Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

Communication Networks

Prof. Laurent Vanbever

Solution: Exercises 8 – Go-Back-N and DNS

8.1 Go-Back-N Warm-Up Questions

Sender and receiver keep separate windows and buffers for sent and received segments.

a) Compare how the sender and the receiver advance their respective windows.

b) Which segments does the *sender* buffer? When can segments be removed from the buffer?

c) A *receiver* typically buffers out-of-order segments. What is the advantage of such a buffer?

Solution:

a) The sender can advance it's window when an in-order acknowledgment arrives. The receiver can advance it's window with each in-order segment.

b) The sender needs to buffer all sent, but unacknowledged, segments. When the segment with the lowest sequence number is acknowledged, it can be removed.

c) An out-of-order buffer helps to reduce unnecessary retransmission of already received segments.

Cumulative ACKs acknowledge that all segments *up to* the acknowledged segment have been received.

d) Why are cumulative ACKs used? Do they help with lost data segments, lost ACKs, or both?

Solution:

d) Cumulative ACKs improve the behaviour for lost ACKs, as each ACK covers all previous ones, which can help to avoid retransmitting already received segments due to lost ACKs.

When *Fast Retransmit* is used, the sender retransmits a segment after duplicate ACKs.

e) How does Fast Retransmit improve performance?

f) Compare Fast Retransmit in the case of mild congestion (some segment losses) and heavy congestion (nearly all segments lost).

Solution:

e) With duplicate ACKs, the sender can detect isolated segment loss and can quickly resend segments, without waiting for timeouts, thus improving performance.

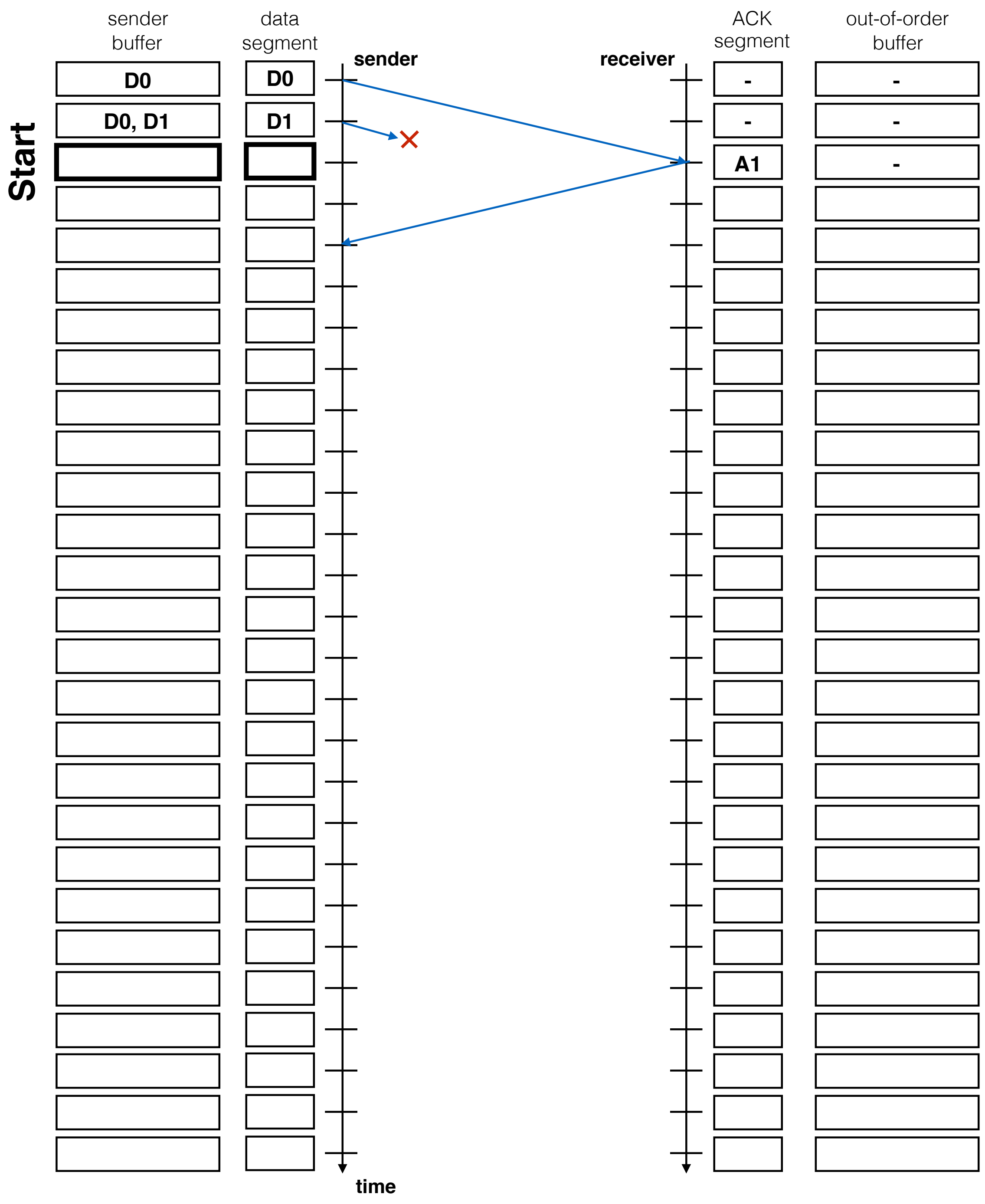
f) Fast Retransmit works best for mild congestion, as explained above. In the case of heavy congestion, not enough segments may arrive to even receive a significant number of duplicate ACKs. In this case, the sender needs to wait for the timeout.

