Protecting Network Quality of Service Against Denial of Service Attacks

Douglas S. Reeves ♦ S. Felix Wu

NC State / UC Davis / MCNC

DISCEX II

Anaheim, California
June 12-14, 2001

Outline

- Motivation, Scope
- Two Solutions
  - Prevention of DoS through resource pricing and authorization
  - Detecting compromised routers through end-system measurements
- Summary

Quality of Service in Networks

- QoS: end-to-end delay, delay jitter, and packet loss rate
  - at many time scales and resource granularities
- QoS: A new *capability*
  - ability to specify and receive a desired QoS
- QoS: A new *vulnerability*
  - #1 Misuse of resource reservations by "normal" users
  - #2 Attack the QoS protocols by hackers
- Some remedies
  - counterincentives / limits on greedy behavior
  - intrusion detection techniques

The ARQoS Project

To prevent some attacks on QoS, and to detect those we can't prevent

1. Resource pricing (at many levels)
2. Authentication of QoS protocols
3. Security policy checking and VPN configuration
4. Intrusion detection for DiffServ and TCP

The ARQoS Project

To prevent some attacks on QoS, and to detect those we can't prevent

1. Resource pricing (at many levels)
2. Authentication of QoS protocols
3. Security policy checking and VPN configuration
4. Intrusion detection for DiffServ and TCP

Solution #1: Resource Pricing

- How share bandwidth / cpu cycles / ... during times of scarcity (e.g., under attack)?
- Conventionally: hard-code a policy (TCP congestion control, time-of-day pricing for telephones, ...)
- Better: implement a "policy-neutral" mechanism that can be customized
  - set a "price" for each resource, users "pay" according to ability and needs

Solution #1: Resource Pricing

- Steps

  iterate

  - measure demand for the resource
  - compute new prices (higher demand → higher price)
  - distribute new prices to users
  - adjust demand in response to price

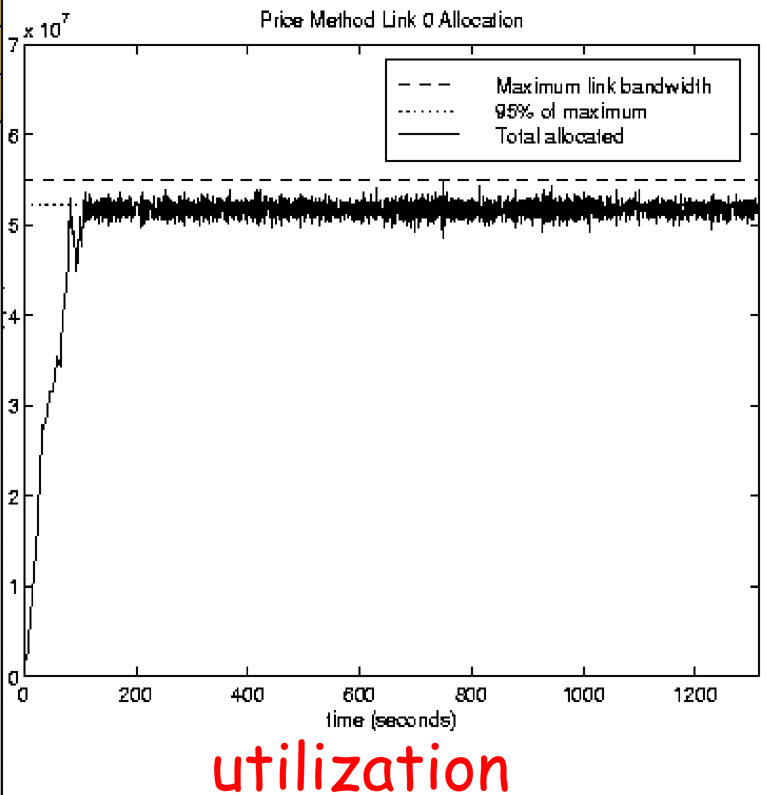
- "Appropriate" timescale / resource granularity for pricing?

