System Programming

C programming manual: lab 8

2018 - 2019

Bachelor Electronics/ICT

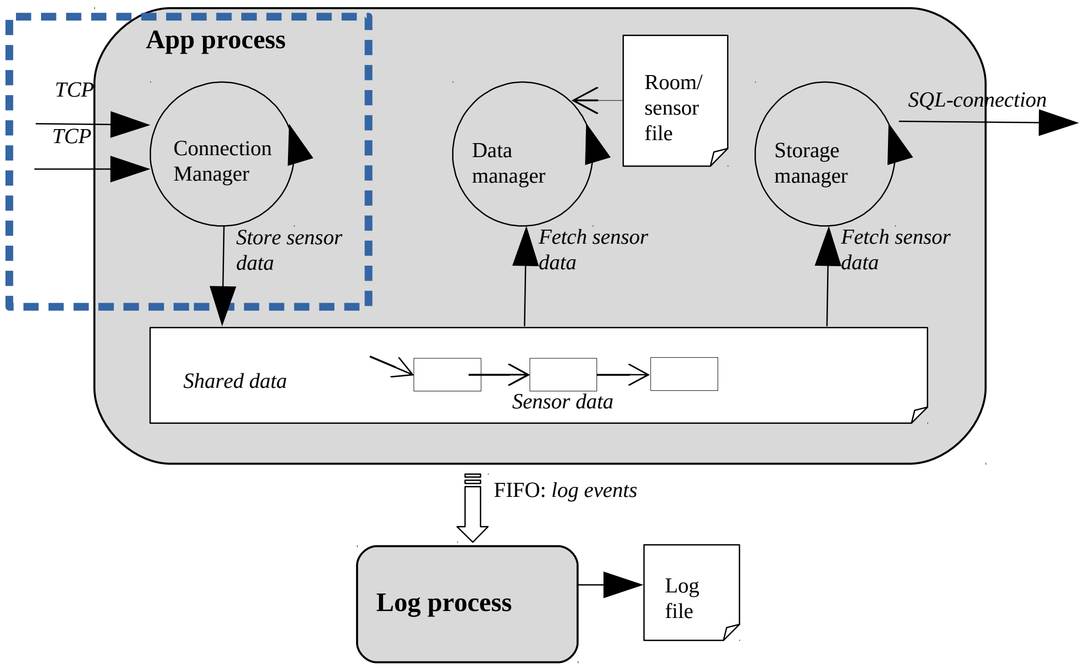
Course coördinator: Luc Vandeurzen
Lab coaches: Stef Desmet
Tim Stas
Jeroen Van Aken
Luc Vandeurzen

Last update: October 23, 2018

C programming

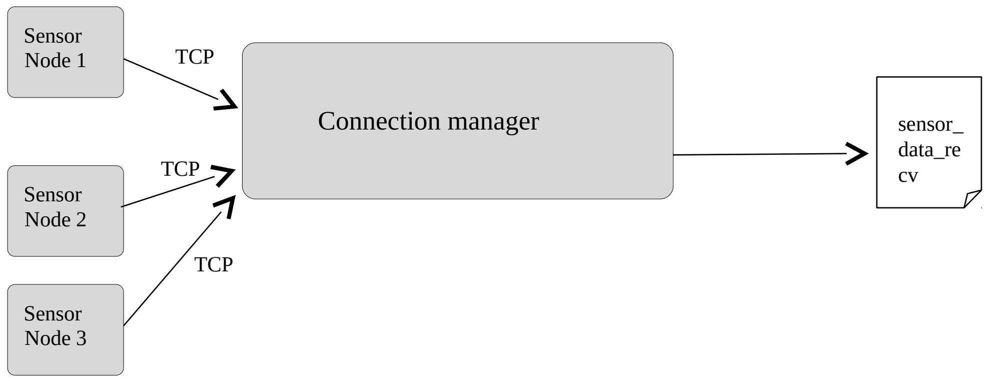
Lab targets:programming with processes and TCP/IP socket communication

For your information
The picture below visually sketches the final assignment of this course. The relationship of this lab to the final assignment is indicated by the dashed blue line.



