Communication Networks
Spring 2022

Q&A Session

Coralie Busse-Grawitz
Tobias Bühler

ETH Zürich
August 15 2022
Today’s Q&A session

- Important concepts
- Answering received questions
- Individual questions
Today’s Q&A session

- Important concepts
- Answering received questions
- Individual questions
Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)
Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)
Since when bits arrive they must make it to the application, all the layers exist on a host.
Routers act as **L3 gateway**
as such they implement L2 and L3
Switches act as **L2 gateway** as such they only implement L2.
Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)
Discussed in the second part

A lot of your questions are related to BGP
Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)
Go-Back-N (GBN)

Sender
0 1 2 3 4 5 6 7 8 9

Receiver
0 1 2

ACK'ed
unACK'ed

ready to send
waiting
Go-Back-N (GBN)

Sender 0 1 2 3 4 5 6 7 8 9
Ack'ed

Receiver 0 1 2
unAck'ed

ready to send

waiting

Timeout: retransmit N segments
Different types of ACKs

**Cumulative ACKs:** acknowledges all previous data segments
But no information which data segment triggered it
TCP uses cumulative ACKs

**Individual ACKs:** no connection to previous data segments
Over time sender can detect missing data segments
Often used together with per-packet timers
Different types of ACKs - SACK

**SACK** (Selective ACKnowledgement): combines advantages
- Bigger ACKs and more complicated implementation
- Detailed view of the buffer at the receiver side

**Important:** acknowledged segments in the SACK header are **not** removed from the sender window/buffer
Important TCP features

- TCP performs an initial handshake and session tear down

- TCP is stream oriented (sends/receives a byte stream)

- TCP retransmits packets after receiving duplicated ACKs

- TCP performs flow control (to protect the receiver)

- TCP performs congestion control (to protect the network)

- TCP often sends and receives bytes at the same time
The TCP CC algorithm tries to solve three problems

It can **estimate** the available bandwidth

It can **adapt** the bandwidth usage of a flow

And it tries to share the available bandwidth **fairly** between flows

The algorithm discussed in the lecture uses AIMD

Increase slowly, decrease very quickly

There exists a large number of different CWND algorithms

All with their own benefits and problems
Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)
traceroute is a tool to reconstruct the path of traffic

current out of context

traceroute exploits the time-to-live (TTL),
as routers return an “ICMP Time exceeded” when it is zero

traceroute sends three packets with the same TTL
to explore multiple paths (hint: loadbalancing)

traceroute increases the TTL with every round
to explore the path step-by-step
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute

TTL=1

TTL=2
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
Simple illustration of traceroute
traceroute output

```bash
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
1  internetbox (192.168.1.1) 5.690 ms 4.478 ms 5.014 ms
2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1) 4.980 ms 10.336 ms 3.920 ms
3   * * *
4   * * *
5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155) 9.031 ms 4.717 ms 4.757 ms
6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38) 4.986 ms 4.130 ms 4.925 ms
7   ip4.gtt.net (212.115.128.45) 5.004 ms 4.504 ms 9.982 ms
8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214) 90.148 ms 89.701 ms 90.097 ms
9   ip4.gtt.net (209.120.137.218) 89.987 ms 89.607 ms 90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108) 89.754 ms 129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110) 89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116) 89.454 ms 89.948 ms 90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38) 90.985 ms 89.724 ms 90.310 ms
15  * * *
16  * * *
```
traceroute output

→ ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets

1 internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
2 1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
* * *
3
4  *
5  *
6  *
7  *
8  *
9  *
10  *
11 nyugwa-ptp-dmgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
12 nyugwa-ptp-dmgwvb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
13 nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
14  *
15  *
16  *