8.2 Go-Back-N (Exam Question 2017)

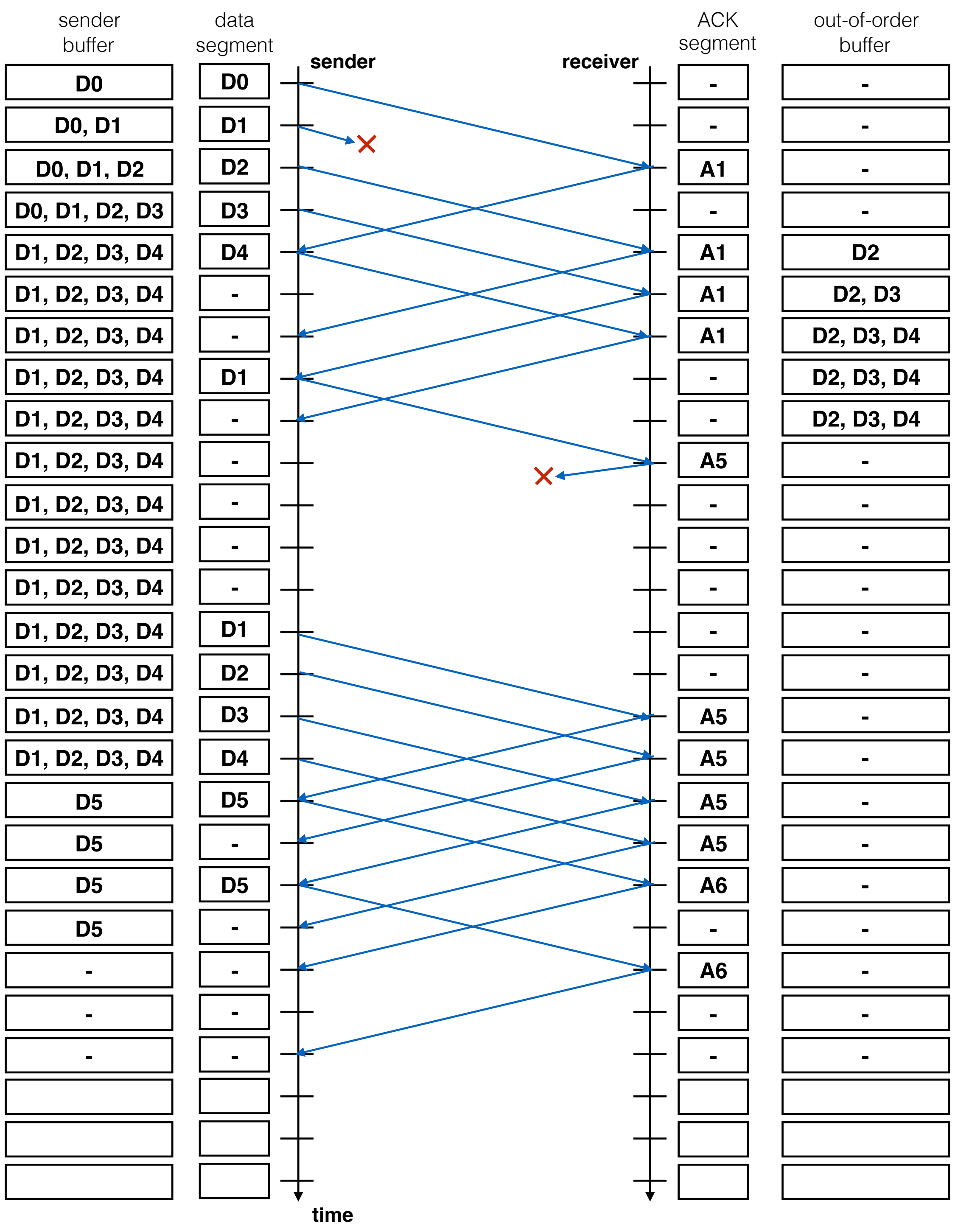
Consider a Go-Back-N (GBN) protocol with the following implementation choices for sender and receiver.

