

2nd Mini-Project: File Transfer

Reliable Data Transfer

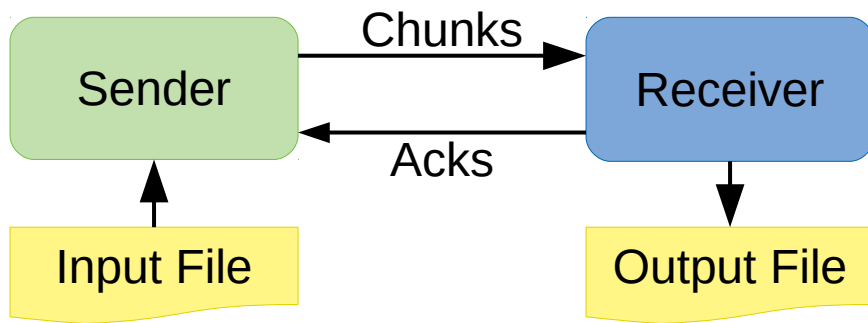
Overview

What you'll learn:

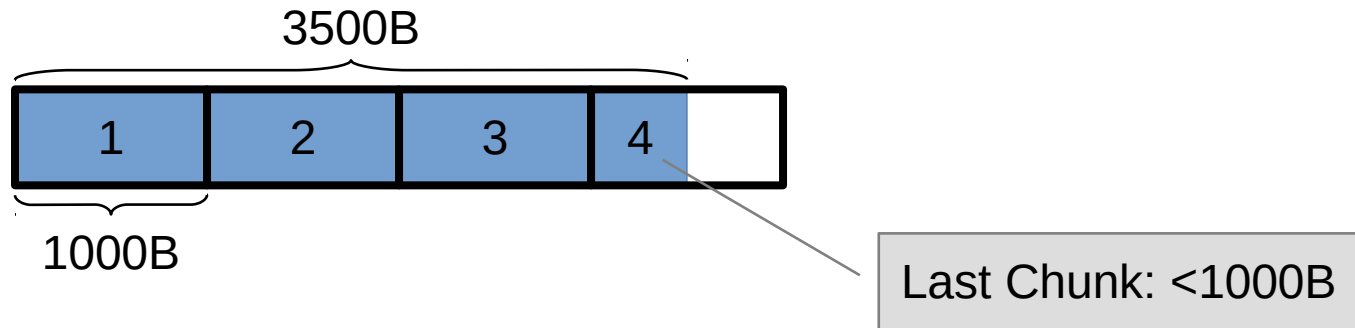
- Reliable data xfer
- UDP sockets

Create file transfer system

- **File Sender**
- **File Receiver**



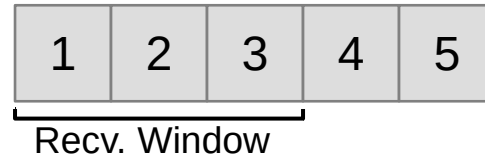
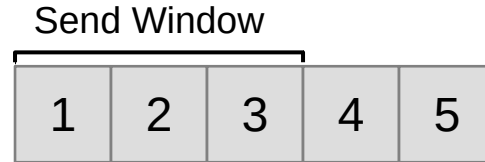
Overview: Files to Chunks



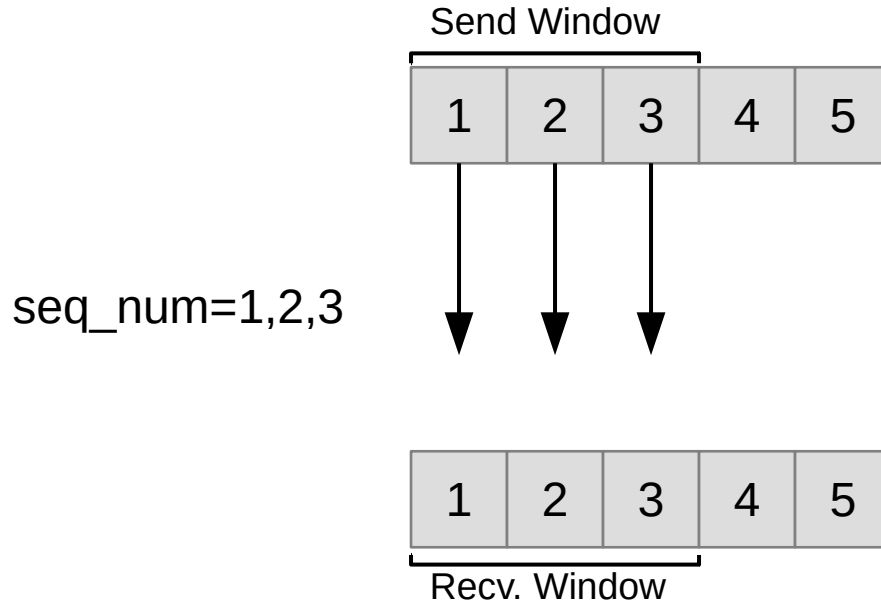
```
typedef struct __attribute__((__packed__)) data_pkt_t {  
    uint32_t seq_num;  
    char data[1000];  
} data_pkt_t;
```

```
typedef struct __attribute__((__packed__)) ack_pkt_t {  
    uint32_t seq_num;  
    uint32_t selective_acks;  
} ack_pkt_t;
```