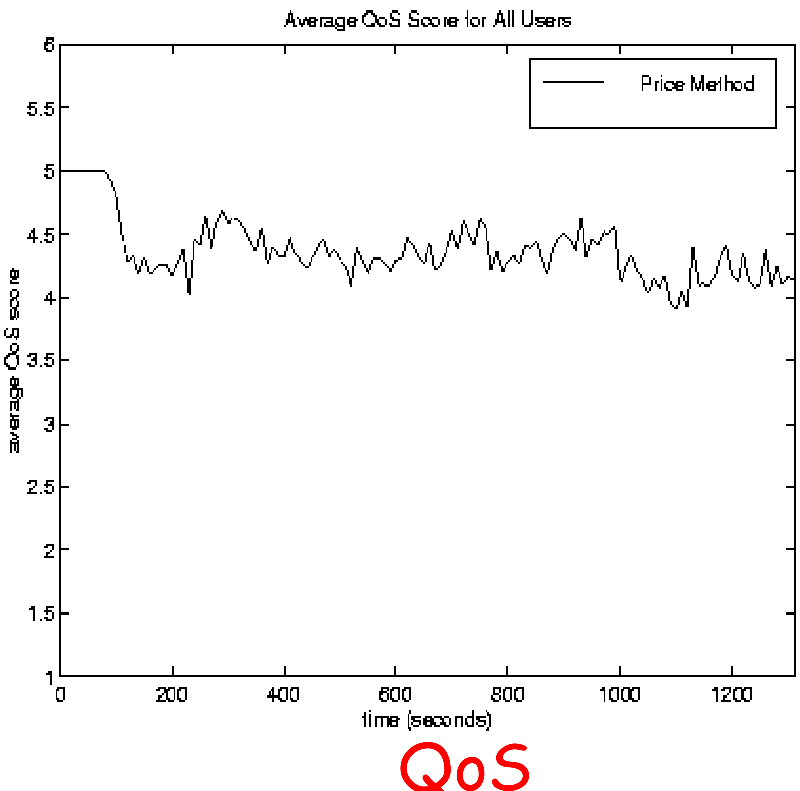
Benefits

1. Discourages / limits excessive resource consumption
2. Policies: weighted max-min fair, proportional fair, maximum aggregate utility, ...
3. Distributed, scalable, asynchronous
4. Provable convergence and optimality
5. Low communication and computation
6. High resource utilization
7. Dynamically adapts to demand

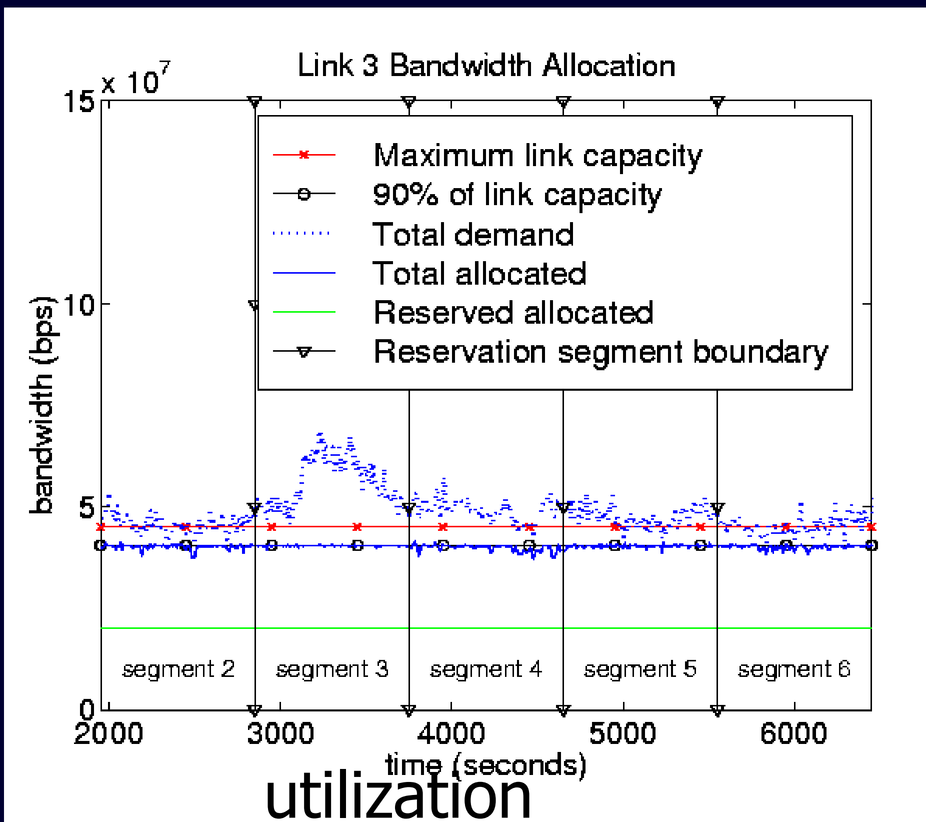
Example 1: "Spot" Market for "Elastic" Applications

