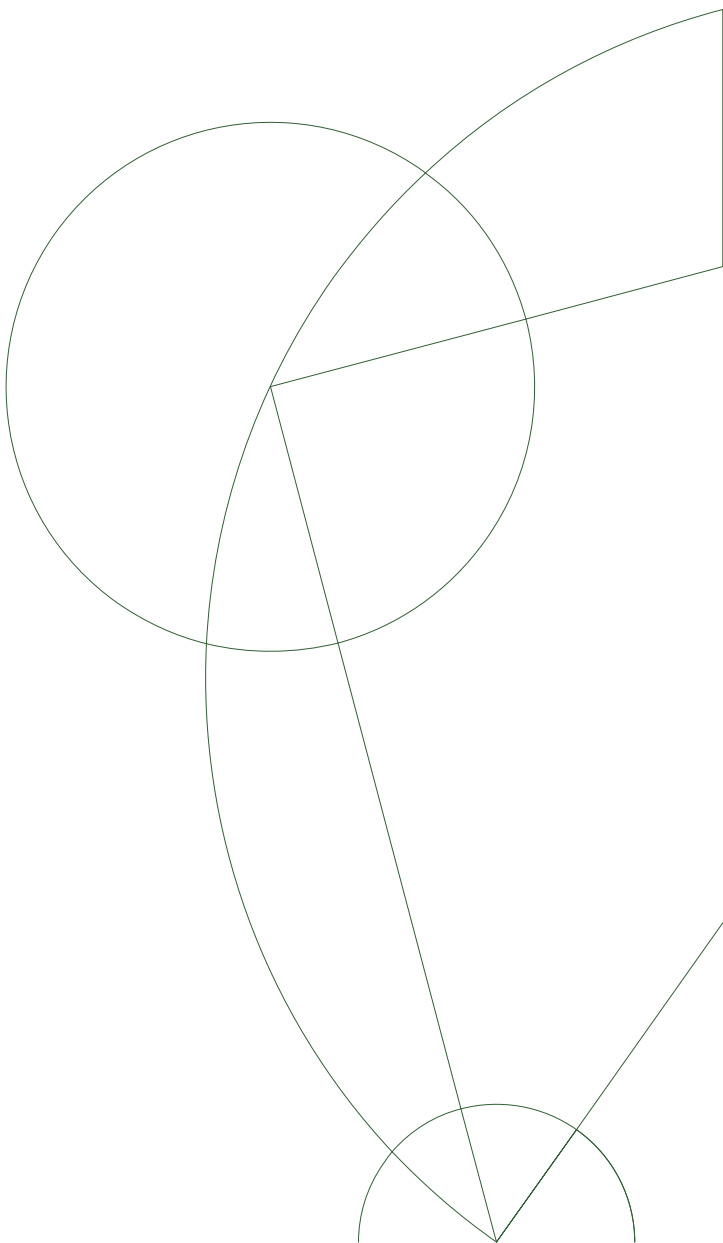
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Assignment 7: Computer Networking-2

CompSys 2017
Department of Computer Science
University of Copenhagen

Peter, Tobias & Xueying

7. January 2018



1 Theoretical part

1.1 Transport protocols

1.1.1 TCP reliability and utilization

Part 1:

It is necessary to have the three way handshake in TCP, if one wants to suffice with less (or none), UDP will be an option. The first handshake happens when the client picks an initial sequence number (ISN) and sends it to the server as a connection request. Without this step, the connection will not be establish. The server picks an ISN for itself as well. After receiving and synchronizing the client's ISN, the server sends its ISN to the client and acknowledges that it should be expecting client's ISN+1 for next. In the third handshake, the client acknowledges server's ISN. These handshakes are all needed, since they ensure the transferred data are trackable by theirs sequence number and increase the reliability of the communication.

Part 2:

TCP is a connection-oriented protocol which has high reliability for every transmission. When a TCP connection is established, two parties can send data to each other simultaneously at high speed since TCP provides the coordination.