Exercise 1: *Processes and network communication*

I



Implement a connection manager that listens for incoming sensor data from sensor nodes and writes all processed data to a **file called 'sensor data_recv'**. TCP sockets are used for the communication between the connection manager and the sensor nodes. We refer to the course "Data communication and computer networks" for more information on TCP and socket calls. For testing purposes, the local loopback network 127.0.0.1 is preferred.
The connection manager may **not** assume a maximum amount of sensors at start up. In fact, the number of sensors connecting to the sensor gateway is not constant and changes over time. A sensor node opens a TCP connection to the connection manager and this TCP connection remains open during the entire lifetime of the sensor node. All data (packet format as defined in lab 6) is sent over this TCP connection. The code for a sensor node is given, together with example code of a simple server. In the given server code, however, only one client connection is supported at the same time. A sensor node is started up with the sensor node ID, the sleep time (in seconds) between to measurements, the IP-address and port number of the connection manager as command line arguments, e.g.: ./sensor 101 60 127.0.0.1 1234 will start a new sensor node that measures every 60 seconds the temperature and sends the result tagged with sensor ID 101 and a timestamp to the connection manager listening at 127.0.0.1:1234.
The connection manager should be able to handle multiple TCP connections at the same time. The connection manager is started up as a terminal application with a command-line argument to define the port number on which it has to listen for incoming connections, e.g.:

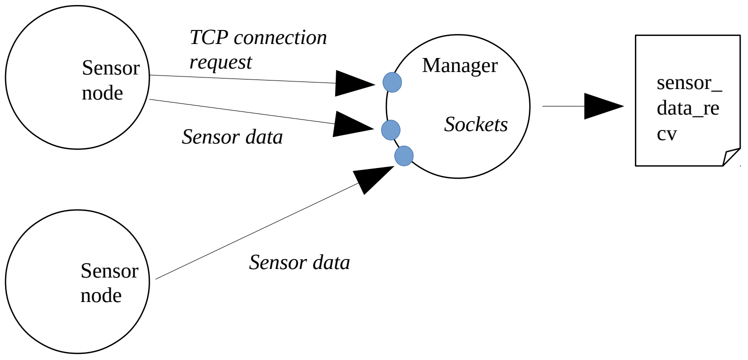
```
> ./a.out 1234
```

starts a connection manager listening on port 1234.

A single process sensor gateway

PLEASE UPLOAD YOUR SOLUTION OF THIS EXERCISE AS A ***ZIP FILE*** *ON https://labtools.groept.be .*

YOU ARE STRONGLY ENCOURAGED TO TEST AND DEBUG YOUR SOLUTION UNTIL THE CRITERIA FOR THIS EXERCISE AS DESCRIBED ON https://labtools.groept.be/ ARE SATISFIED!



In this solution only one process is used to handle multiple TCP sockets. Each time a new TCP connection request is received, the accept() socket call will return a new socket to handle this TCP connection. That means that the main process has to listen to a set of sockets (one for each connected sensor node and one to listen for new connection requests) at the same time. Remember that socket operations (send, receive, accept, …) are by default working in blocking mode such that a problem arises when the sensor connection manager is blocked on one socket while at the same time there is incoming data on another one. Multiplexing I/O using poll() or select() is a classical solution to solve this kind of problem. Recall that a maximum amount of sensor nodes may **not** be assumed and that sensor nodes can open or close connections at any time. The connection manager keeps track of a 'last active' timestamp for every socket connection to a sensor node. If no activity was monitored on such a socket after TIMEOUT time, the socket is closed and removed from the data structure maintaining all sockets. If no socket connections to sensor nodes exist anymore (because they are closed or timed out), the connection manager waits for another TIMEOUT time on new connections from sensor nodes before finally terminating. It should be possible to define TIMEOUT at compile-time with the preprocessor directive TIMEOUT=<some_value> where <some_value> is expressed in seconds. Compilation should fail and an appropriate error message should be printed when this preprocessor directive is not set at compile-time. Also remember that the connection manager is started up a command-line argument to define the port number on which it has to listen for incoming connections.