Communication Networks

Spring 2019

Tobias Bühler

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

May 6 2019

Introduction and demo

General information

Python and Git tutorial

Eric and Hendrik

Introduction and demo

Python and Git tutorial

General information

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

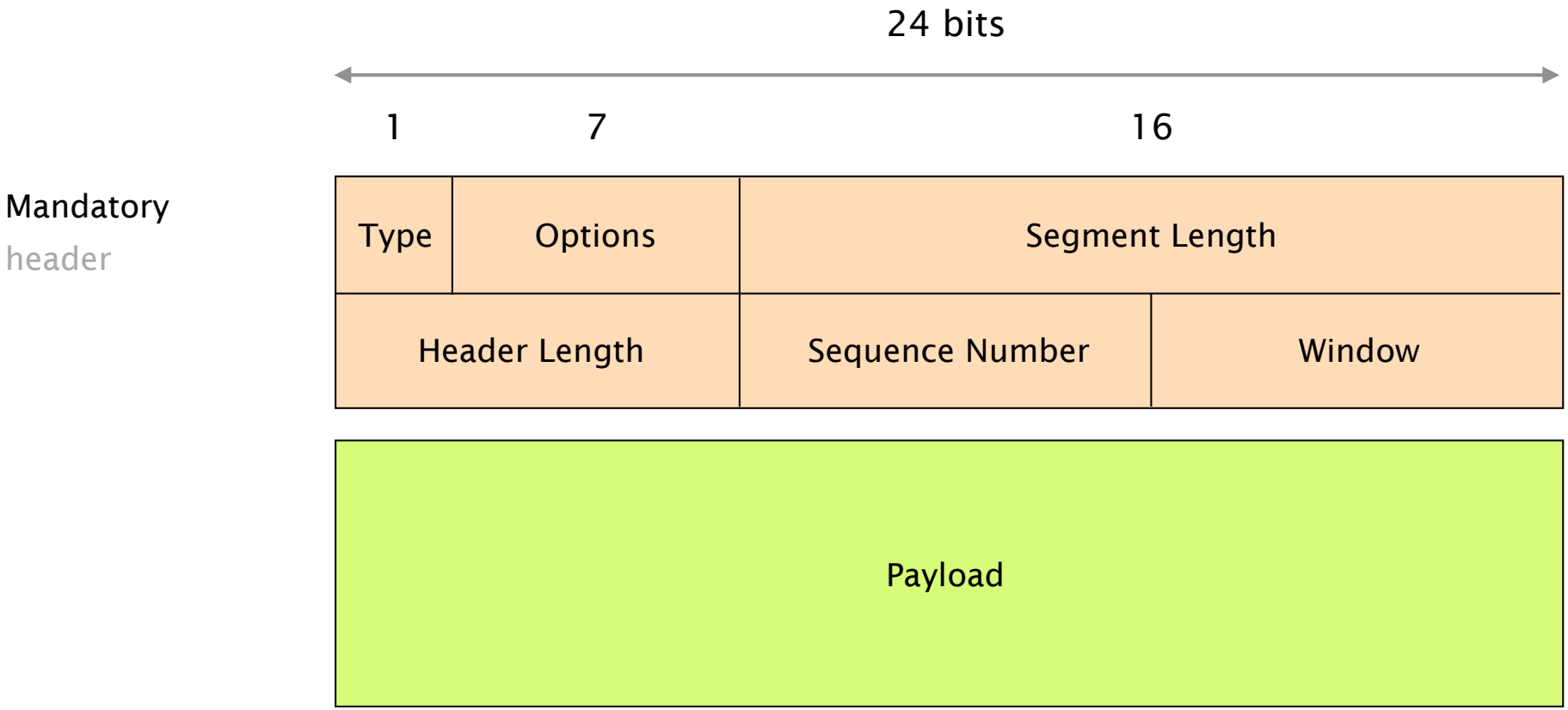
recover from packet loss
and reordering

Implement your own Reliable Transport Protocol

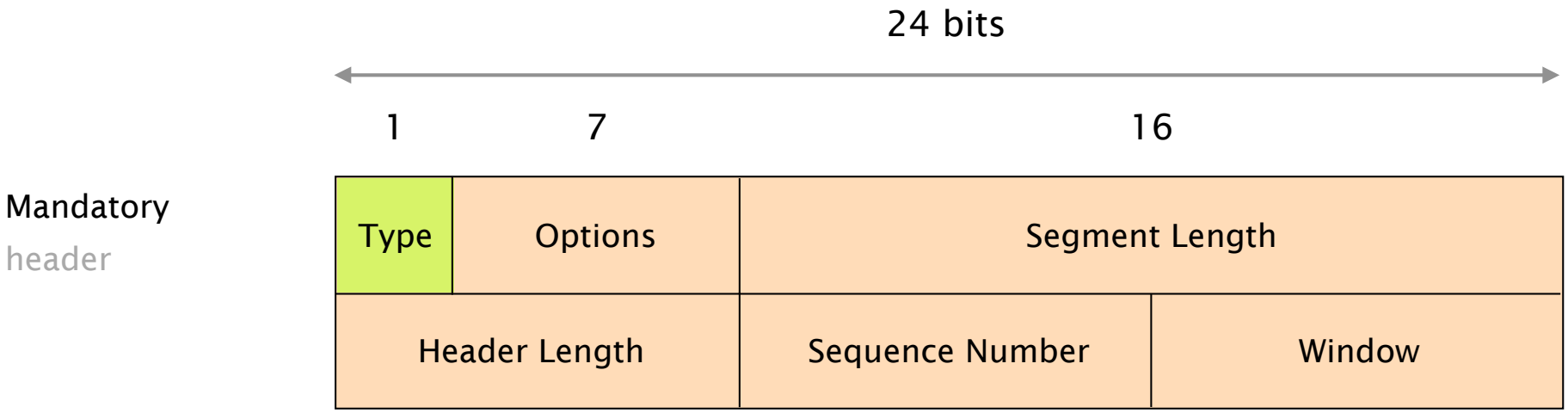
recover from packet loss
and reordering

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



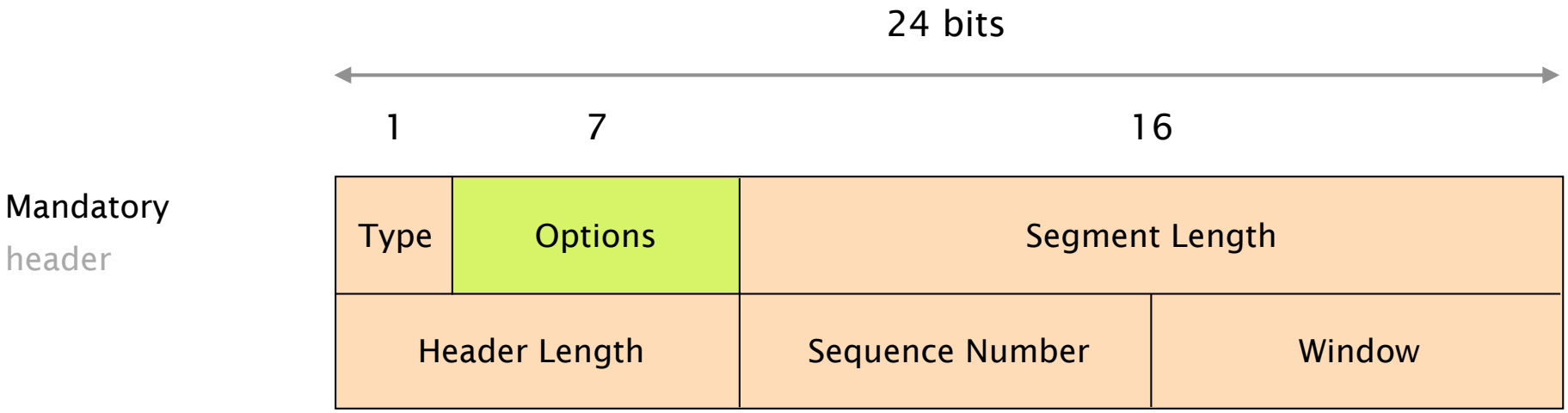
The header of our Go-Back-N protocol is 6 bytes long