TTL values/hops
traceroute output

traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
3  * * *
4  * * *
5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *

name and IP address of the hop
traceroute output

traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
3  *  *  *
4  *  *  *
5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  *  *  *
11  nyugwa-tp-tp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-tp-tp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  *  *  *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  *  *  *
16  *  *  *

not all hops reply
traceroute output

traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
1  internetbox (192.168.1.1) 5.690 ms 4.478 ms 5.014 ms
2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1) 4.980 ms 10.336 ms 3.920 ms
3  *  *  *
4  *  *  *
6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38) 4.986 ms 4.130 ms 4.925 ms
7  ip4.gtt.net (212.115.128.45) 5.004 ms 4.504 ms 9.982 ms
8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214) 90.148 ms 89.701 ms 90.097 ms
9  ip4.gtt.net (209.120.137.218) 89.987 ms 89.607 ms 90.059 ms
10  *  *  *
11  nyugwa-ptp-dmzgw-vl3081.net.nyu.edu (128.122.254.108) 89.754 ms 129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110) 89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116) 89.454 ms 89.948 ms 90.312 ms
13  *  *  *
14  wsqdrgwa-vl902.net.nyu.edu (128.122.1.38) 90.985 ms 89.724 ms 90.310 ms
15  *  *  *
16  *  *  *

RTT for all three probes
**ping** is a tool to test reachability of hosts

**ping** relies on ICMP echo request messages 
and the destination sends ICMP echo reply messages back

**ping** measures the RTT from source to destination, 
reports packet loss and provides a statistical summary
Destination address

```plaintext
(base) rbirkner@RJBMBP ➜  ping sydney.edu.au
PING sydney.edu.au (129.78.5.8): 56 data bytes
64 bytes from 129.78.5.8: icmp_seq=0 ttl=235 time=344.896 ms
64 bytes from 129.78.5.8: icmp_seq=1 ttl=235 time=363.529 ms
64 bytes from 129.78.5.8: icmp_seq=2 ttl=235 time=338.075 ms
64 bytes from 129.78.5.8: icmp_seq=3 ttl=235 time=318.827 ms
64 bytes from 129.78.5.8: icmp_seq=4 ttl=235 time=318.279 ms
64 bytes from 129.78.5.8: icmp_seq=5 ttl=235 time=318.923 ms
64 bytes from 129.78.5.8: icmp_seq=6 ttl=235 time=318.162 ms
64 bytes from 129.78.5.8: icmp_seq=7 ttl=235 time=318.173 ms
64 bytes from 129.78.5.8: icmp_seq=8 ttl=235 time=406.951 ms
64 bytes from 129.78.5.8: icmp_seq=9 ttl=235 time=325.697 ms
64 bytes from 129.78.5.8: icmp_seq=10 ttl=235 time=351.016 ms
```
<table>
<thead>
<tr>
<th>Test</th>
<th>RTT Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Round Trip Time Both directions!</td>
</tr>
<tr>
<td>#11</td>
<td></td>
</tr>
</tbody>
</table>
Today’s Q&A session

- Important concepts
- Answering received questions
- Individual questions
How does BGP update when the network changes?

Which messages are exchanged?

E.g., due to a link failure
On the wire, BGP is a rather simple protocol composed of four basic messages:

<table>
<thead>
<tr>
<th>Type</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>establish TCP-based BGP sessions</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>report unusual conditions</td>
</tr>
<tr>
<td>UPDATE</td>
<td>inform neighbor of a new best route</td>
</tr>
<tr>
<td></td>
<td>a change in the best route</td>
</tr>
<tr>
<td></td>
<td>the removal of the best route</td>
</tr>
<tr>
<td>KEEPALIVE</td>
<td>inform neighbor that the connection is alive</td>
</tr>
</tbody>
</table>
On the wire, BGP is a rather simple protocol composed of four basic messages

<table>
<thead>
<tr>
<th>Type</th>
<th>Used to</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>establish TCP-based BGP sessions</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>report unusual conditions</td>
</tr>
<tr>
<td>UPDATE</td>
<td>inform neighbor of a new best route</td>
</tr>
<tr>
<td></td>
<td>a change in the best route</td>
</tr>
<tr>
<td></td>
<td>the removal of the best route</td>
</tr>
<tr>
<td>KEEPALIVE</td>
<td>inform neighbor that the connection is alive</td>
</tr>
</tbody>
</table>
On the wire, BGP is a rather simple protocol composed of four basic messages

<table>
<thead>
<tr>
<th>type</th>
<th>used to…</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>establish TCP-based BGP sessions</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>report unusual conditions</td>
</tr>
<tr>
<td>UPDATE</td>
<td>inform neighbor of a new best route</td>
</tr>
<tr>
<td></td>
<td>a change in the best route</td>
</tr>
<tr>
<td></td>
<td>the removal of the best route</td>
</tr>
<tr>
<td>KEEPALIVE</td>
<td>inform neighbor that the connection is alive</td>
</tr>
</tbody>
</table>
There are two types of UPDATEs

