Communication Networks
Exercise 9

Last week’s exercise

Important lecture topics

Introduction to this week’s exercise

Time to solve the exercise
Task 8.3: BGP Hijack

AS path poisoning gives the hijacker some control over which ASes are/are not affected by the hijack
Task 8.3: BGP Hijack

AS path poisoning gives the hijacker some control over which ASes are/are not affected by the hijack.

20.0.0.0/23 - AS path: F
20.0.2.0/23 - AS path: F
Task 8.3: BGP Hijack

AS path poisoning gives the hijacker some control over which ASes are/are not affected by the hijack
Communication Networks
Exercise 9

Last week’s exercise

Important lecture topics

Introduction to this week’s exercise

Time to solve the exercise
The Go-Back-N protocol
The Go-Back-N Protocol
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with a sliding window, cumulative ACKs, timeouts and retransmissions

Sender

Receiver
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

<table>
<thead>
<tr>
<th>Sender</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

sliding window
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

ready to send

```
0 1 2 3
```

Sender

Receiver
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

cumulative ACKs
make up for losses

Sender

0 1 2 3 4 5 6 7 8 9

Receiver

0 1 2 3
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

When a timeout occurs, the sender retransmits all segments in the window.
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

When a timeout occurs, the sender retransmits all segments in the window.
The Go-Back-N Protocol

a simple reliable transport protocol with
a sliding window, cumulative ACKs, timeouts and retransmissions

Sender

Receiver
Physical and virtual ports
A **port** can describe two completely different concepts

A physical port on a switch or router (interface)

A logical (virtual) port on a host to demultiplex incoming data
Physical ports
Physical ports

Physical interface on a device
Often numbered from 1…N
Physical ports

Important if you configure a device (compare routing project)

These ports are normally **not** visible in a packet header

We also saw these ports in the Spanning Tree algorithm
Constructing a Spanning Tree in a nutshell

Switches...

elect a root switch
the one with the smallest identifier

determine if each interface is
on the shortest-path from the root
and disable it if not
For this switches exchange
Bridge Protocol Data Unit (BDPU) messages

Each switch $X$ iteratively sends

$BPDU(Y, d, X)$ to each neighboring switch

the switch ID it considers as root

the # hops to reach it
Initially, each switch proposes itself as root and sends \((X,0,X)\) on all its interfaces.

Upon receiving \((Y, d, X)\), checks if \(Y\) is a better root. If so, considers \(Y\) as the new root, and floods the updated message.

Switches compute their distance to the root, for each port, simply add 1 to the distance received, if shorter, flood.

Switches disable interfaces not on the shortest path.
Upon receiving different BPDUs from different switches with the same cost, pick the BPDU with the lower switch sender ID.

Upon receiving different BPDUs from a neighboring switch, pick the BPDU with the lowest port ID (e.g., port 2 < port 3).
Upon receiving ≠ BPDUs from ≠ switches with = cost
Pick the BPDU with the lower switch sender ID

Upon receiving ≠ BPDUs from a neighboring switch
Pick the BPDU with the lowest port ID (e.g. port 2 < port 3)

This switch receives two BPDUs from its neighbor
Both have the same cost
One is received over port 1, the other over port 4
The switch picks the one from port 1
Logical (virtual) ports on a host
Logical (virtual) ports on a host

Host/server

App 1  App 3

App 2  App 4

A single interface with IP 1.2.3.4
Logical (virtual) ports on a host

How does the host (the transport layer) know to which application it has to forward incoming packets?

A single interface with IP 1.2.3.4

Host/server

App 1

App 2

App 3

App 4
Logical (virtual) ports on a host

How does the host (the transport layer) know to which application it has to forward incoming packets?

Each application listens on a different logical port.
Logical (virtual) ports on a host

How does the host (the transport layer) know to which application it has to forward incoming packets?

Each application listens on a different logical port.