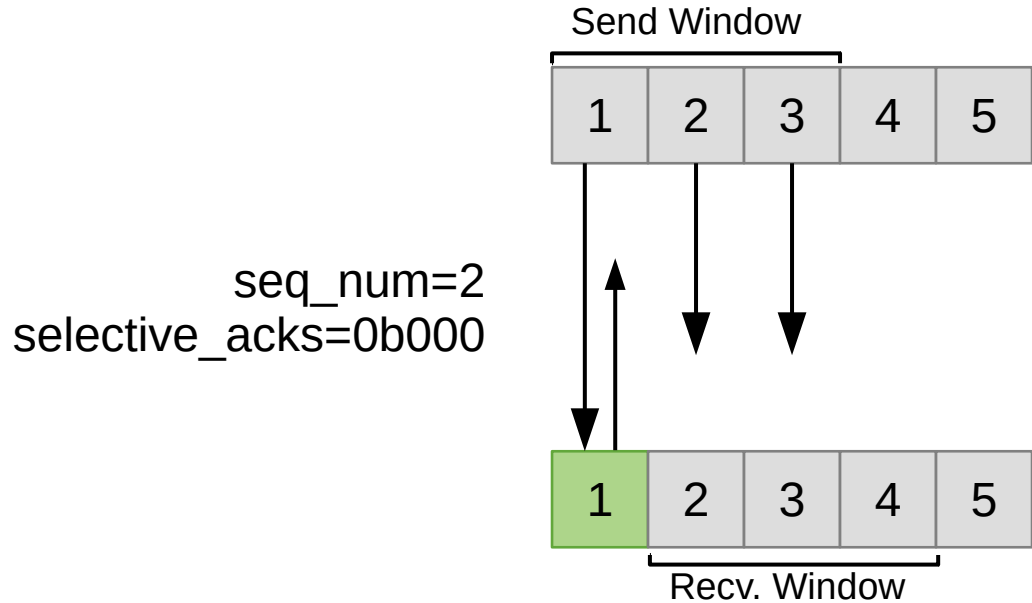
Overview: Reliable Data Transfer



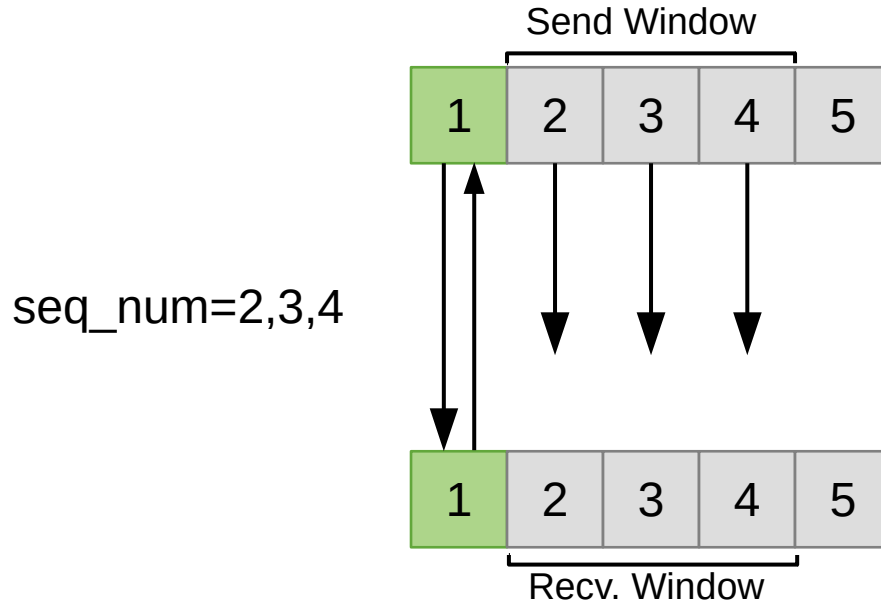
Overview: Reliable Data Transfer



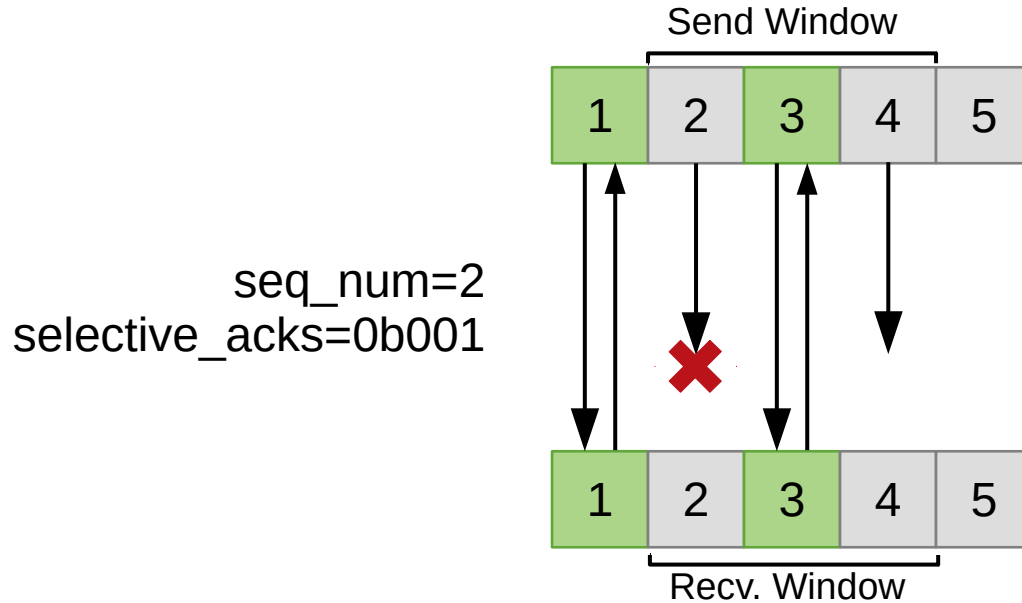
Overview: Reliable Data Transfer



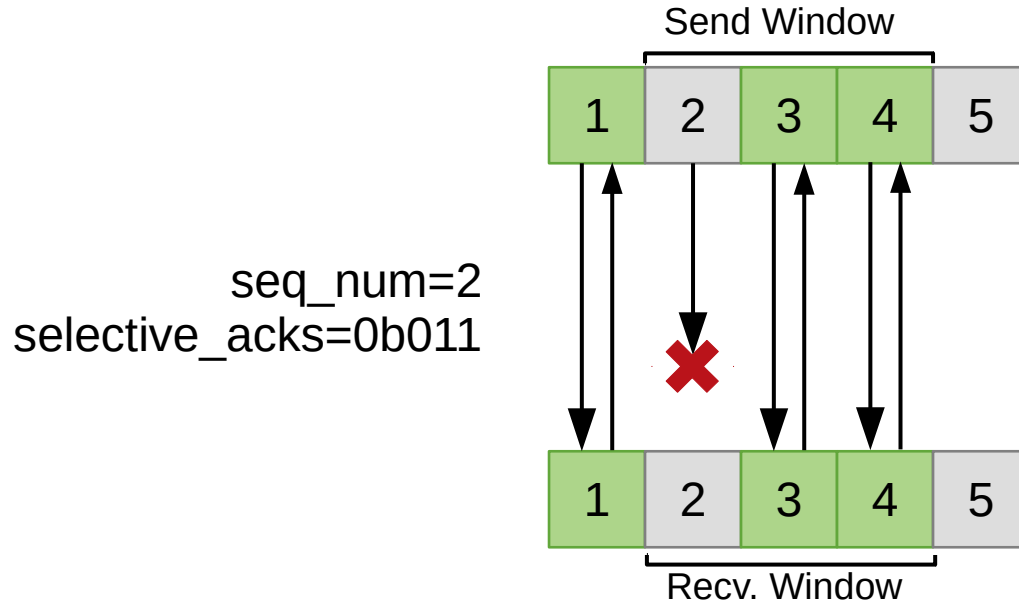
Overview: Reliable Data Transfer



Overview: Reliable Data Transfer



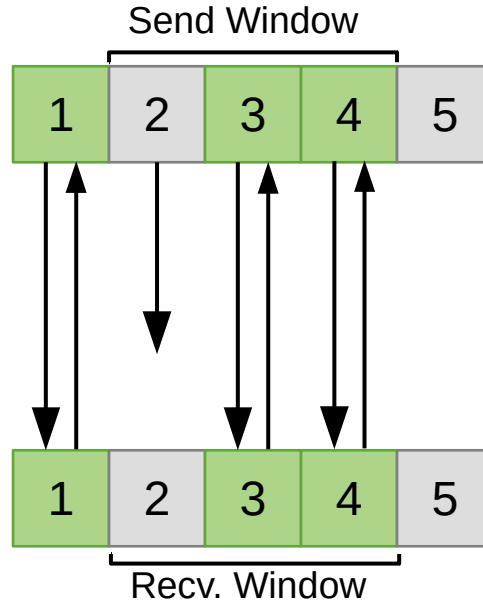