UPDATE inform neighbor of

a new best route
a change in the best route
the removal of the best route
There are two types of UPDATEs

- UPDATE
  - inform neighbor of
    - a new best route
    - a change in the best route
    - the removal of the best route

Advertisements
There are two types of UPDATEs

UPDATE inform neighbor of

Advertisements —— a new best route

Withdrawal —— a change in the best route

Withdrawal —— the removal of the best route
Let’s look at an example (adapted from exercise 7.2)
Forwarding paths without any failures

AS 4 -> AS 1 -> AS 2
AS 3 -> AS 0 -> AS 2
82.130.64.0/21
Let’s focus on AS 0, we assume it has 4 routers
Most routers know one route towards 82.130.64.0/21, the router connected to AS 1 knows two routes.
Now the link between AS 0 and AS 2 fails
The directly connected routers detect the failure first
The router closest to the failure withdraws its current best route

route from a peer is forwarded to customer
The router closest to the failure withdraws its current best route.

AS 0

82.130.64.0/21

route from a peer is forwarded to customer

AS 1

82.130.64.0/21

[0 2]

preferred

AS 2

82.130.64.0/21

[0 2]
The router connected to AS 1 announces its new best route

route from a peer is forwarded to customer
Every router is once again able to reach the prefix

route from a peer
is forwarded to customer
Updated forwarding paths

AS 4 <-> AS 1
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS 0</td>
<td></td>
<td>AS 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|AS 3|     |     | 82.130.64.0/21
Now the link between AS 0 and AS 2 is working again
AS 0 receives an update from AS 2
The router connected to AS 2 propagates this route as it is preferred.
At this point, every router knows about both routes

route from a peer is forwarded to customer

AS path

82.130.64.0/21 [0 2] ← preferred
82.130.64.0/21 [0 1 2]
Finally, the router connected to AS 1 withdraws its second, non-best route.

This withdrawal does not need to be forwarded to the peer. The previous update implicitly removed the route.
We are once again in the initial state.
High-level recap of the inner workings of the BGP protocol

How does the decision process work?
All acceptable routes

Best route to each destination

IP forwarding table
Given the set of all acceptable routes for each prefix, the BGP Decision process elects a single route. BGP is often referred to as a single path protocol.
Prefer routes…

with higher LOCAL-PREF

with shorter AS-PATH length

with lower MED

learned via eBGP instead of iBGP

with lower IGP metric to the next-hop

with smaller egress IP address (tie-break)
An AS influences the traffic by modifying route attributes
<table>
<thead>
<tr>
<th>Attributes</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXT-HOP</td>
<td>egress point identification</td>
</tr>
<tr>
<td>AS-PATH</td>
<td>loop avoidance</td>
</tr>
<tr>
<td></td>
<td>outbound traffic control</td>
</tr>
<tr>
<td></td>
<td>inbound traffic control</td>
</tr>
<tr>
<td>LOCAL-PREF</td>
<td>outbound traffic control</td>
</tr>
<tr>
<td>MED</td>
<td>inbound traffic control</td>
</tr>
</tbody>
</table>
How relevant are the group projects for the exam?
How relevant are the group projects for the exam?

We will not ask you to write correct FRRouting configs
But you should be able to describe something in pseudo code

We could also show you a simple FRRouting config snippet
You should be able to understand what is happening

You do not need to write Python program code
Again, we could ask to explain something in pseudo code
How does the routing project relate to what we just saw?

