Lab 01: More reliable transmission on top of UDP

Distributed & network programming

A program that would be able to reliably copy a file from one machine to another

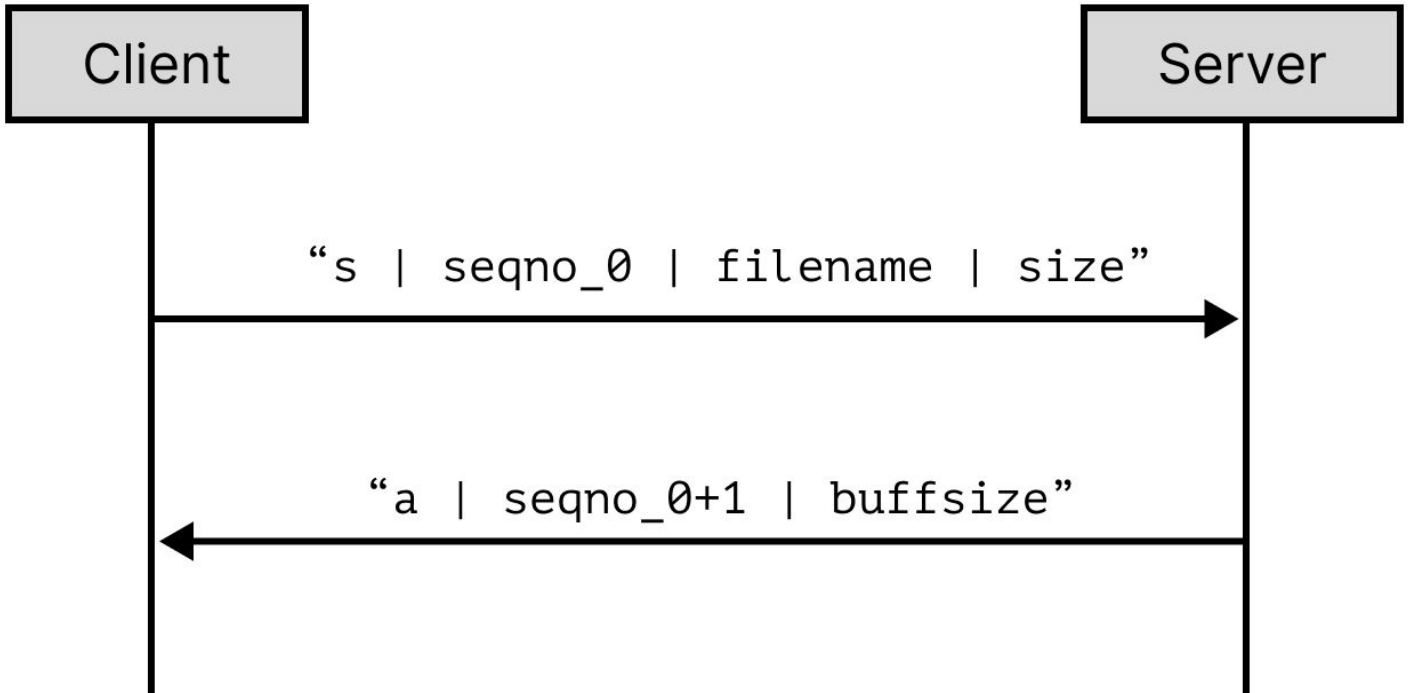
```
python3 server.py 12300


python3 client.py \
   server-hostname:12300 \
   path/to/local/file.jpg \
   filename-on-server.jpg
```