Overview: Reliable Data Transfer



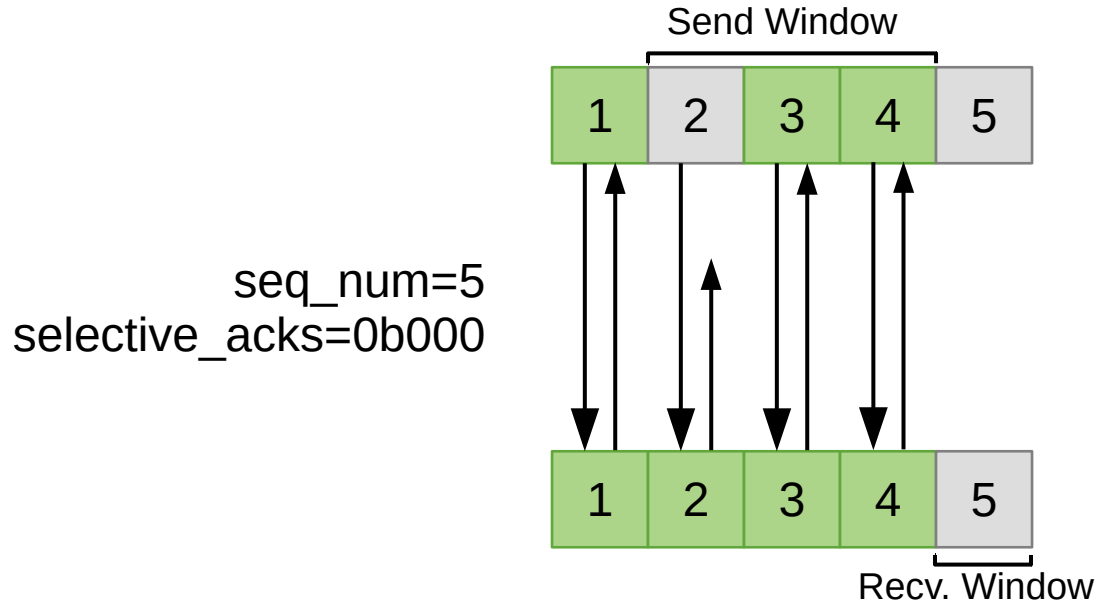
Overview: Reliable Data Transfer

*** 1s Later ***

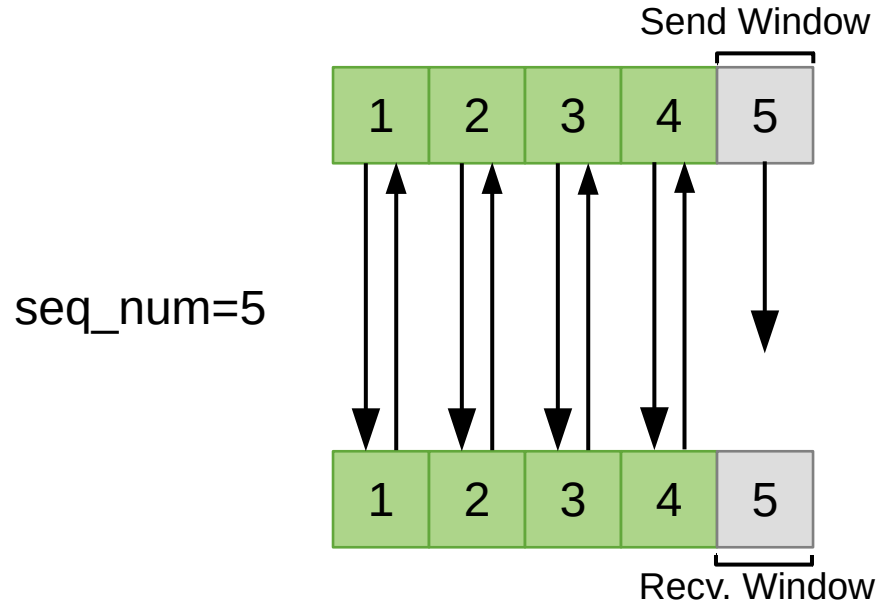
seq_num=2



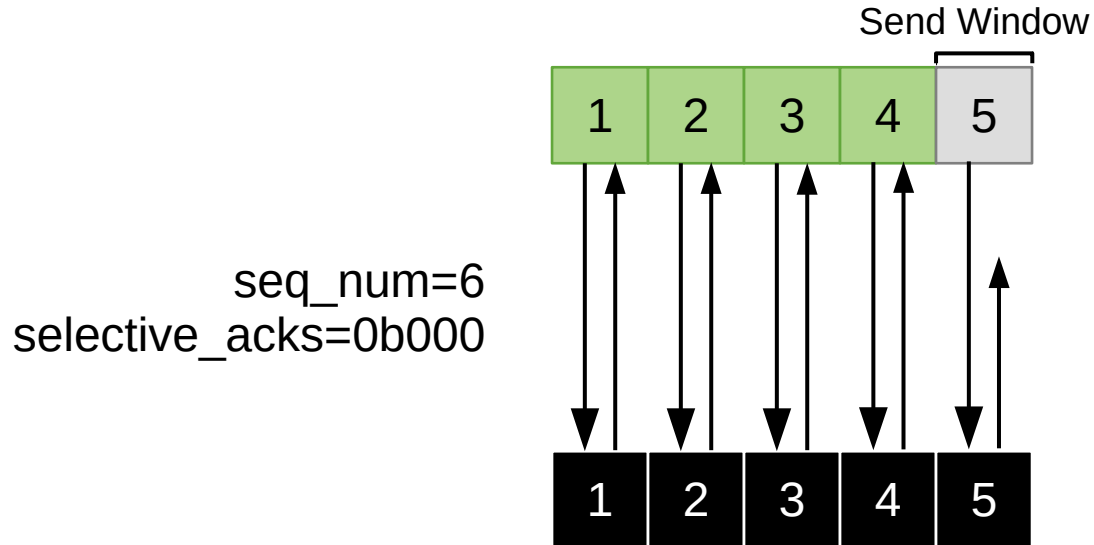
Overview: Reliable Data Transfer



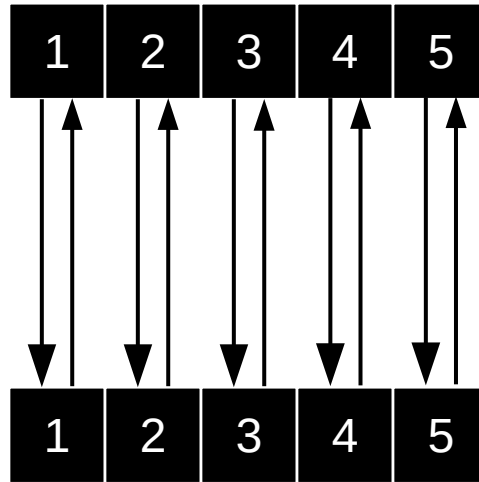
Overview: Reliable Data Transfer



Overview: Reliable Data Transfer



