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

  measure demand for the resource

  compute new prices (higher demand $\rightarrow$ 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

![Link 3 Bandwidth Allocation](chart1)

- Maximum link capacity
- 90% of link capacity
- Total demand
- Total allocated
- Reserved allocated
- Reservation segment boundary

![Average QoS Score](chart2)

- Prefer cheaper
- Prefer reserved

![utilization](chart3)

![QoS](chart4)
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

Network Owner #1
- PolicyDB
- PS
- resource price
- bid
- COPS-RSVP
- RSVP

Network Owner #2
- PolicyDB
- PS
- resource price
- bid
- COPS-RSVP
- RSVP

U1
- bid
- authorized bid amount
- SIP

R1
- RSVP

R2
- RSVP

R3
- RSVP

U2
- bid

Service Provider
- UserDB
- BillDB
- ability to pay

Symbols:
- \( U_x \) = user
- \( R_x \) = router
- \( SM \) = Session Manager
  (= authorization server)
- \( PS \) = policy server
- \( BillDB \) = billing database
- \( UserDB \) = subscriber database
- \( PolicyDB \) = policy database
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
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

Heidelberg

Probability

NCU

Probability

SingNet

Probability

UIUC

Probability
Results: Impact on Session Duration

7 packets are dropped among more than 4000 packets in a connection
## Results: Session Duration Metric

<table>
<thead>
<tr>
<th>Delay nbin=3</th>
<th>Heidelberg</th>
<th>NCU</th>
<th>SingNet</th>
<th>UIUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DR</td>
<td>MR</td>
<td>DR</td>
<td>MR</td>
</tr>
<tr>
<td>Normal*</td>
<td>1.6%</td>
<td>-</td>
<td>7.5%</td>
<td>-</td>
</tr>
<tr>
<td>PerPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(10, 4, 5)</td>
<td>97.4%</td>
<td>2.6%</td>
<td>95.2%</td>
<td>4.8%</td>
</tr>
<tr>
<td>(20, 4, 5)</td>
<td>99.2%</td>
<td>0.8%</td>
<td>98.5%</td>
<td>1.5%</td>
</tr>
<tr>
<td>(40, 4, 5)</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(20, 20, 5)</td>
<td>96.3%</td>
<td>3.7%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(20, 100, 5)</td>
<td>100%</td>
<td>0%</td>
<td>95.3%</td>
<td>4.7%</td>
</tr>
<tr>
<td>(20, 200, 5)</td>
<td>98.6%</td>
<td>1.4%</td>
<td>99%</td>
<td>1%</td>
</tr>
<tr>
<td>(100, 40, 5)</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>RetPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5, 5)</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>RanPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>74.5%</td>
<td>25.5%</td>
<td>26.8%</td>
<td>73.2%</td>
</tr>
<tr>
<td>40</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Intermittent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25.6%</td>
<td>74.4%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>(10, 4, 5)</td>
<td>50</td>
<td>0%</td>
<td>100%</td>
<td>24.9%</td>
</tr>
<tr>
<td></td>
<td>3.7%</td>
<td>96.3%</td>
<td>97.3%</td>
<td>2.7%</td>
</tr>
</tbody>
</table>
## Results: Dropped Packet Position Metric

<table>
<thead>
<tr>
<th>Position nbin=5</th>
<th>Heidelberg</th>
<th>NCU</th>
<th>SingNet</th>
<th>UIUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DR</td>
<td>MR</td>
<td>DR</td>
<td>MR</td>
</tr>
<tr>
<td>Normal*</td>
<td></td>
<td>4.0%</td>
<td>5.4%</td>
<td>3.5%</td>
</tr>
<tr>
<td>PerPD (10, 4, 5)</td>
<td>99.7%</td>
<td>0.3%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(20, 4, 5)</td>
<td>100%</td>
<td>0%</td>
<td>98.1%</td>
<td>1.9%</td>
</tr>
<tr>
<td>(40, 4, 5)</td>
<td>96.6%</td>
<td>3.4%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(20, 20, 5)</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(20, 100, 5)</td>
<td>98.9%</td>
<td>1.1%</td>
<td>99.2%</td>
<td>0.8%</td>
</tr>
<tr>
<td>(20, 200, 5)</td>
<td>0%</td>
<td>100%</td>
<td>76.5%</td>
<td>23.5%</td>
</tr>
<tr>
<td>(100, 40, 5)</td>
<td>0.2%</td>
<td>99.8%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>RetPD (5, 5)</td>
<td>84.9%</td>
<td>15.1%</td>
<td>81.1%</td>
<td>18.9%</td>
</tr>
<tr>
<td>RanPD 10</td>
<td>0%</td>
<td>100%</td>
<td>42.3%</td>
<td>57.7%</td>
</tr>
<tr>
<td>40</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Intermittent (10, 4, 5)</td>
<td>98.6%</td>
<td>1.4%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>(5)</td>
<td>34.1%</td>
<td>65.9%</td>
<td>11.8%</td>
<td>88.2%</td>
</tr>
</tbody>
</table>
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