PL Assignment: Design Network Protocol

Assignment Introduction

The Design Network Protocol assignment consists of two parts. In the first part you will design a network protocol that will be implemented by other student(s). In the second part you will implement the protocol designed by someone else. Both parts of the assignment will be graded by a fellow student. You will be assigned all tasks individually or as a team depending on your preference. When submitting files, it is important that you do not include any personal identifying information. Also, files need to be 2MB or smaller.

To sign up as a team, you will notify the professor of your intention and provide the name of your partner. Only the 'designated' team member will receive an account in the PL system that they will share with the other team member to submit the work required. Therefore, team members will be sharing an account by sharing the username/password with each other. (Do not use your UCID password for the PL system.)

The assignment has quality control steps built-in. To ensure that the protocol design can be implemented, there will be several tasks designed to improve its quality: (A) Two students/teams will comment on the initial protocol design. (B) Based on these comments, you will update your initial protocol design. (C) The updated protocol will be reviewed again to ensure that it can be implemented. If it cannot be implemented, it will be returned for revision, possibly multiple times until approved or until the due date. If the protocol has not been approved in time, this step will be bypassed, you'll receive no credit for it, and the professor will provide a different protocol design for another student to implement instead.

As another quality control step, two students/teams will grade the implementation with a third consolidating the grades if needed. After receiving your grade, you will have until the due date to dispute it to the professor by regrading your own implementation.

It is important to remember that the majority of the steps in the assignment are done by the students and thus if you do not complete your task, there will be another student who won't be able to continue the assignment as they will be waiting for you to finish. To ensure that the process runs smoothly, please stick to the deadlines so that the process moves along. There will be a grace period of 6 hours for most task deadlines (unless otherwise specified) after which you will be removed from the process and replaced by another student. If you get removed from the task, you will not be able to participate in subsequent tasks. If you have any issues please consult with the PL webmasters (Erick: efs3@njit.edu, Prof. Bieber: bieber@njit.edu) to have those issues sorted out.

PL Assignment Tasks

The following are the tasks to be done as part of the assignment

Design a Network Protocol

Based on the following instructions you will design a network protocol for another student to implement. The due date for this task is October 26th at midnight.

Instructions

Specify a protocol that communicates over UDP with an Internet of Things (IoT) device, a light bulb. The protocol must allow a client program to perform the following operations on a light bulb server:

1. *Turn the light bulb on and set the color (if the light bulb is already on and color the same, the operation should work without error)*
2. *Change the light color*
3. *Determine the status of the light: on/off, color*
4. *Turn the light off (If the light bulb is already off, the operation should work without error)*

The protocol must include a mechanism for:

1. *Error reporting: e.g. the bulb is not functioning, color specified is not supported, message format is not recognized*
2. *Reliability: Provide a response that the operation worked successfully. Also specify the behavior of the client when no response is received.*

The protocol must specify:

1. *Message Format (fields in the message, size in bytes, possible values)*
2. *Message Semantics (what function each message type performs, and possible results)*
3. *Trace output that client and server programs must print to demonstrate the program is working including examples.*
4. Command-line arguments that specify IP address and port the server is running on and enable any test case to be executed (in arbitrary order)

Note: See the DNS assignment for an example of a protocol specification, including how to draw and describe a message format.

The protocol design will be judged on the following:

- *Is the specification complete? Does the protocol meet all requirements specified above, and support all operations?*
- *Is the specification clear and unambiguous? Can a reader of the specification implement the protocol? If the client and server were written by two different programmers, would the two programs interoperate?*

IMPORTANT: There is no one way to design a protocol. Therefore, it is expected that the precise message format and protocol behavior will differ between authors of the specifications. However, your design will ***not*** *be able to proceed to the next step unless your design is complete and unambiguous. You will be given a chance to revise your protocol based on feedback. No credit will be given if the protocol design is not approved by the last due date.*

NOTE: Before you submit your file, make sure you rename the file “Protocol_Design_Draft” and remove any personal identifying information.

Comment on Network Protocol Design

Please go to the **“Design a Network Protocol”** task and download the file named *“Protocol_Design_Draft.”* Based on instructions provided on how to design a network protocol, please provide feedback on how to improve the protocol design so that:

- It meets all the requirements
- It is implementable so that the client and server can operate with each other

In your comments, make sure to also follow the following instructions

Instructions

To approve the specification, make sure that the answers to the following questions would be clear to an implementor of the specification:

1. *What message type and fields are used to request each of the following operations?*
   - *1.1. Turn the light bulb on and set the color*
   - *1.2. Change the light color*
   - *1.3. Determine the status of the light: on/off, color*
   - *1.4. Turn the light off*
2. *What message type and fields are returned in response to a request for each of the above operations?*
   - *2.1. When the operation is successful*
   - *2.2. When the operation is unsuccessful (either because there is something wrong with the message format in the request or an operation cannot be performed e.g. the light does not support the color specified or the bulb is broken)*
3. *Is it clear how to format message requests and responses, including*
   - *3.1. The name and purpose of each field*
   - *3.2. Date type of each field (string, number)*
   - *3.3. Valid values for each field*
   - *3.4. Size in bytes of each field*
4. *Is it clear how the client or server behaves in the following error cases:*
   - *4.1. A request or response packet is dropped for whatever reason*
   - *4.2. A request or response packet is duplicated for whatever reason*

In short, does the specification provide sufficient information to implement the protocol? If the client and server were written by two different programmers based only on reading the specification, would the two programs interoperate?