Overview: Reliable Data Transfer



Overview: Reliable Data Transfer

Notes

- Chunks start at 1
- Fields in network order
- Ack = recv. window
 - seq_num = base
 - selective_acks skips first (will always be 0)

RDT Modes and Window Size

- Stop-and-Wait
 - Send = 1, Receive = 1
- Go-Back-N
 - Send = N, Receive = 1
- Selective Repeat
 - Send = N, Receive = $M \leq N$

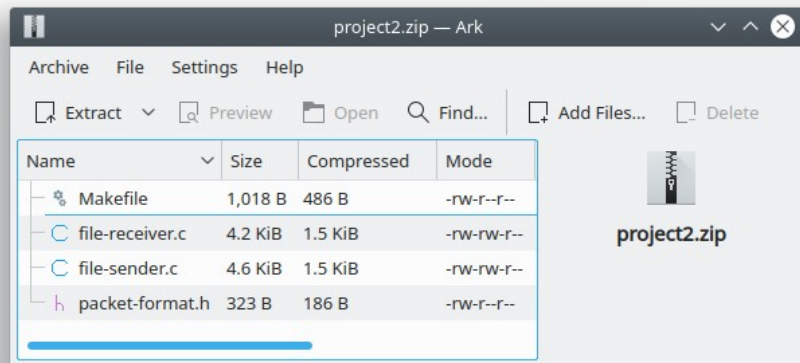
File Transfer in Action

```
project-reliable-xfer: bash — Konsole
File Edit View Bookmarks Settings Help
david@Kubuntu:~/project-reliable-xfer$ echo Test > send.dat
david@Kubuntu:~/project-reliable-xfer$ ./file-sender send.dat localhost 1234 1
Sending segment 1.
Received ACK 2 / 00000000.
david@Kubuntu:~/project-reliable-xfer$
```