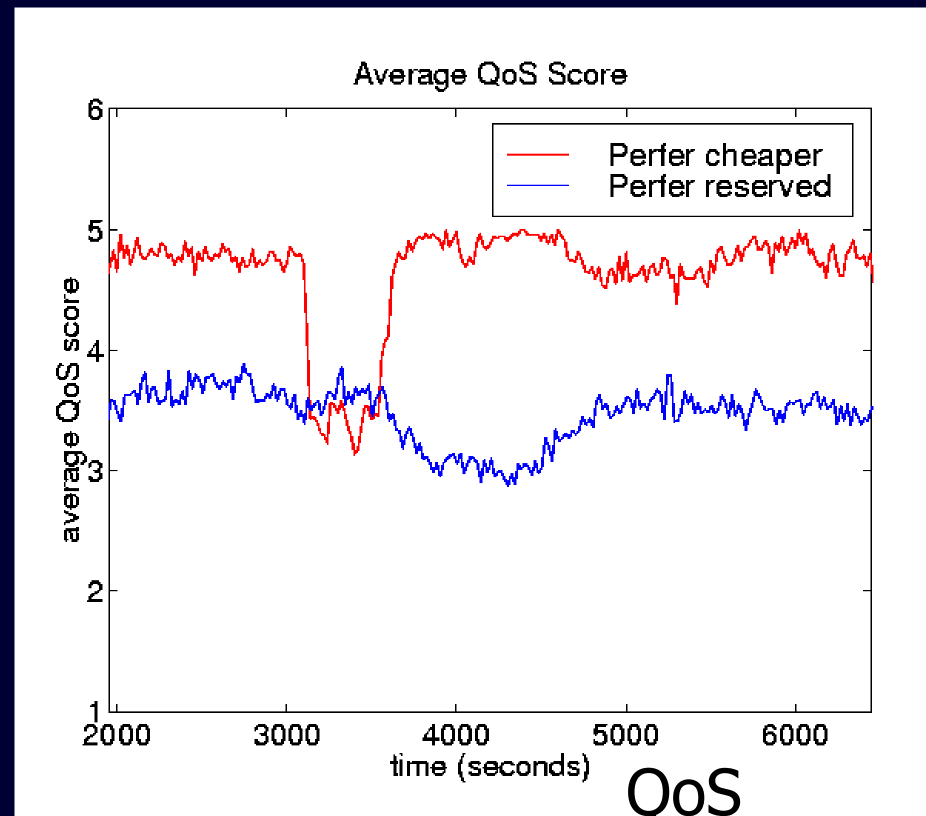
- 160 users, MPEG (VBR) video traffic, benchmark network



