Jacobs University Bremen

Lab Class 2

Automation

Course CO23-320203

CLAMV account and Simulink

- For all CLAMV account access problems contact Achim Gelessus a.gelessus@jacobs-university.de
- You can run Matlab / Simulink directly from your machine by connecting to CLAMV:
  - In terminal: `ssh <your_username>@10.70.2.36 -X`
  - run all commands directly from command line, ex. `matlab &`

(it works on a linux or a virtual machine. A separate x-window server must be installed on windows)

- You can also install the Matlab suite from the following sources:
- Windows: enter the following address in the file manager \\clabfs2.clamv.jacobs-university.de\windows
- Linux:

```
sudo mount 10.72.1.12:/disk/software/LinuxInst /mnt && sudo
/mnt/Matlab/Matlab_R2017b/install
```

Cyclic Redundancy Check - recap

- Calculation of a verification value by applying a specific mathematical function on the message body
  - This value is added to and travels together with the data packet
  - At the destination, the communication peer verifies if he gets the same values in its calculations
- CRC is just one of the commonly used methods
- We have seen it in Modbus (with a generation polynomial 0xA001)

Class exercise 1

- Get the code from

  https://marine.jacobs-university.de/inf/auto2019/Lab02/AutomationLab02Code.zip

- The code for this exercise is in the subfolder Ex1
- Compile the program with the following command:

```
gcc check_crc.c crc.c -std=c99 -o check_crc
```

- Run the executable from command line:

```
./check_crc
```

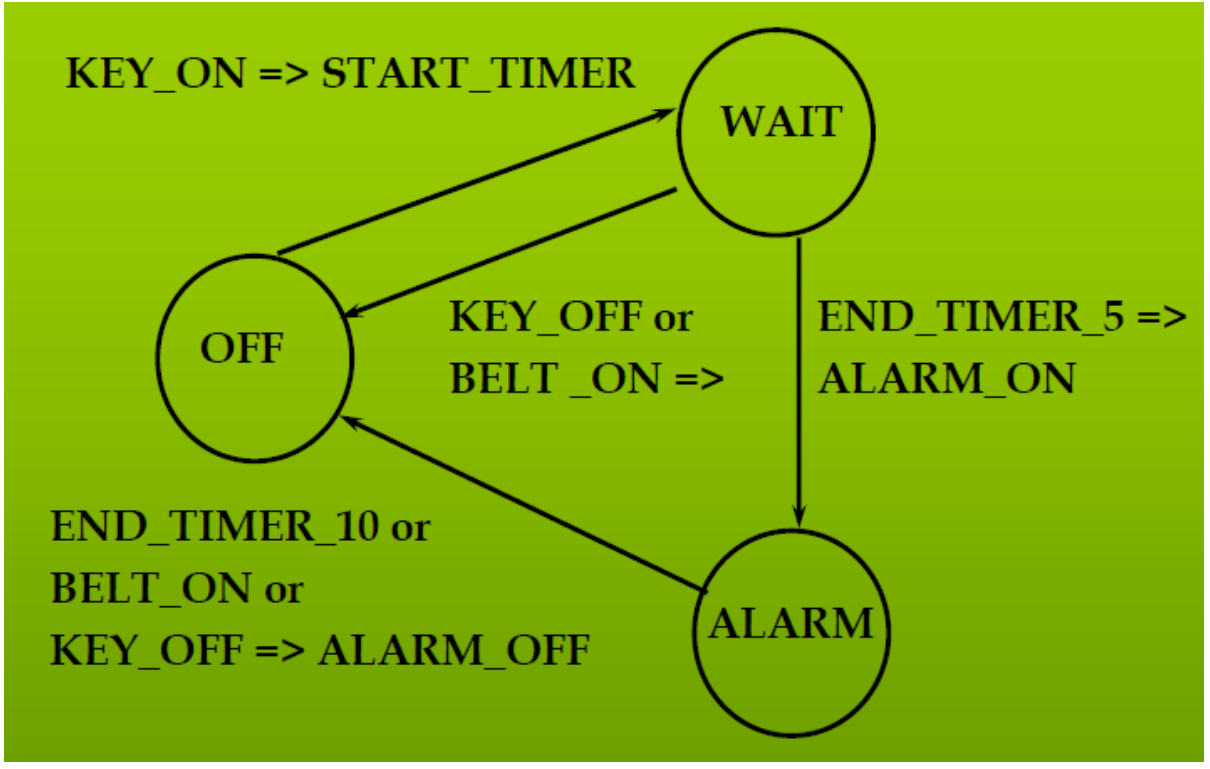
- It runs a simple CRC check on the packet body (first two bytes) and compares it to the CRC values stored in the message (last two bytes)
- Objective: by modifying at least one of the message bytes
  - Make messages #0 and #2 pass the failed CRC
  - Make messages #1 and #3 fail the CRC test

