General information

Introduction and demo

Python and Git tutorial

Eric and Hendrik
General information

Introduction and demo

Python and Git tutorial
Two important pillars of today’s Internet

Internet-wide routing
Covered in the first project

Reliable transport
Main focus of the second project
Implement your own Reliable Transport Protocol

recover from packet loss
and reordering
Implement your own **Reliable** Transport Protocol

recover from packet loss and reordering

**Part 1**

*Simple Go-Back-N implementation*

Retransmit all packets after a timeout

**Part 2**

*Support for Selective Repeat*

Fast retransmission after duplicated ACKs

**Part 3**

*Support for Selective Acknowledgements (SACK)*

SACK contains blocks of correctly received segments
The header of our Go-Back-N protocol is **6 bytes** long.

<table>
<thead>
<tr>
<th>1</th>
<th>7</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Options</td>
<td>Segment Length</td>
</tr>
<tr>
<td>Header Length</td>
<td>Sequence Number</td>
<td>Window</td>
</tr>
</tbody>
</table>

Payload
The header of our Go-Back-N protocol is 6 bytes long.

0 = DATA segment
1 = ACK segment
The header of our Go-Back-N protocol is 6 bytes long.

0 = no SACK support
1 = SACK support
The header of our Go-Back-N protocol is \textit{6 bytes} long

Length of the payload. Normally, 64 bytes. Only last segment could be smaller.
The header of our Go-Back-N protocol is 6 bytes long.

<table>
<thead>
<tr>
<th>Type</th>
<th>Options</th>
<th>Segment Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Length</td>
<td>Sequence Number</td>
<td>Window</td>
</tr>
</tbody>
</table>

Total length of the header. In bytes.
The header of our Go-Back-N protocol is 6 bytes long

In DATA: segment sequence number. Starts at 0
In ACK: next expected in-sequence segment
The header of our Go-Back-N protocol is **6 bytes** long.

Sender respectively receiver window size.
In number of segments.
For SACK we need an **optional** header
For SACK we need an **optional** header

Number of SACK blocks in the optional header
Between 1 and 3
For SACK we need an **optional** header

<table>
<thead>
<tr>
<th></th>
<th>Block Length</th>
<th>Left edge 1st block</th>
<th>Length 1st block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td></td>
<td>Left edge 2nd block</td>
<td>Length 2nd block</td>
</tr>
<tr>
<td>Padding</td>
<td></td>
<td>Left edge 3rd block</td>
<td>Length 3rd block</td>
</tr>
</tbody>
</table>

Start of the first block
For SACK we need an **optional** header

<table>
<thead>
<tr>
<th>Optional header</th>
<th>8</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Length</td>
<td>Left edge 1st block</td>
<td>Length 1st block</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Left edge 2nd block</td>
<td>Length 2nd block</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Left edge 3rd block</td>
<td>Length 3rd block</td>
<td></td>
</tr>
</tbody>
</table>

Length of the first block. In number of segments
A block with one segment has size 1
For SACK we need an **optional** header

Padding for better alignment
SACK example - Receiver

Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header:

SACK header:
Correctly received segments: 0, 1, 2
Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17
Mandatory header: ACK number: 3
SACK header:
SACK example - Receiver

Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header: ACK number: 3

SACK header:

<table>
<thead>
<tr>
<th>#blocks</th>
<th>start b1</th>
<th>size b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>start b2</td>
<td>size b2</td>
</tr>
<tr>
<td>Padding</td>
<td>start b3</td>
<td>size b3</td>
</tr>
</tbody>
</table>
SACK example - Receiver

Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header: ACK number: 3

SACK header:

<table>
<thead>
<tr>
<th>#blocks</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>start b2</td>
<td>size b2</td>
</tr>
<tr>
<td>Padding</td>
<td>start b3</td>
<td>size b3</td>
</tr>
</tbody>
</table>
Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header:

SACK header:

<table>
<thead>
<tr>
<th></th>
<th>#blocks</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>start b3</td>
<td>size b3</td>
<td></td>
</tr>
</tbody>
</table>
SACK example - Receiver

Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header:

SACK header:

<table>
<thead>
<tr>
<th>#blocks</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Padding</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>
SACK example - Receiver

Correctly received segments: 0, 1, 2
Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header:

SACK header:

<table>
<thead>
<tr>
<th>#blocks</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Padding</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>
SACK example - Receiver

Correctly received segments: 0, 1, 2

Buffered out-of-order segments: 4, 5, 8, 10, 11, 12, 13, 15, 16, 17

Mandatory header: ACK number: 3

SACK header:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Padding</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Padding</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>
There are multiple options to test your implementation

Run your sender against your receiver
Should be your main focus

Test with the implementation of another group
Good way to find out if you followed all the specifications

**Optionally**, use our test framework
Passing all the tests does not guarantee a 6
A new VM waits for you

All the scrips are already on your VM
Use scp or git to transfer files

You keep your group number from the first project
**Important:** VM port number is 3000 + group number

Use the password from the routing project
Let’s see how the **final** sender and receiver should look like
If you have questions

Ask on Slack
Please use the #transport_project channel

Visit an exercise sessions
Or an additional Q&A session
Final comments

Deadline: May 31 2019
Friday, end of the semester

Read the assignment text carefully
Make sure you follow all the specifications

Do not copy code from other groups
We will check your code with automated tools
Introduction and demo

Python and Git tutorial

Eric and Hendrik