Distributed Edge-to-Edge QoS Monitoring Methods to Manage Traffic Flows in IP Networks Supporting Differentiated Services

October 22, 2001

Jae-Young Kim
jay@postech.ac.kr

Distributed Processing & Network Management Lab.
Dept. of Computer Science and Engineering
POSTECH

Outline

- **Introduction**
- DiffServ Framework and Its Limitations
- Monitoring Methods for Edge-to-Edge DiffServ Flows
- Applying Proposed Methods to Managing DiffServ Networks
- Edge-to-Edge DiffServ Flow Monitoring System
- Conclusions

Current Internet and QoS Support

- Simple but efficient IP technology
  - connectionless protocol
  - single FIFO queuing / best-effort packet forwarding
  - statistical multiplexing → efficient link utilization
  - shortest path routing → only one route at a time
  - IP over everything
  - Internet users and traffic volumes drastically increased

- No QoS support
  - same service quality for every packet
  - explosive traffic increase / burst traffic patterns
    → congestions at bottleneck points → unexpected delay / loss / jitter
  - no quality guarantee

Internet QoS Frameworks

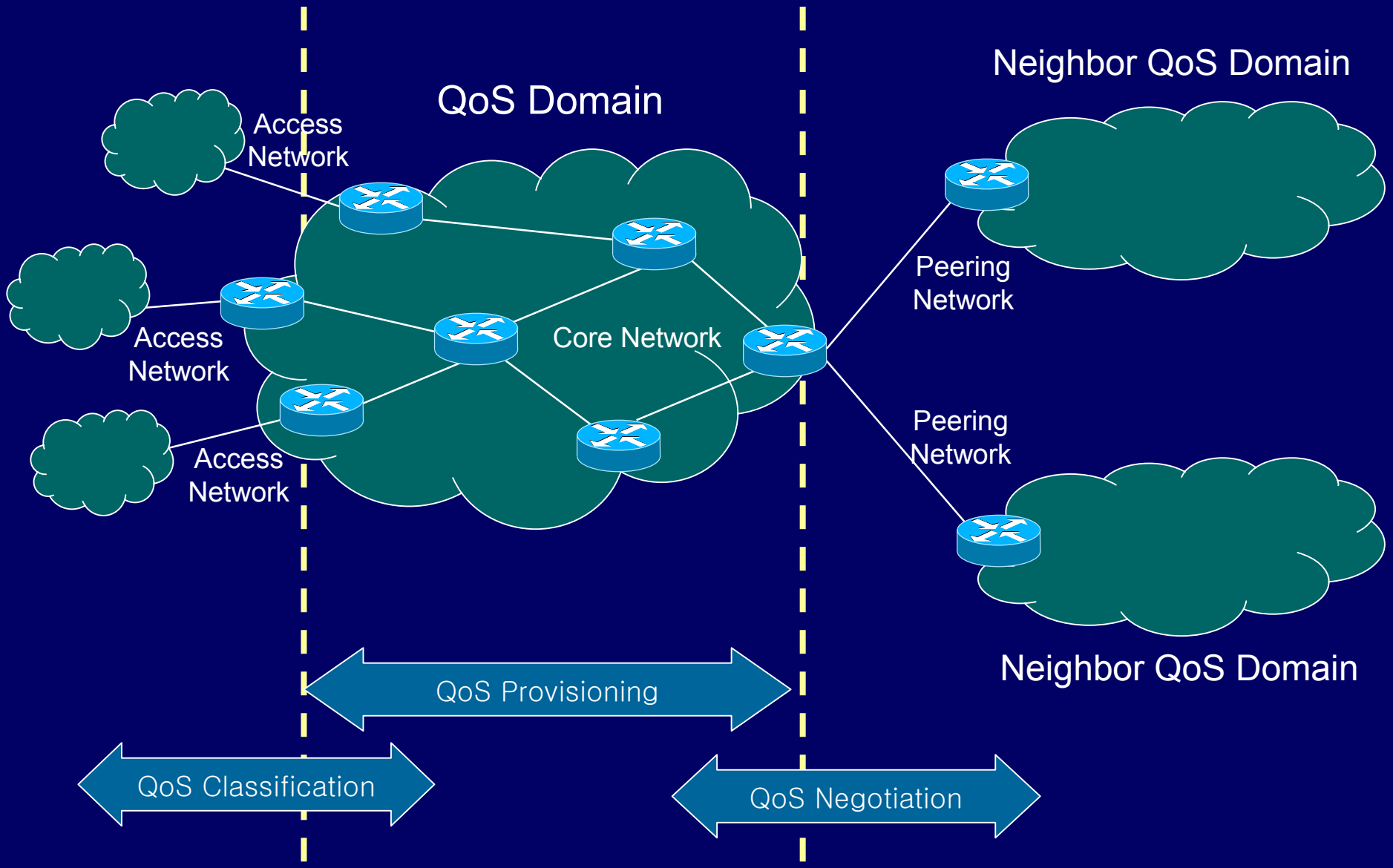
- Need for supporting QoS
  - need for guaranteeing QoS
  - application characteristics change (elastic → QoS-sensitive)
  - users requirements / providers requirements
  - need to provide different service quality to different users & applications

- Current Internet QoS frameworks
  - answers for providing different QoS to different users & applications

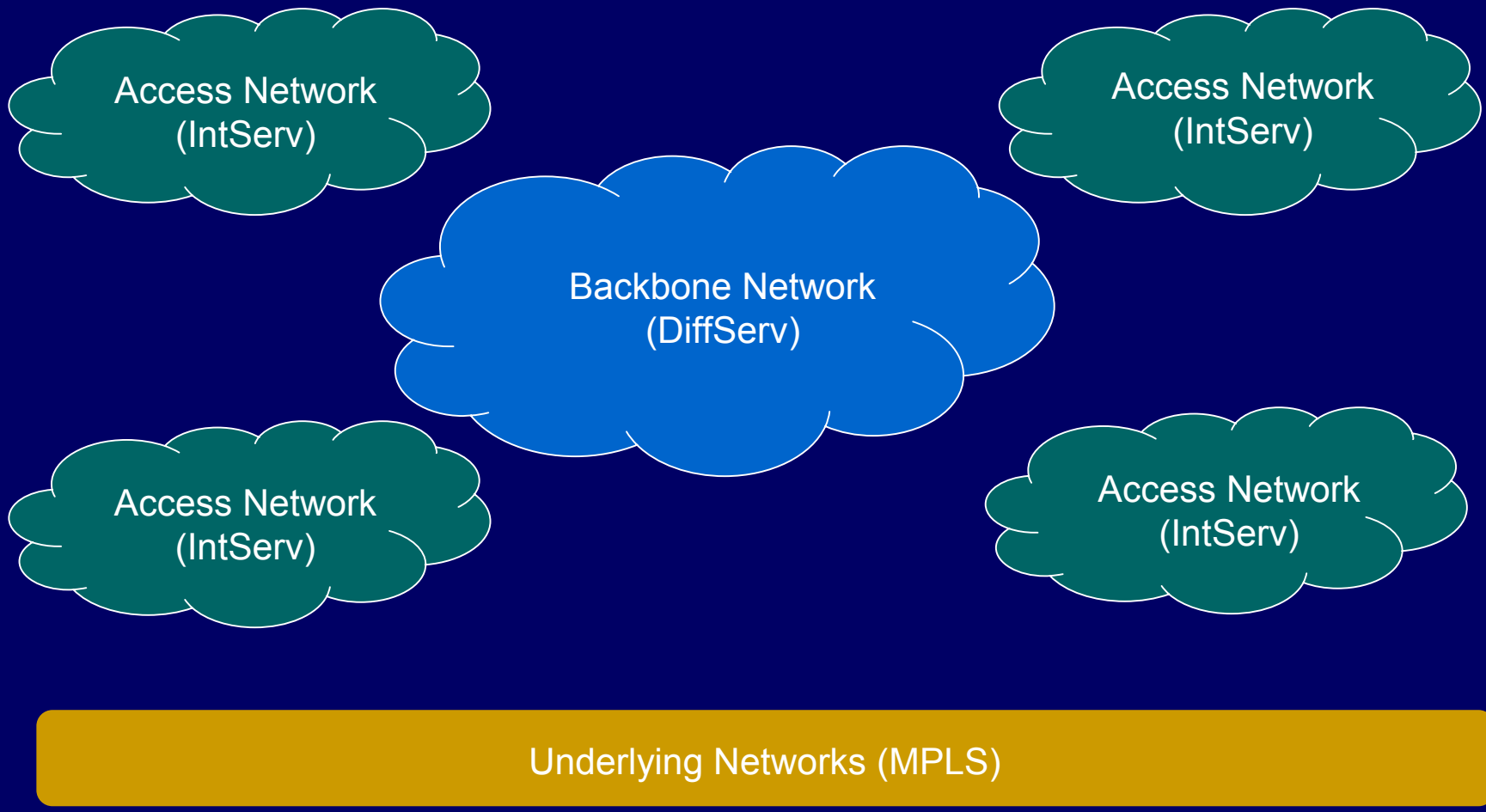
  - IntServ : 1993, RSVP signaling, fine-grained, not scalable
  - DiffServ : 1997, no signaling, edge/core model, coarse-grained, scalable
  - MPLS : 1998, label switching, tool for traffic engineering, LSP, LDP

  - each QoS framework has pros and cons to each other
  - integration of above frameworks

QoS-enabled IP Network Model



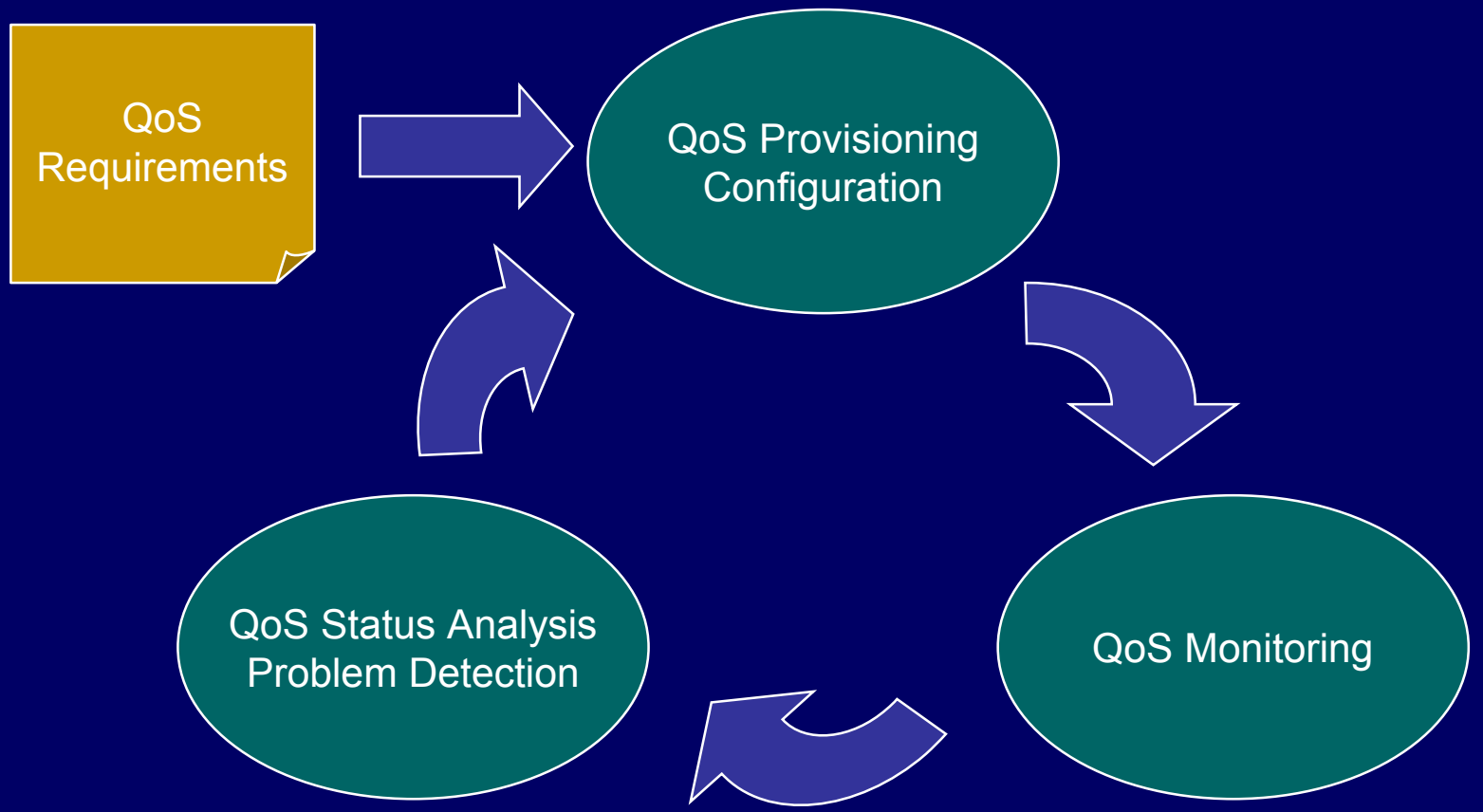
Integration of Internet QoS Frameworks



Need for QoS Monitoring

- Dynamic nature of network status
  - traffic burst and congestion
  - dynamic routing change
- Need to keep up the QoS status change
  - snapshot current QoS status
  - ensure QoS provisioning
  - verify expected QoS
  - understand historical trends
- Need to have information for solving QoS problems
  - report QoS degradation
  - pinpoint QoS degradation locations
  - help to find possible solutions
- Few research work on QoS monitoring
  - most existing work is done on QoS configuration and provisioning

QoS Management Process Cycle



Motivation and Key Ideas

- DiffServ QoS framework
  - edge / core model
  - small number of QoS classes (maximum 64) for aggregated traffic
  - separation of QoS control and routing control
- Edge-to-Edge QoS monitoring
  - E-to-E QoS is useful for managing DiffServ domains in various ways
  - each DiffServ router can monitor QoS classes it handles
  - however, local QoS monitoring doesn't provide E-to-E QoS status
  - how about combining routing and local QoS monitoring together?
- Key ideas
  - E-to-E QoS = routing topology + local QoS monitoring
  - local QoS monitoring (throughput, drop) at each DiffServ router
  - construct E-to-E QoS by combining local QoS and E-to-E routing path
  - modeling E-to-E DiffServ flow and hop-by-hop QoS concatenation rules
  - finer-grained DiffServ : enable different control on different E-to-E flows
    (number of supported QoS classes) X (number of edge routers)^2

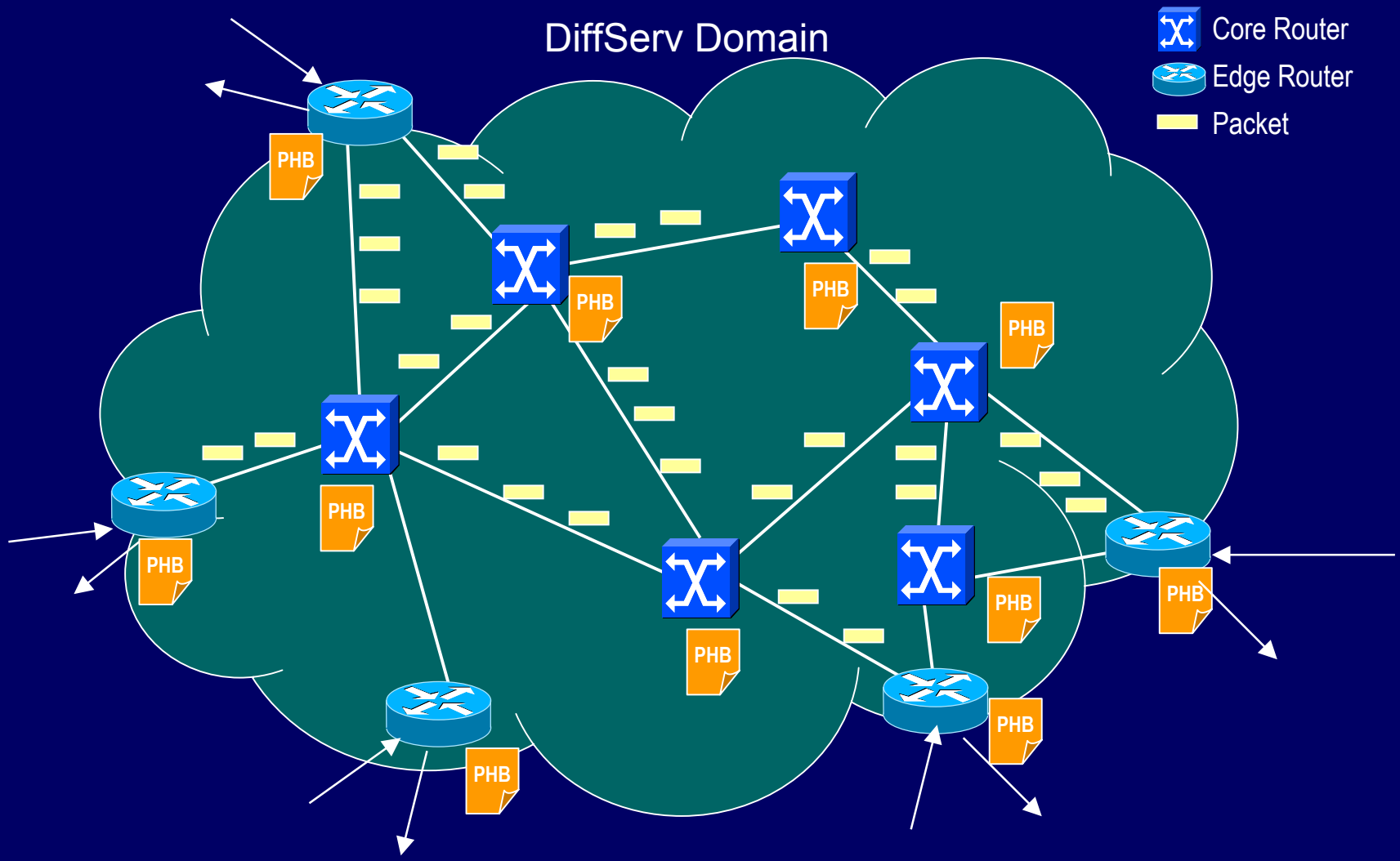
Related Work

- Basic IP traffic monitoring
  - various low-level monitoring tools: tcpdump, ntop, ethereal, …
  - SNMP-based monitoring: MIB2, RMON (from IETF)
  - traffic flow monitoring: RTFM (RFC2720-2724), NetFlow (Cisco)
  - IP performance metric (IPPM): RFC2330
- End-to-End QoS monitoring
  - passive monitoring
    Jiang et al. (University of Singapore), NetScope (AT&T Lab)
  - active probing
    PINGer (Stanford IEPM), MBAC (Measurement-Based Admission Control)
- DiffServ QoS management approach
  - primitive DiffServ management: DSMIB, DSPIB, DSMON (from IETF)
  - Bandwidth Broker architecture (BB): IETF Internet draft, 1997
  - Per-Domain Behavior (PDB): IETF RFC3086
  - Resource Management in DiffServ (RMD) framework: IETF Internet draft, 2001
  - TEQUILA project: European Commission IST, 2000~2002

Outline

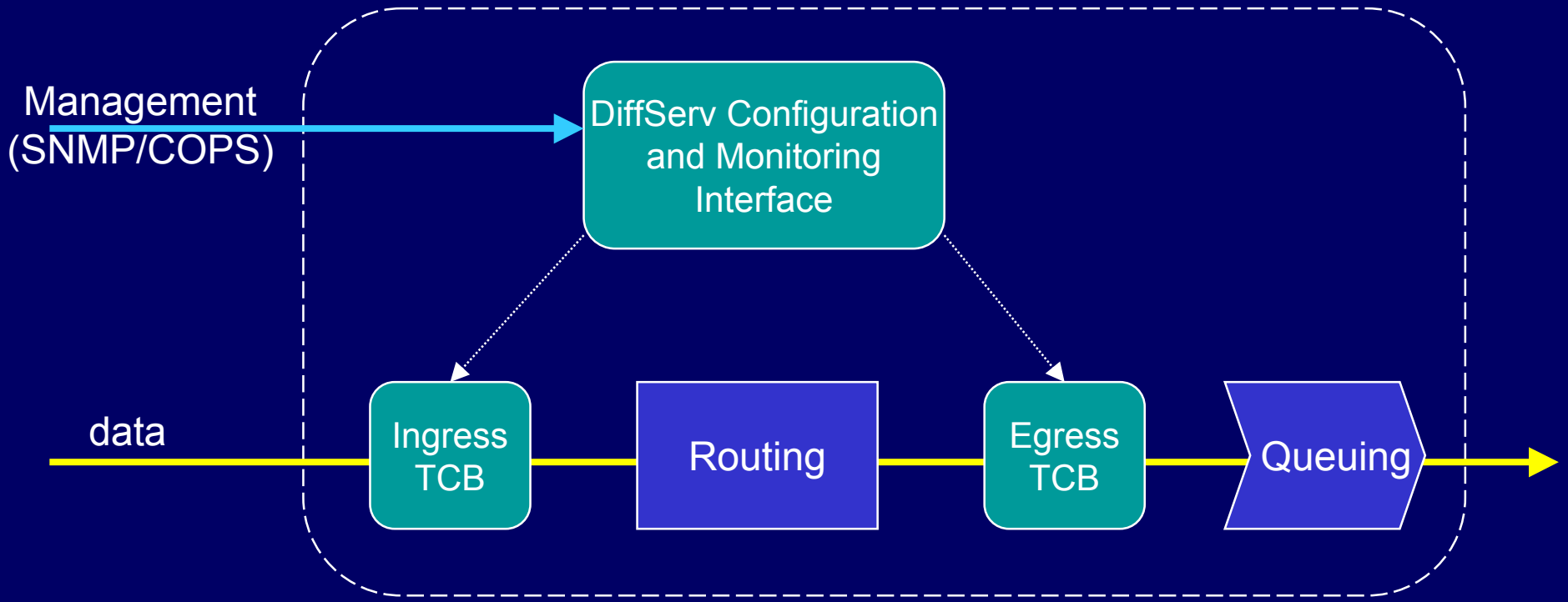
- Introduction
- **DiffServ Framework and Its Limitations**
- Monitoring Methods for Edge-to-Edge DiffServ Flows
- Applying Proposed Methods to Managing DiffServ Networks
- Edge-to-Edge DiffServ Flow Monitoring System
- Conclusions