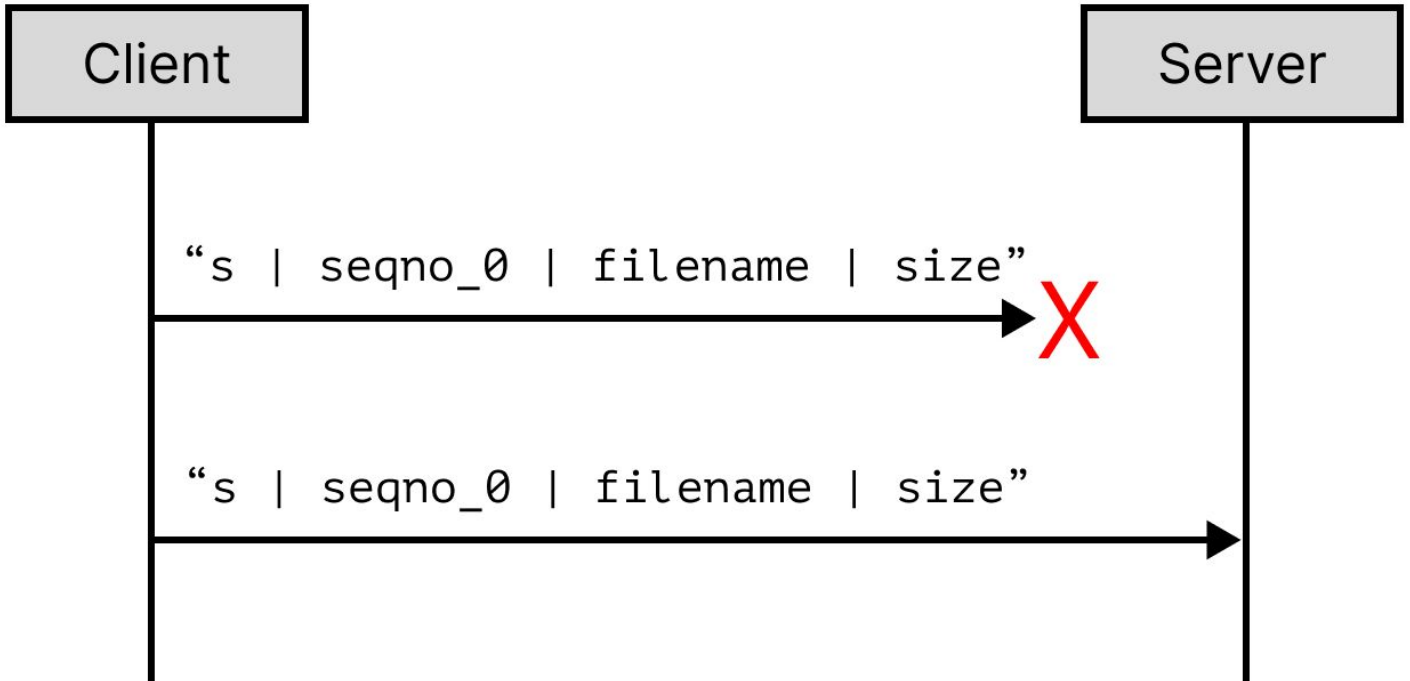
General structure of the protocol

1. Stage 1: Establish a session
2. Stage 2: Deliver chunks of the file

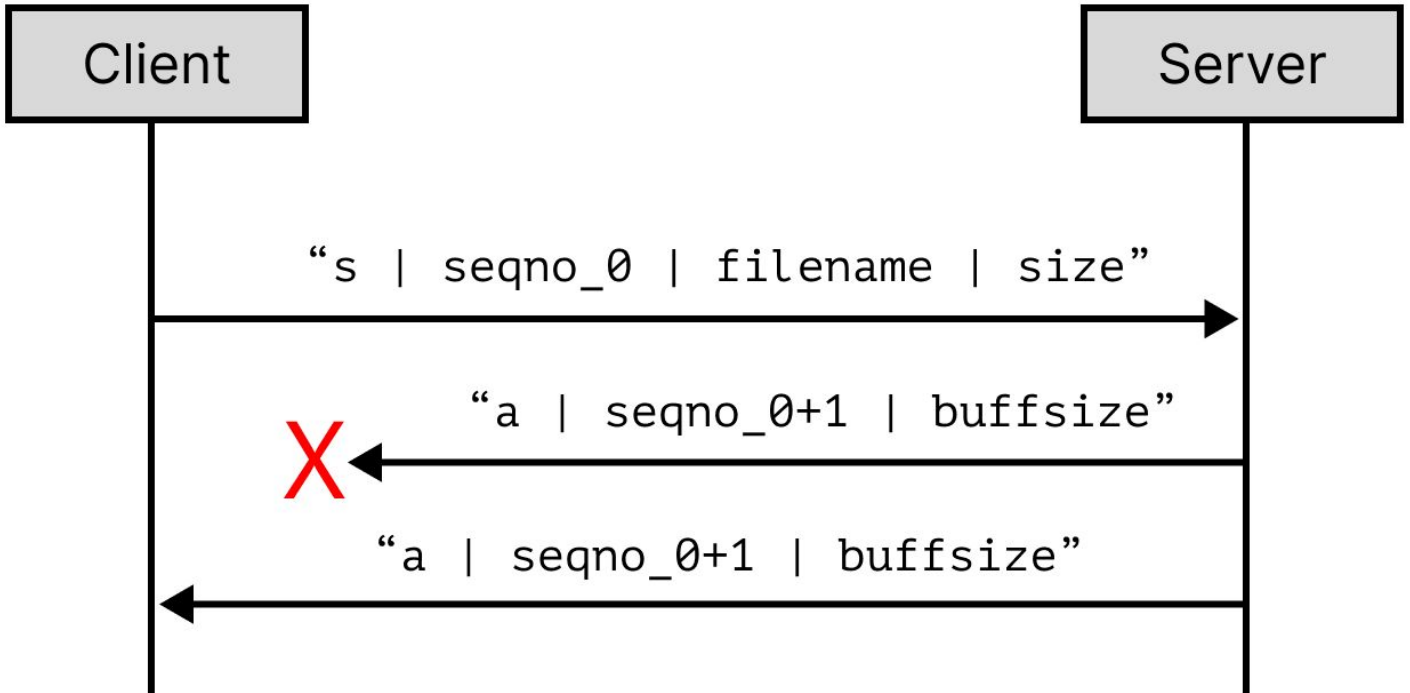
Stage 1: Establish a session



Stage 1: Establish a session, start message was lost



Stage 1: Establish a session, ack message was lost



Can we distinguish between lost message and delayed message?