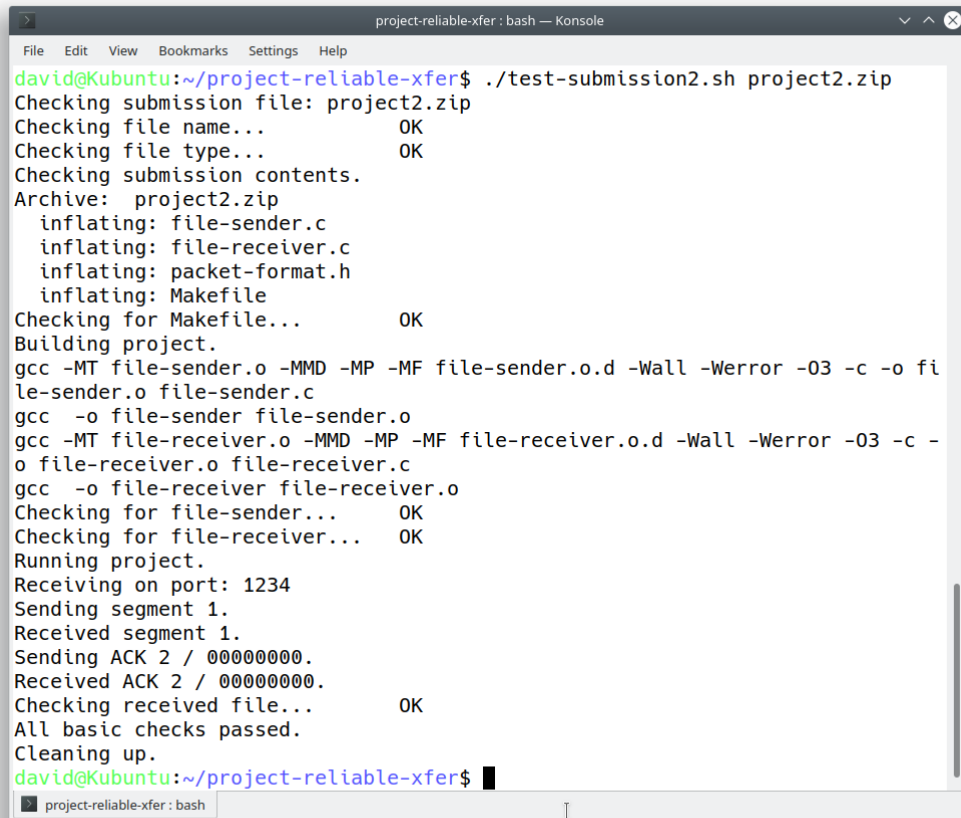
```
project-reliable-xfer: bash — Konsole <2>
File Edit View Bookmarks Settings Help
david@Kubuntu:~/project-reliable-xfer$ ./file-receiver receive.dat 1234 1
Receiving on port: 1234
Received segment 1.
Sending ACK 2 / 00000000.
david@Kubuntu:~/project-reliable-xfer$ cat receive.dat
Test
david@Kubuntu:~/project-reliable-xfer$
```


Submission

- Submit project2.zip to Fénix
 - Code
 - Makefile in base folder
 - No build artifacts
- Must build with **make**
 - Generate file-sender & file-receiver



Pre-Submission Checks

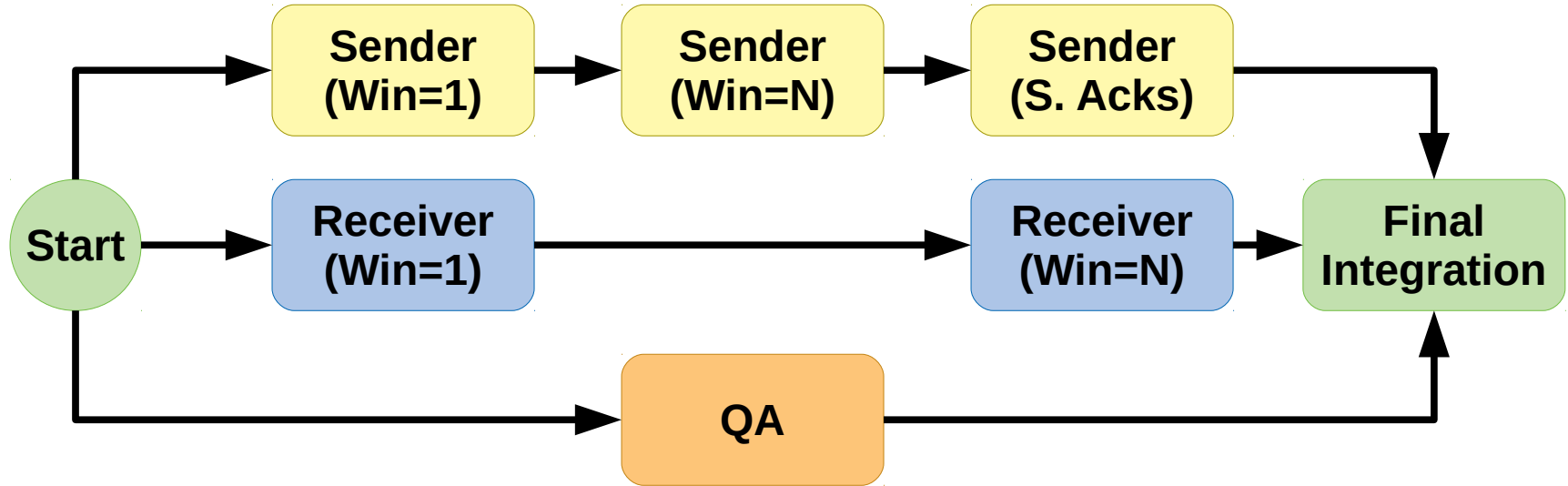
A terminal window titled "project-reliable-xfer: bash — Konsole" showing the execution of a script. The script performs various checks on "project2.zip", including file name, type, and contents. It then builds and runs a project, simulating a file transfer over a network. The output shows successful completion of all checks and the transfer process.

```
project-reliable-xfer: bash — Konsole
File Edit View Bookmarks Settings Help
david@Kubuntu:~/project-reliable-xfer$ ./test-submission2.sh project2.zip
Checking submission file: project2.zip
Checking file name... OK
Checking file type... OK
Checking submission contents.
Archive: project2.zip
  inflating: file-sender.c
  inflating: file-receiver.c
  inflating: packet-format.h
  inflating: Makefile
Checking for Makefile... OK
Building project.
gcc -MT file-sender.o -MMD -MP -MF file-sender.o.d -Wall -Werror -O3 -c -o fi
le-sender.o file-sender.c
gcc -o file-sender file-sender.o
gcc -MT file-receiver.o -MMD -MP -MF file-receiver.o.d -Wall -Werror -O3 -c -
o file-receiver.o file-receiver.c
gcc -o file-receiver file-receiver.o
Checking for file-sender... OK
Checking for file-receiver... OK
Running project.
Receiving on port: 1234
Sending segment 1.
Received segment 1.
Sending ACK 2 / 00000000.
Received ACK 2 / 00000000.
Checking received file... OK
All basic checks passed.
Cleaning up.
david@Kubuntu:~/project-reliable-xfer$
```