Concepts of DiffServ Framework



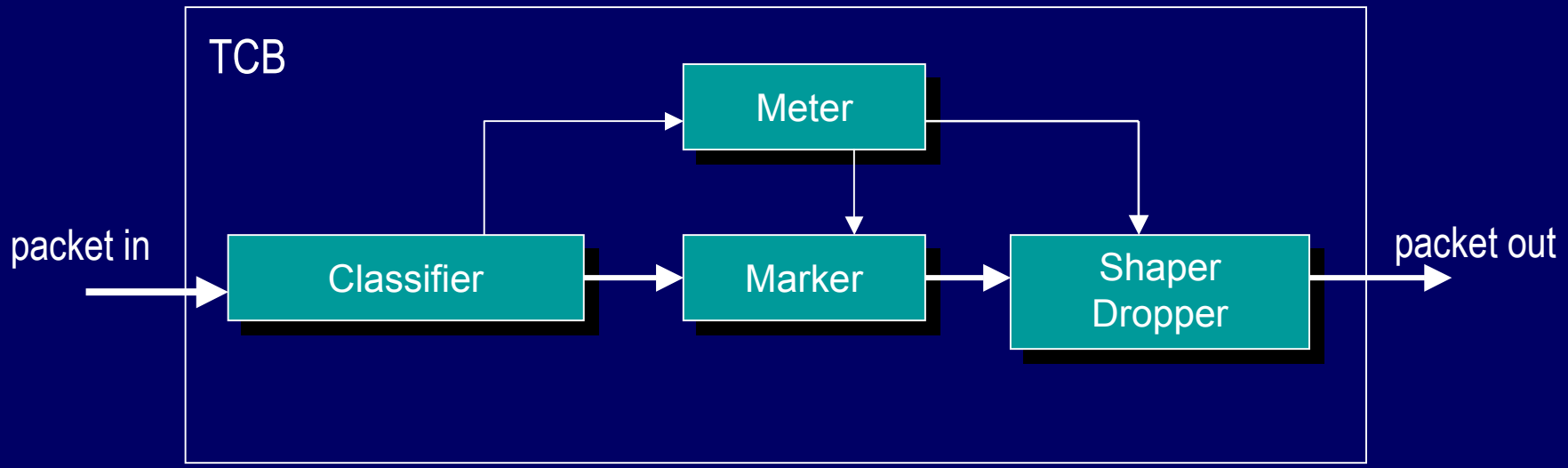
- Edge / Core model
- Per-Hop Behavior (PHB)
- Limited number of QoS classes for aggregated traffic

Functional Blocks of a DiffServ Router

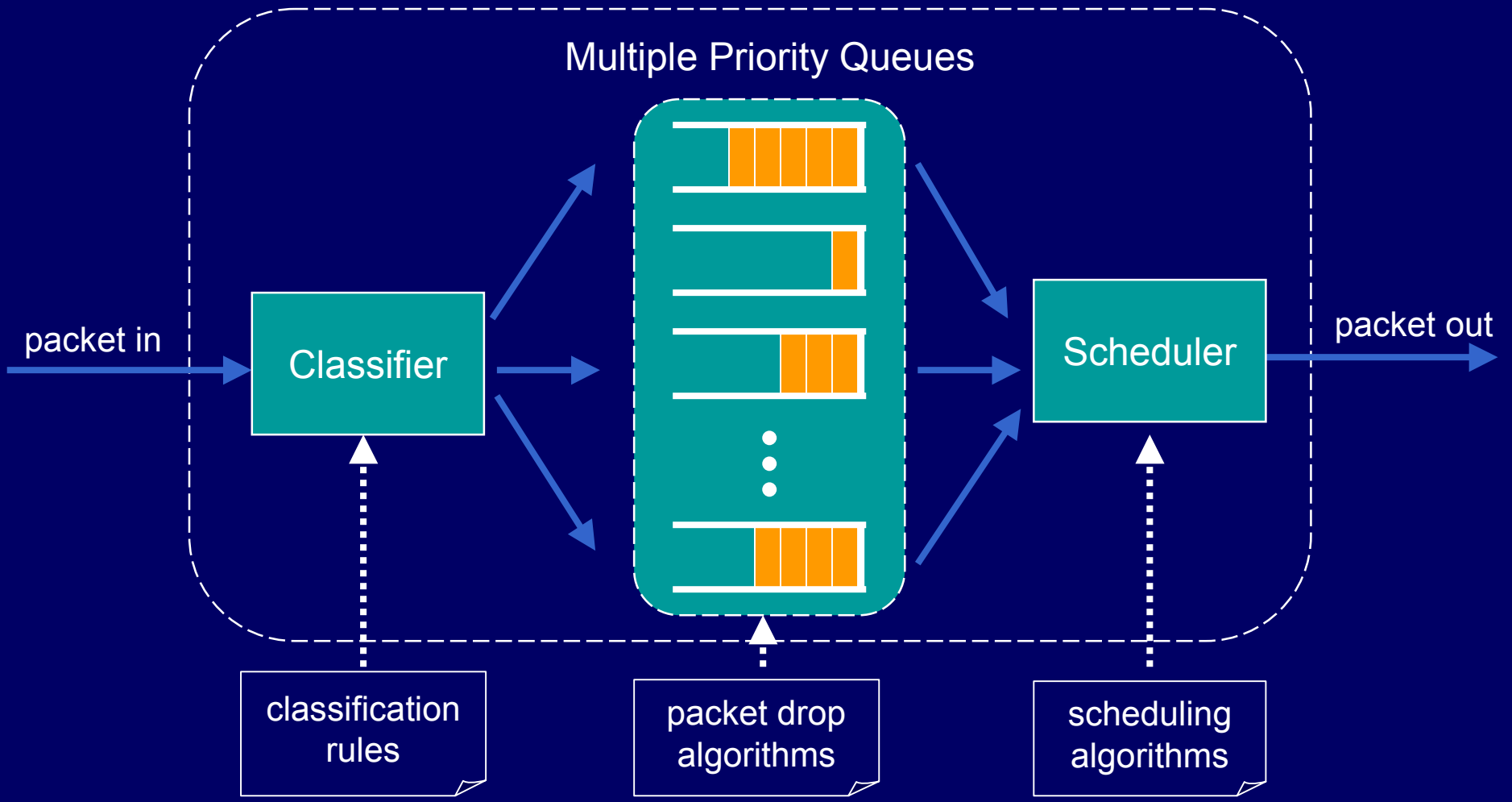


TCB : Traffic Conditioning Block

Traffic Conditioning Block



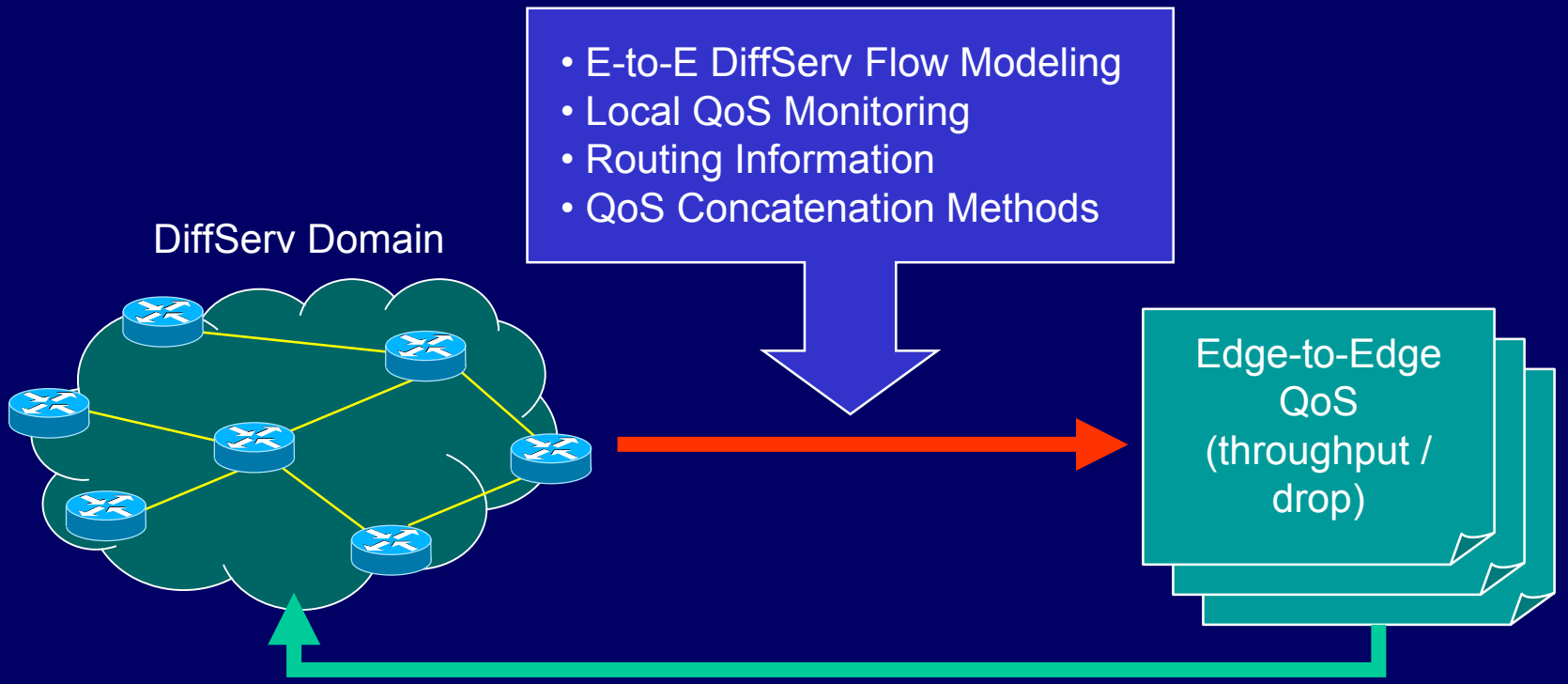
Conceptual Model of a TCB



Why Monitoring E-to-E DiffServ QoS?

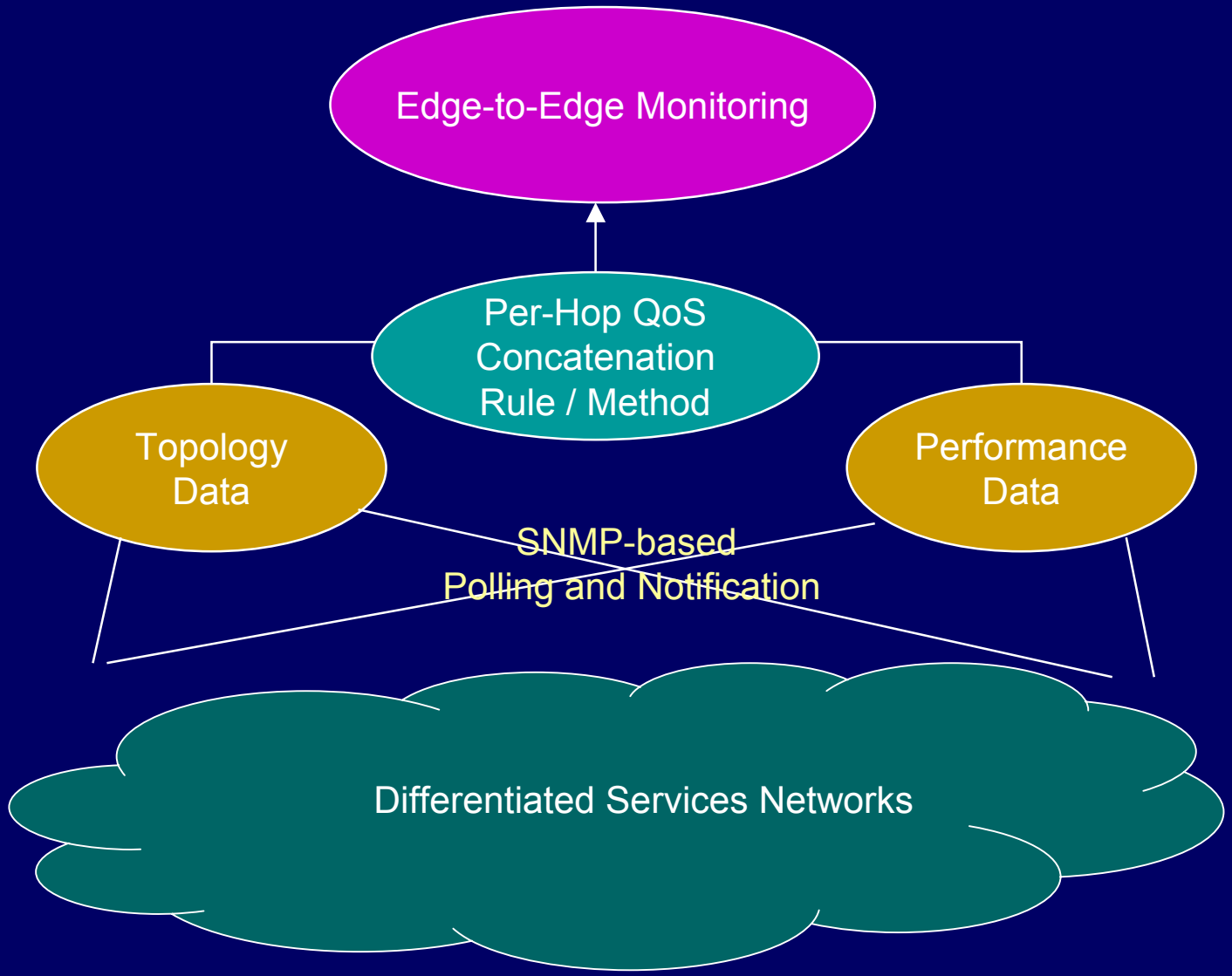
- Tool for deciding QoS provisioning parameters
  - it is hard to decide proper QoS parameters when provisioning DiffServ network
  - multiple ingress / egress situation results in complicated merge / split points
  - periodic monitoring results help to decide proper parameters
- Report detailed network status
  - E-to-E DiffServ QoS information provides detailed snapshot of network status
  - network operators can understand the QoS of the specific E-to-E connection
  - PDB can be checked by the E-to-E QoS information
- Pinpoint reasons of bottleneck situation
  - when bottleneck occur, the bandwidth-consuming E-to-E flow can be identified
  - this information can be used for traffic engineering decision
- Accounting and pricing
  - usage accounting needs E-to-E QoS information
  - pricing model can be built by the monitored information
- End-to-end QoS
  - edge-to-edge QoS can be a building block for end-to-end QoS
  - used for intradomain SLA negotiation

Key Ideas



- Tool for Provisioning
- Detailed Network Status Report
- Bottleneck Detection and Resolution
- Finer-Grained DiffServ

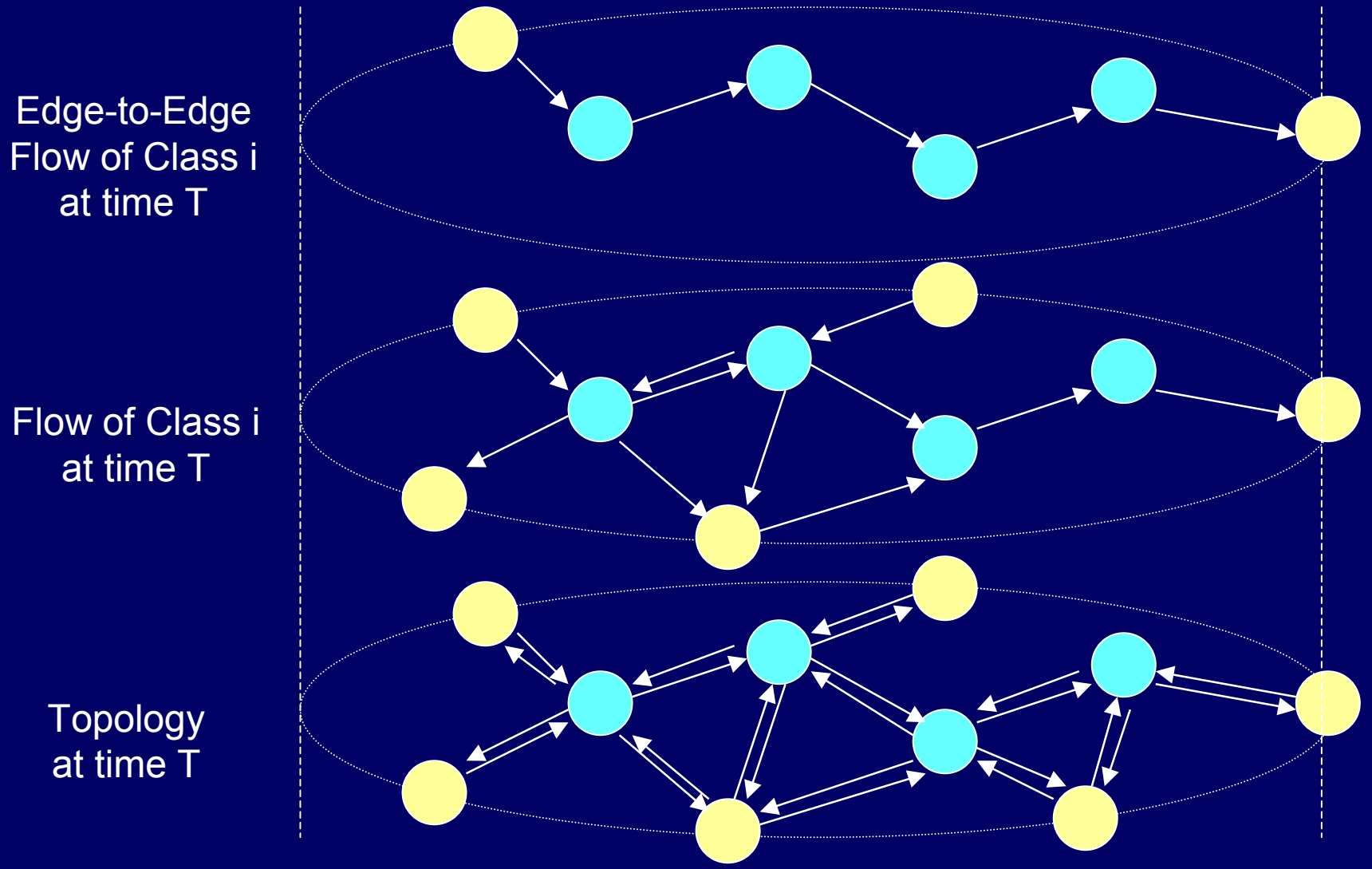
Proposed Monitoring Architecture



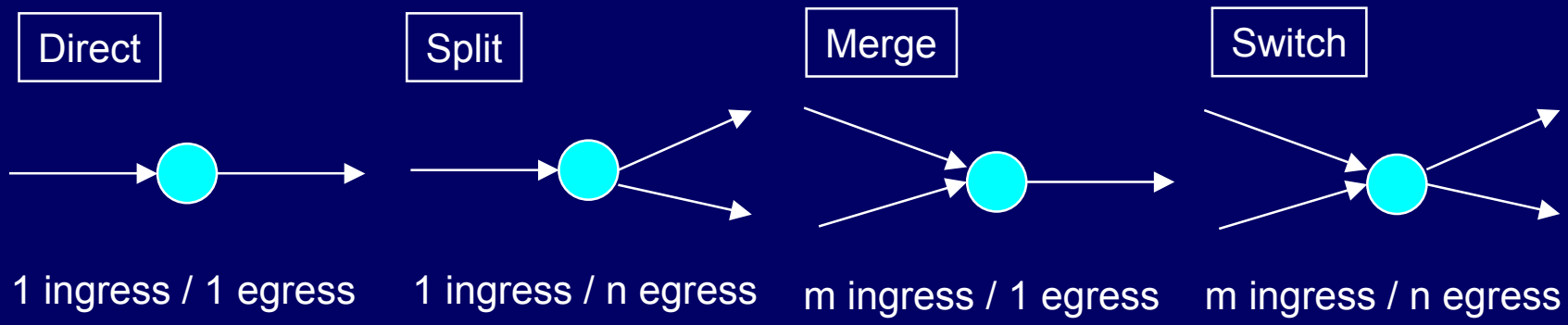
Outline

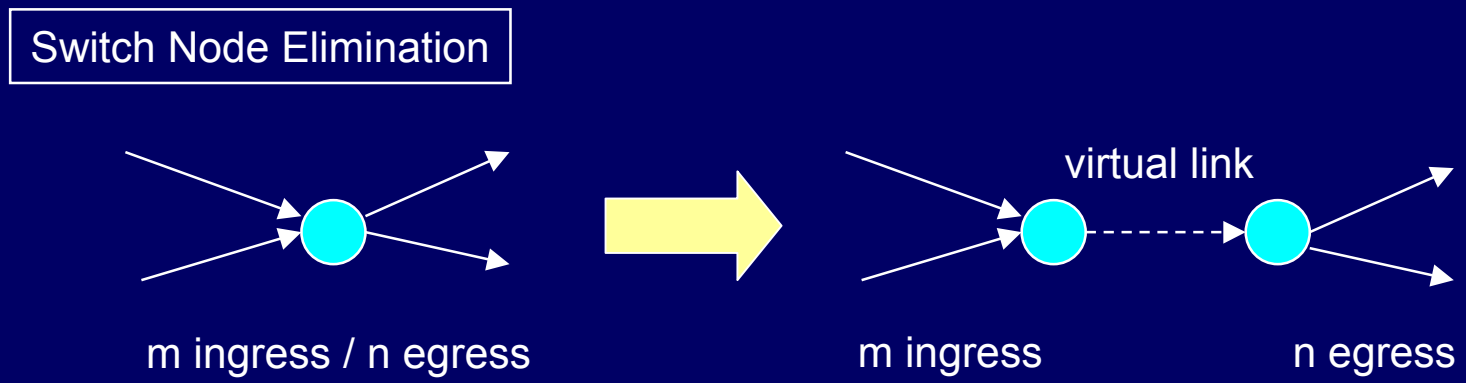
- Introduction
- DiffServ Framework and Its Limitations
- **Monitoring Methods for Edge-to-Edge DiffServ Flows**
- Applying Proposed Methods to Managing DiffServ Networks
- Edge-to-Edge DiffServ Flow Monitoring System
- Conclusions