Where do the "route-maps" come into play?
All acceptable routes

BGP Decision Process

Best route to each destination

forwarding entries

IP packets

IP forwarding table
All acceptable routes

BGP Decision Process

Best route to each destination

IP forwarding table

Input filters
Attribute Manipulation
Input filters
Attribute Manipulation
Input filters
Attribute Manipulation

Output filters
Attribute Manipulation
Output filters
Attribute Manipulation
Output filters
Attribute Manipulation

Neighbor 1
Neighbor 2
Neighbor n

BGP sessions
Adj-RIB-In
Adj-RIB-Out
BGP sessions

Loc-Rib

all routes shown in the looking glass

"in" route maps

forwarding entries

"out" route maps

in route maps

IP packets

IP packets
Let’s look at some examples

The shown examples are not correct FRRouting configs!
But they roughly show the level we would expect at the exam
Policy enforcement with local pref and BGP communities

IN route-map
c connected to a customer

OUT route-map
c connected to a provider
Policy enforcement with local pref and BGP communities

**IN** route-map connected to a **customer**

- set local pref to 1000
- add community 10:500

**OUT** route-map connected to a **provider**
Policy enforcement with local pref and BGP communities

**IN** route-map connected to a **customer**

for all routes:

- set local pref to 1000
- add community 10:500

**OUT** route-map connected to a **provider**

if route has community 10:500:

- send out

if route has community 10:100 or 10:200:

- drop / do not send out

if route is one of our prefixes:

- send out
Policy enforcement with local pref and BGP communities

**IN** route-map connected to a **customer**

for all routes:
- set local pref to 1000
- add community 10:500

influences BGP decision process

as tags for other routers in the same network

**OUT** route-map connected to a **provider**

if route has community 10:500:
- send out

if route has community 10:100 or 10:200:
- drop / do not send out

if route is one of our prefixes:
- send out

our own prefixes are always advertised

drop routes from peers and providers
Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

**OUT route-map**
connected to **peer** in ZURI

**OUT route-map**
connected to **same** peer in BERN
Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

**OUT** route-map connected to peer in ZURI

for our prefix:

-> set MED to 10

**OUT** route-map connected to same peer in BERN
Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

- set MED to 10

**OUT** route-map connected to **same** peer in BERN

for our prefix:

- set MED to 200
Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

We try to influence the BGP decision process of the peer
Remember, lower is better!

\[\text{OUT route-map connected to peer in ZURI}\]

for our prefix:

\[\rightarrow \text{set MED to 10}\]

\[\text{OUT route-map connected to same peer in BERN}\]

for our prefix:

\[\rightarrow \text{set MED to 200}\]
Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

- send out unmodified

**OUT** route-map connected to **same** peer in BERN
Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

- send out unmodified

**OUT** route-map connected to **same** peer in BERN

for our prefix:

- add our AS number 5 times to AS PATH
Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

**OUT route-map**
connected to **peer** in ZURI

for our prefix:

-> send out unmodified

**OUT route-map**
connected to **same** peer in BERN

for our prefix:

-> add our AS number 5 times to AS PATH

This time we influence
the AS PATH length
Could still be overwritten
by peer (e.g., local pref)
Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

-> send out unmodified

**OUT** route-map connected to **same** peer in BERN
Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

-> send out unmodified

**OUT** route-map connected to **same** peer in BERN

for our prefix:

-> drop / do not send out
Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

**OUT** route-map connected to **peer** in ZURI

for our prefix:

- send out unmodified

**OUT** route-map connected to **same** peer in BERN

for our prefix:

- drop / do not send out

We should really only receive traffic via ZURI
But what if this connection fails? We lose all traffic
Exercise 7, task 3c)

given A’s and C’s dumps, draw AS-level topology.

AS relationships, e.g. for A and B?
Exercise 7, task 4)

1. eBGP UPDATE
2. 11.0.0.0/8
3. AS_PATH: 3 1
4. R1’s inbound policy
5. set local-pref: 100
6. R2
7. 5 ms
8. 5
9. OSPF weight
10. R3
11. 5 ms
12. 5
13. R4
14. R4’s inbound policy
15. set local-pref: 200 > 100
16. 11.0.0.0/8
17. AS_PATH: 2 1
18. eBGP UPDATE
19. iBGP full-mesh
20. (not displayed)
Exercise 7, task 4)

R4 sends WITHDRAW and uses route over R1 instead.

how do R3 & R4 know the path over R1?
How many RTT is a TCP handshake?

1 RTT is enough since one can already send data with the last ACK.
Exam 2016 4.c)ii): what changes to the SACK algo?

reminder:
we do not answer any questions related to the old exams
Today’s Q&A session

- Important concepts
- Answering received questions
- Individual questions
Important information

Question deadline this Saturday 20.08.2022

Exam: 27.08.2022 – HIL G 41