0 = DATA segment

1 = ACK segment

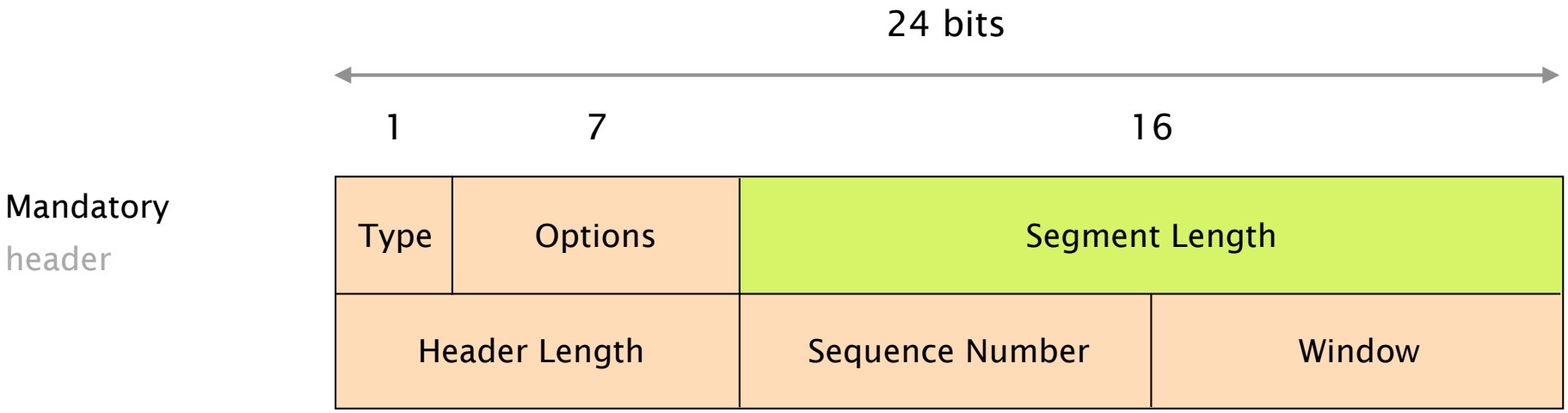
The header of our Go-Back-N protocol is 6 bytes long