Transport protocol headers contain these port numbers.
Ports in UDP/TCP packets

**UDP**

<table>
<thead>
<tr>
<th>SRC port</th>
<th>DST port</th>
</tr>
</thead>
<tbody>
<tr>
<td>checksum</td>
<td>length</td>
</tr>
</tbody>
</table>

**DATA**

**TCP**

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence number</th>
<th>Acknowledgment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HdrLen</th>
<th>0</th>
<th>Flags</th>
<th>Advertised window</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options (variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Data**
Logical (virtual) ports on a host

Incoming packets are multiplexed based on their destination port.
Logical (virtual) ports on a host

Incoming packets are multiplexed based on their destination port.

A single interface with IP 1.2.3.4
More on ports

Ports are 16-bit header fields (max port number: $2^{16} - 1$)

Ports 0–1023 are "well-known" for example port 443 for HTTPS

Ports 1024–65535 are so-called "ephemeral" ports given to clients (picked at random)

For more details look at the lecture slides

UDP and TCP, keywords: ports and sockets
Example: The Internet with NAT (lecture week 5)
Example: The Internet with NAT

Local Network
192.168.0.0/16
192.168.3.4
192.168.3.5

Server
5.6.7.8
port 80

Internet

NAT / R

9.10.11.12

IP:port
src 192.168.3.4:3001
dst 5.6.7.8:80

Local Network
192.168.0.0/16
Example: The Internet with NAT

- **Server**: 5.6.7.8, port 80
  - Application listens on port 80

- **Internet**
  - **R**
  - **NAT/R**

- **Local Network**: 192.168.0.0/16
  - **192.168.3.4**
  - **192.168.3.5**

- **Network 9.10.11.12**
  - **192.168.3.4**
  - **192.168.3.5**

- **Ephemeral Port**: 192.168.3.4:3001
  - Client listens on this port

- **Well-Known Port**: 5.6.7.8:80
  - Port 80

- ** NAT Configuration**: 
  - 

- **Reverse Path Forwarding (RPF)**
Example: The Internet with NAT

The NAT replaces source port and IP. Port 3001 with a new port 5000 and IP 192.168.3.4 with a public IP 9.10.11.12.
Example: The Internet with NAT

The packet reaches the correct application as it contains destination port 80

The NAT table is as follows:

- 192.168.3.4:3001 → 9.10.11.12:5000
- 9.10.11.12:5000 → 192.168.3.4:3001

Server
5.6.7.8
port 80

Network Diagram:
- Server (5.6.7.8) connected to the Internet through a NAT device (R).
- The NAT device maps 9.10.11.12:5000 to 192.168.3.4:3001.
- The packet reaches the correct application because it contains destination port 80.
Example: The Internet with NAT

The answer from the server goes towards destination port 5000
Example: The Internet with NAT

The NAT performs the reverse translation
Example: The Internet with NAT

The packet reaches the correct application on the client listening on port 3001.

5.6.7.8:80
192.168.3.4:3001

Server
5.6.7.8
port 80

NAT table
192.168.3.4:3001 ↔ 9.10.11.12:5000

Local Network
192.168.0.0/16
Example: The Internet with NAT

Important: all the here shown ports do not belong to a physical interface on a device. They are all logical ports on hosts.
Communication Networks

Exercise 9

Last week’s exercise

Important lecture topics

Introduction to this week’s exercise

Time to solve the exercise
Task 9.1: Reliable versus Unreliable Transport

Simple introduction question

Consider the information from the lecture slides
Task 9.2: Negative Acknowledgements

Instead of acknowledging what we received ...

... the receiver could also acknowledge not-received data

In which scenarios does this (not) work well?
Task 9.3: Fairness

In this question we consider a max-min fair allocation

Have a look at lecture slides 78-81 in

04_concepts_reliable_transport.pdf
Task 9.4: Understanding Go-Back-N’s Behavior

Consult the introduction slides we just discussed
Task 9.5: Reliable Transport

Draw time-sequence diagrams

sender

0

…

receiver

10 Mbps link
100 ms propagation delay
10000 bits in data segment
ACK size very small
Communication Networks

Exercise 9

Last week’s exercise

Important lecture topics

Introduction to this week’s exercise

Time to solve the exercise