Modeling Topology / Flow / E-to-E Flow

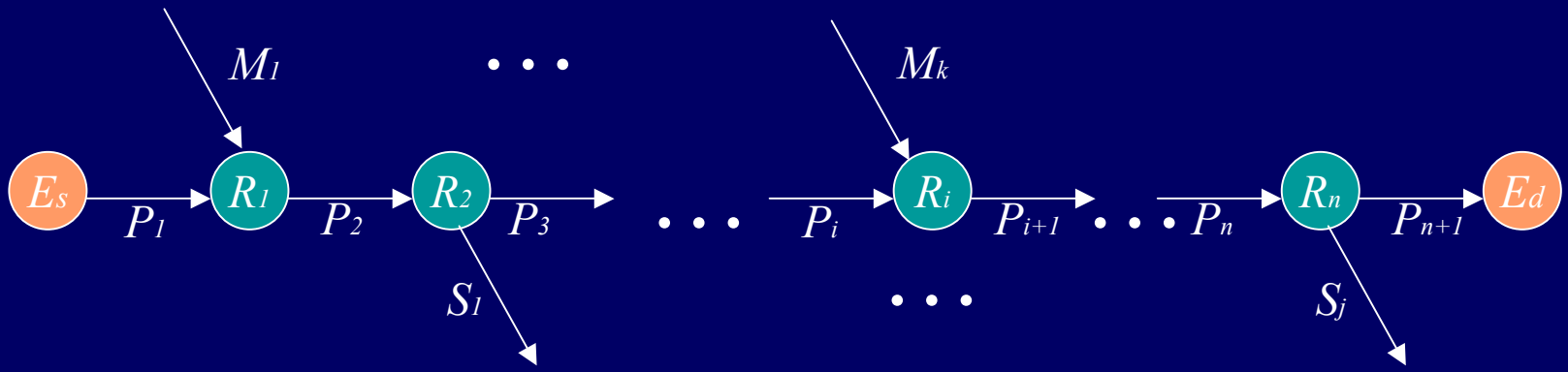


Graphical Representation of DiffServ Nodes





Representation of an E-to-E DiffServ Flow

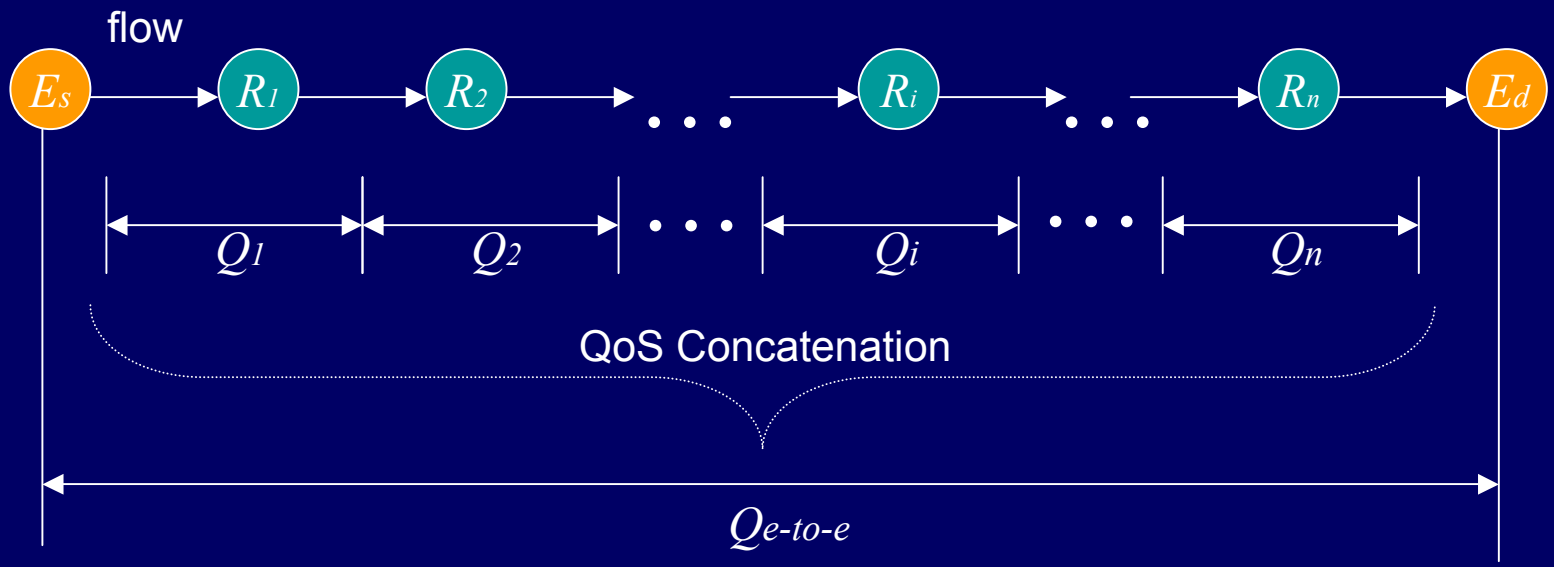


Es : source (ingress) edge router
Ed : destination (egress) edge router
Ri : core router

Pi : path link
Mk : merge link
Sj : split link

• each link has QoS information Q(Ci,Li), where Ci is an aggregate class and Li is a link
• Q(Ci,Li) contains throughput and drop rate of traffic in aggregate class Ci in the link Li within monitoring interval T
• In steady state (throughput from sources + throughput from merge links) equals to (throughput into sinks + throughput into split links + drops in transit)

Distributed QoS Monitoring and Concatenation

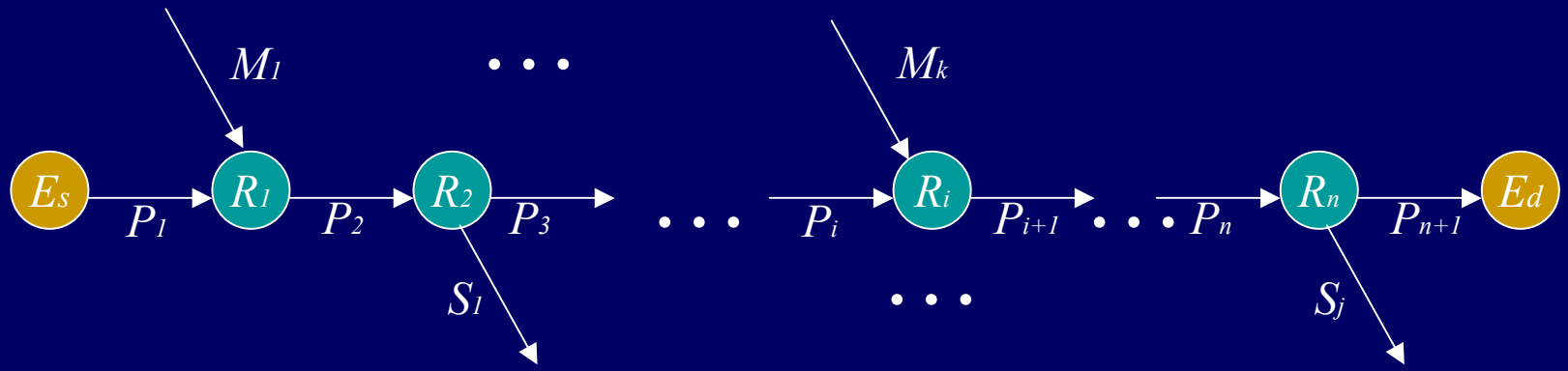


E-to-E QoS Concatenation Methods

- Purpose
  - methods for combining locally-observed QoS parameters
  - local QoS parameters + edge-to-edge routing path
  - simple and scalable methods
- Two different methods
  - edge-to-edge QoS with min/max bound
  - edge-to-edge QoS with ingress marking / egress counting
- Two phases
  - first phase: calculate edge-to-edge throughput (forward routing path)
  - second phase: calculate edge-to-edge drop by using BSR (backward routing path)

$$\text{Bandwidth Share Ratio (BSR)} = \frac{\text{amount of edge-to-edge throughput}}{\text{amount of aggregated throughput of a link}}$$

Min/Max Bound for E-to-E DiffServ Flow



• From the local QoS information, the max/min throughput of an edge-to-edge DiffServ flow can be obtained by following rules.

Max E-to-E Throughput = min [throughput at Pi]
Min E-to-E Throughput = throughput at P1 - Σ (throughput at Sj)

QoS Concatenation Algorithm (Min/Max)

- First phase: calculate edge-to-edge throughput

```
TH_max = TH_min = throughput of P1

for ( each path link, Pi ) {
    BSR_max = TH_max / throughput of all merged links
    DROP_max = BSR_max * Di
    TH_max = TH_max - DROP_max

    BSR_min = TH_min / throughput of all merged links
    DROP_min = BSR_min * Di
    TH_min = TH_min - DROP_max

    TH_max = min( TH_max, throughput of Pi+1 )
    if ( Ri is a split node ) {
        TH_min = TH_min - throughput of all split links
    } else {
        TH_min = min( TH_min, throughput of Pi+1 )
    }
}

final_throughput_max = TH_max
final_throughput_min = TH_min
```

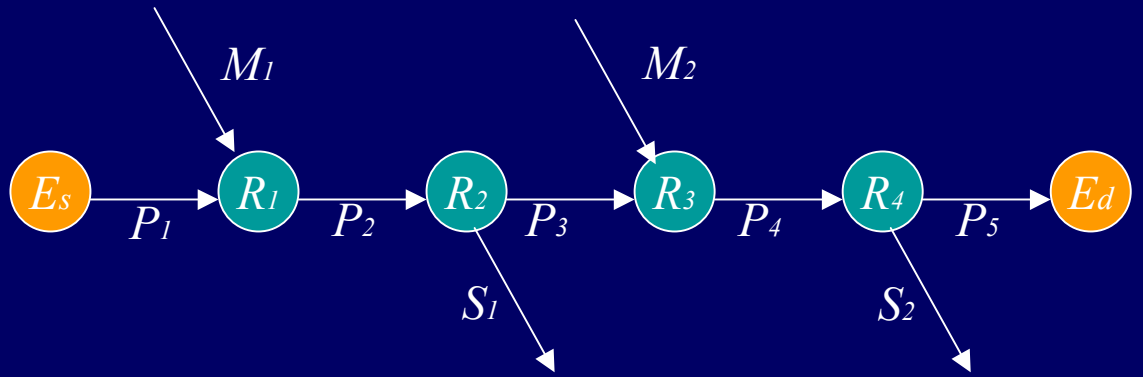
QoS Concatenation Algorithm (Min/Max)

- Second phase: calculate edge-to-edge drop

```
TH_max = final_throughput_max
TH_min = final_throughput_min

for ( each path link, Pi, in reverse order ) {
    BSR_max = TH_max / throughput of Pi
    DROP_max = BSR_max * Di
    TH_max = TH_max + DROP_max

    BSR_min = TH_min / throughput of Pi
    DROP_min = BSR_min * Di
    TH_min = TH_min + DROP_min
}

initial_throughput_max = TH_max
initial_throughput_min = TH_min
```

Example of E-to-E DiffServ Flows

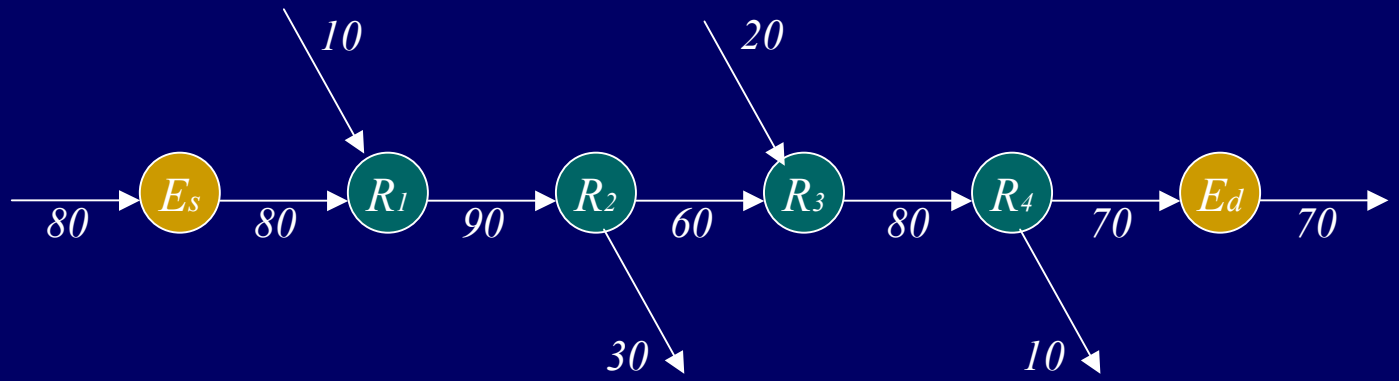
Example Topology



Example Cases

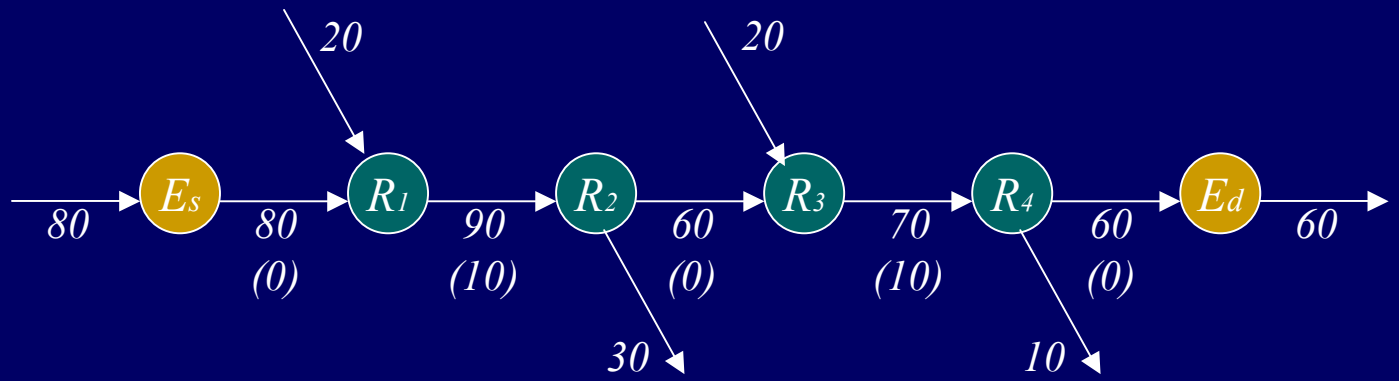
- Case 1 : with no drop rate
  - there is no packet drop in every link, non-congested situation
- Case 2 : with drop rate
  - some of links have packet drops due to congestion

Case 1 : with no drop rate



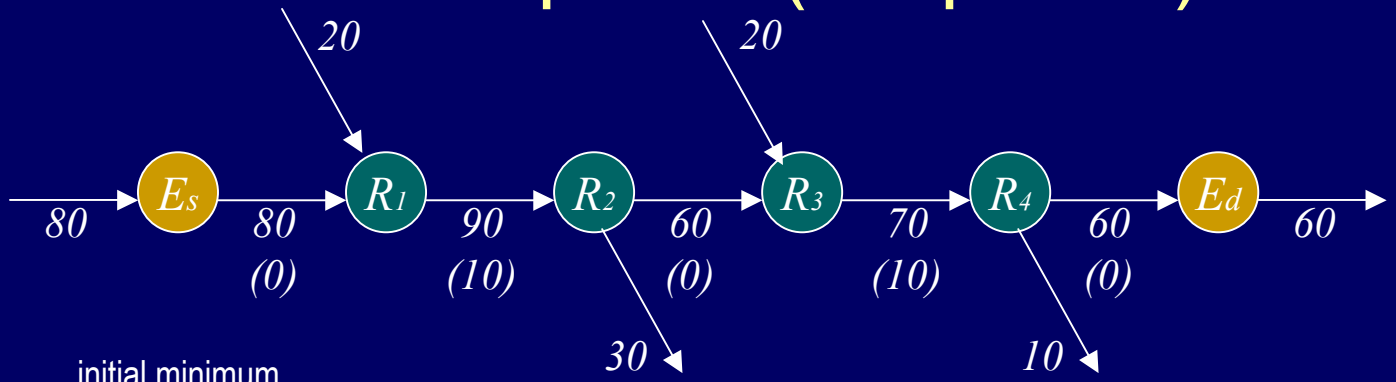
Min Throughput	80	80	50	50	40	final minimum
Min BSR		40/90	40/60	40/80	40/70	
Max Throughput	80	80	60	60	60	final maximum
Max BSR		60/90	60/60	60/80	60/70	

Case 2 : with drop rate (1st phase)



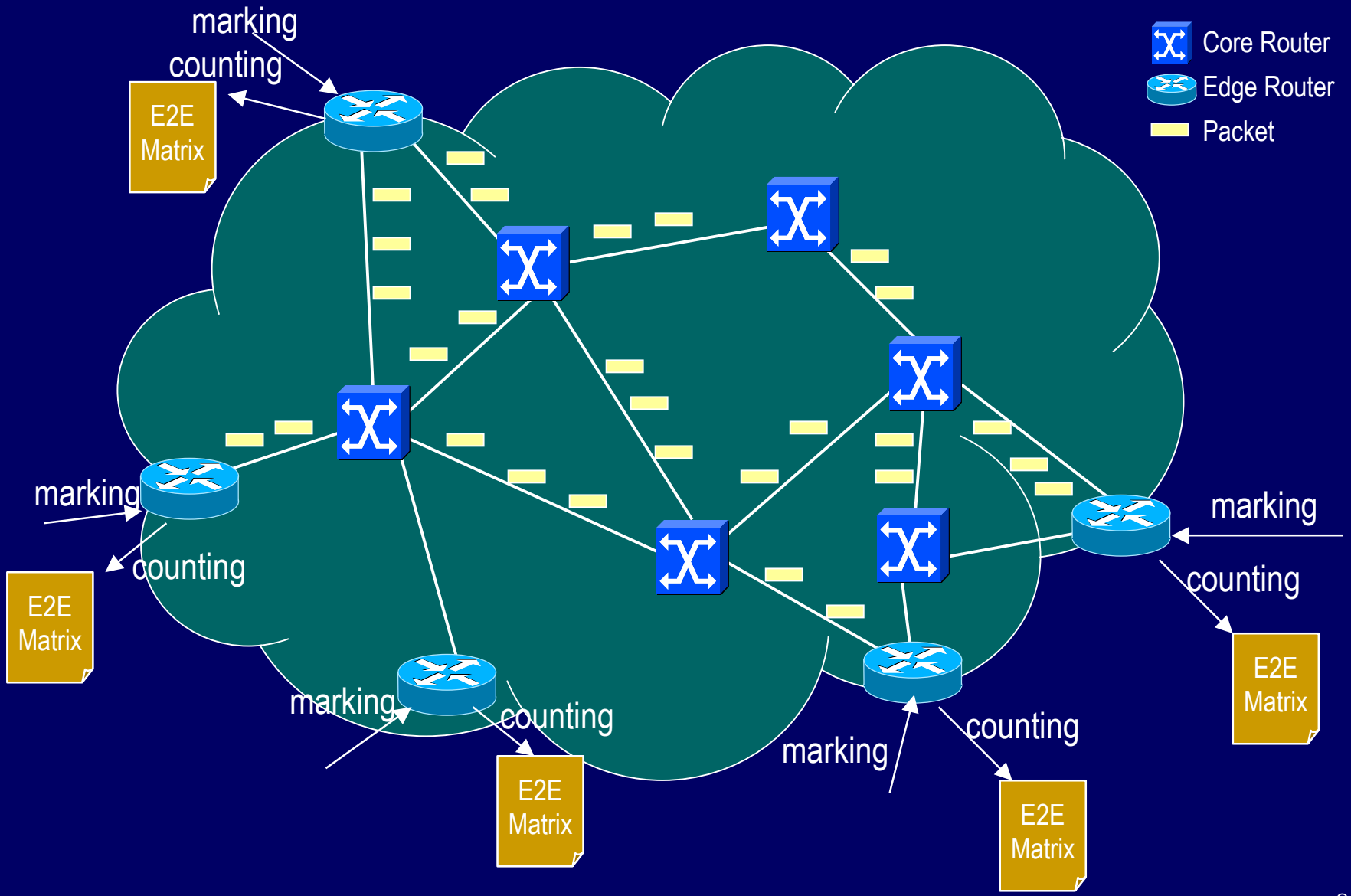
Min Throughput	80	72	42	37	27	final minimum
Min BSR		80/100		42/80		
Drop at Link	0	4/5 × 10	0	42/80 × 10	0	
Total Drop for Min	0	8	8	13	13	
Max Throughput	80	72	60	52	52	final maximum
Max BSR		80/100		60/80		
Drop at Link	0	4/5 × 10	0	6/8 × 10	0	
Total Drop for Max	0	8	8	16	16	

