Congestion control aims at solving three problems:

1. Bandwidth estimation
   - How to adjust the bandwidth of a single flow to the bottleneck bandwidth?
   - Could be 1 Mbps or 1 Gbps...

2. Bandwidth adaptation
   - How to adjust the bandwidth of a single flow to variation of the bottleneck bandwidth?

3. Fairness
   - How to share bandwidth "fairly" among flows, without overloading the network.

Congestion control differs from flow control both are provided by TCP though:

- Flow control prevents one fast sender from overloading a slow receiver.
- Congestion control prevents a set of senders from overloading the network.

The sender adapts its sending rate based on these two windows:

- Receiving Window (RWND): How many bytes can be sent without overflowing the receiver buffer? Based on the receiver input.
- Congestion Window (CWND): How many bytes can be sent without overflowing the routers? Based on network conditions.
- Sender Window: $\min(CWND, RWND)$.

The two key mechanisms of Congestion Control:

- Detecting congestion
- Reacting to congestion
The 2 key mechanisms of Congestion Control

Detecting losses can be done using ACKs or timeouts, the two signals differ in their degree of severity:
- duplicated ACKs: mild congestion signal, packets are still making it
- timeout: severe congestion signal, multiple consequent losses

TCP approach is to gently increase when not congested and to rapidly decrease when congested.

It depends on the problem we are solving...

TCP approach is to gently increase when not congested and to rapidly decrease when congested.

What increase/decrease function should we use?
It depends on the problem we are solving...

Congestion control aims at solving three problems:

1. Bandwidth estimation: How to adjust the bandwidth of a single flow to the bottleneck bandwidth?
   - Could be 1 Mbps or 1 Gbps...
2. Bandwidth adaptation: How to adjust the bandwidth of a single flow to variation of the bottleneck bandwidth?
3. Fairness: How to share bandwidth "fairly" among flows, without overloading the network?

Initially, you want to quickly get a first-order estimate of the available bandwidth:

Intuition: Start slow but rapidly increase until a packet drop occurs

Increase policy:
- \( cwnd = 1 \) initially
- \( cwnd = 1 \) upon receipt of an ACK

#1 Bandwidth estimation: How to adjust the bandwidth of a single flow to the bottleneck bandwidth?
- Could be 1 Mbps or 1 Gbps...

#2 Bandwidth adaptation: How to adjust the bandwidth of a single flow to variation of the bottleneck bandwidth?
AIMD converge to fairness and efficiency, it then fluctuates around the optimum (in a stable way)

Congestion control makes TCP throughput look like a “sawtooth”

We now have completed the transport layer (!)

This week on Communication Networks

Routing Project
Reliable Transport Project
Recap, demo and final results
Introduction and demo
Python and Git tutorial

Routing Project
Reliable Transport Project
Recap, demo and final results
Introduction and demo
Python and Git tutorial
Recap

73%
Proportion of valid BGP paths in your mini-Internet

*From 15 traceroutes launched between random pairs of ASes

Your mini-Internet works!
and common services can run on top of it

For this project, you basically did what an actual network operator has to do
Including debugging and monitoring your configuration and connectivity

There was often multiple ways to answer the questions
and we found some interesting answers
Enabling authentication in OSPF

Group BGP neighbors to simplify configuration

Multiple valid answers for question 3.3

How we have built the mini-Internet
How we have built the mini-Internet

between VMs: 192.168.56.0/24

Communication Networks 2018
Routing Project

Except the grades within ~2weeks from now

Implement your own **Reliable** Transport Protocol

recover from packet loss
and reordering

Let's see how the final sender and receiver should look like

**Part 1**
Simple Go-Back-N implementation
Retransmit all packets after a timeout

**Part 2**
Support for Selective Repeat
Fast retransmission after repeated 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>Type</th>
<th>Options</th>
<th>Segment Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Mandatory header**
- **Payload**

Sequence number overflow

- **NBITS**
- **overflow**
- **application examples**
Sequence number overflow

\( \text{NBITS} \) controls the maximum sequence number

- assuming \( \text{NBITS}=3 \):
  \[ 2^{\text{NBITS}} - 1 = 7 \]

\( \cdots 5, 6, 7, 0, 1, 2, \cdots \)

- application examples:
  - ACK number, SACK header blocks, retransmission, ...