1.1.2 Reliability vs overhead

Part 1:

Beside the special three-way-handshake, TCP connection adds both parties' ISN and ACK number, urgent pointer, windows, flags, options and some other information along with the transferred data, comparing to UDP sends only the segments, length, checksum and both parties' port. Adding the mentioned info, the reliability is increased.

Part 2:

TCP connections are reliable because its error-checking mechanism:

- Detect bit errors by using checksum
- Detect missing data by adding sequence number
- Recover from lost data by preforming re-transmission

This mechanism guarantees reliability, ordering and data integrity. Without it initializing sequence number, all the data packets are independent of each other, therefore they have no inherent order and they are untraceable if any of them are damaged or lost. TCP marks all packets with sequence numbers, packets will be checked if they are received both intact and in the same order in which they were sent.

Packets are not always delivered. For instance, when the recipient is offline, the program will send an error message that the connection is lost.

1.2 TCP: Principles and practice

1.2.1 TCP headers

Part 1:

The purpose of the RST-bit is to stop the TCP-connection instantly, and it varies from the FIN-bit since the RST-bit does not need an acknowledgement before the session terminates. A RST package may be send, if the connection is suddenly being interrupted. A half-open connection may happen if a one device losses/stops the connection, without the acknowledgement from the other device.

The sequence and acknowledgement number headers refers to how much data each side has been sending and received. The relation between a server and a client's sequence and acknowledgement numbers is quite important, since the sequence number is included on every transmitted packed and acknowledged by the opposite host as the acknowledgement number to verify that the data was transported successfully. In other words the relation between a server and a client acknowledgement and sequence number is that they used to verify that data has been correctly send and received.

The purpose of the receive window in TCP, is flow control, a positive window size indicates that a receiver is willing to receive a certain amount of data.

If the window size of the receiver is 0. Then no information will be received. The sender will receive a TCP Zero Window, indicating that the receiver's window size is 0.

Part 2:

Client received the SYN-ACK-packet means: 1.connection is established from the client side; 2.the server has received and synchronized client's ISN, it is ready for the next packet from client. Hence the client can initiate its transfer of data along its ACK packet after having received the SYN-ACK-packet, without interfering the transfer order.

1.2.2 Flow and Congestion Control

Part 1:

The congestion control of TCP is an end-to-end congestion control and not a network assisted congestion control.

Part 2:

TCP determines the transmission rate by having a congestion window, which is maintained by the sender.

Part 3:

The way the TCP sender calculates the value of the congestion window is by making sure that the sender cannot exceed the minimum of cwnd and rwnd:

$$LastByteSent - LastByteAcked \leq min\{cwnd, rwnd\}$$

Part 4:

The sender needs to implement the fast re-transmit algorithm, to allow for fast re-transmit and the receiver will have to implement a fast recovery algorithm to support the re-transmit. The sender should receive 3 ACKS before using fast re-transmit.

2 Programming Part

2.1 Compiling and running the code

The chat client library and test programs may be compiled and tested via a UNIX-like command line in the following way:

1. Unzip `src.zip` and navigate to `.../src/`.
2. Compile the library and programs by running this command:
   `make`
3. Run the `name_server` program in one command line using this command:
   `./name_server`
4. Run the `peer` program in another command line using this command:
   `./peer`

The user will now be able to interact with the `name_server` via the `peer`-client using the following commands:

- `/login <nick> <password> <IP> <Port>` : the user must first log into the server.
- `/exit` : the user may exit the `peer`-program if they are not logged in to the `name_server`.
- `/logout` : the user may logout, if they are logged into the server.
- `/lookup` : the user may request a list of all other online users, if the requesting user is logged into the server.
- `/lookup <nick>` : the user may request the status and information of a specific user via their *nick*, if the requesting user is logged into the server.

2.2 Testing

2.3 Implementation