Remote temperature sensor device

BASIC DEVICE DESCRIPTION

The device connects to a server through a wireless network and the Internet. In basic form, it does not use any encryption of its own. Since you will not be using an actual device, you can instead simply write a program that behaves as such a device should. You should generate random, but plausible temperature readings from your program and interact as specified with the remote server.

Every T seconds (T=3 by default), the device takes a temperature reading. It sends that reading to the server in the following format:

"[DevID] TEMP = ##.#" where "##.#" is replaced by a reading to one decimal place. [DevID] should be a unique 9 decimal digit device identifier. Devices should be configured with one particular ID, which should never change.

Readings are sent even if they are repetitions of the previous reading, by default.

Temperature can be measured and reported in either Fahrenheit or Centigrade. Fahrenheit is used by default. Every reading should be stored in a log file, on a separate line.

COMMANDS

The device should accept the following commands from the server. Extraneous characters at the end of a command should result in the command being ignored. For commands accepting a parameter, non-valid values of the parameter should result in the command being ignored. No error messages will be sent to the server when commands are ignored. Each command received, whether valid or invalid, should be saved into the same log file as the temperature readings. The entire command received should be written to the log, each on its own line. Invalid commands should have the characters " I" (space capital-I) appended to the end of the received command, to indicate they were invalid.

TURN OFF - The device should turn off, which means the program should stop taking readings or responding to commands. It is OK to simply exit your program when you receive the OFF command and ignore any further commands sent from the server. The program only needs to start again when you explicitly run it. Nothing sent to the device while it is turned off should be logged or even read. But you should insure that after handling the OFF command, whether by exiting the program or doing something else, you should be able to reconnect to the server at another time.

Format – "OFF"

STOP – The device should stop taking and reporting readings. However, the program should continue running and should continue to accept remote commands. If another STOP command is received while the device is in stop mode, it should be treated as a no-op, but should be logged.

Format – "STOP"

START - If the device is in stop mode, the device should start taking readings and should report them. It should restart in whatever state it was in when it stopped, meaning the interval of measurement and reporting will remain the same, the choice of Fahrenheit or Centigrade should be consistent, and whether the temperature is displayed locally should be the same, unless a command changing those settings was received while the device was stopped. If the device is not in stop mode when a START command is received, the command should be treated as a no-op, but should be logged.

Format – "START"

CHANGE SCALE – The device should start using the scale specified in the command to report its temperature reading. It should continue using this scale across STOP and START commands. CHANGE SCALE commands received while the device is stopped should be honored, but no readings should be sent until a START command is received. A TURN OFF command followed by a restart should start it in the Fahrenheit scale, which is the default scale. A CHANGE SCALE command requesting the same scale as is currently in use should be treated as a no-op, but should be logged.

Format – "SCALE=[F/C]" where "F" means change to Fahrenheit and "C" means change to Centigrade.

Examples: SCALE = F

SCALE = C

CHANGE PERIOD – The device should change the period at which it measures temperature. The period is measured in seconds and should be an integer value between 1 and 3600. If the requested period is the same as the current frequency, the command should be treated as a no-op, but should be logged. CHANGE PERIOD commands received while the device is stopped should be honored by changing the reporting period, but no reports should be sent until the device receives a START command.

Format - "PERIOD=####" where "####" is a number between 1 and 3600. The number may have leading zeros, but should be properly interpreted whether it does or does not have them. A number out of range should be treated as an error, meaning it will be logged with an invalid indication, but will not result in the current period of the device readings being changed.

Examples: PERIOD = 5

PERIOD = 010

PERIOD = 2400

PROTOCOL

Your program should set up a TCP connection to the server using a socket mechanism and should maintain and use that connection as long as the server is reachable. The device should initially contact the server on a specified port and send a message containing the student's 9 digit ID number. This number should subsequently be used as the DEV_ID, mentioned above.

On top of TCP, the device should be prepared to accept a command at any time and act upon it when received. Commands will not require the device to send a response. Also, as long as the device is in measurement mode, it should send a message on that socket in the format described earlier reporting the temperature read from its sensor at the specified period. There will be no application-level acknowledgement of the measurement messages. The device should not send any messages to the server on this socket other than the initial setup and subsequent measurement messages.

The server will run on a particular machine and port. I will provide students with these details. The server will send the client a set of commands, terminating with the OFF command. The student's client code must properly send device readings to the server throughout and respond properly to all commands sent.

THE TLS VERSION

After completing the initial version of the device software, students must add cryptography to the protocol by running it over TLS. A separate TLS server will be made available. Each student should add the necessary TLS functionality to his client code to allow him to connect to the secure server. This secure connection will encrypt the traffic in both directions, both the readings sent from client to server and the commands sent from server to client.

The secure server will also send the client a set of commands, which must be performed properly, while also sending the server all readings. Again, the set of commands will terminate with an OFF command.

LAB 4 COMPONENTS

In addition to this document, I will provide the DNS name of the server to be used and the numerical ports used for the regular and TLS version of the server. The device emulation written by the student can run on any system, as long as it interacts correctly with the server. It can be run on the SEAS Linux servers, at least. The students will not be provided with the servers' source code nor will they be given code skeletons for developing the client code.

DELIVERABLES

The student must deliver the basic client code, a log for the activities of the basic client/server interaction as outlined above, the TLS version of the client code, and a log for the activities of the secure client/server interactions