Case 2 : with drop rate (2nd phase)



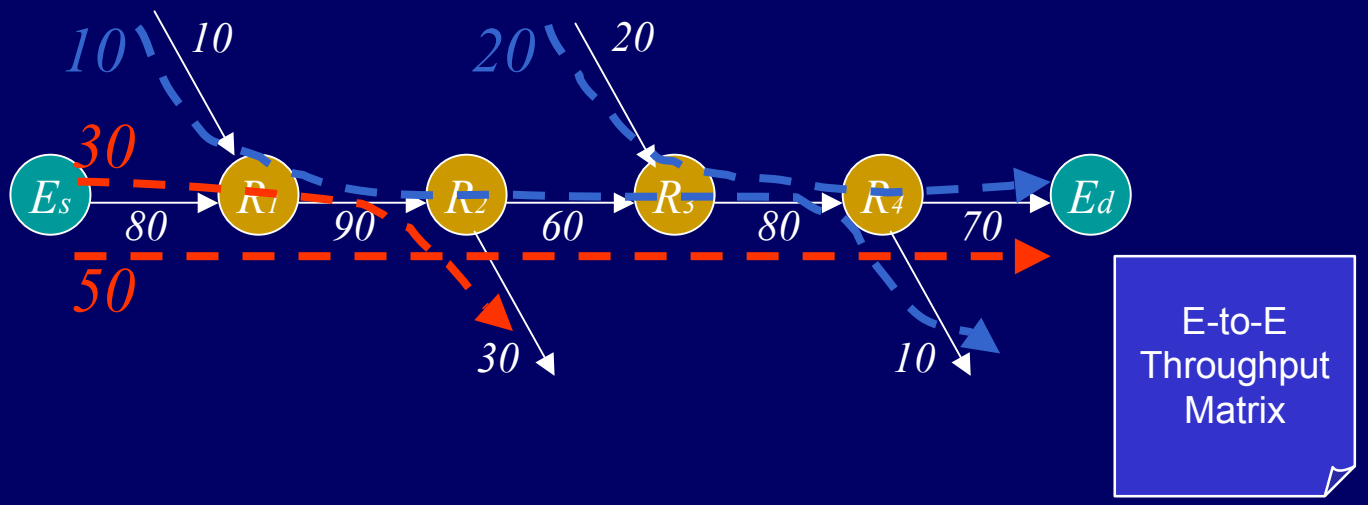
	initial minimum					
Min Throughput	34	31	31	27	27	final minimum
Min BSR		31/90		27/70		
Drop at Link	0	31/90 × 10	0	27/70 × 10	0	
Total Drop for Min	7	7	4	4	0	
	initial maximum					
Max Throughput	66	59	59	52	52	final maximum
Max BSR		59/90		52/70		
Drop at Link	0	59/90 × 10	0	52/70 × 10	0	
Total Drop for Max	14	14	7	7	0	

E-to-E QoS with Ingress Marking / Egress Counting



With E-to-E Throughput Matrix

• If every ingress edge router can distinguish the amount of traffic to every egress edge router, we can monitor the status of real E-to-E flows in more detail.



Ingress Marking / Egress Counting (Cont.)

- Mechanism
  - additional marking in IP packet header
  - every ingress edge router marks its ID on packets leaving
  - every egress edge router checks the ID of incoming packets and counts the number of packets/bytes for each ingress edge
  - each egress edge router can build an E-to-E throughput matrix for E-to-E flows from every ingress edge router to the egress edge router
  - with this E-to-E throughput matrix, E-to-E throughput is easily obtained

- Benefits
  - easy and correct way to measure real E-to-E throughput
  - simple marking at ingress edge routers
  - affordable load of counting at egress edge routers
  - no modification at core routers

Ingress Marking / Egress Counting (Cont.)

- Implementation Consideration

1) Modified DS field

– the number of current standard DSCP is only 21 of 64 available

– if we use currently-unused 2-bit in DS field with a proper encoding rule, 8 edge ID can be incorporated in DS field

– if we limit the number of service classes in a DiffServ domain, the number of available edge ID can be increased

2) Use of MPLS header

– for general implementation with unlimited number of DiffServ classes and edge routers, MPLS shim header can be used

– this needs MPLS capability in every DiffServ router

QoS Concatenation Algorithm (IM/EC)

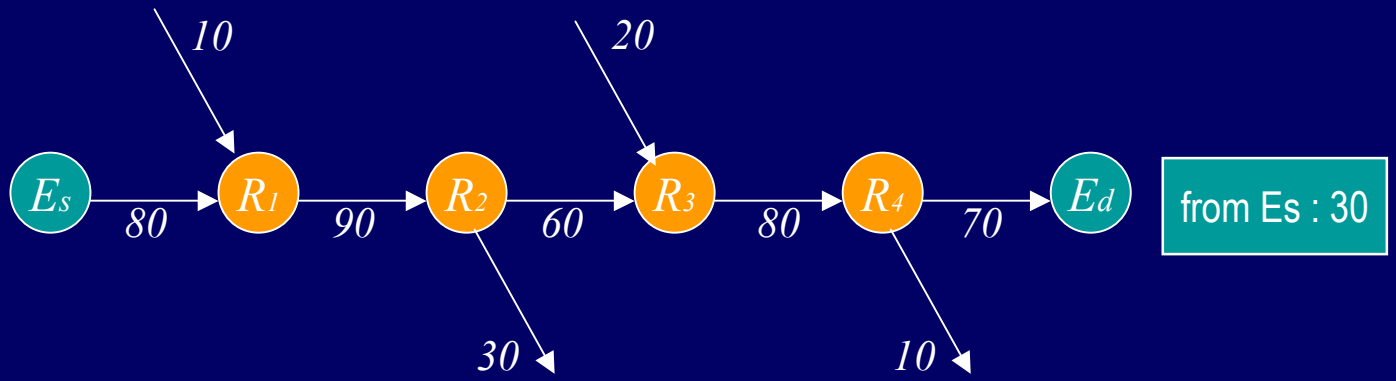
- First phase

```
TH = edge-to-edge throughput calculated from the e-to-e matrix
```

- Second phase

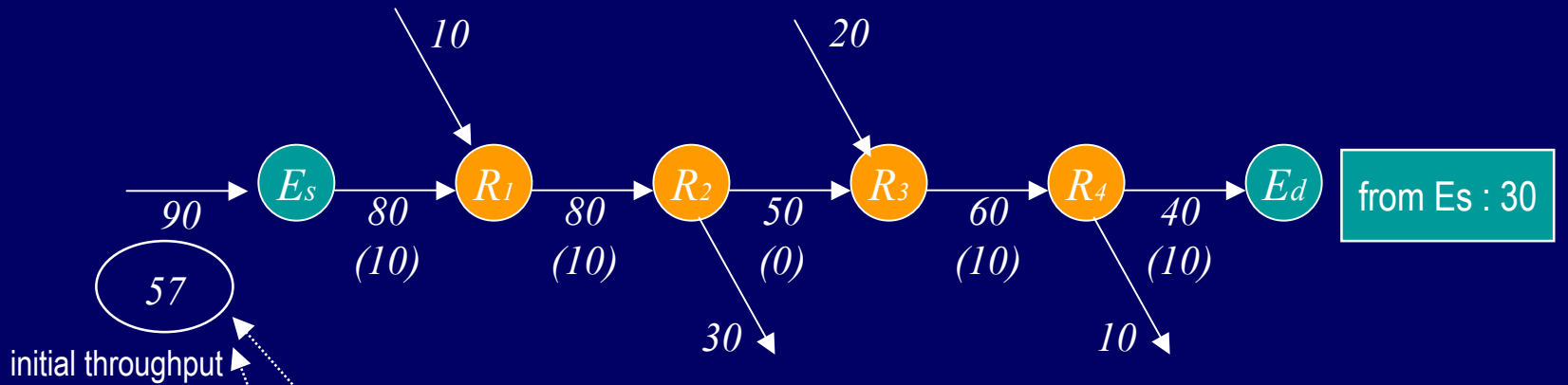
```
for ( each path link, Pi, in reverse order ) {
    BSR = TH / throughput of Pi
    DROP = BSR * Di
    TH = TH + DROP
}
initial_throughput = TH
```

Case 1 : with no packet drop



Throughput	30	30	30	30	30
BSR	30/80	30/90	30/60	30/80	30/70

Case 2 : with packet drops

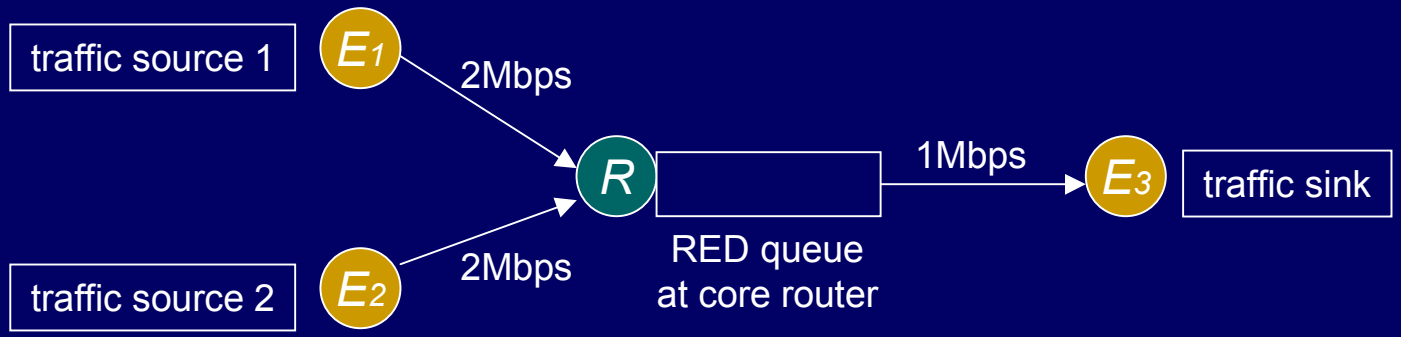


Throughput	50	44	44	38	30
BSR	64/80	44/80	44/50	38/60	30/40
Drop at Link	7	6	0	6	8
Total Drop	27	20	14	14	8

Assumptions for the Monitoring Methods

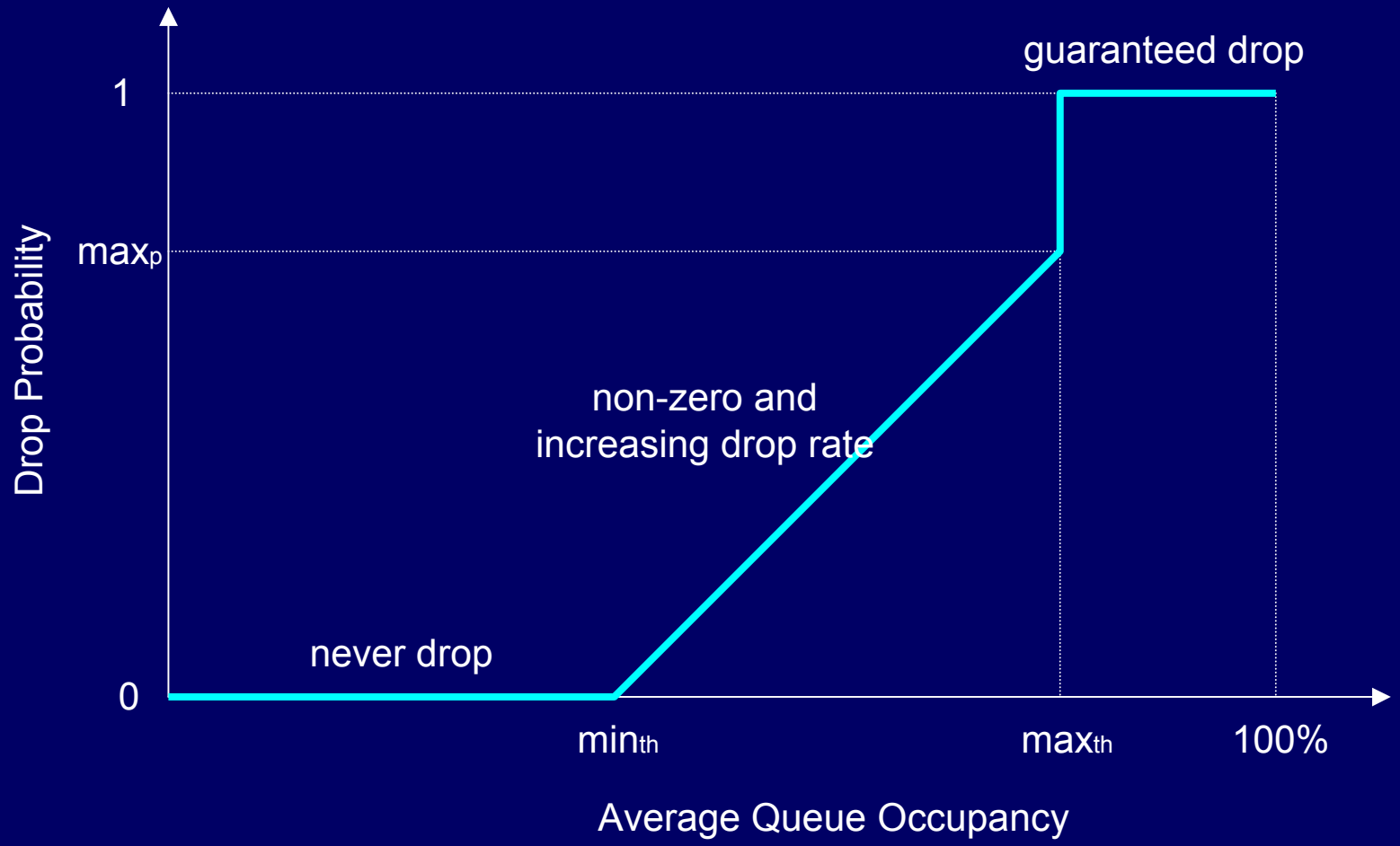
- Routing information
  - each DiffServ router has a routing table
  - dynamic routing change is reflected on the routing table
- Per-class statistics
  - each DiffServ router keeps records of DiffServ traffic of each class
  - counter for each packet transmitted and dropped
- Monitoring interval
  - longer than edge-to-edge delay
  - shorter than routing change
- Proportional drop
  - drop probability of a connection should be proportional to the amount of traffic share of the total throughput
  - for calculating edge-to-edge drop rates by following the proposed methods

Simulation for Checking the Proportional Drop Characteristic at DiffServ RED Queue



- Purpose
  - to check if an RED queue has proportional drop characteristics
  - S. Floyd and V. Jacobson, IEEE/ACM Transactions on Networking, August 1993
- Simulation
  - two traffic sources are merged at an RED queue
  - change BSR of two sources and count the drop rates of each traffic flow
  - total amount of traffic flows is 1.2Mbps, while the bottleneck link capacity is 1Mbps

RED Queue Drop Characteristics

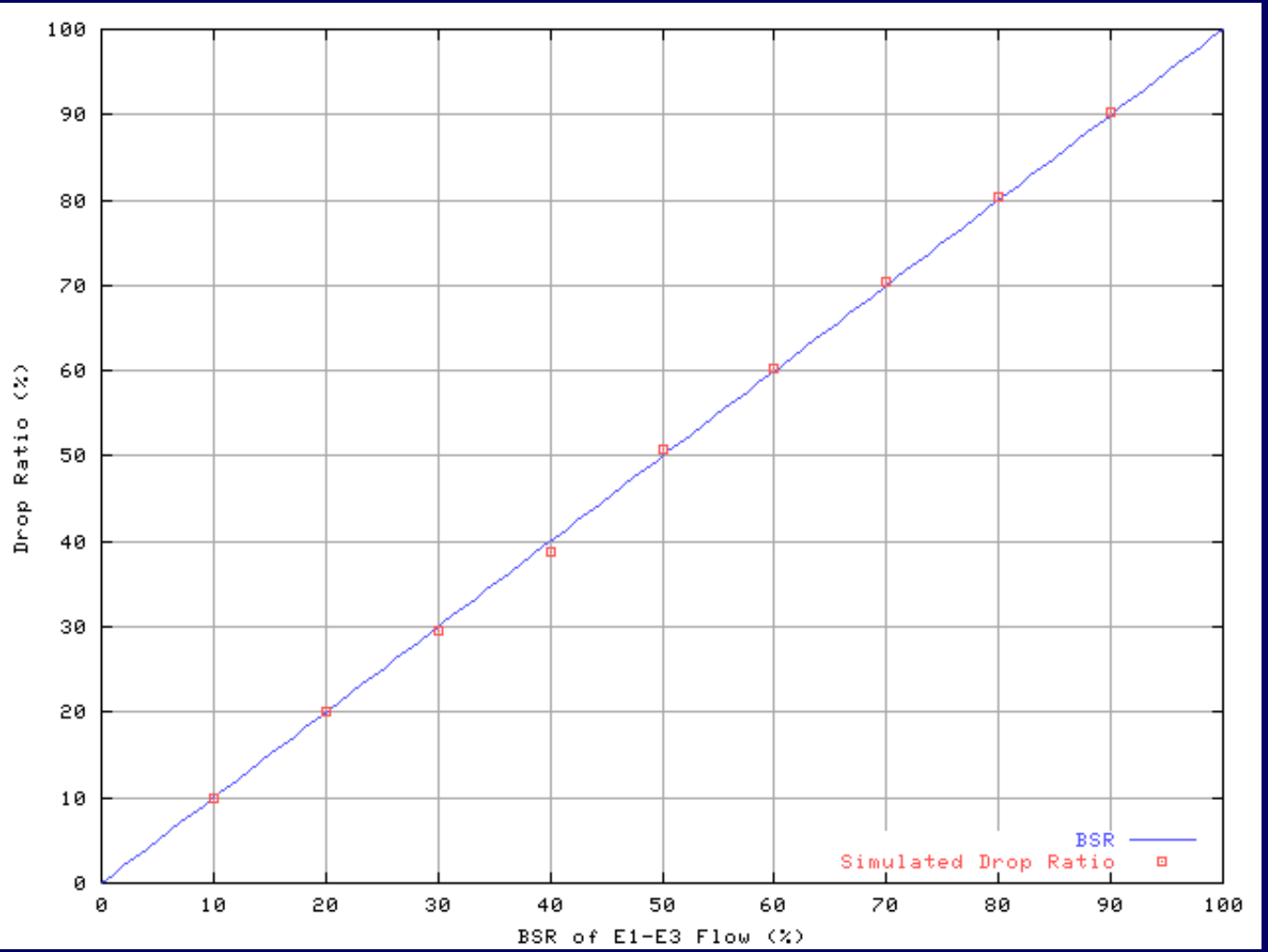


min$_{th}$: minimum threshold
max$_{th}$: maximum threshold

Proportional Drop Simulation Result

BSR	E1→E3	E2→E3	Total RED Drop (packet)	Drop from E1 (packet)	Drop from E2 (packet)
1:9	0.12Mbps	1.08Mbps	3238	325 (10.04%)	2913 (89.06%)
2:8	0.24Mbps	0.96Mbps	3219	649 (20.16%)	2570 (79.84%)
3:7	0.36Mbps	0.84Mbps	3246	956 (29.45%)	2290 (70.55%)
4:6	0.48Mbps	0.72Mbps	3219	1246 (38.71%)	1973 (61.29%)
5:5	0.6Mbps	0.6Mbps	3267	1658 (50.75%)	1609 (49.25%)
6:4	0.72Mbps	0.48Mbps	3229	1944 (60.20%)	1285 (39.80%)
7:3	0.84Mbps	0.36Mbps	3227	2271 (70.37%)	956 (29.63%)
8:2	0.96Mbps	0.24Mbps	3210	2581 (80.40%)	629 (19.60%)
9:1	1.08Mbps	0.12Mbps	3391	3065 (90.39%)	326 (9.61%)

Proportional Drop at RED Queue



- Result
  - the RED queue proportionally drops packets from different traffic sources