Can we distinguish between lost message and delayed message?

Not really

Can we distinguish between lost message and delayed message?

Not really

What happens if we retry a message that we thought was lost, but was actually delayed?

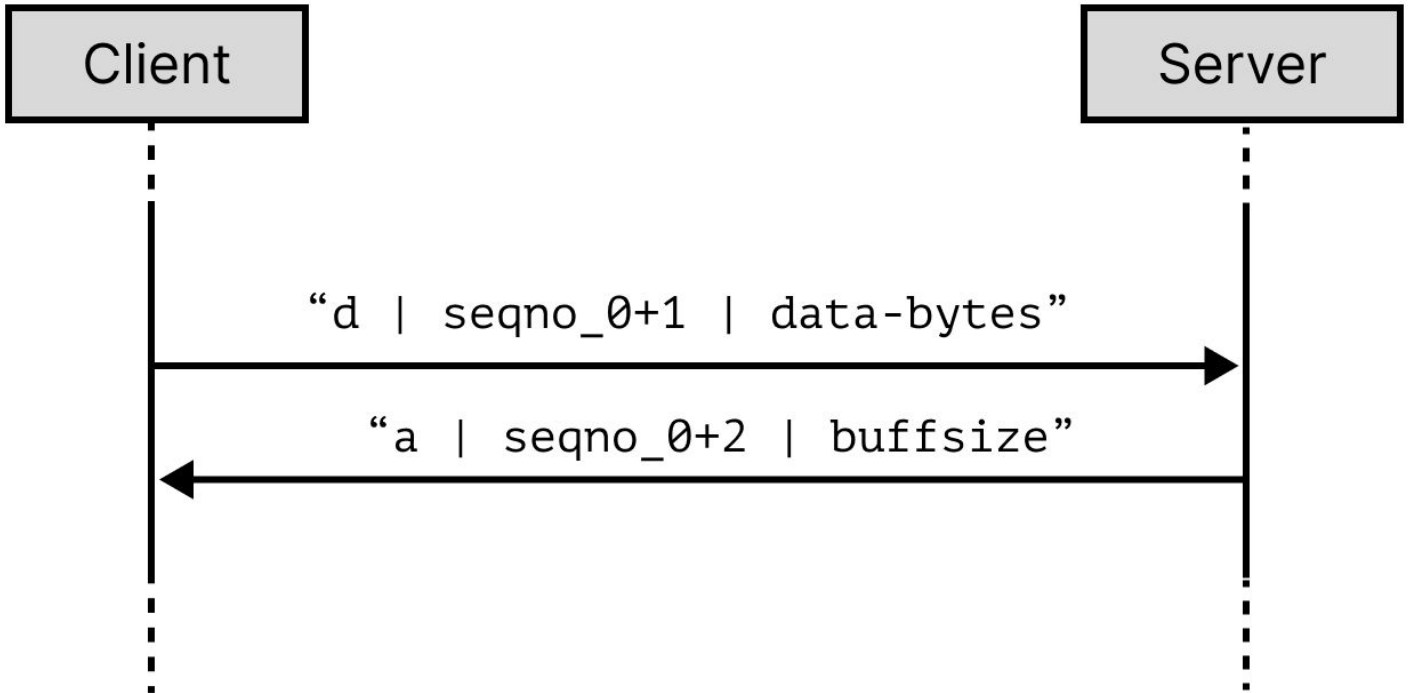
Can we distinguish between lost message and delayed message?