The Go-Back-N sender waits for a timeout before segments are retransmitted

<table>
<thead>
<tr>
<th>Sent segments:</th>
<th>0 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver behavior:</td>
<td>0 - 2 3 4 6</td>
</tr>
<tr>
<td>Out-or-order segments are dropped</td>
<td></td>
</tr>
<tr>
<td>Sent ACKs:</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>Retransmission:</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Selective Repeat can increase the performance

<table>
<thead>
<tr>
<th>Sent segments:</th>
<th>0 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver behavior:</td>
<td>0 - 2 3 4 5</td>
</tr>
<tr>
<td>Out-or-order segments are buffered</td>
<td></td>
</tr>
<tr>
<td>Sent ACKs:</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>Retransmission:</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

For SACK we need an optional header

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th><strong>Options</strong></th>
<th><strong>Segment Length</strong></th>
<th><strong>Header Length</strong></th>
<th><strong>Sequence Number</strong></th>
<th><strong>Window</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mandatory header</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optional header</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Length</td>
<td>Left edge 1st block</td>
<td>Length 1st block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Left edge 2nd block</td>
<td>Length 2nd block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Left edge 3rd block</td>
<td>Length 3rd block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packeted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optional header:

- Black Length
  - Left edge 1st block
  - Length 1st block
- Padding
  - Left edge 2nd block
  - Length 2nd block
- Packeted
  - Left edge 3rd block
  - Length 3rd block

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

Start of the first block

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: 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:
SACK header:
### SACK example - Receiver

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Correctly received segments: 0, 1, 2
Buffered out-of-order segments: 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17
Mandatory header: ACK number: 3
SACK header: 

### SACK example - Receiver

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</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: ACK number: 3
SACK header: 

### SACK example - Receiver

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</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: ACK number: 3
SACK header: 

### SACK example - Sender

Receiver SACK header:

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

ACK number: 3
ACK - block 1: 3
block 1 - block 2: 6, 7
block 2 - block 3: 9
after block 3: no retransmission

### SACK example - Sender

Receiver SACK header:

<table>
<thead>
<tr>
<th>Blocks</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

ACK number: 3
ACK - block 1: 3
block 1 - block 2: 6, 7
block 2 - block 3: 9
after block 3: no retransmission
To test your implementation…

... run your sender against your receiver
... test with the implementation of another group
... optionally, use our test framework

Ask your questions on Slack (#transport_project) or visit an exercise session

Tobias Bühler (@buehlert)
Maximilian Schütte (@Maximilian (TA))
Alexander Dietmüller (@Alexander (TA))
Rüdiger Brünker (@rbirkner)
Roland Meier (@roland)
Thomas Holterbach (@thomas_holterbach)

Next week on Communication Networks

This Thursday: Ascension Day
Monday: Applications: DNS and HTTP

The Hitchhiker’s Guide to Efficient Python Development

Communication Networks
Spring 2018
ETH Zürich

Contents

# Why we use Python

# Stop wasting time: Editors, Linters, File Sync

# Get to know the framework

# Avoiding Catastrophe: Version Control

# Git made easy: GitLab and SourceTree

Python
Slither along with your friendly neighbourhood snake!
Reasons to choose Python

- Interpreted Language
- Many packages available
- Simple yet powerful Syntax / Beginner Friendly
- Often used in academia and science

Learn the Basics BEFORE You Start!

We promise the basics will pay off...

Learn the Basics BEFORE You Start!

One afternoon on learnpython.org should suffice

- If you skip the preparation, bugs may go unnoticed and cost you points
- Also you will spend much more time on debugging than you would have to learn the python basics

Learning Python for Everyone

- Interactive Getting Started Guide
  - http://www.learnpython.org/
  - Short Intro
    - https://developers.google.com/edu/python/
  - Not So Short Intro
    - https://docs.python.org/3/tutorial/index.html
  - Detailed Intro
    - https://learnpythonthehardway.org/python3/
  - Free Video Series for Beginners
    - https://developers.google.com/edu/python/tutorials/learncode-py/tutorial-2-linked-list
  - Udemy Lecture for Beginners
    - https://www.udemy.com/course/learn-python-bootcamp/

Learning Python for Pros


Python 2.7 or 3.x?

- Python 2.7 is slowly dying
- Python 3.x is cleaner, better, faster, stronger...

Which Python Shall It Be?

Two major distributions to consider...
Which Python Shall It Be?

CPython from python.org
- The “default” distribution
- Is installed on the VMs
- Comes only with standard library
- Pip packet manager

Anaconda by Continuum Analytics
- Optimized for data science and large scale science apps
- Derived from CPython
- Ships with a big library of science related packages
- Uses Conda packet manager
- But also supports pip

VSCode & PyLint
It's 2018, get your development workflow together!

Editing on the console is cumbersome...
... but sometimes useful for quick fixes!