0 = no SACK support

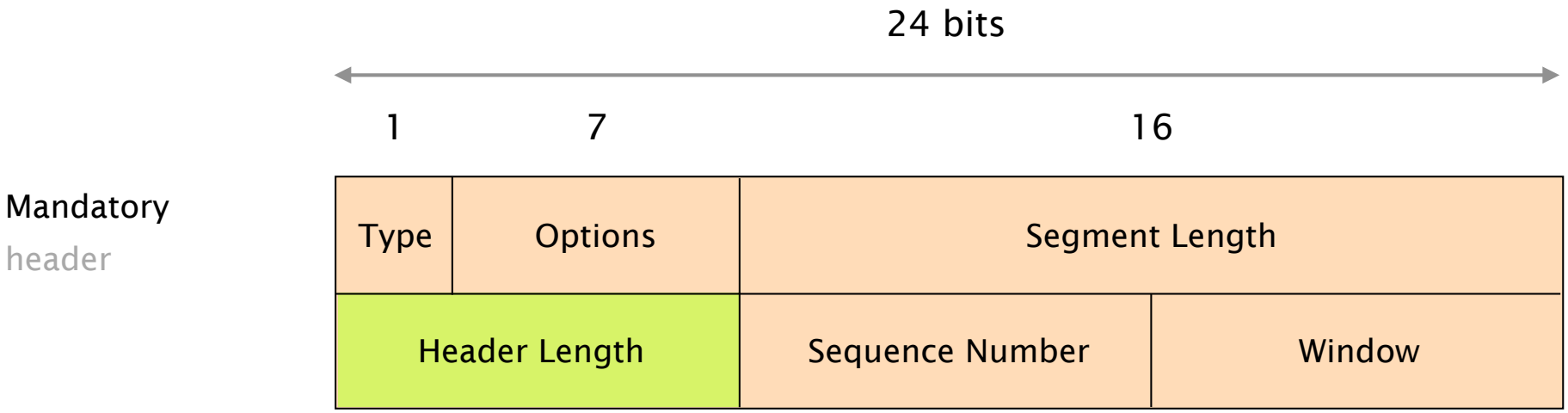
1 = SACK support

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



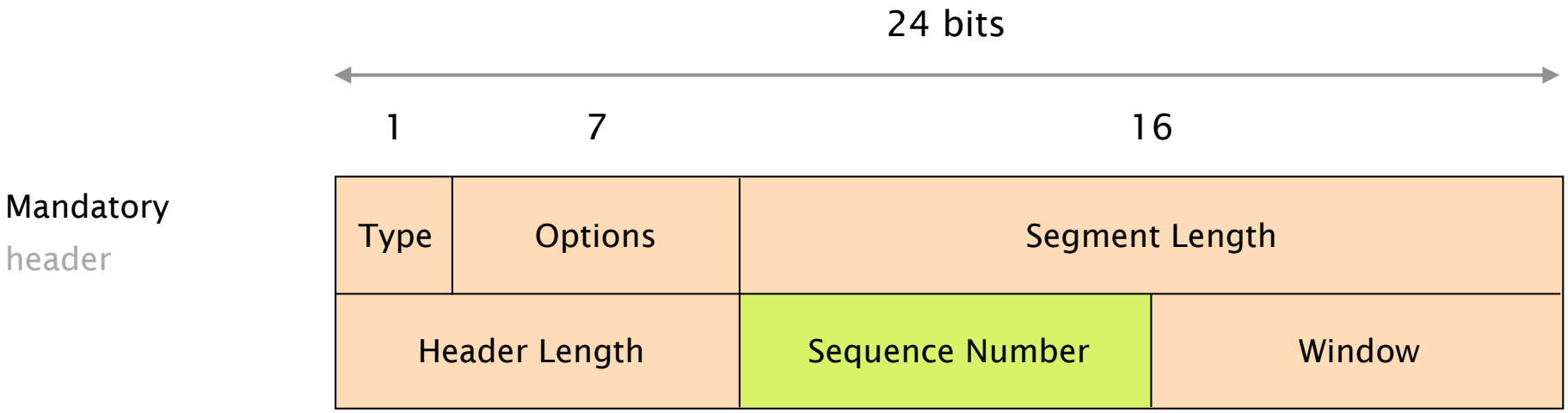
Length of the payload. Normally, 64 bytes.
Only last segment could be smaller

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



Total length of the header.
In bytes

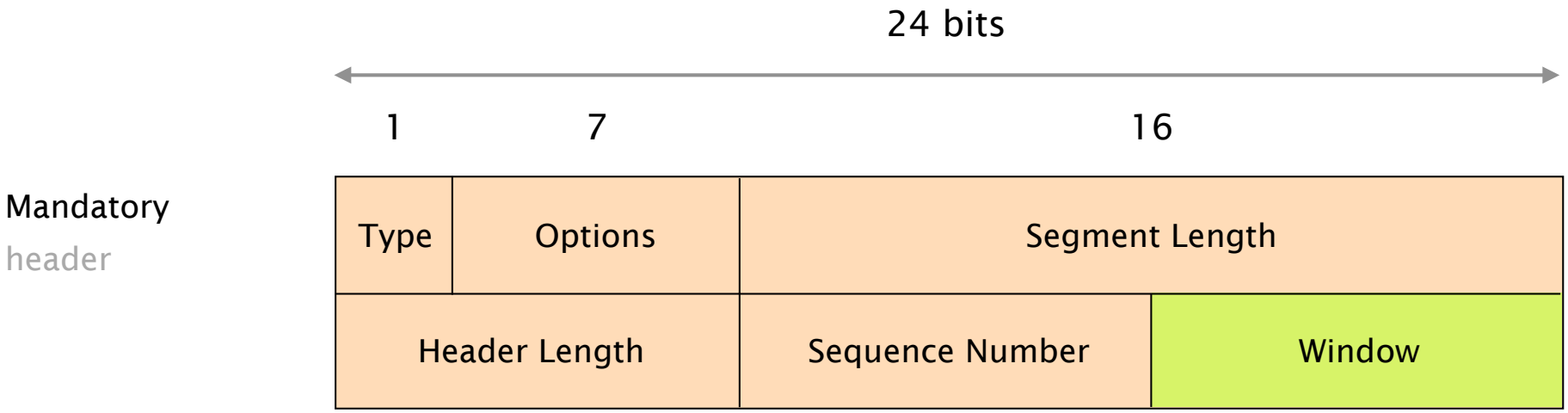
The header of our Go-Back-N protocol is 6 bytes long