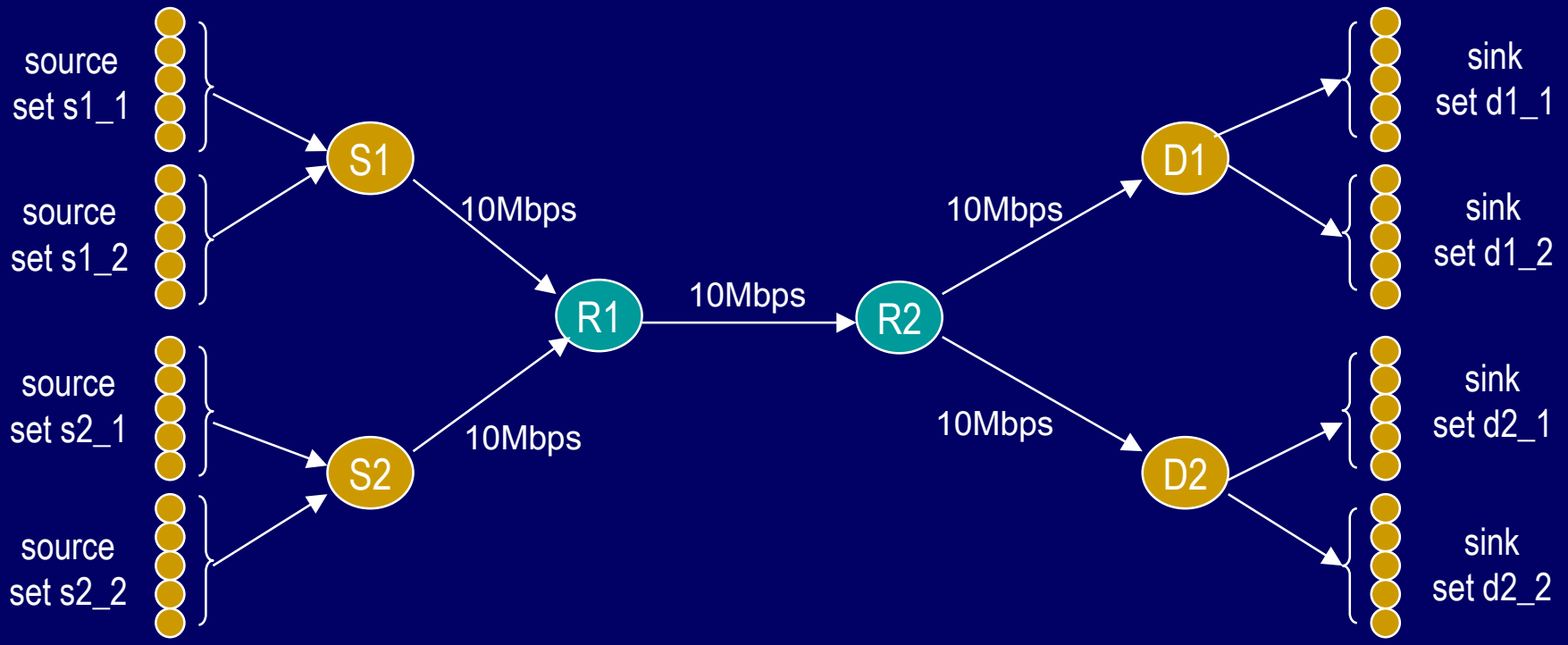
Outline

- Introduction
- DiffServ Framework and Its Limitations
- Monitoring Methods for Edge-to-Edge DiffServ Flows
- **Applying Proposed Methods to Managing DiffServ Networks**
- Edge-to-Edge DiffServ Flow Monitoring System
- Conclusions

Applications of Edge-to-Edge QoS in Managing DiffServ Networks

- Network Status Reporting
  - new information : E-to-E QoS status
- Dynamic Adaptive Provisioning
  - decide appropriate provisioning parameters periodically
- Bottleneck Detection and Resolution
  - pinpoint bottleneck link
  - search for bandwidth-consuming E-to-E flows
- Others
  - monitoring Per-Domain Behavior (PDB)
  - intradomain management
  - accounting / pricing

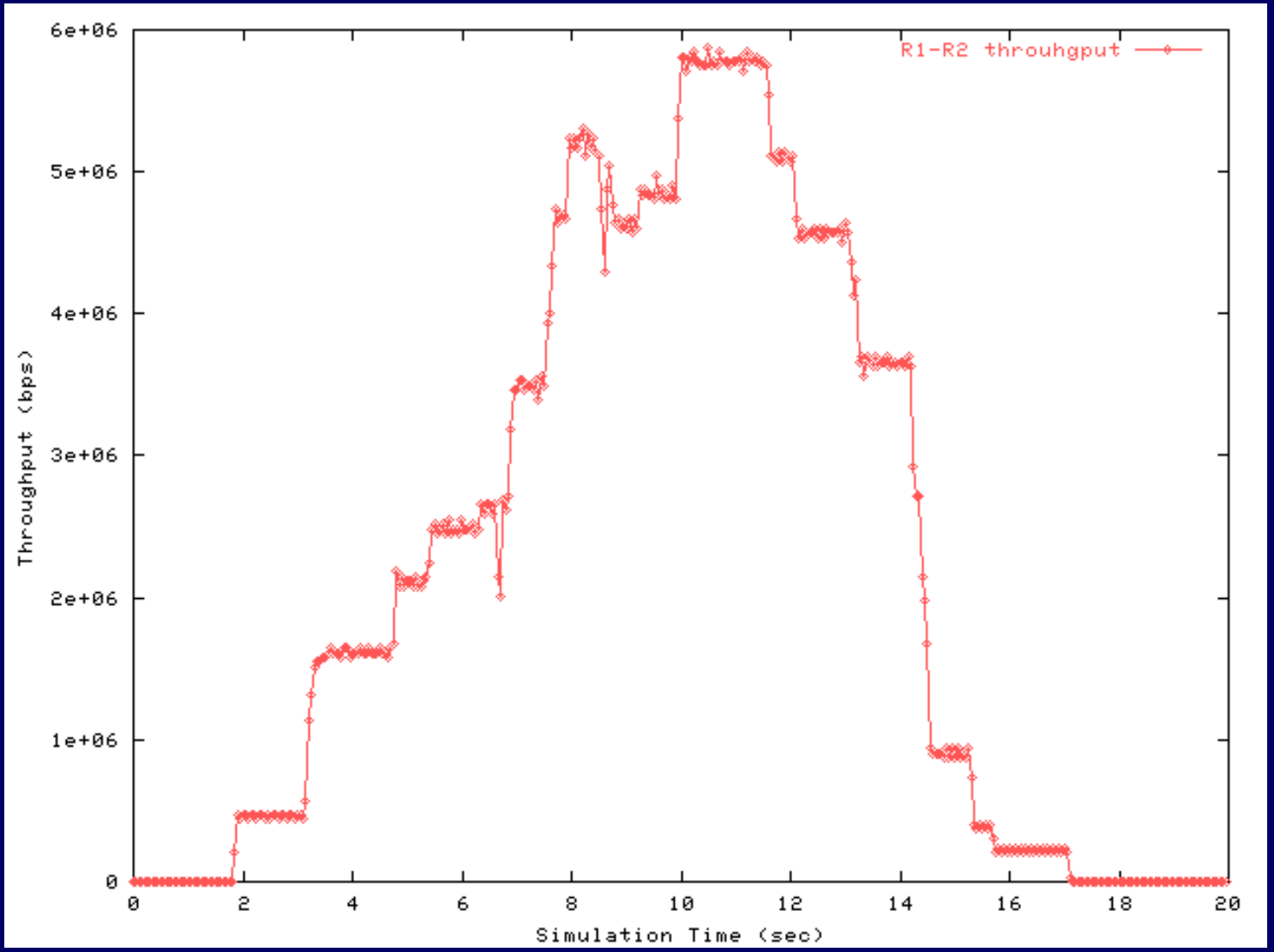
Simulation Model for Monitoring DiffServ E-to-E Flows



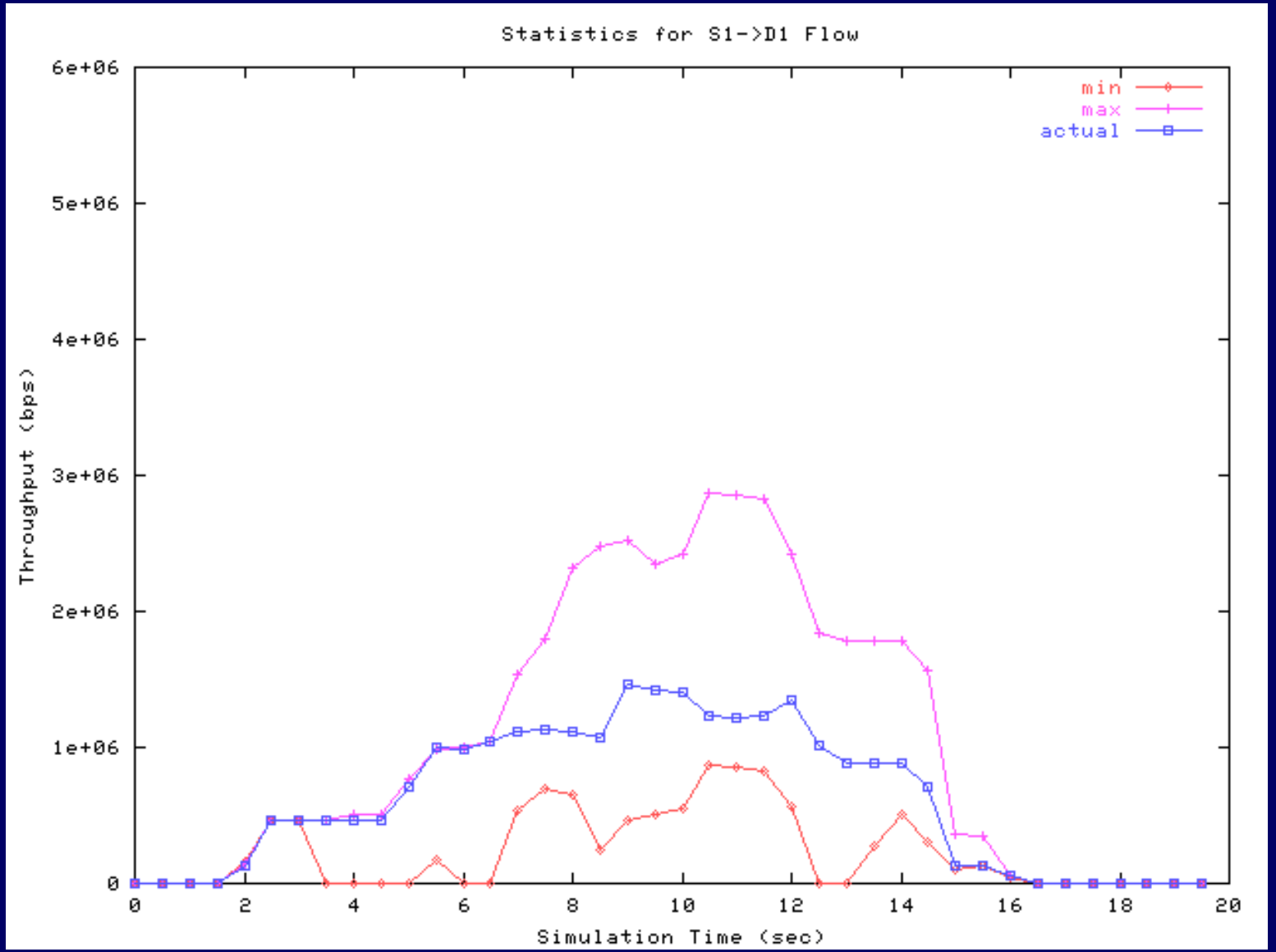
- Each source set has 5 independent CBR traffic generators (64bps ~ 1Mbps).
- There are 4 different E-to-E DiffServ flows (S1→D1, S1→D2, S2→D1, S2→D2)
- There is no drop in transit routing paths (non-congested situation)

- Example 1: similar amount of bandwidths are shared among four E-to-E flows
- Example 2: one dominant bandwidth-consuming E-to-E flow (S1→D2)

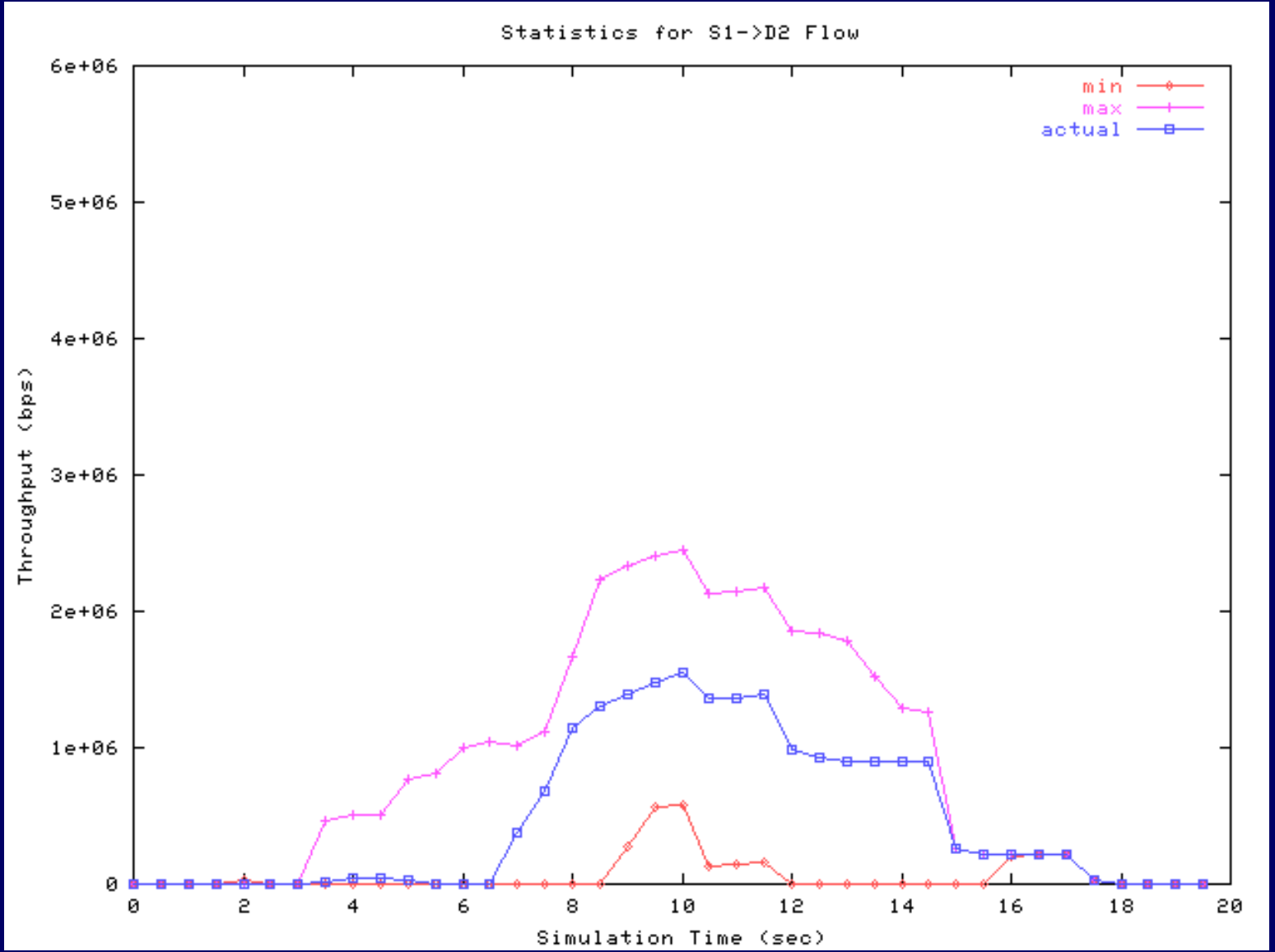
Ex. 1: R1 → R2 Aggregated Throughput



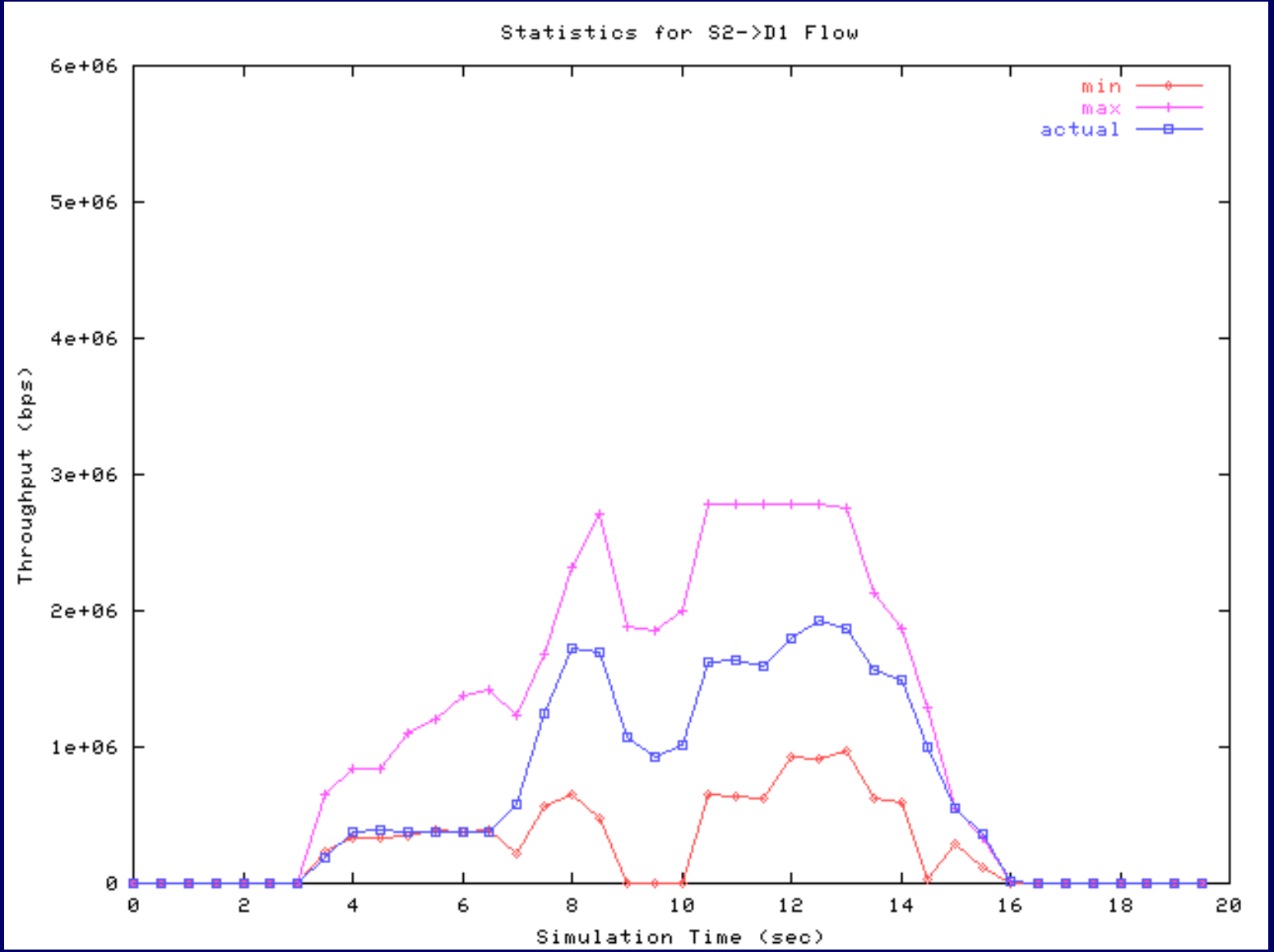
Edge-to-Edge Throughput (S1→D1)



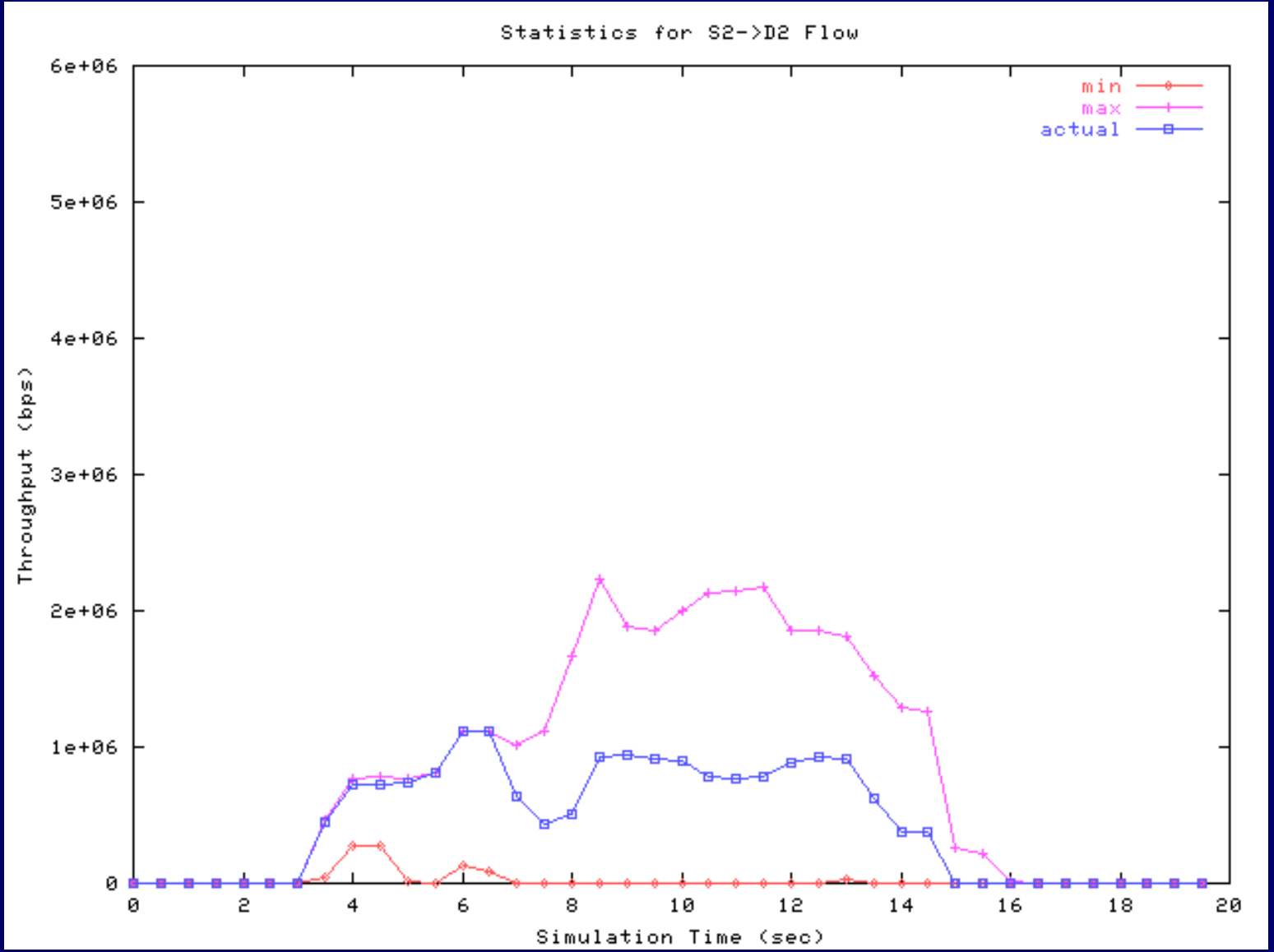
Edge-to-Edge Throughput (S1→D2)