Advice: Debugging

- Standard output/error will be ignored during grading
 - `printf(...)`
- Debug tools also available
 - **log-packets.c**: Packet logging & fault injection
 - **generate-msc.sh**: Log analysis & MSC generation
- Testing
 - Look into **test-submission2.sh / run.sh** for ideas.

Advice: Task Breakdown



Advice: MSC Generation

```
gcc -shared -fPIC -Wall -Werror -O3 \  
    -o log-packets.so log-packets.c -ldl
```

```
LD_PRELOAD = "./log-packets.so" \  
    SEND_DELAY="500" \  
    DROP_PATTERN="01" \  
    PACKET_LOG="sender.log" \  
./file-sender ...
```

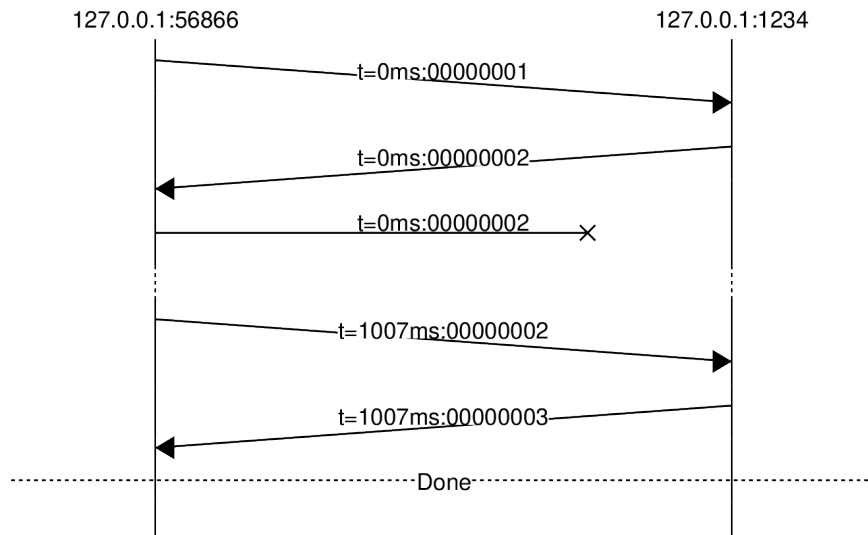
```
./generate-msc.sh msc.eps sender.log receiver.log
```