In DATA: segment sequence number. Starts at 0

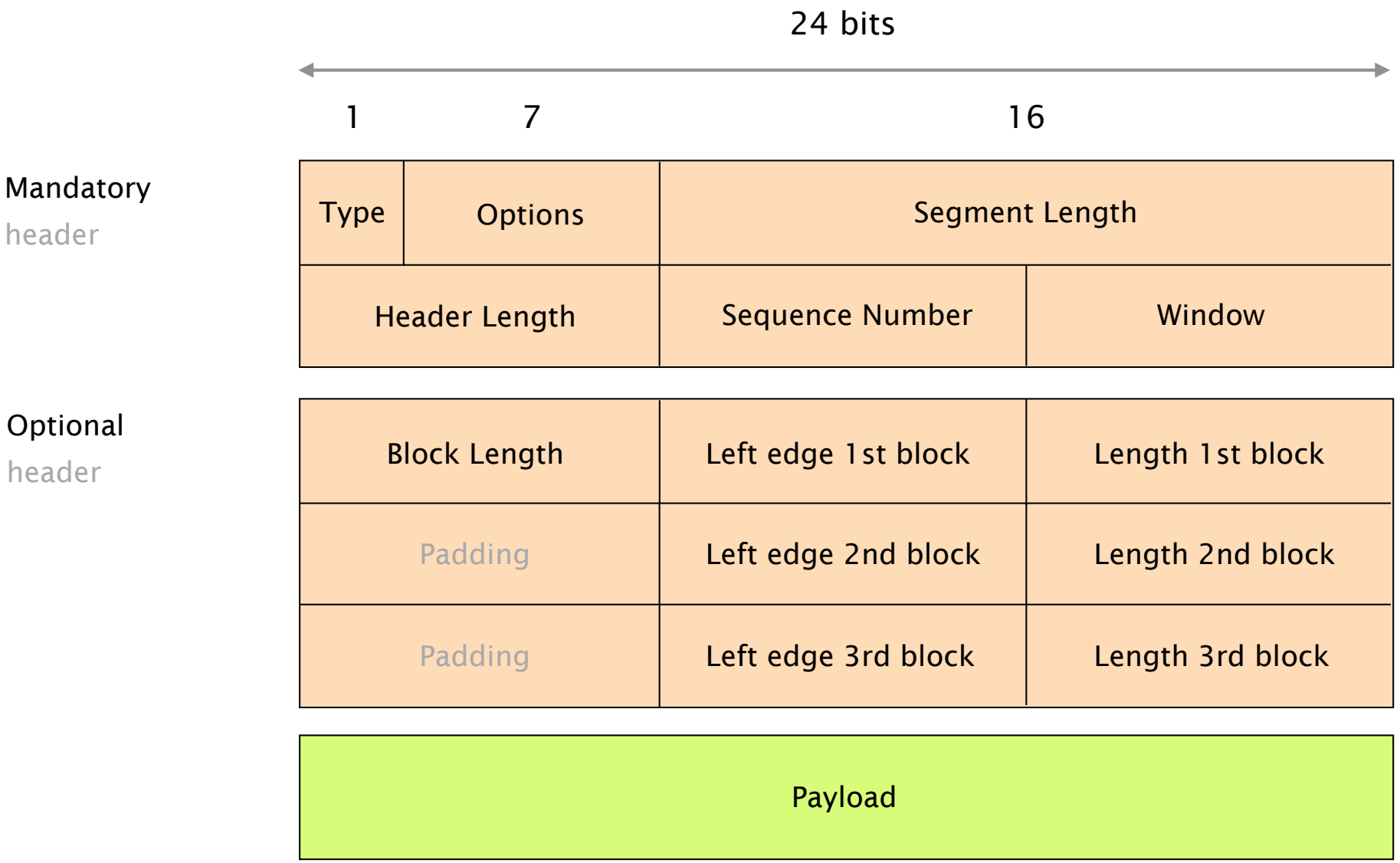
In ACK: next expected in-sequence segment