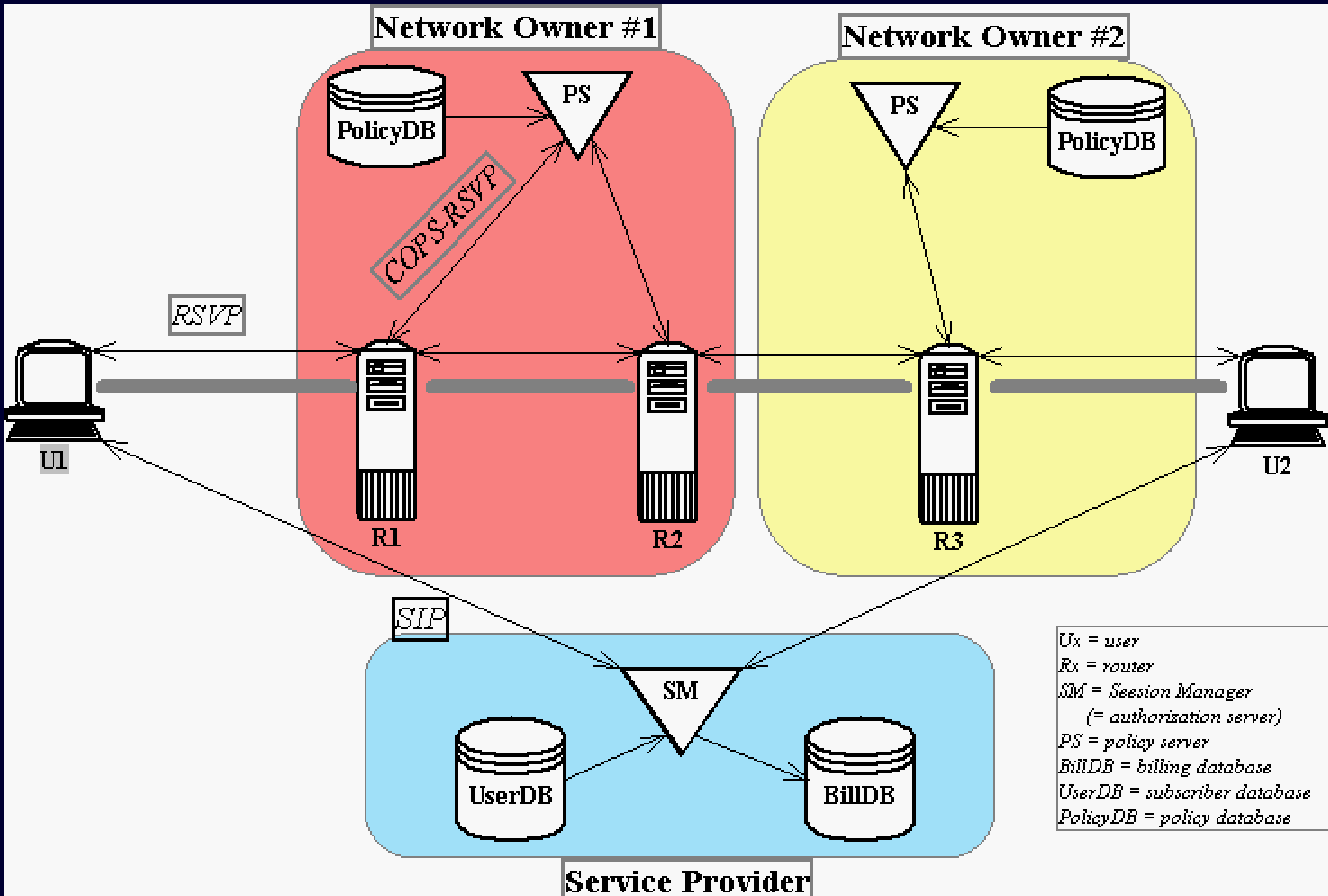


Example 2: "Reservation" Market for Inelastic Applications





Network Relationships



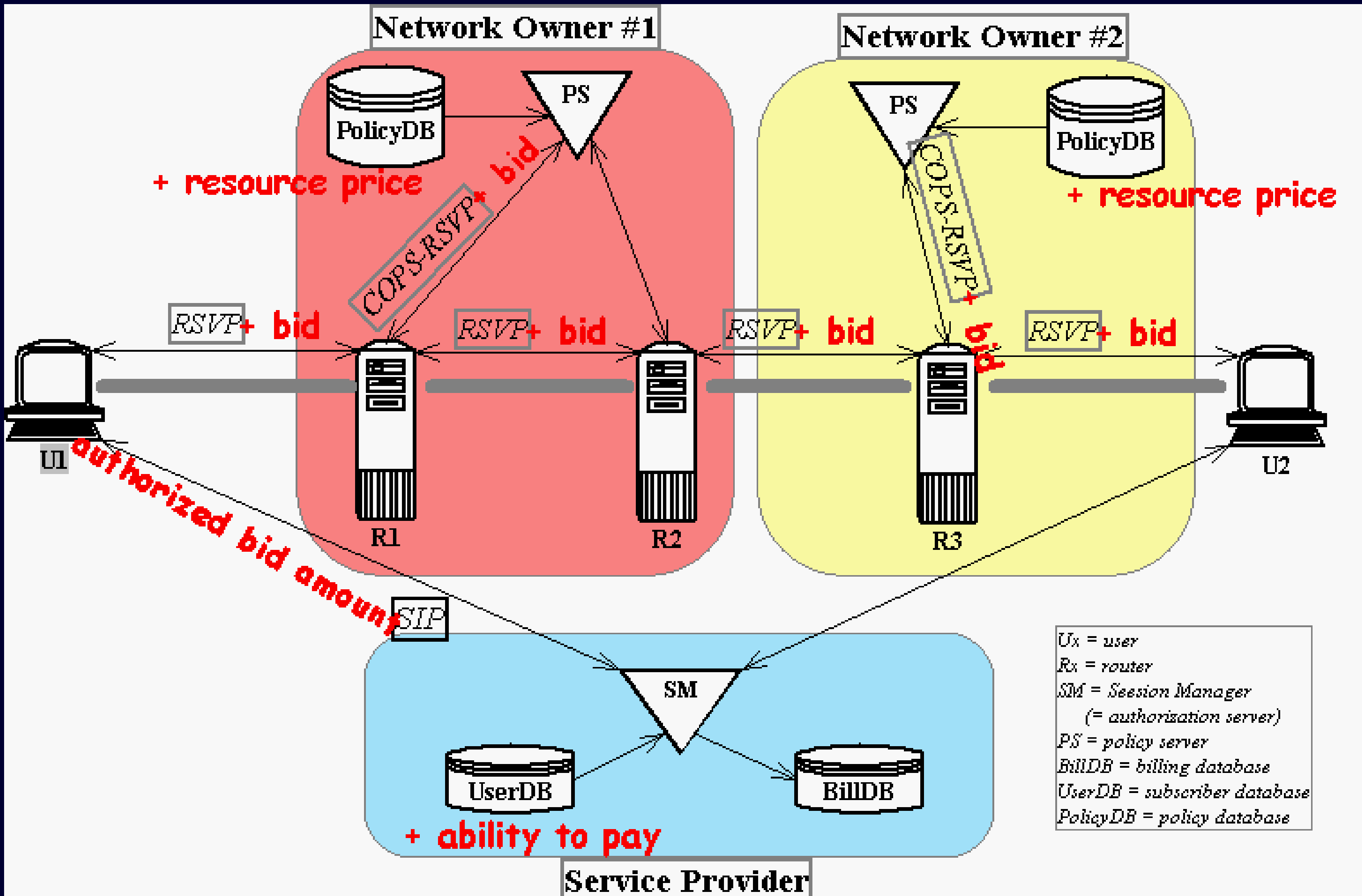
Resource Authorization

- No one entity owns the whole network any more
- Businesses won't share information or allow external control