- The sender and receiver window have a size of 4;
- The receiver saves out-of-order segments in an (infinite) buffer and removes them as soon as the missing segment(s) arrive;
- The receiver uses cumulative ACKs which acknowledge all previous segments and point to the next expected data segment;
- The sender uses Fast Retransmit. After three duplicate ACKs, the sender immediately retransmits the corresponding data segment. For instance, if the sender gets the following ACKs [A1, A1, A1], it will immediately retransmit the data segment D1;
- For each tick in the diagram below, the sender can send one data segment and the receiver can send one ACK. Sender and receiver will first analyze the incoming packet and then send a data segment/ACK;
- The sender uses a retransmission timer of 5 ticks. Each time it sends a data segment or receives an ACK, the timer is reset. After a timeout, the sender retransmits all current segments in its sender buffer (in order, one segment per tick);
- A data segment or ACK needs two ticks to travel to the other end of the connection. See the given start in the diagram.

Your task: Use the diagram on the next page to draw the successful transmission of 6 data segments (D0 to D5) if the **first data segment** (D1, already indicated) **is lost** as well as **ACK A5 is lost the first time it is sent**. For each tick, indicate which data segment or ACK is transmitted (if any) as well as the content of the sender and out-of-order buffer.


sender buffer
data segment
sender
receiver
ACK segment
out-of-order buffer
Start
D0
D0
-
-
D0, D1
D1
-
-
A1
-
time

Solution:



sender buffer	data segment	ACK segment	out-of-order buffer
D0	D0	-	-
D0, D1	D1	-	-
D0, D1, D2	D2	A1	-
D0, D1, D2, D3	D3	-	-
D1, D2, D3, D4	D4	A1	D2
D1, D2, D3, D4	-	A1	D2, D3
D1, D2, D3, D4	-	A1	D2, D3, D4
D1, D2, D3, D4	D1	-	D2, D3, D4
D1, D2, D3, D4	-	-	D2, D3, D4
D1, D2, D3, D4	-	A5	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	-	-	-
D1, D2, D3, D4	D1	-	-
D1, D2, D3, D4	D2	-	-
D1, D2, D3, D4	D3	A5	-
D1, D2, D3, D4	D4	A5	-
D5	D5	A5	-
D5	-	A5	-
D5	D5	A6	-
D5	-	-	-
-	-	A6	-
-	-	-	-
-	-	-	-

8.3 Local DNS server

On Linux and Mac computers you can use the command line tool `dig` to perform DNS lookups. The corresponding tool for Windows is `nslookup`. First, perform a lookup for `nyu.edu` using your default DNS server by running the command:

```
dig nyu.edu
```

```
nslookup nyu.edu
```

- What is the IP address of the server behind `nyu.edu`?