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



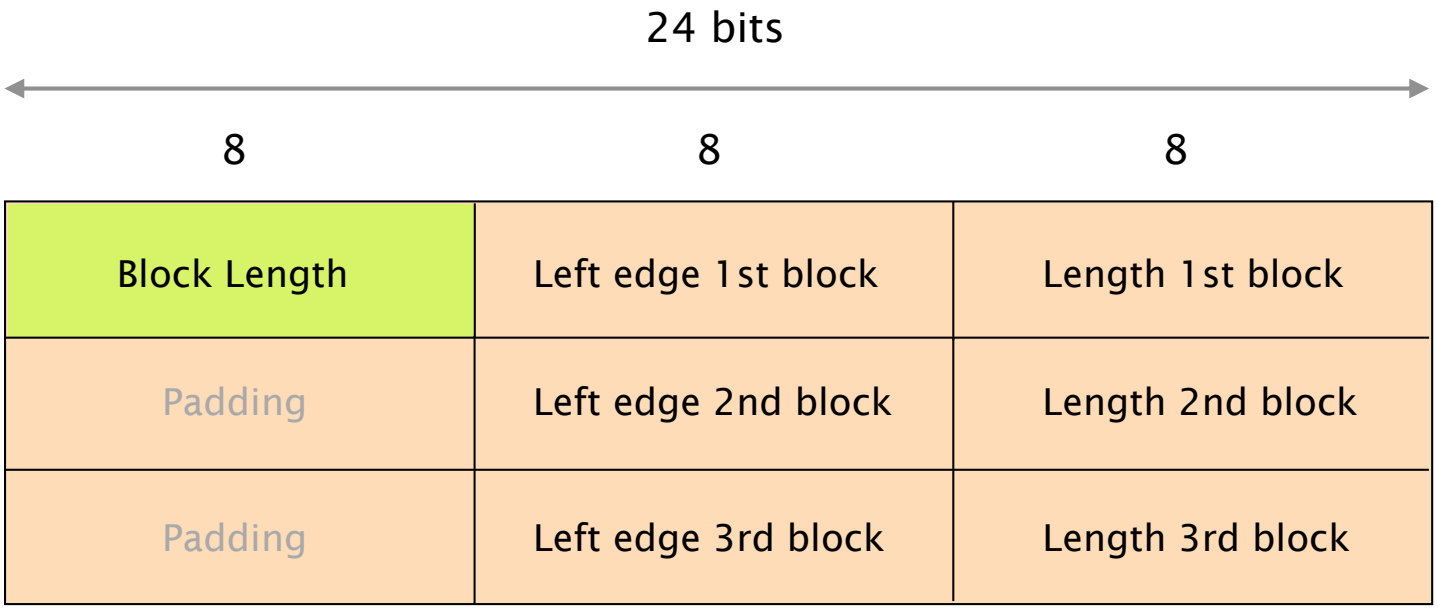
Sender respectively receiver window size.
In number of segments