Pricing Implementation Requirements

1. User requests a resource amount, and submits a bid
2. Bid is authorized / authenticated by a service manager (call server)
3. Request+bid is submitted to the resource manager (policy server)
4. Resource manager consults current price and accepts or rejects bid
5. User is notified, resource is reserved

Pricing Implementation



Authorization (cont.)

- Must protect against forgery, modification, stockpiling, etc. of authorization "tickets"
- Appropriate for heterogeneous networks, mobile users, ...
- “Establish trust before allocating resources”

Solution #2: Detecting Compromised Routers

- "Good" routers drop packets because of congestion
  - packet drop rate highly variable
- "Bad" routers drop packets to interfere with quality
- Can these be distinguished?

Approach: Anomaly Detection at the End Systems (Hosts)

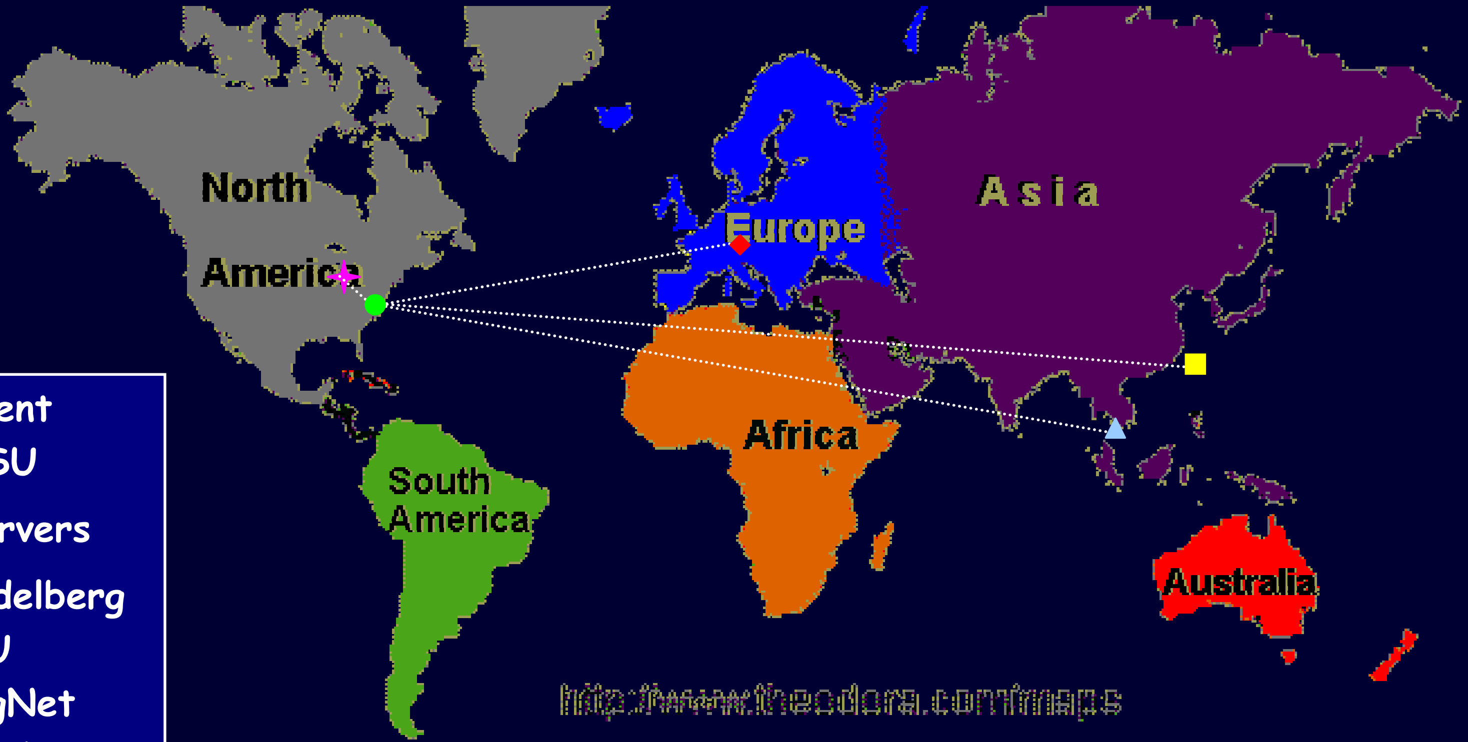
- Measure "normal" TCP behavior *at the host systems* (i.e., no router cooperation required)
- Construct a statistical profile
  - Q-test detection mechanism
  - developed by SRI (NIDES-STAT)
- Compare *observed* TCP behavior to *expected* profile, and flag anomalies

Details

- Possible dropping attack “patterns”
  - random
  - periodic
  - intermittent
  - retransmitted packets only
- Metrics
  - number of packets dropped
  - which packets dropped
  - session duration

Will It Work in Practice?