Not really

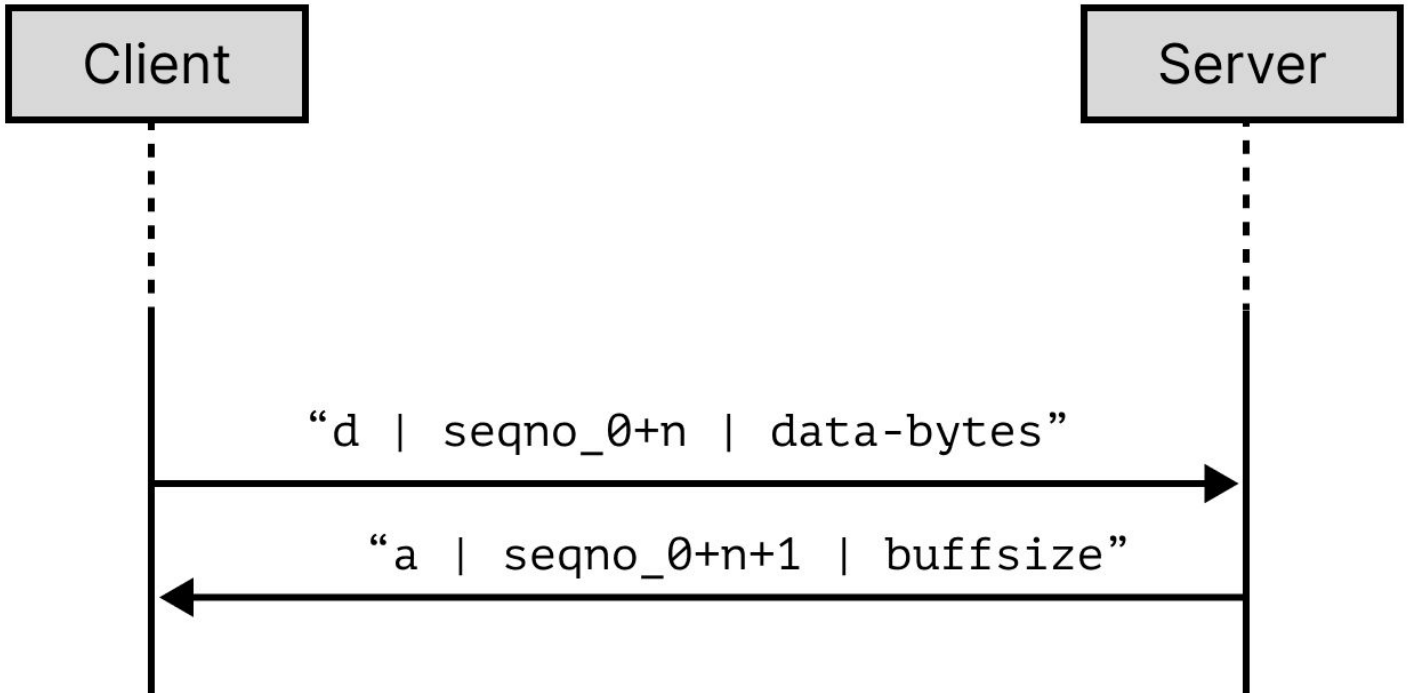
What happens if we retry a message that we thought was lost, but was actually delayed?

