RSB) requires 124 to 192 bytes
  – if reservation is modified, old state is also stored

Evaluation

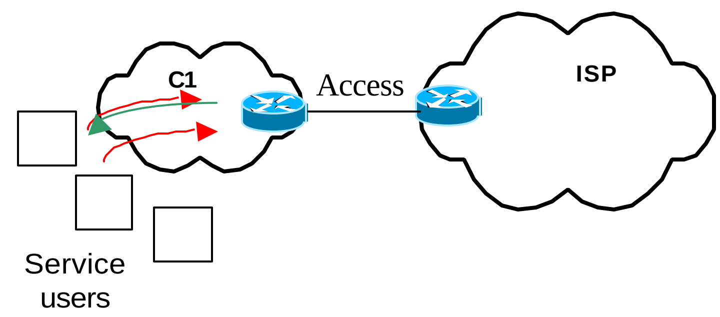
- Scalability
  - edge nodes deal only with traffic aggregates
  - state information include PSB and RSB corresponding to each request and reservation
    - O (N) where N is number of customer flows
- Management complexity
  - Access lists at the edges for policy and admission control

Evaluation...

- Management complexity
  - state information reduced if ingress routers decide to permit or deny request
  - no single point of failure
  - Inter-provider Interaction not essential due to receiver proxy

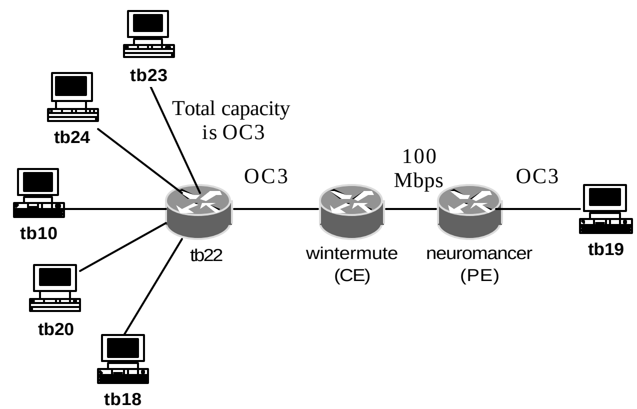
Access link

- Simplicity Vs Fairness
  - if hosts unaware of reserved rate, fairness cannot be guaranteed
- Receiver proxy for host enabled reservations



Access link (Fairness)

- Class 1
  - 10Mbps, 8 MTU sized burst
  - UDP (tb10 & tb20)
- Class 2
  - 20Mbps, 50 MTU burst
  - TCP (tb23 & tb24)
- Class 3
  - 70Mbps, no constraints
  - TCP (tb18) and all out of profile packets from other classes



Fairness Results

	Class 1 T'put (Mbps)		Class 2 T'put (Mbps)		Class 3 T'put (Mbps)
Flow #	tb10	tb20	tb21	tb23	tb18
1	2.494	2.487	4.108	4.265	11.048
2	2.428	2.423	4.313	4.339	11.061
3					11.638
4					11.615
	4.922	4.910	8.421	8.064	45.362
Total	9.832		16.485		45.362

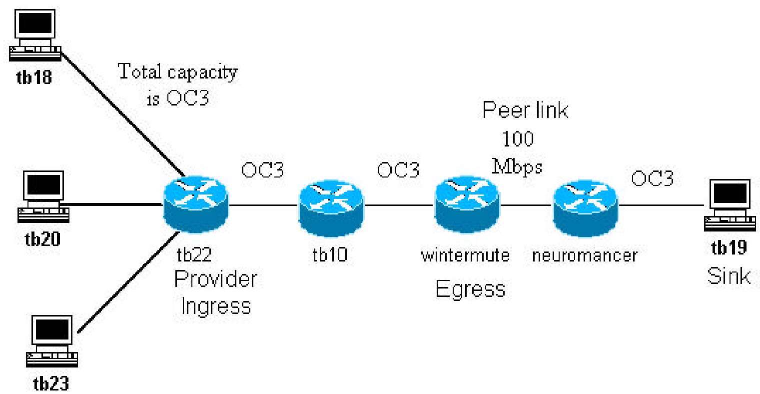
Class 1(b=3125, t=10ms), Class 2 (b=12500, w=12500), Class 3 (b=28750, w=32000)

	Class 1 T'put (Mbps)		Class 2 T'put (Mbps)		Class 3 T'put (Mbps)
Flow #	tb10	tb20	tb21	tb23	tb18
1	2.495	2.499	10.141	1.759	11.080
2	2.337	2.499	1.757	1.821	11.861
3					11.900
4					10.082
	4.832	4.998	11.898	2.580	43.923
Total	9.830		14.478		43.923

Class 1(b=3125, t=10ms), Class 2 (b=12500, w=12500) and (b=32500, w=32500), Class3 (b=28750, w=32000)

Peering points

- Class 1
  - 10Mbps, 8 MTU sized burst
  - NetSpec UDP burst (tb20)
- Class 2
  - 20Mbps, 50 MTU burst
  - NetSpec TCP full (tb23)
- Class 3
  - 70Mbps, no constraints
  - NetSpec TCP full (tb18)



Evaluation

Flow	Class 1 T'put (Mbps)	Class 2 T'put (Mbps)	Class 3 T'put (Mbps)
1	2.499	4.907	7.428
2	2.498	4.892	7.431
3	2.498	4.894	7.150
4	2.498	4.887	7.133
Total	9.993	19.480	29.142

Class1(b=3125, t=10ms), Class2 (b=22500, w=22500), class3 (b=28750, w=32000)

Flow	Class 1 T'put (Mbps)	Class 2 T'put (Mbps)	Class 3 T'put (Mbps)
1	2.498	4.041	8.216
2	2.497	4.047	8.214
3	2.497	4.060	8.208
4	2.495	4.026	8.209
Total	9.987	16.174	32.847

Class3 (b=38750, w=42000)

Flow	Class 1 T'put (Mbps)	Class 2 T'put (Mbps)	Class 3 T'put (Mbps)
1	2.491	3.004	9.114
2	2.386	3.000	9.116
3	2.438	2.984	9.068
4	2.491	2.491	9.071
Total	9.806	11.960	36.369

Class 3 (b=48750, w=52000)

Observations

- The overall performance degraded due to packet classification and queuing
  - may not be a problem with specialized router hardware
- Traffic used to test Class 2 is TCP, hence throughput reduced due to TCP back-off
  - due to two priority levels and WRR mechanism of CBQ
  - increasing share to 40Mbps but rate limiting to 20Mbps solved the problem

Conclusions

- QoS techniques needed at high utilization points of network
  - access and peering points
  - no guarantees on delay and jitter
  - introduce QoS at originating and receiving access points; if not effective, reserve at peer links
- End-to-end dynamic negotiation easier if domains travel not more than 2 transit AS

Future Work

- Measurement and analysis at a ‘real’ access and peer links
- Implementation supports traffic control using CBQ
  – could be extended to support WFQ in Linux

Questions ?