- Established TCP connections to 4 FTP sites around the world



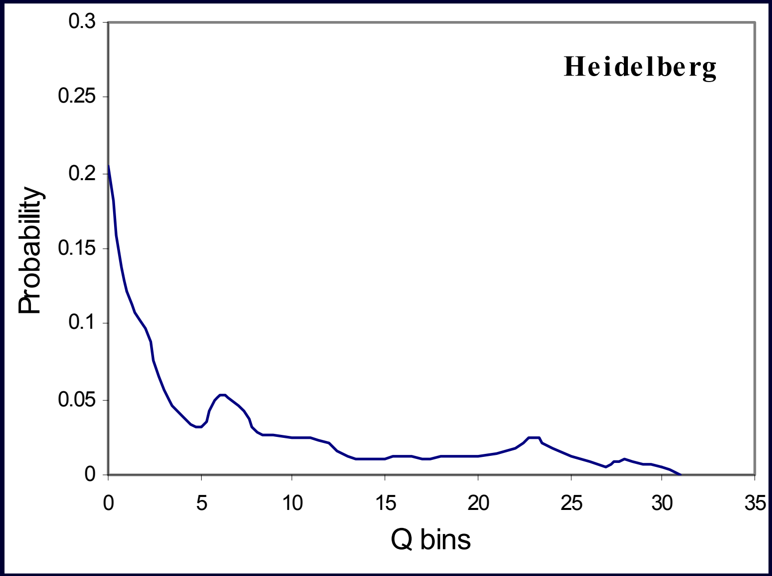
FTP Client
- NCSU

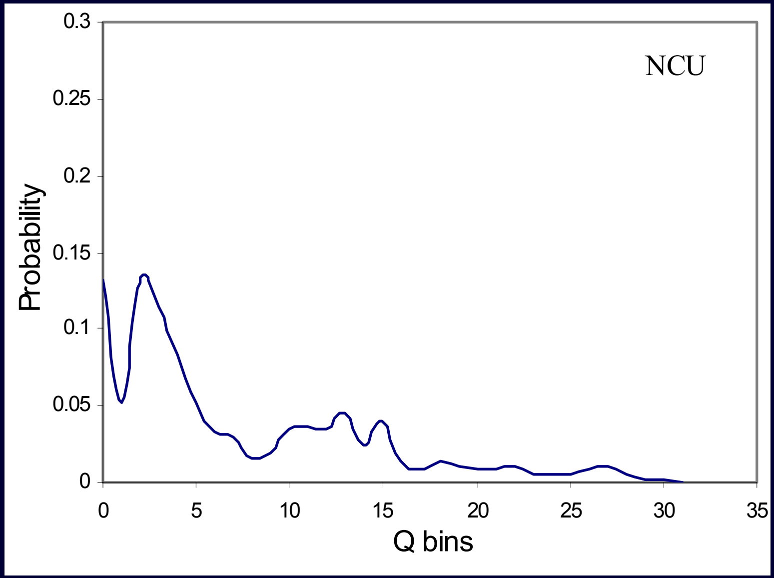
FTP Servers
- Heidelberg
- NCU
- SingNet
- UIUC