For SACK we need an optional header



For SACK we need an optional header

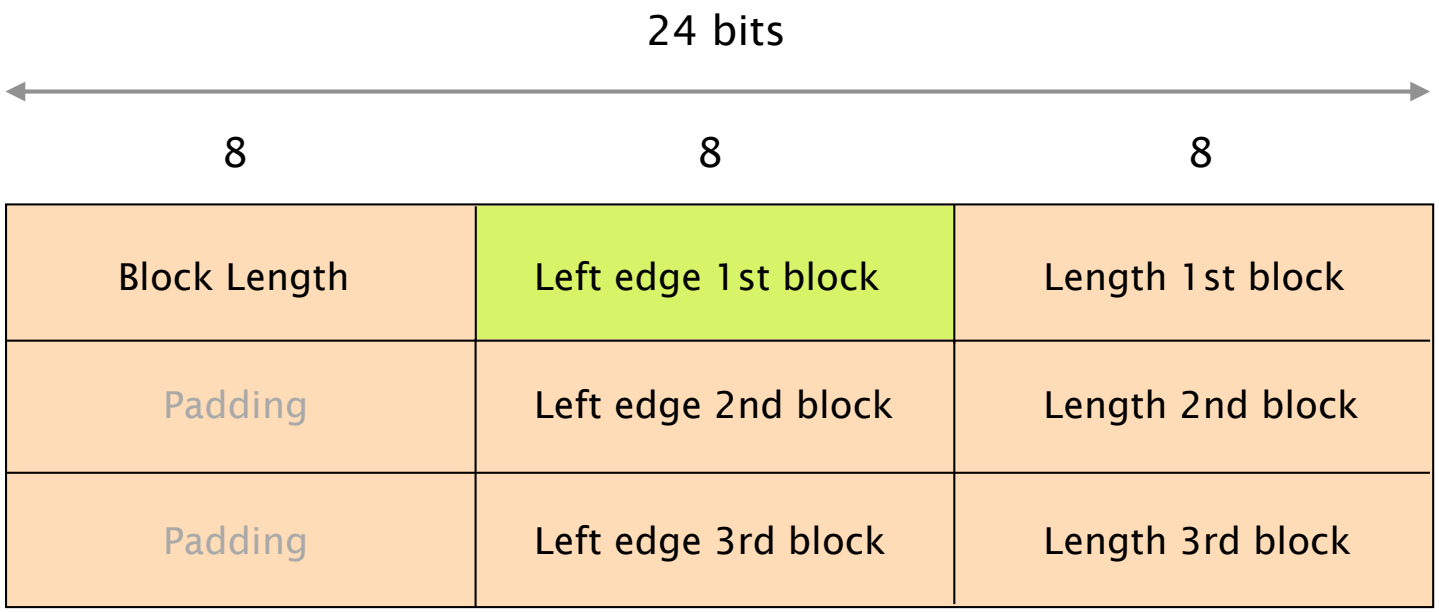
Optional
header



Number of SACK blocks in the optional header
Between 1 and 3

For SACK we need an optional header

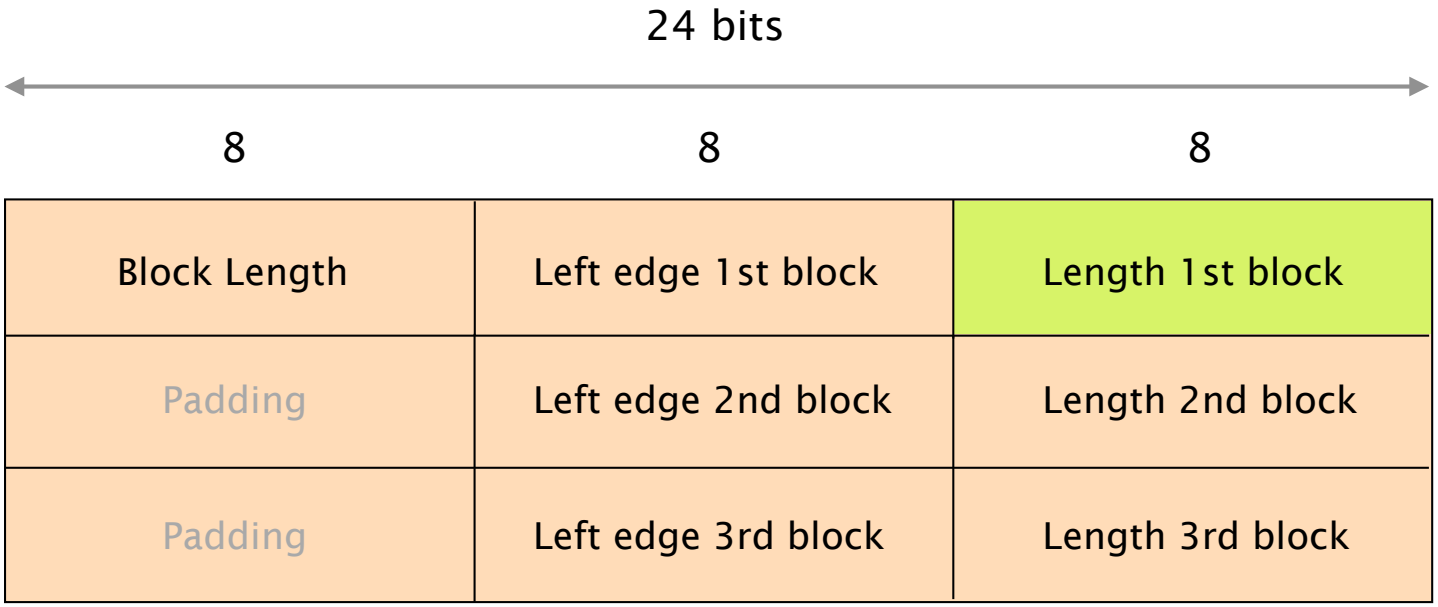
Optional header



Start of the first block

For SACK we need an optional header

Optional
header

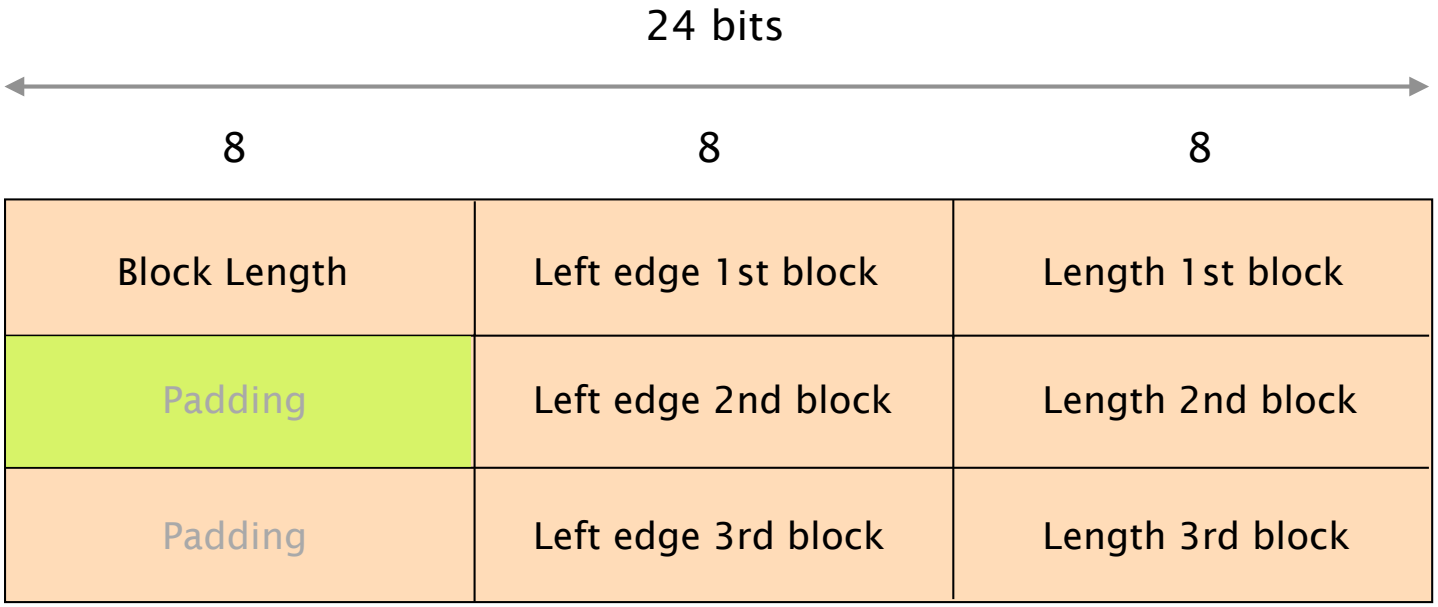


Length of the first block. In number of segments

A block with one segment has size 1

For SACK we need an optional header

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:

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:

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:

#blocks	start b1	size b1
Padding	start b2	size b2
Padding	start b3	size b3

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:

#blocks	4	2
Padding	start b2	size b2
Padding	start b3	size b3

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:

#blocks	4	2
Padding	8	1
Padding	start b3	size b3

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:

#blocks	4	2
Padding	8	1
Padding	10	4

SACK example - Receiver

Correctly received segments: 0, 1, 2

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

no space

Mandatory header: ACK number: 3

SACK header:

#blocks	4	2
Padding	8	1
Padding	10	4

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:

3	4	2
Padding	8	1
Padding	10	4

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