Class exercise 1

- Submit your source check_crc.c files to moodle by Friday 17 May

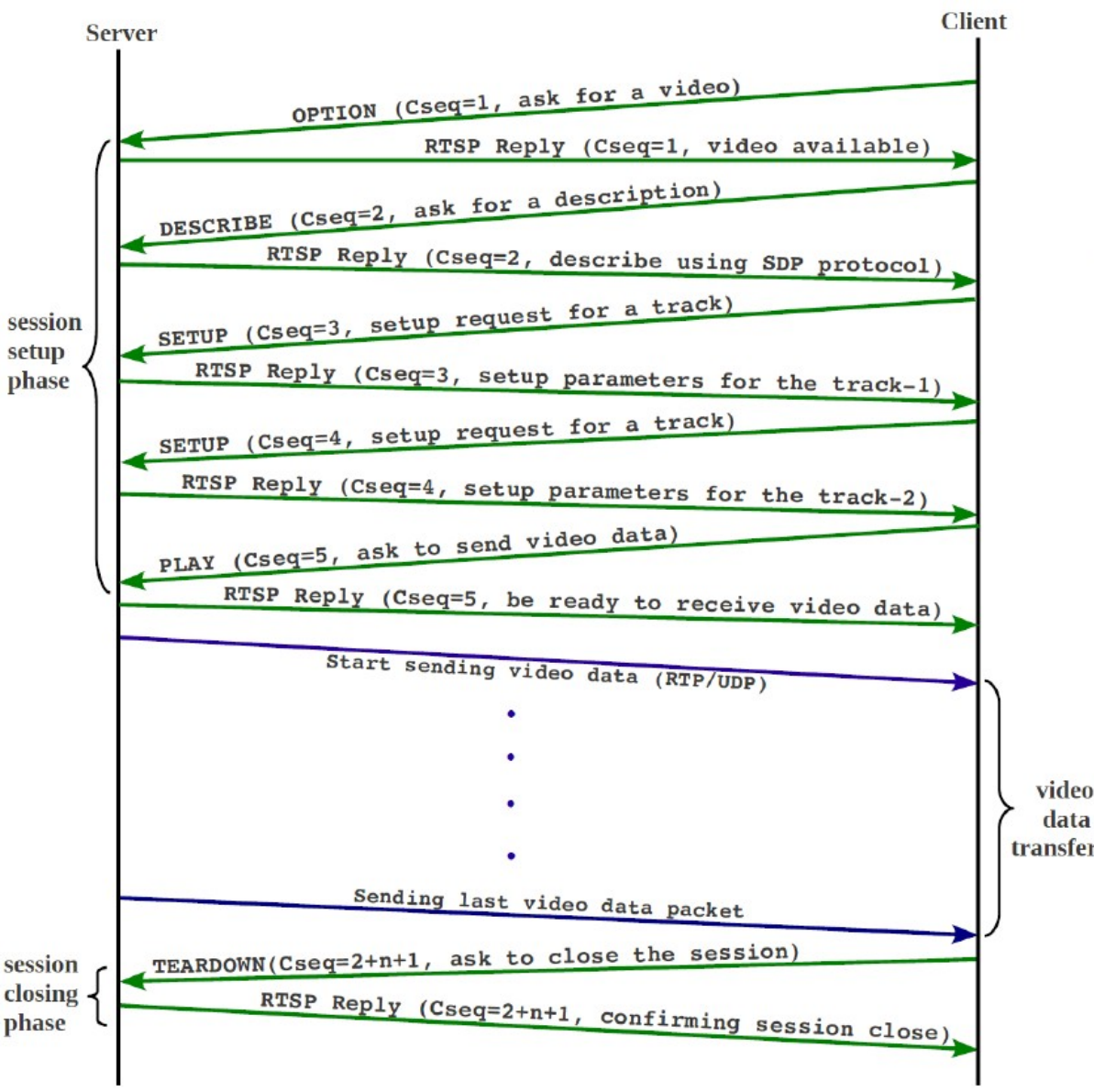
  (address: https://moodle.jacobs-university.de)
- With technical problems, please contact the course TA Daniel Nieto Rodriguez <d.nietorodriguez@jacobs-university.de>
- In case of trouble with moodle send the file to s.krupinski@jacobs-university.de **before the deadline**

State machines - recap



Protocol as a state machine

- Automation protocols, discussed in the last class are defined by packet formats, data types, etc. but also by the procedure of data exchange
- The latter can be expressed or implemented as a state machine



Class exercise 2

- The code for this exercise is in the subfolder Ex2
- Compile the server with the following command:

```
gcc server.c crc.c  -std=c99 -o server
```

- Run the executable from command line:

```
./server
```

- Compile the server with the following command:

```
gcc client.c crc.c  -std=c99 -o client
```

- Run the executable from command line in another window:

```
./client
```

Class exercise 2

- Exercise back story:
  - The server code is similar to what could be running on a smart sensor or an embedded device
  - The client code is a code for interrogating slave nodes that could realistically be found on an data acquisition PC
- Question
  - Where do I need to put the CRC function in the client program in order to verify the incoming messages for integrity?
- Task/Submission: draw the state machine diagram representing what is happening on the client side and submit the document on moodle (same conditions as exercise 1)