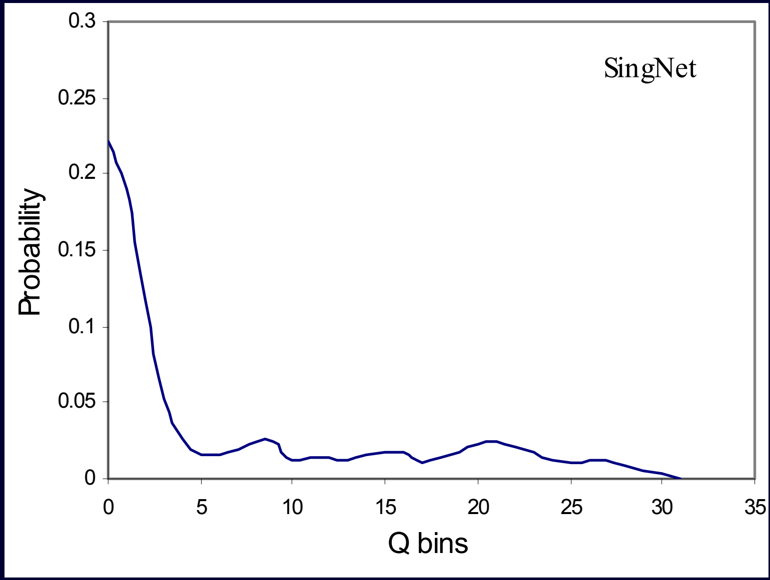
Experiments

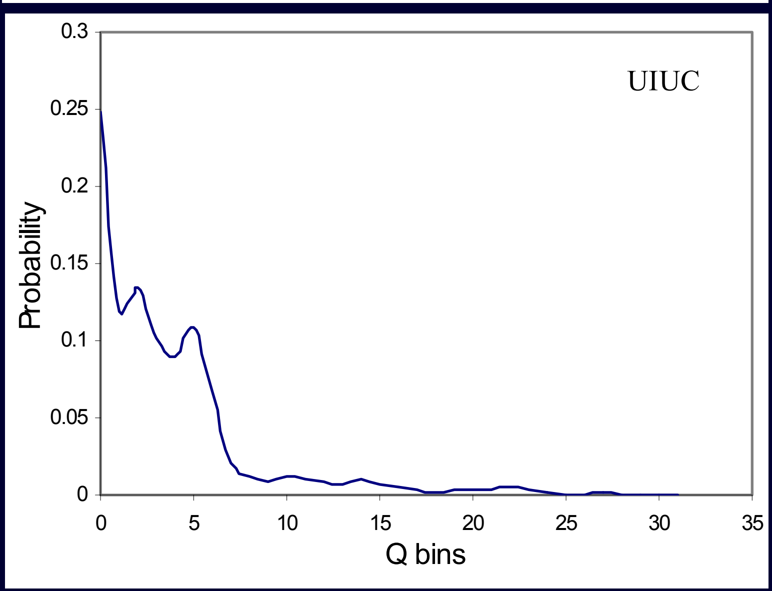
- Established a profile over 2 week period, substantial variability observed
- Compromised a router in our testbed to drop packets maliciously
- Compared observed behavior with profile