Many good Python IDEs available!
- Sublime Text
- Visual Studio Code
- Atom.io
- JetBrains PyCharm
- Eclipse PyDev

Many good Python IDEs available!
#Any of the above will do, you the one you know and adapt it to the project!
#Top three are basic and can be used for many programming languages
#PyCharm is the most powerful Python IDE and even free for ETH students (professional edition)

Integrated Development Environment Benefits
#Easy to set up and getting started
#Come with many supporting tools out of the box
#IntelliSense, Syntax Checker / Linter, Auto completion...
#GUI based debugging is much faster and easier

Linters
#A Linter performs static code analysis
#It points out...
#... errors in your code
#... redundant code
#... code that can be optimized
#... changes that improve the readability of your code
#Use it so you don’t have to spend hours chasing typos!

Secure File Transfer Protocol (SFTP)
#Available via extension for Visual Studio Code
#Makes transferring files from / to the VM super easy
#Extension shows you differences between local and vm code
# Step-by-Step Installation Reference

- Install CPython 3.x or Anaconda / Miniconda 3.x
  - [https://www.python.org/downloads/](https://www.python.org/downloads/)
  - [https://www.anaconda.com/download/](https://www.anaconda.com/download/)
- Install Visual Studio Code
  - [https://code.visualstudio.com/](https://code.visualstudio.com/)
- Start Visual Studio Code and click on the extensions icon on the left
- Search for and install Python (ms-python.python) and sftp (liximomo.sftp)
- Reload after BOTH installations have finished

---

# Configure Python and PyLint in VSCode

- Press F1 and enter "Python: Select Interpreter"
- Choose the python version that you just installed.
- On Mac, use the one in /usr/local, NOT the system installation!
- Press F1 again and enter "Python: Select Linter" and choose "PyLint"
- The first time you open a python file, you will receive a message box in the bottom right corner saying that PyLint is not installed. Press "install" to do so.
- On Mac, go will be installed if not installed already.

---

# Configure sftp and Download Code Reference

- In VSCode, open a folder where you want your project files to be located.
- Press F1 and enter "SFTP: Config"
- A config file will pop up. Enter the details to your VM, as shown on the next slide. Providing a password is optional.
- The config will be stored in a subfolder `.vscode` and can be edited anytime.
- Right-click in the VSCode file browser and use the SFTP features like "download", "upload", or "sync".
- In general, the plugin is conservative when it comes to destructive operations. See Extension Info page for more details.

---

# The Project Skeleton

You don’t need to start from scratch...

---

# Sending and Receiving Packets in Python

```python
from scapy.all import send, IP, TCP
Payload = b"This is some binary test data."
packet = IP(src="192.168.0.1", dst="8.8.8.8") / TCP() / payload
send(packet)
```

Combine headers with the division operator

---

# SFTP Example Config

```json
{
  "protocol": "sftp",
  "host": "samichlaus.ehez.ch",
  "username": "root",
  "port": 3000+YOUR-GROUP-NUMBER,
  "remotePath": ".", "ignore": ".*"
}
```

Don’t forget this! It makes sure that you just copy the project related files!
Sending and Receiving Packets in Python

```python
print(packet.summary())
print(packet.show())
from scapy.all import IP
ip_header = packet.getlayer(IP)
source_address = ip_header.src
payload = ip_header.payload
```

Define Your Own Header

```python
from scapy.all import Packet, bind_layers, BitEnumField, BitField

class GBN(Packet):
    name = 'GBN'
    fields_desc = [
        BitEnumField("type", 0, 1, {0: "data", 1: "ack"}),
        BitField("options", 0, 7),
    ]

# Tell Scapy where to look for the header when receiving a packet
bind_layers(IP, GBN, frag=0, proto=222)
```

Our GBN Automaton is powered by Scapy

```
from scapy.all import Automaton, ATMT
class GBNSender(Automaton):
    @ATMT.state(initial=1)
    def BEGIN(self):
        raise self.SEND()
    @ATMT.state()
    def SEND(self):
        # Your code!

Transitions are triggered by exceptions
```

Where to start?

- The GBN header is already defined...
  you’ll need to extend it in later questions
- The automaton skeleton is fully implemented...
  no new states or transitions needed
- The receiver already works for the first question...
  complete the sender, check receiver for inspiration

Version Control

```
Everyon works on the same file and uploads it to the server.
The version uploaded last overwrites all other changes.
```

Without git

```
Tracks Changes in Source Code
```

```
```

```
With git

Everyone works on the same file and pushes the changes to the git repository.

All changes are combined, nothing is lost.

**git Workflow**

1. Create Repository
2. Invite Group Members

```
git clone <repository>
codecodecodecode...
git commit
```

1. git pull
2. git push

Try it yourself and learn more:
http://try.github.io/
https://backlog.com/git-tutorial/

**SourceTree & Gitlab**

... because no matter what they say, GUI matters.