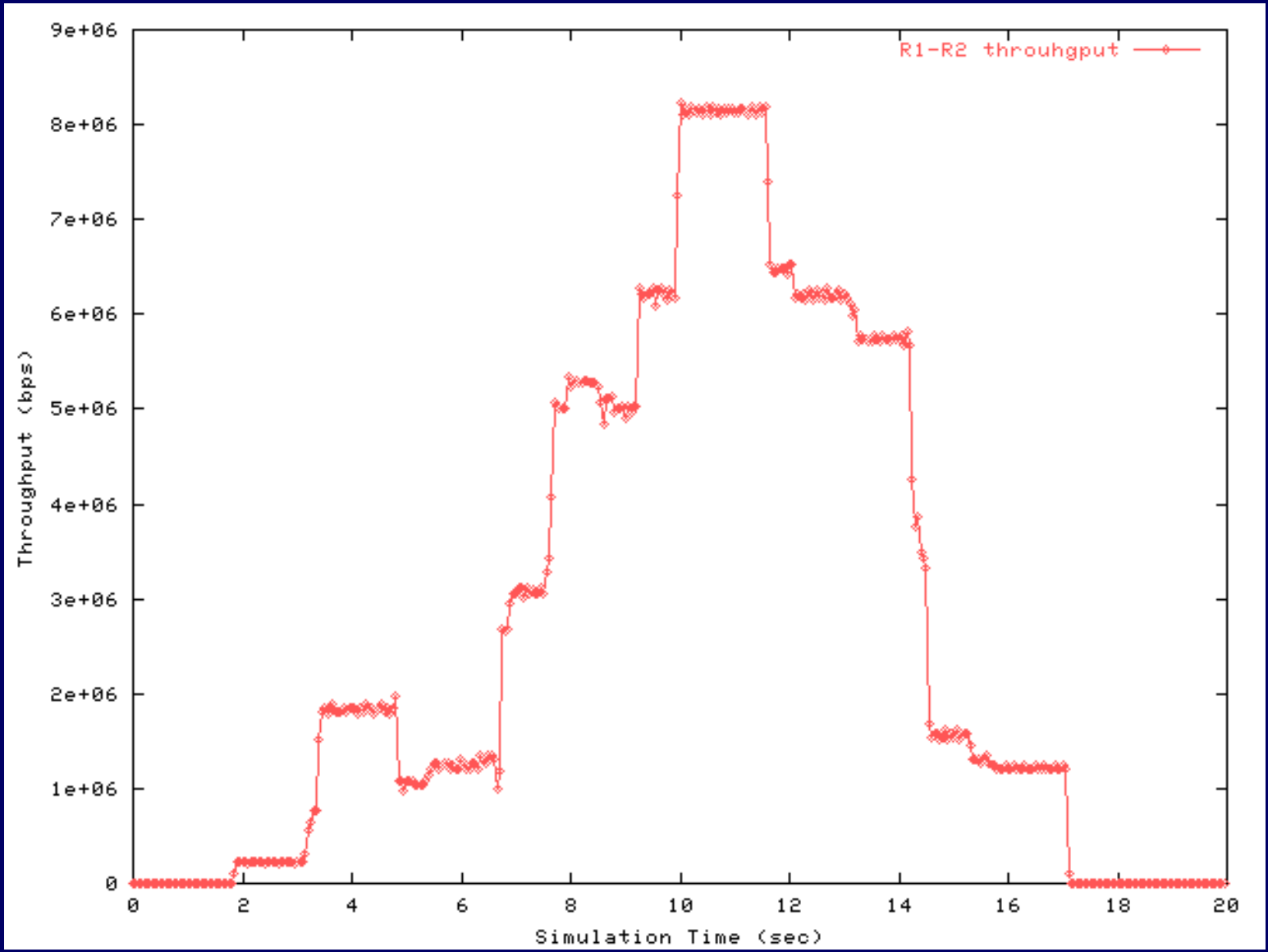
Edge-to-Edge Throughput (S2→D1)



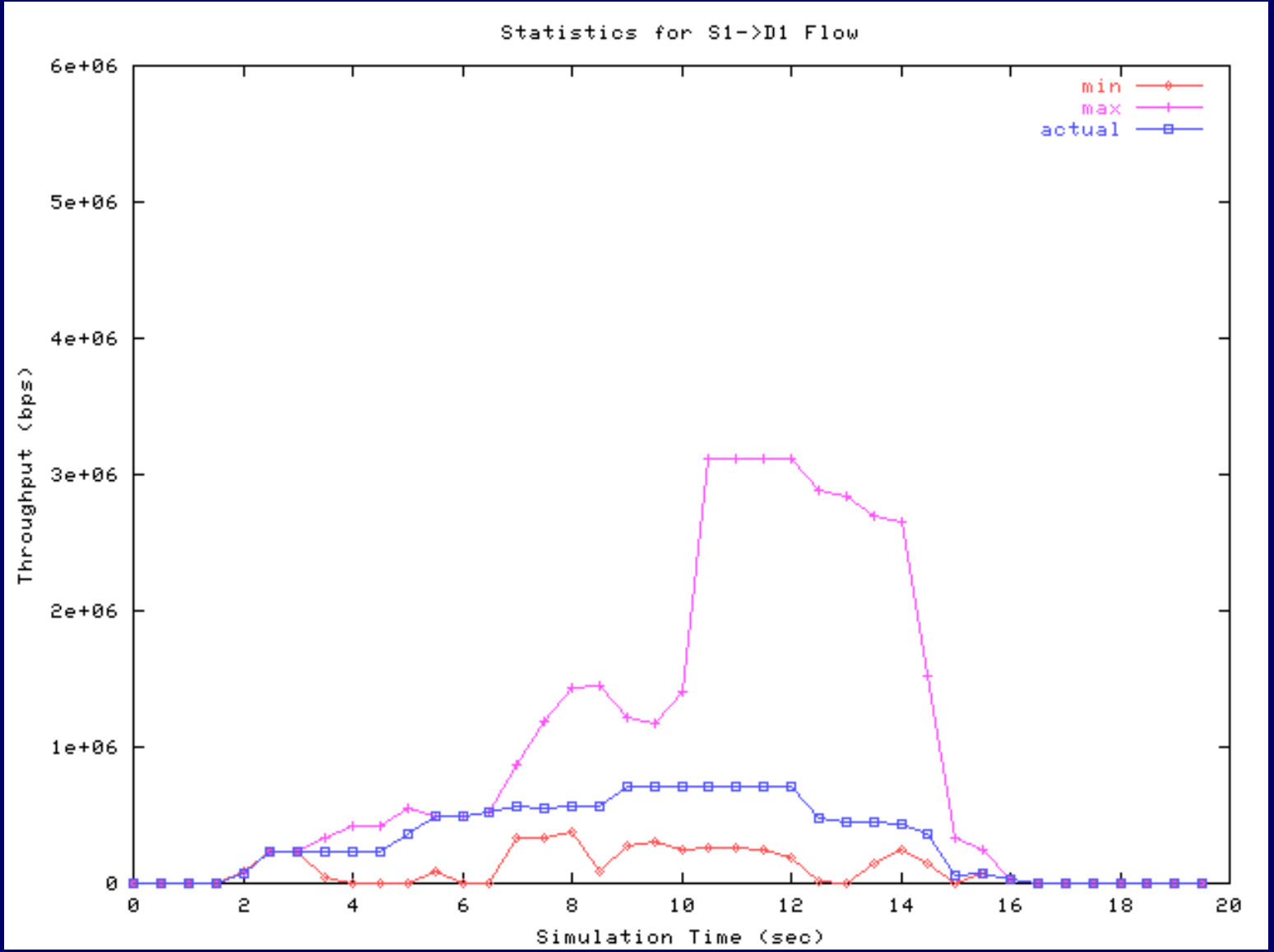
Edge-to-Edge Throughput (S2→D2)



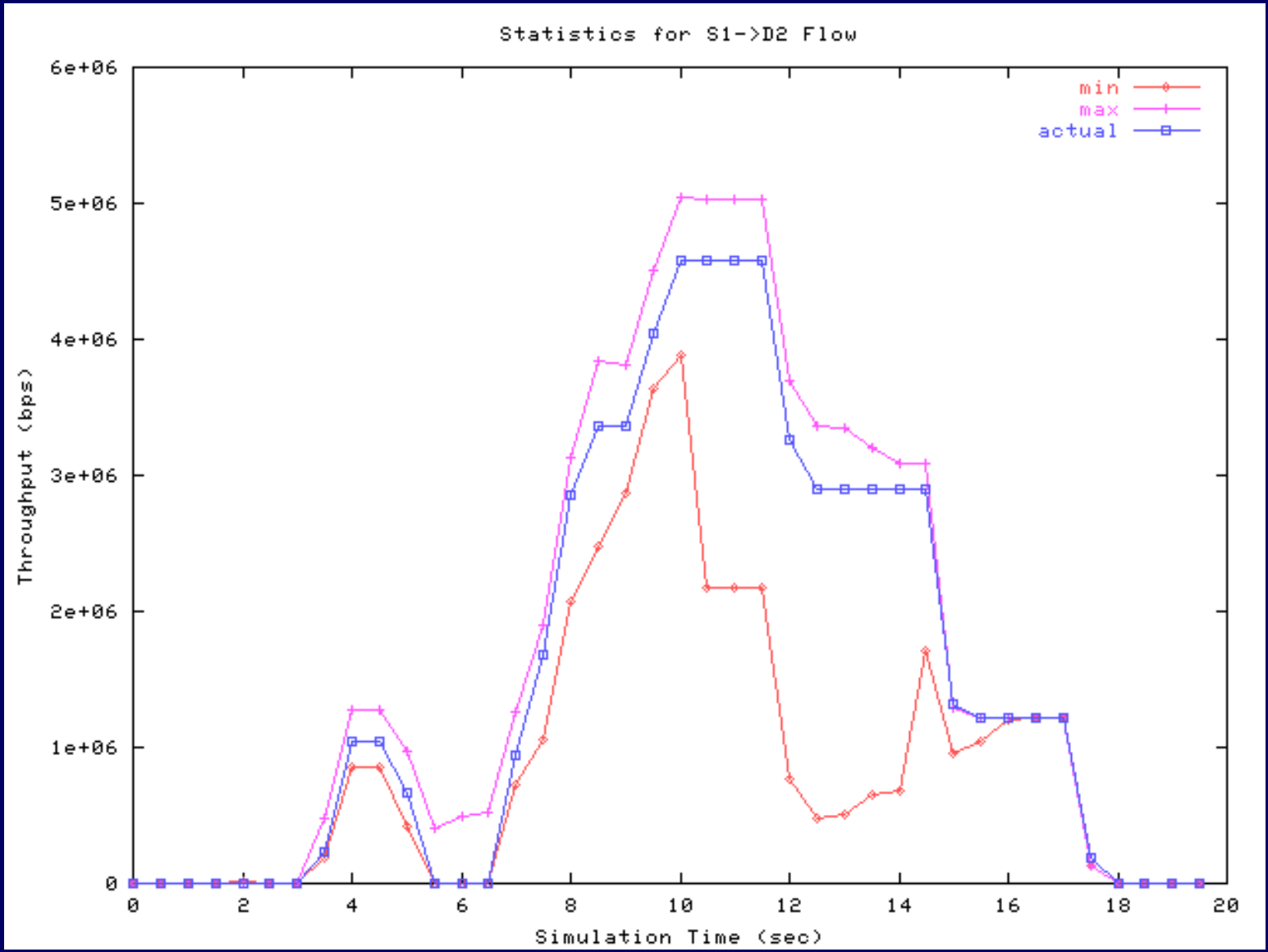
Ex. 2: R1 → R2 Aggregated Throughput



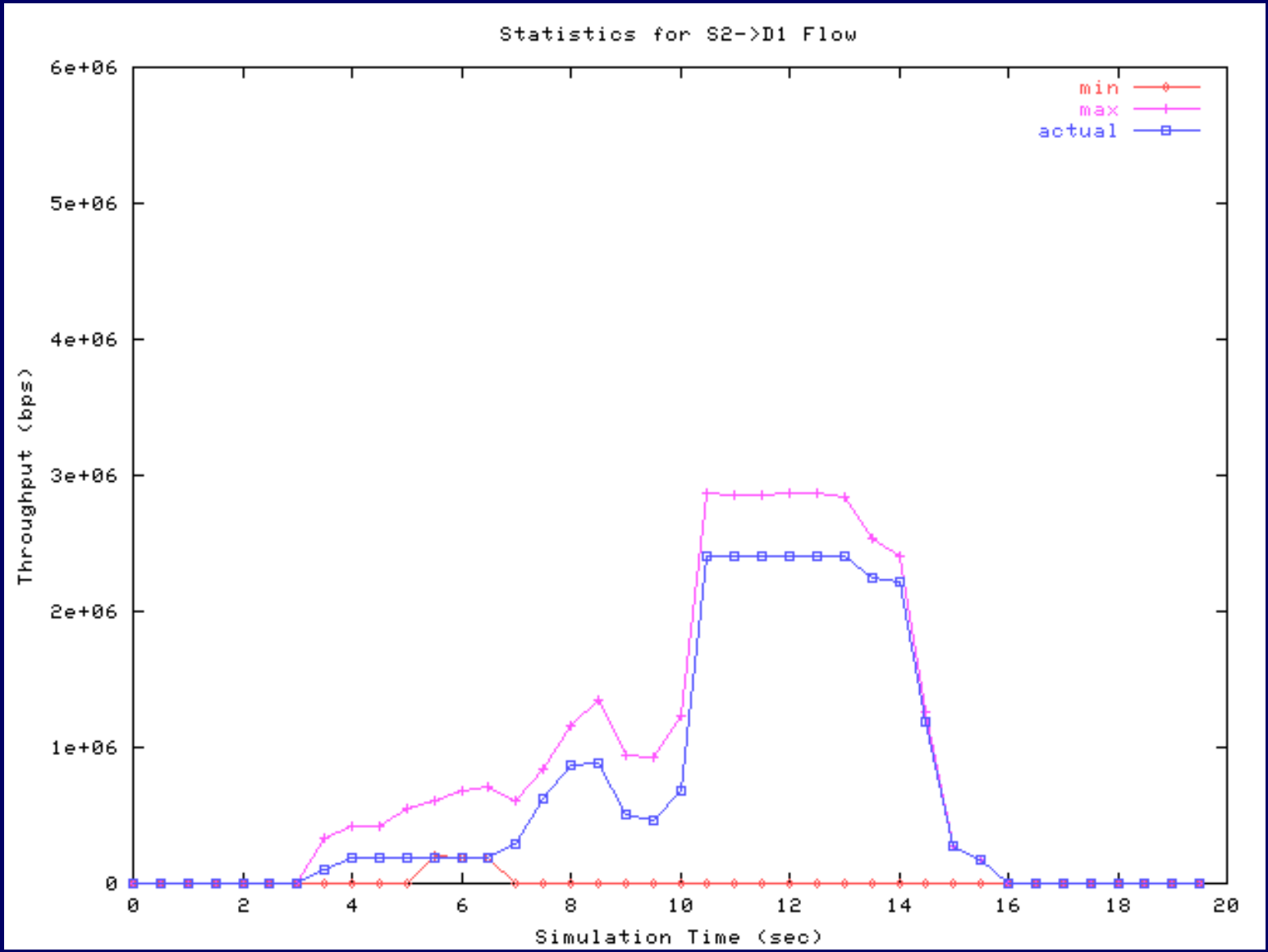
Edge-to-Edge Throughput (S1→D1)



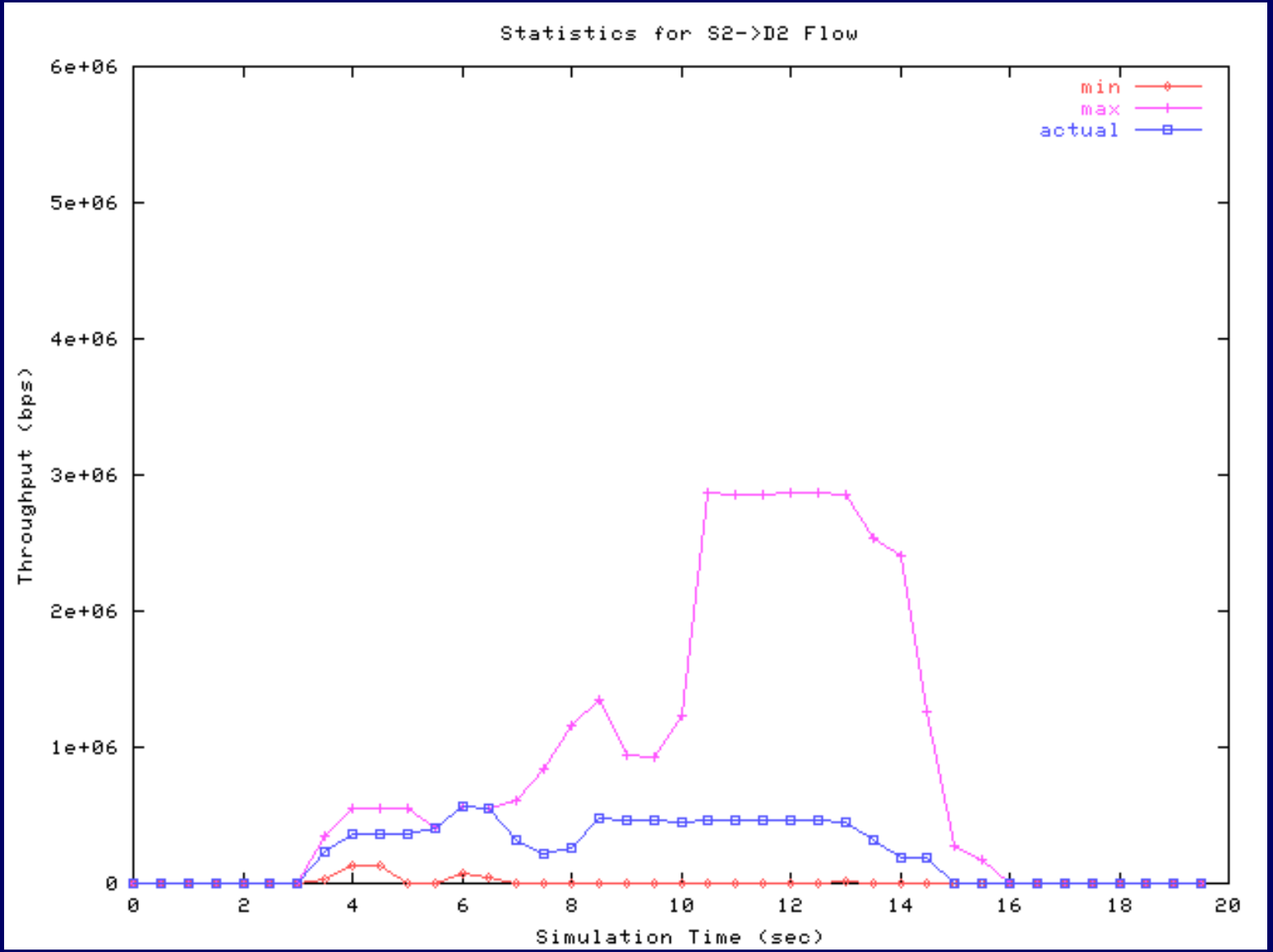
Edge-to-Edge Throughput (S1→D2)



Edge-to-Edge Throughput (S2→D1)