Example Profiles: Session Duration

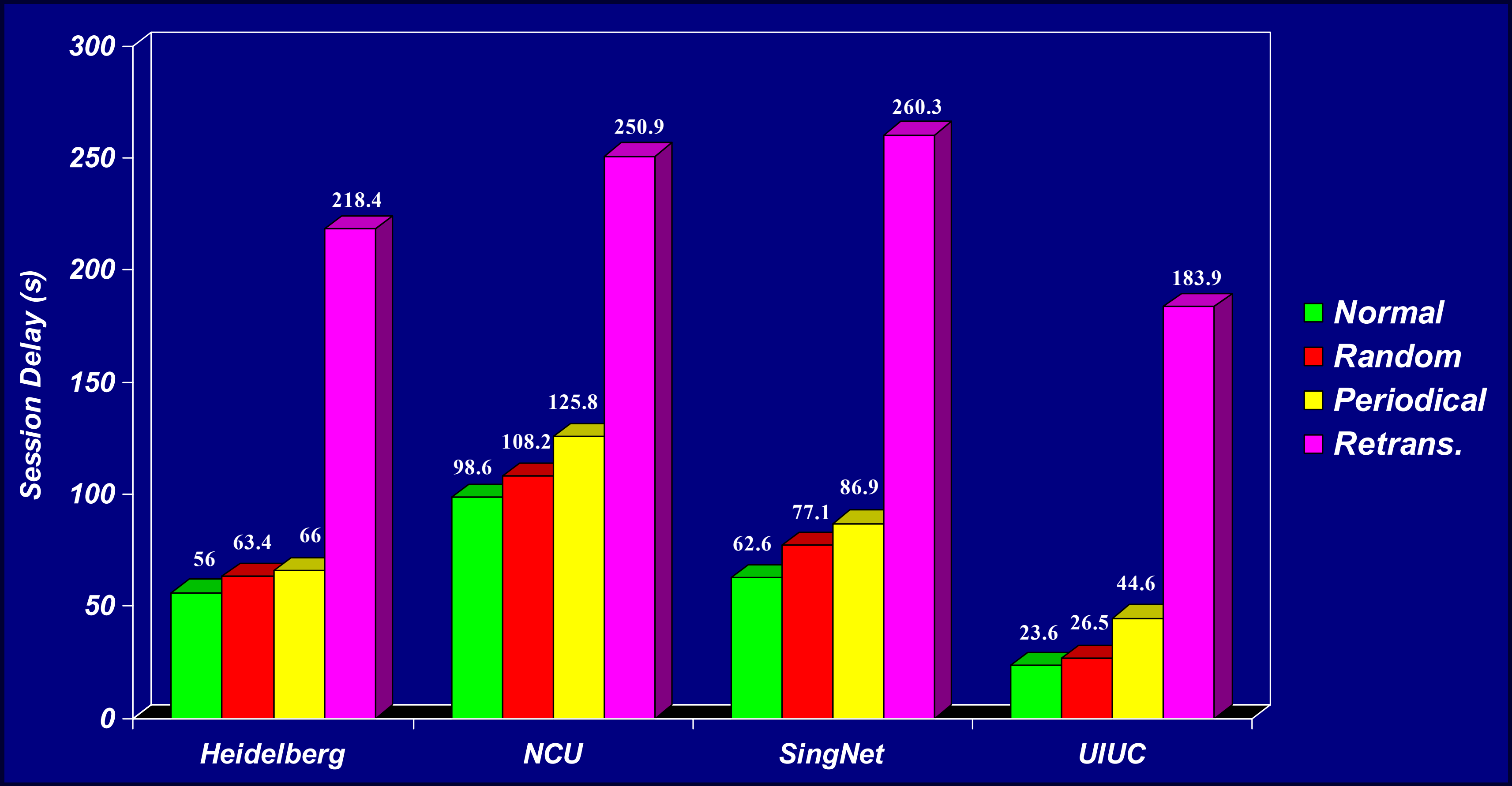








Results: Impact on Session Duration



7 packets are dropped among more than 4000 packets in a connection

Results: Session Duration Metric

Delay nbin=3		Heidelberg		NCU		SingNet		UIUC	
		DR	MR	DR	MR	DR	MR	DR	MR
Normal*	-	1.6%	-	7.5%	-	2.1%	-	7.9%	-
PerPD	(10, 4, 5)	97.4%	2.6%	95.2%	4.8%	94.5%	5.5%	99.2%	0.8%
	(20, 4, 5)	99.2%	0.8%	98.5%	1.5%	100%	0%	100%	0%
	(40, 4, 5)	100%	0%	100%	0%	100%	0%	100%	0%
	(20, 20, 5)	96.3%	3.7%	100%	0%	92.6%	7.4%	98.9%	1.1%
	(20, 100, 5)	100%	0%	95.3%	4.7%	98.7%	1.3%	100%	0%
	(20, 200, 5)	98.6%	1.4%	99%	1%	97.1%	2.9%	100%	0%
	(100, 40, 5)	100%	0%	100%	0%	100%	0%	100%	0%
RetPD	(5, 5)	100%	0%	100%	0%	100%	0%	100%	0%
RanPD	10	74.5%	25.5%	26.8%	73.2%	67.9%	32.1%	99.5%	0.5%
	40	100%	0%	100%	0%	100%	0%	100%	0%
Intermittent (10, 4, 5)	5	25.6%	74.4%	0%	100%	0%	100%	97.3%	2.7%
	50	0%	100%	24.9%	75.1%	0%	100%	3.7%	96.3%

Results: Dropped Packet Position Metric

Position nbin=5		Heidelberg		NCU		SingNet		UIUC	
		DR	MR	DR	MR	DR	MR	DR	MR
Normal*	-	4.0%	-	5.4%	-	3.5%	-	6.5%	-
PerPD	(10, 4, 5)	99.7%	0.3%	100%	0%	100%	0.0%	100%	0%
	(20, 4, 5)	100%	0%	98.1%	1.9%	99.2%	0.8%	100%	0%
	(40, 4, 5)	96.6%	3.4%	100%	0%	100%	0%	98.5%	1.5%
	(20, 20, 5)	100%	0%	100%	0%	100%	0 %	100%	0%
	(20, 100, 5)	98.9%	1.1%.	99.2%	0.8%	99.6%	0.4%	99.1%	0.9%
	(20, 200, 5)	0%	100%	76.5%	23.5%	1.5%	98.5%	98.3%	1.7%
	(100, 40, 5)	0.2%	99.8%	0%	100%	0%	100%	100%	0%
RetPD	(5, 5)	84.9%	15.1%	81.1%	18.9%	94.3%	5.7%	97.4%	2.6%
RanPD	10	0%	100%	42.3%	57.7%	0%	100%	0%	100%
	40	0%	100%	0%	100%	0%	100%	0%	100%
Intermittent (10, 4, 5)	5	98.6%	1.4%	100%	0%	98.2%	1.8%	100%	0%
	50	34.1%	65.9%	11.8%	88.2%	89.4%	10.6%	94.9%	5.1%

Summary

- QoS must be protected, or it will be attacked *as soon as it is deployed*
- Pricing provides precise, flexible, low overhead control of resource allocation
- Compromised routers that drop packets maliciously can be detected *by end systems* fairly easily
- ARQoS project tackling several other security issues
  - detection of attacks on DiffServ in core networks
  - synthesis of VPNs to implement security policy
  - applications of pricing
  - protection of reliable multicast