See: run.sh

Advice: MSCs

Stop-and-Wait

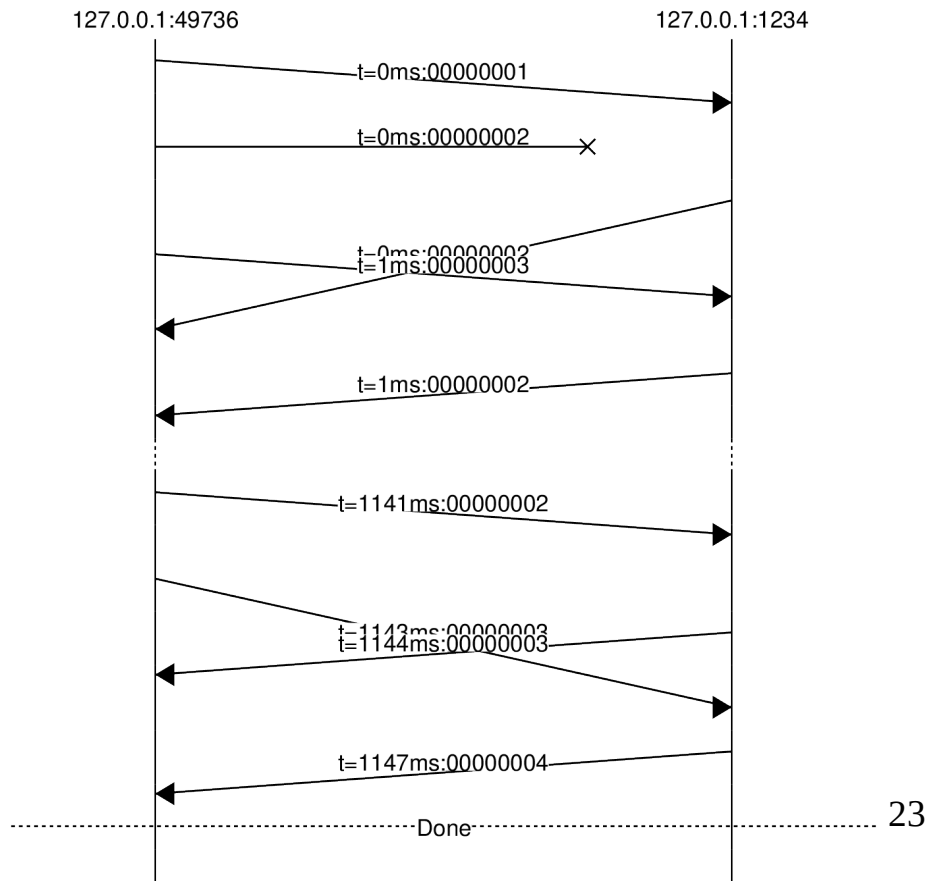
- 2 Chunks
- Sender
 - DROP_PATTERN="01"
 - Send Window = 1
- Receiver
 - DROP_PATTERN=""
 - Receive Window = 1



Advice: MSCs

Go-Back-N

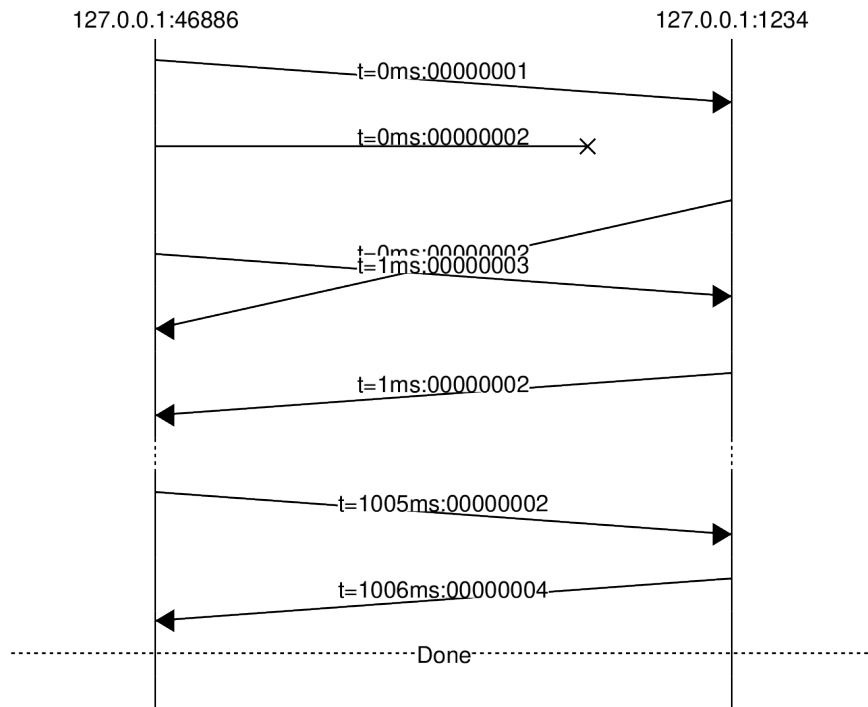
- 3 Chunks
- Sender
 - DROP_PATTERN="01"
 - Send Window = 3
- Receiver
 - DROP_PATTERN=""
 - Receive Window = 1



Advice: MSCs

Selective-Repeat

- 3 Chunks
- Sender
 - DROP_PATTERN="01"
 - Send Window = 3
- Receiver
 - DROP_PATTERN=""
 - Receive Window = 3



Advice: MSCs

Improv

- How Many Chunks?
- Sender
 - DROP_PATTERN="?"
 - Send Window = ?
- Receiver
 - DROP_PATTERN="?"
 - Receive Window = ?