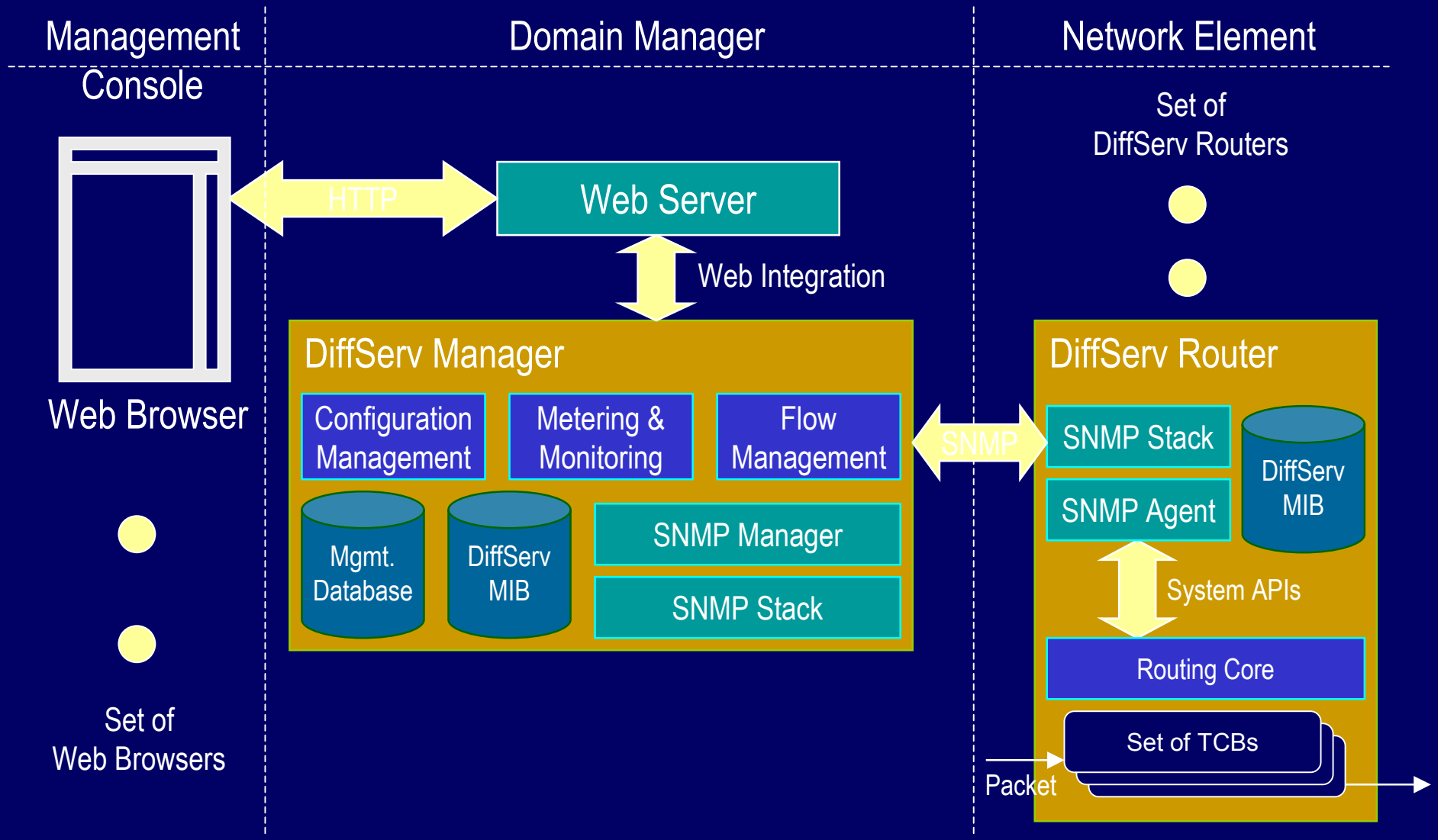
Edge-to-Edge Throughput (S2→D2)



Outline

- Introduction
- DiffServ Framework and Its Limitations
- Monitoring Methods for Edge-to-Edge DiffServ Flows
- Applying Proposed Methods to Managing DiffServ Networks
- **Edge-to-Edge DiffServ Flow Monitoring System**
- Conclusions

Design Architecture

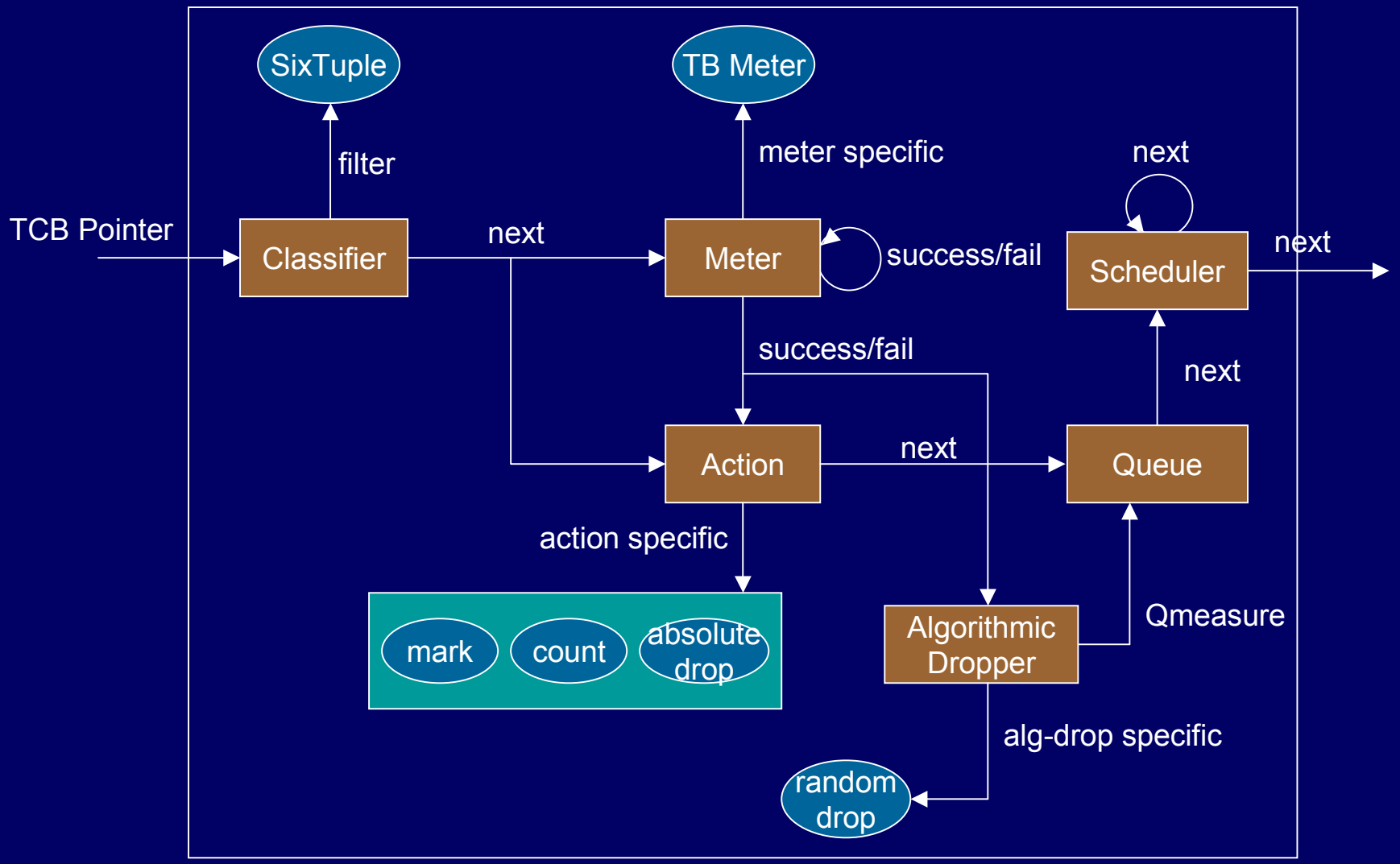


IETF DiffServ MIB

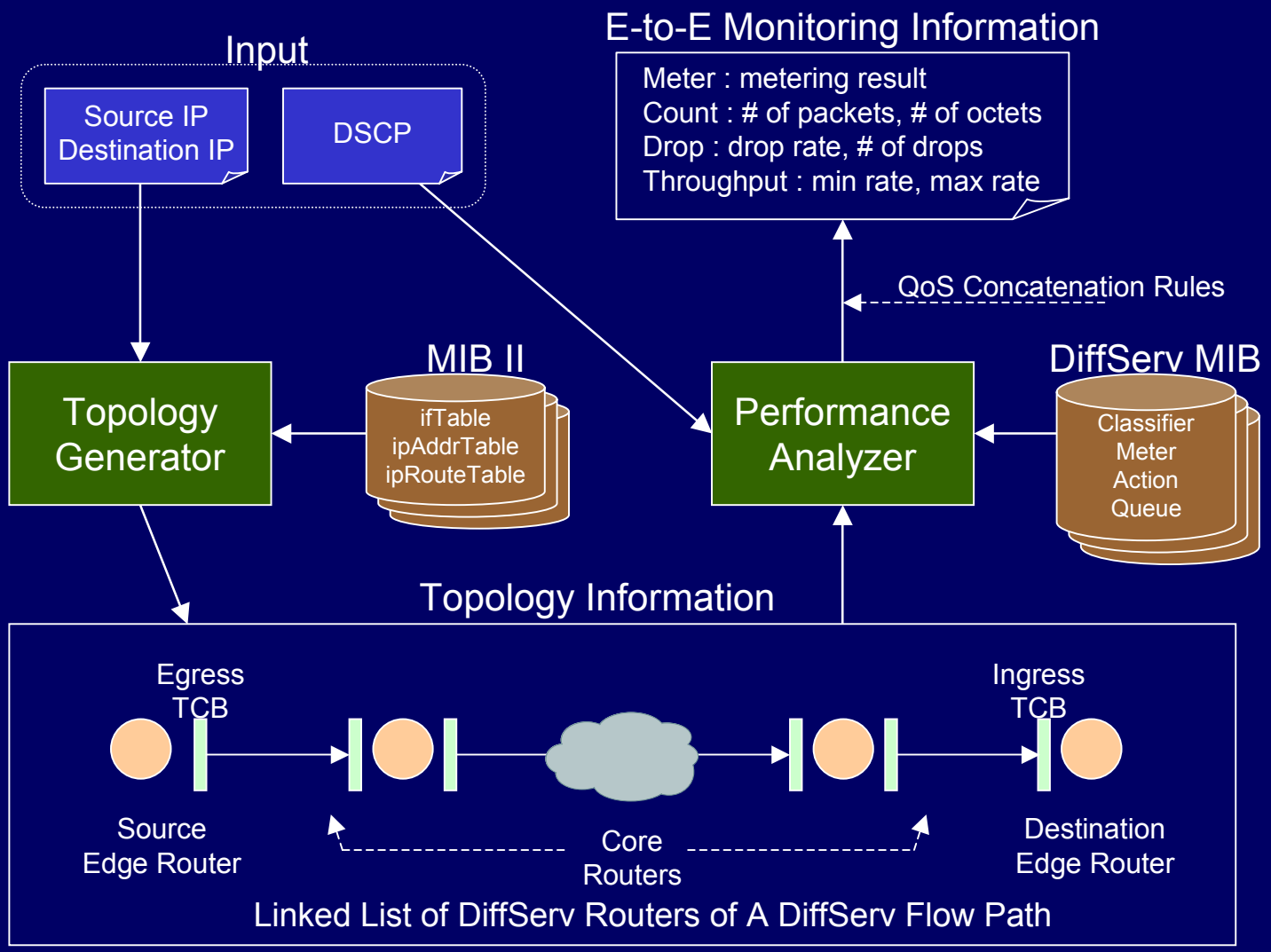
- For managing DiffServ router in SNMP framework
  - being standardized
  - IETF Internet draft : Ver1 (July 1999) ~ Ver14 (October 2001)

Element	Table Name	Description
Classifier	Classifier	general classification parameters
	SixTupleClfr	5-tuple information + DSCP value
Meter	Meter	general metering parameters
	TBMeter	token bucket meter
Action	Action	general action parameters
	MarkAct	marker action
	CountAct	counter action
	AbsoluteDrop	absolute drop action
Queue	AlgDrop	algorithmic dropper parameters
	RandomDrop	random dropper parameters
	Queue	queuing parameters
	Scheduler	scheduling parameters