  Solution: Note that the actual IP address can depend on the local DNS server you use. We got the following answer:

  ```
  ;; ANSWER SECTION:
  nyu.edu.   60   IN   A   216.165.47.10
  ```

Now, perform the same lookup, but use one of the DNS root servers (e.g., `a.root-servers.net`) by running:

```
dig @a.root-servers.net nyu.edu
```

```
nslookup nyu.edu a.root-servers.net
```

- Why does the answer differ compared to the one from your local DNS server?

  Solution: The request is not sent to a open DNS resolver, but to a DNS server that only provides answers about its own zone. Therefore, the root DNS server only points you to the name servers responsible for the next zone in the hierarchy, the edu zone.

  We got the following answer:

  ```
  ;; AUTHORITY SECTION:
  edu.  172800  IN  NS  a.edu-servers.net.
  edu.  172800  IN  NS  c.edu-servers.net.
  edu.  172800  IN  NS  d.edu-servers.net.
  ```

- How would you proceed with this answer to find the IP address behind `nyu.edu`?

  Solution: Now that we know which servers are responsible for the edu zone, we can continue step-by-step just like your local DNS server would. Next, we would send a request to one of the edu name servers:

  ```
  dig @a.edu-servers.net nyu.edu
  ```

  The reply points us to the name servers in charge of the zone of NYU. By sending a request to them, we finally get the IP address behind the URL `nyu.edu`.

8.4 Local vs. authoritative DNS server

Perform a DNS query for `uzh.ch` using first the authoritative DNS server (`ns1.uzh.ch`) and then your local server.

Note: When using `nslookup` on Windows, you need to specify the `-debug` flag to get the relevant information for this task. For example:

```
nslookup -debug uzh.ch
```

- Compare the `ANSWER SECTION` of the responses. Can you see differences between the answers from your local DNS server and the authoritative server? Run the query to your local server multiple times to make the differences more obvious.

  Solution: The answers differ in the time to live (TTL). While the TTL is constant in the replies from the authoritative DNS server, it varies in the replies from the local server.

- What is the reason for this difference?

  Solution: The local DNS server caches replies to requests. To ensure that it does not keep outdated information in its cache, each authoritative name server attaches a TTL to its replies. The TTL tells the local DNS server how long it can store the reply in the cache and use it to reply to requests.

- As you have seen in the lecture, DNS can be used to balance the incoming load. What are the considerations one has to make when using DNS load balancing with respect to the TTL?

  Solution: With low TTLs we can ensure that we can shift the load quickly. However, low TTLs also mean that our authoritative DNS server will get many more requests.

8.5 Multiple answers

Whenever a client (e.g., your computer) receives multiple IP addresses as answer to a DNS lookup, it picks the very first one. Only if that one does not work, it tries the next one in order.

When you run `dig yahoo.com`, you receive multiple IP addresses as an answer compared to, for example, `dig google.com`.

Can you think of a reason for providing multiple IP addresses? Run the lookup for `yahoo.com` multiple times.

Solution: (i) Resiliency: Should the first IP addresses not be reachable, you can try with the second one without having to do another DNS request.

(ii) Load of the authoritative DNS server: By giving multiple answers, you have a higher chance that one of the answers works for the entire lifetime of the reply (defined TTL value). Your DNS server will therefore get less frequent requests compared to a DNS server which only answers with one entry and a low TTL value.

(iii) Load balancing: You can use it as one way to do load balancing.