Duplicates

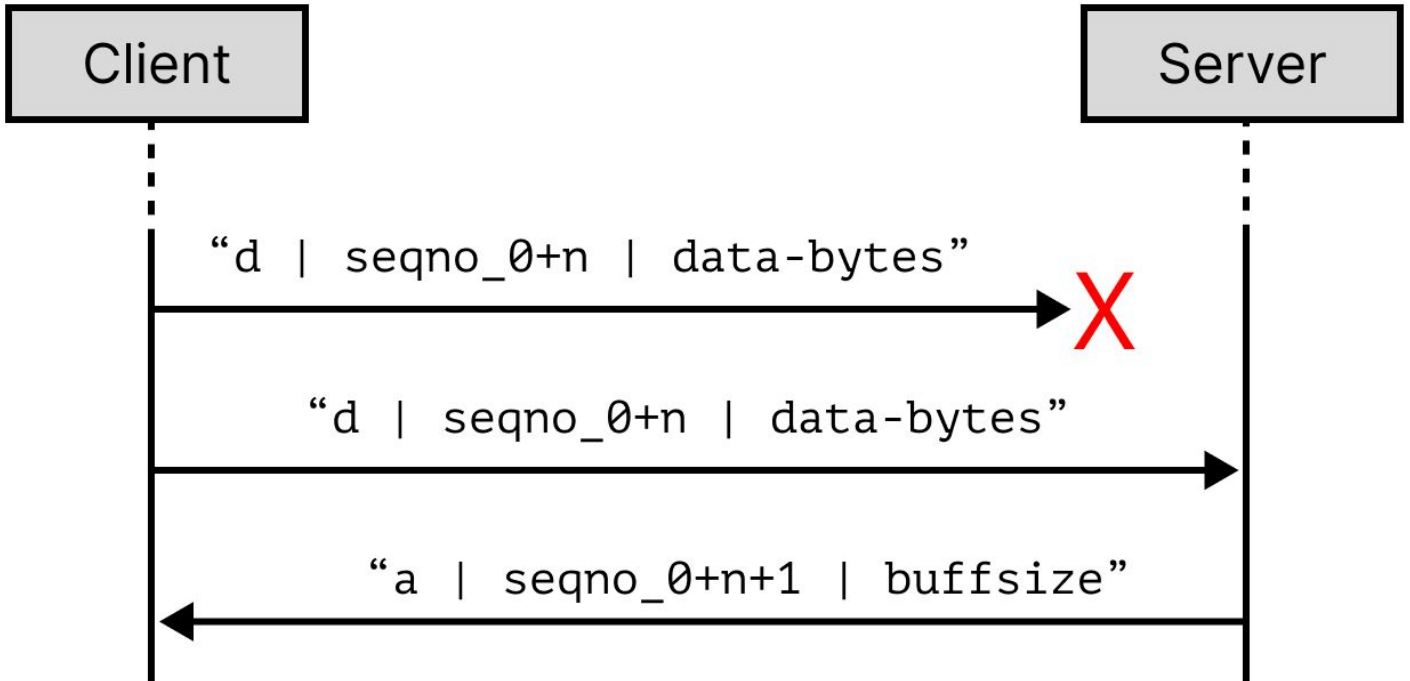
Stage 2: Deliver chunks of the file



Stage 2: Deliver chunks of the file



Stage 2: Deliver chunks of the file



Last data message probably wouldn’t be exactly buffsize, that’s alright.

Once all chunks are delivered, output file should assembled on server side with specified filename.

To test against

```
bad_server 12300


good_client \
   server-hostname:12300 \
   path/to/local/file.jpg \
   filename-on-server.jpg
```

Useful methods

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.settimeout(3000)
# bind using specific network interface to specific port
s.bind((address, port))
# use any available port
s.bind(('', 0))
msg, client_address = s.recvfrom(BUFFSIZE)
s.sendto(message, client_address)
```

Assemble and parse packets

```
packet = f"d | {seqno} | ".encode() + data_chunk
```

```
prefix, rest = packet.split(" | ".encode(), 1)
# prefix gonna be b"d"
seqno, data = rest.split(" | ".encode(), 1)
seqno = int(seqno.decode())
```