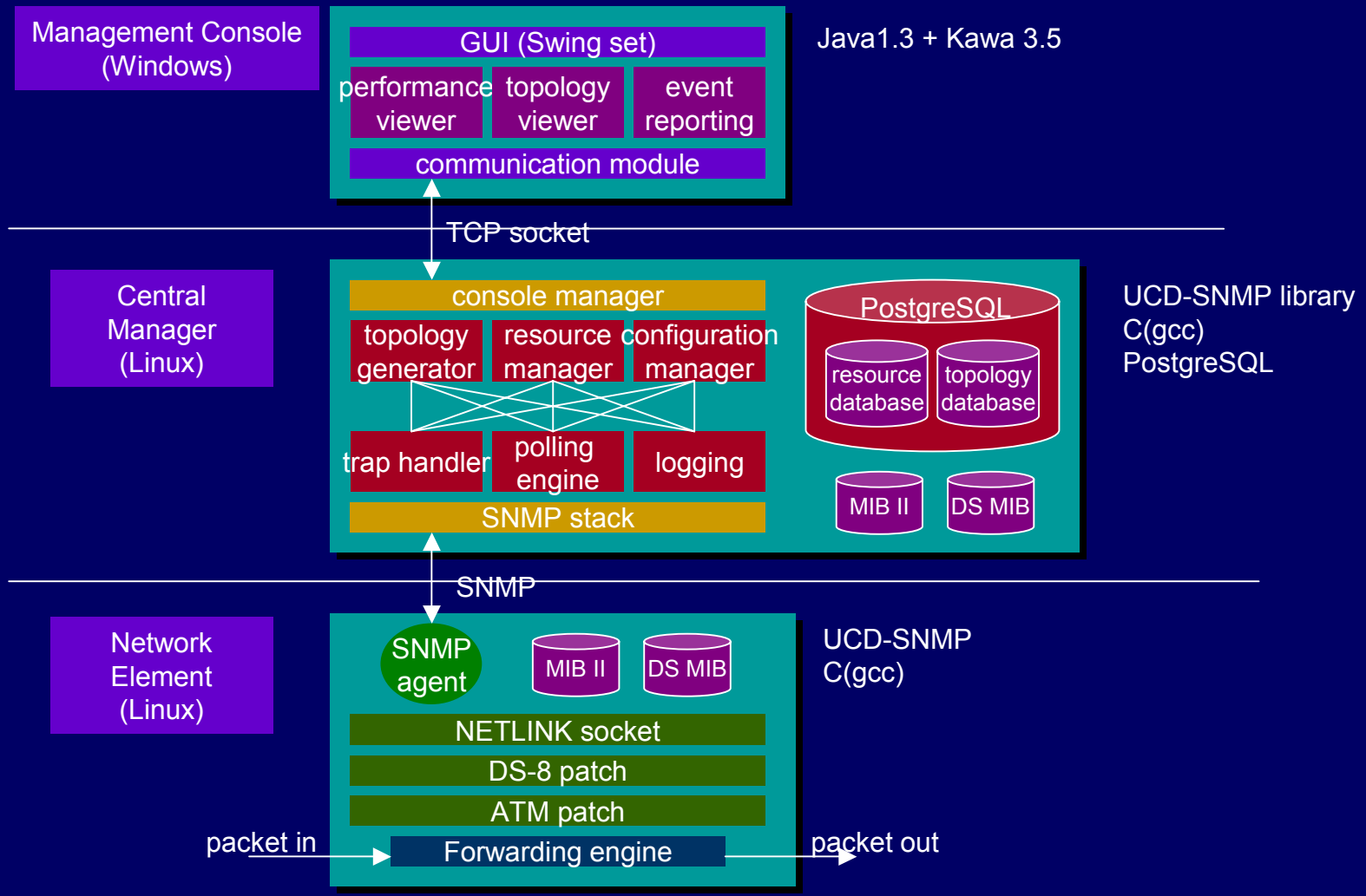
Detailed Architecture of DiffServ MIB



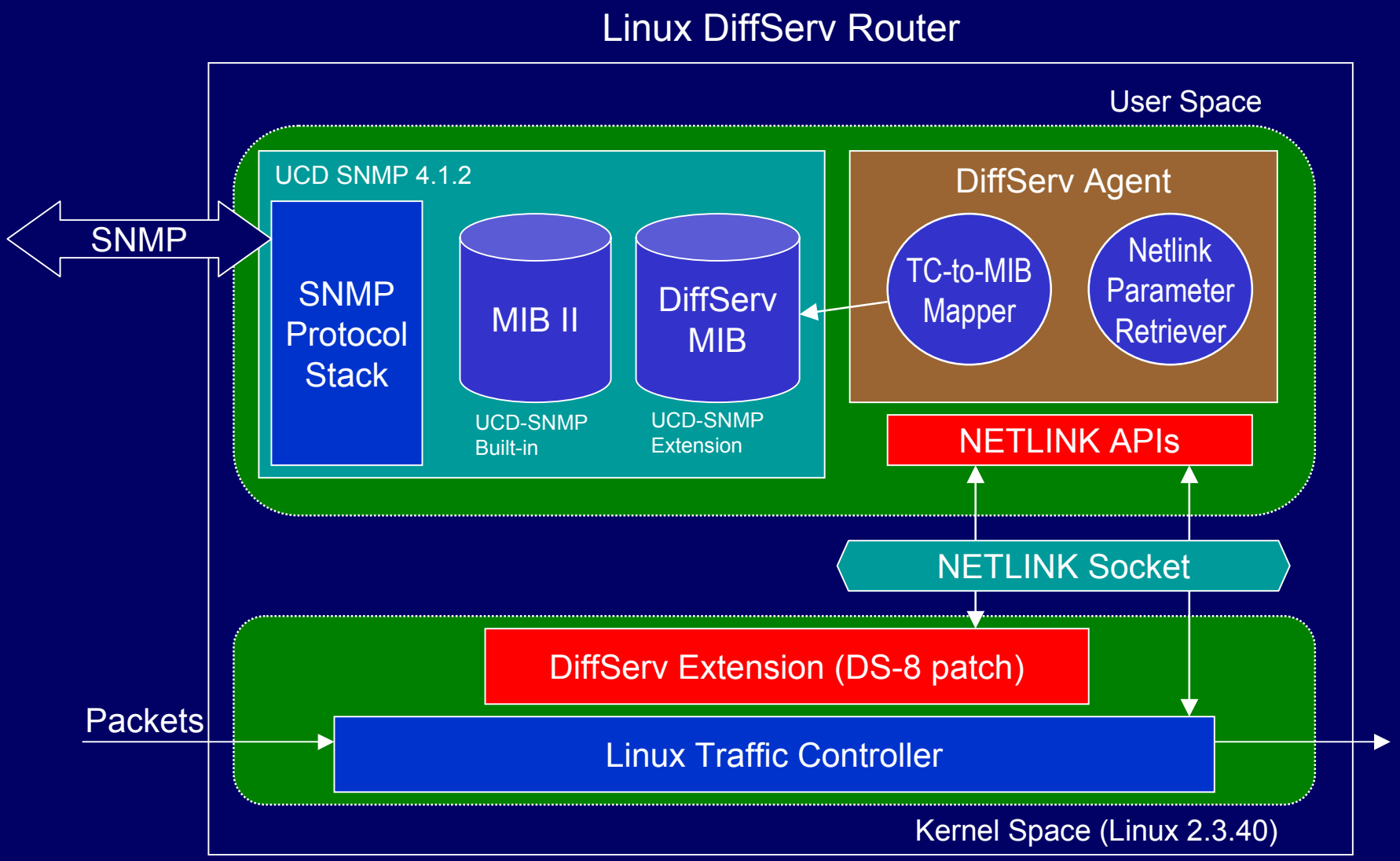
Edge-to-Edge DiffServ Flow Construction



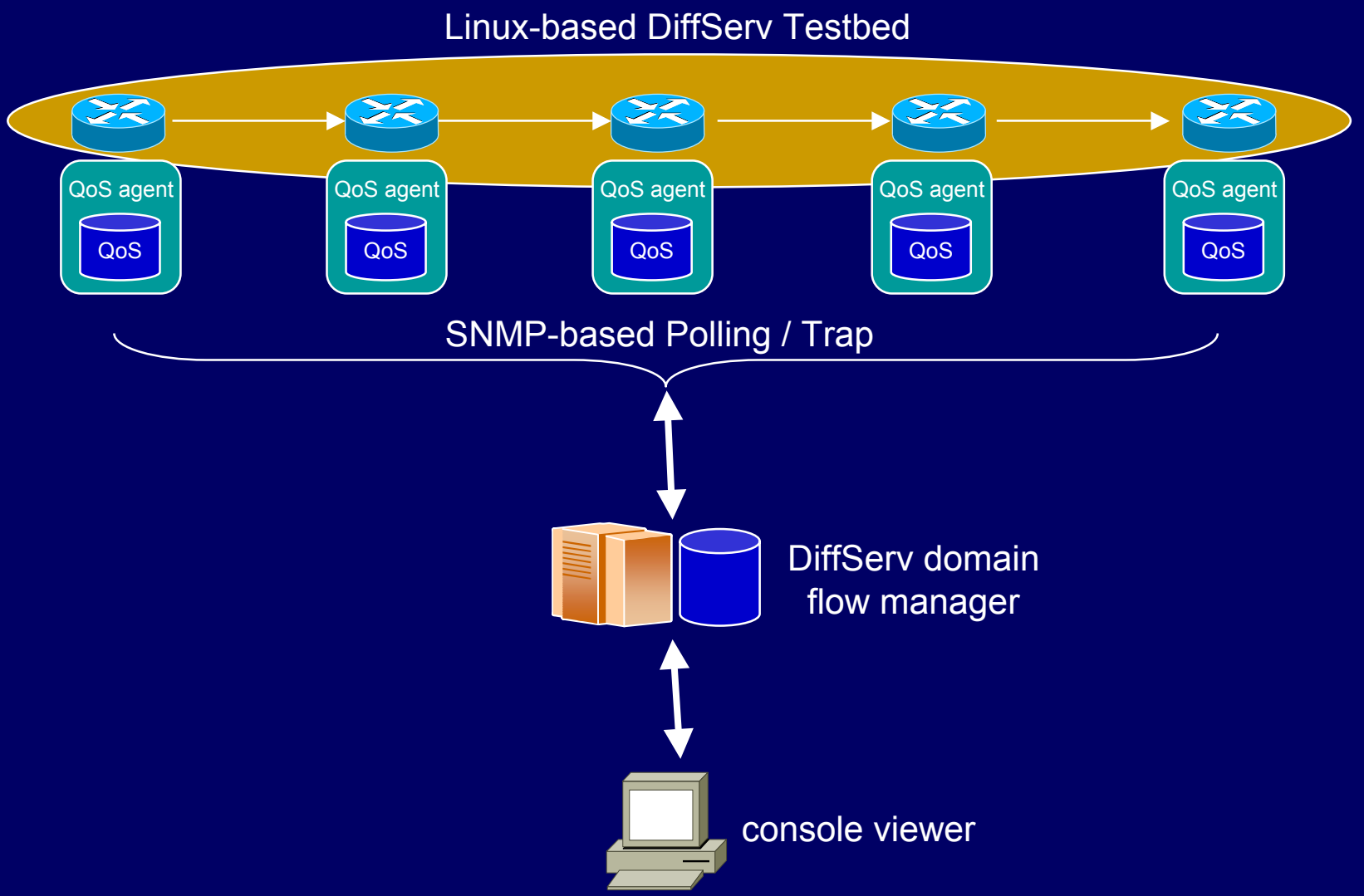
Implementation Architecture



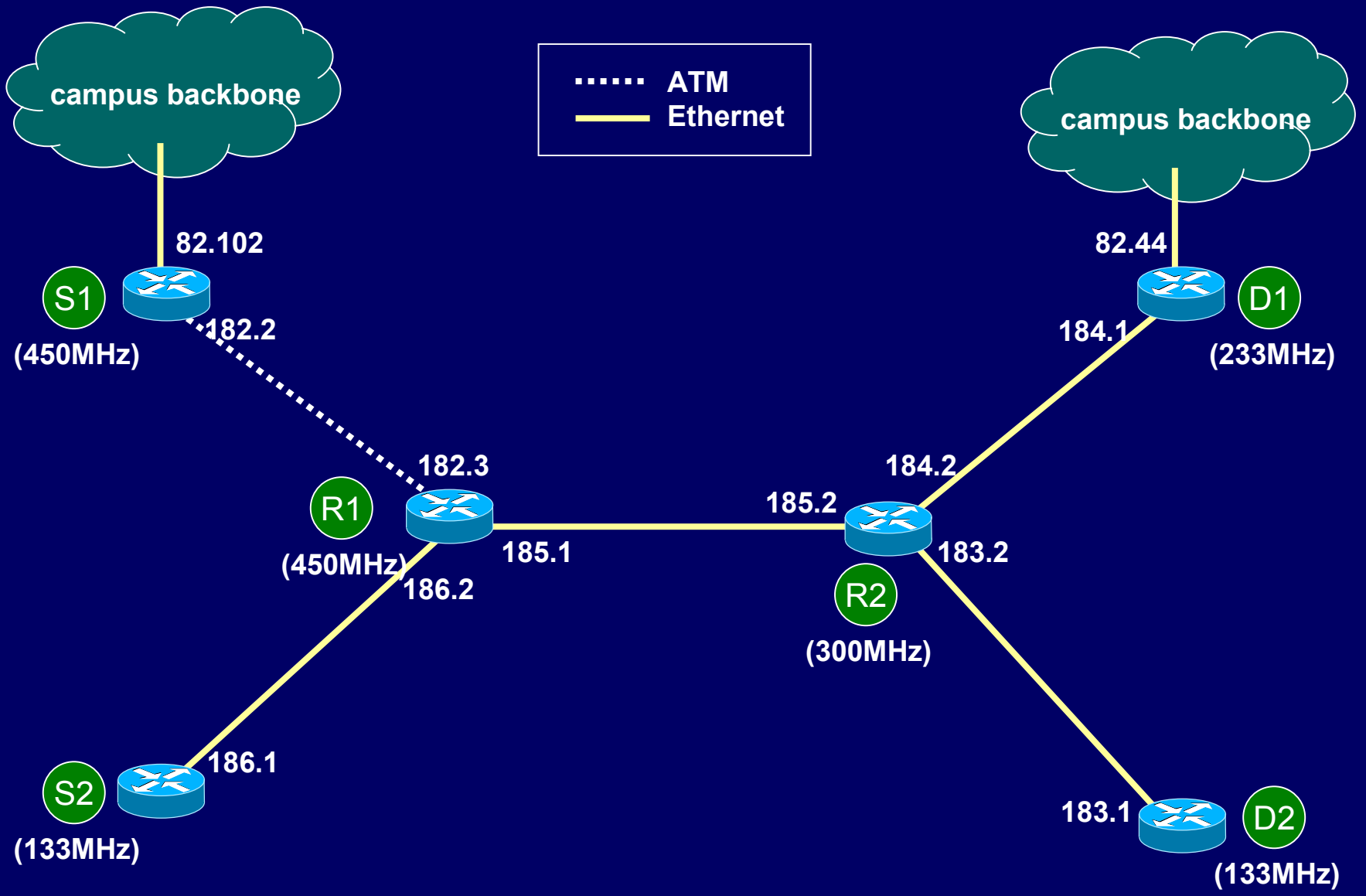
SNMP Agent Architecture in Linux-based DiffServ Router



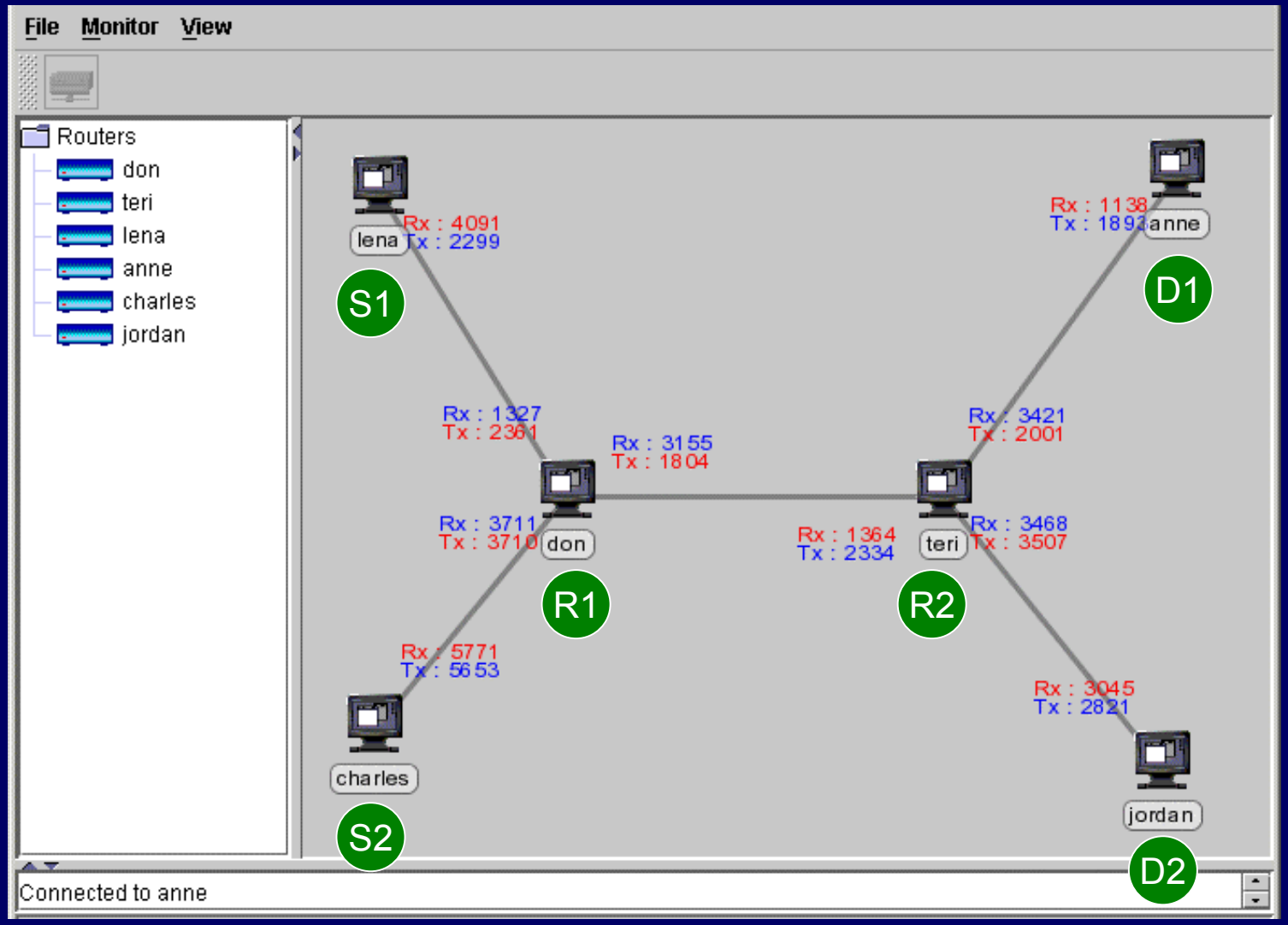
Implementation on a Linux-based Testbed



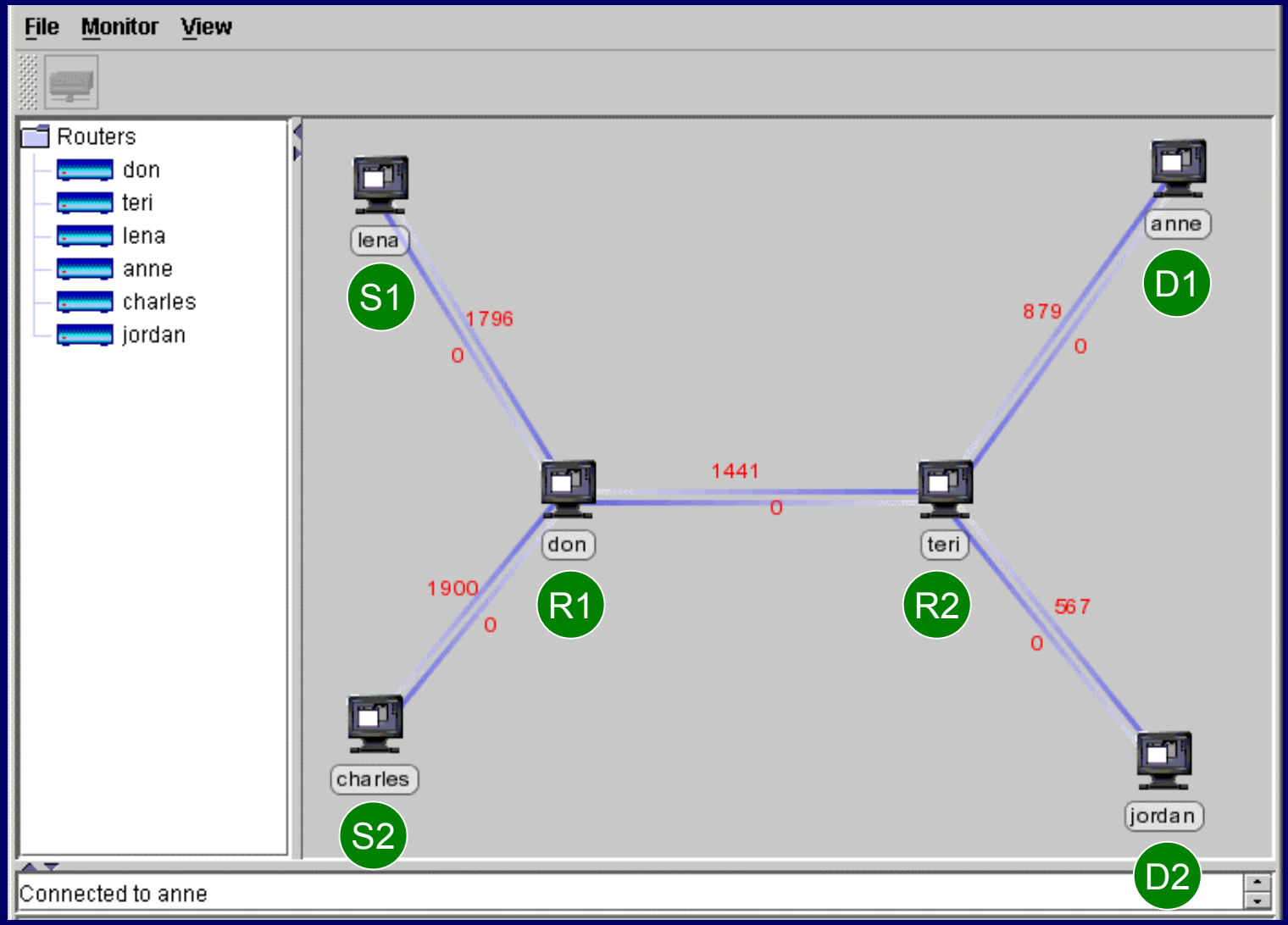
DiffServ Testbed Configuration



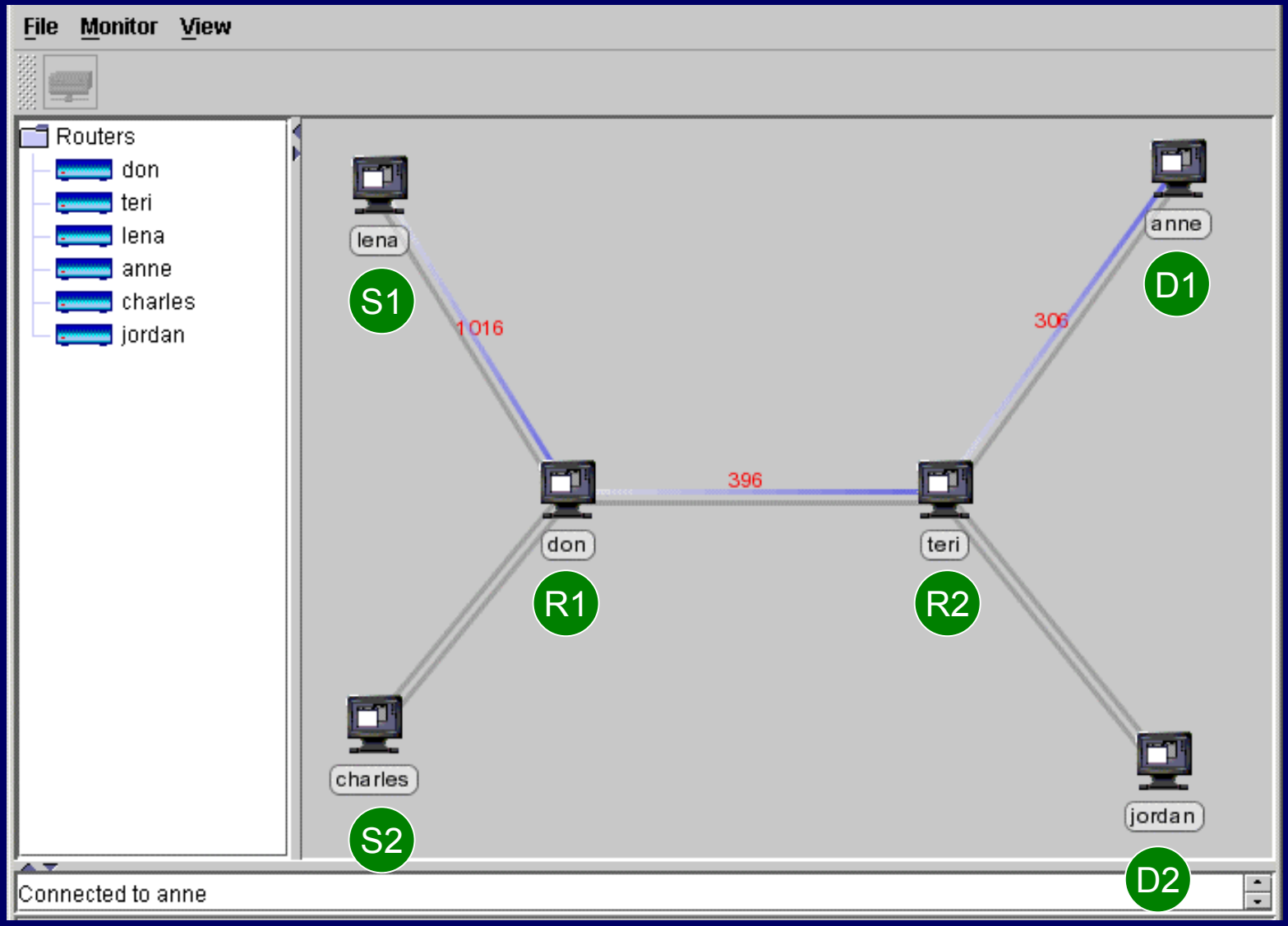
DiffServ Domain Flow Monitoring System



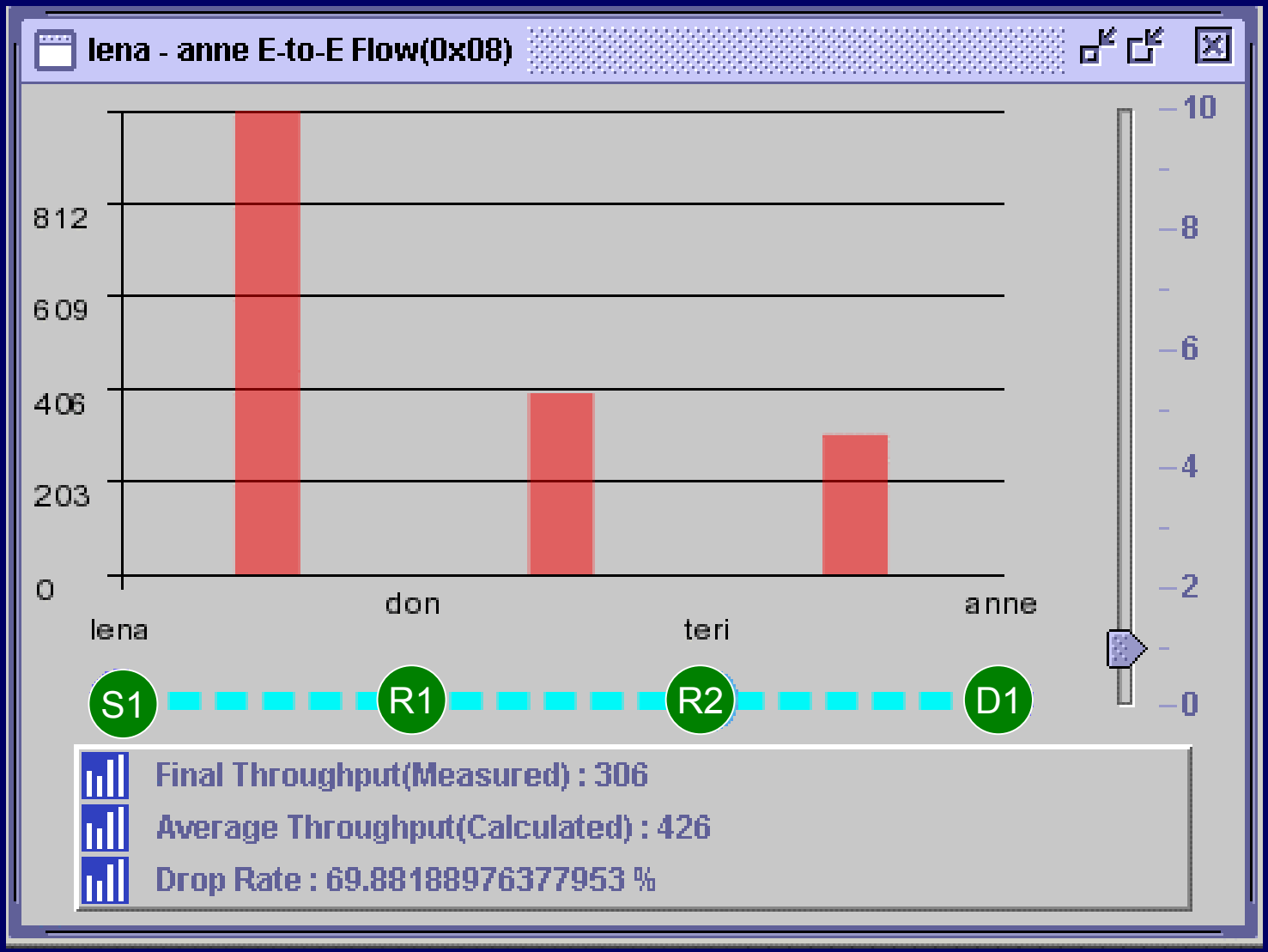
DiffServ Domain Flow Monitoring System



DiffServ Domain Flow Monitoring System



DiffServ Domain Flow Monitoring System



Outline

- Introduction
- DiffServ Framework and Its Limitations
- Monitoring Methods for Edge-to-Edge DiffServ Flows
- Applying Proposed Methods to Managing DiffServ Networks
- Edge-to-Edge DiffServ Flow Monitoring System
- **Conclusions**

Summary

- Need for Internet QoS
  - Internet explosion / QoS requirements
  - Internet QoS frameworks are suggested
- DiffServ and its limitations
  - DiffServ is a scalable backbone solutions in IP QoS frameworks
  - detailed management framework is needed
  - QoS monitoring is one of important requirements
- Edge-to-edge QoS monitoring of DiffServ flows
  - modeling edge-to-edge DiffServ flows
  - routing information + local QoS measurements → edge-to-edge QoS
  - two monitoring methods
  - applications of edge-to-edge QoS monitoring
- Developing an edge-to-edge DiffServ flow monitoring system
  - necessary when DiffServ is deployed in IP backbones
  - managing DiffServ routers in SNMP framework is suggested
  - working and testing on a Linux-based testbed

Contributions

- Various applications of edge-to-edge DiffServ flows
  - a useful building block for sophisticated management functions
  - configuration and provisioning
  - end-to-end traffic performance and QoS monitoring
  - traffic path controlling according to dynamic load changes
  - accounting and billing according to SLA's
- Realization of an SNMP-based DiffServ flow monitoring system
  - guide for similar system development
  - SNMP-based DiffServ QoS monitoring agent with DSMIB
- Finer-grained DiffServ
  - enable different controls on different ingress/egress DiffServ flows
  - efficient and scalable monitoring methods for more flexible QoS control

Future Work

- Complete QoS management framework
  - integration with the policy framework (interworking with COPS/PIB)
  - feedback to reconfiguration
- Extend to various high-level management functions
  - service level agreements monitoring
  - accounting / billing
  - interdomain QoS negotiation
- Bidirectional edge-to-edge DiffServ flows
  - network service is provided with bidirectional communications
  - how to model and monitor bidirectional edge-to-edge DiffServ flows

Thank You.