If not, provide concise feedback regarding where the specification is incomplete or unclear. Provide suggestions on what is needed to complete or clarify the protocol design.

Your feedback should be objective, clear and be as specific as possible so that the protocol designer(s) can make use of it to improve their protocol design.

The due date for this task is October 29th at midnight.

Update the Network Protocol

Based on the feedback provided on the **"Comment on Network Protocol Design"** task, update your protocol design so that it meets that requirements outlined by the instructor and is implementable. The due date for this task is November 2nd at midnight.

Before you submit your file, make sure you rename the file *"Updated_Design_Draft"* and remove any personal identifying information.

NOTE: After you submit this, another student will need to approve it, confirming that it is implementable. That student may return it to you and provide feedback for revising it. You must do this as many times as necessary for that student to approve it, or you will get a zero grade for the protocol. You must check your PL dashboard periodically to see if the task has been returned to you for revision! When resubmitting the revised draft, please delete the previous file and upload a new one. Please also add a version number to it. (i.e., *Updated_Design_Draft_2, Updated_Design_Draft_3* …).

Approve the Protocol Design that You Will Implement

NOTE: Do NOT upload any file in this task.

Please go to **"Update the Network Protocol"** task and download the *"Updated_Design_Draft"* file (or *"Updated_Design_Draft_X"* if there have been multiple revisions.)

You must evaluate this file and decide whether it is fully implementable as written. (If so, you will be implementing it!) If it is fully implementable, click on the "Approve" button. If it is not implementable as written, then provide feedback for how it must be improved and click on the "Return for Revision" button. (In this case the author can revise it and the PL system will notify you when a new version is ready for you to evaluate again.) This will be an iterative process that could potentially require multiple submissions.

The final deadline for this process is November 9th at midnight. However, you should review the updated protocol and send feedback back to the protocol designer(s) by November 4th so that the protocol designer(s) can make the appropriate changes.

If by November 9th the Protocol has not passed approval, the Professor will take over this task to ensure that the faulty protocol is replaced by one that can be implemented. Students that have a protocol design that did not pass approval by the deadline will receive no credit for this part of the assignment.

If the Protocol is not approved, please let the task expire as the Professor will handle it. Do not upload any file in this task nor approve a protocol that cannot be implemented!

Implement the Protocol Design for Server

First check whether there is a file uploaded in the **"Approve Protocol Design"** task above and download the file named *"Replacement_Protocol."* Only if there is no file there, then go to **"Update the Network**

Protocol" task above and download the *"Updated_Design_Draft"* file (or *"Updated_Design_Draft_X"* if there have been multiple revisions). Rename this file *"protocol_design_implemented"* and upload it along with the other implementation files required below.

The due date for this task is November 24th midnight. Please note that the Implementation of the Protocol has been split up into two parts and both have the same deadline. You must first submit this task before the next task "Implement the Protocol Design for Client" is assigned to you.

Instructions
If you are working as a team, one team member implements the server and one team member implements the client. Each team member is responsible for their code and must write and test the program independently of the other team member.

The server code emulates a light bulb. It receives messages from the client requesting the light bulb to be turned on/off and change color, and to provide current status. Implement the server code per the protocol specification. If you are working as a team, use the client code written by the other team member to run all test cases against the server.

Test Cases should demonstrate the following operations:

1. *Turn the light bulb on and set the color (if the light bulb is already on, the operation should make no change)*
2. *Change the light color*
3. *Determine the status of the light: on/off, color*
4. *Turn the light off (If the light bulb is already off, the operation should make no change)*

Negative test cases should include:

5. *The client sends a message with an incorrect message format*
6. *The client sends a message where the color specified is not supported by your server*

Upload the following files:

7. *protocol_design_implemented*
8. *iot-server.py*
9. *Screenshots of the client and server output demonstrating all test cases(.pdf)*
10. *Wiireshark trace demonstrating all test cases (.pcap file)*

NOTE: Before you submit your files, make sure you name them correctly and remove any personally identifying information.

Implement the Protocol Design for Client

You will be implementing the same protocol design for the client as you used in the prior task "Implement the Protocol Design for Server."

The due date for this task is November 24th midnight. Please note that the Implementation of the Protocol has been split up into two parts and both have the same deadline. You must first submit "Implement the Protocol Design for Server" before this task is assigned to you.

Instructions
If you are working as a team, one team member implements the server and one team member implements the client. Each team member is responsible for their code and must write and test the program independently of the other team member.

The client code is used to send operations to the light bulb server. It sends messages to the server requesting the light bulb be turned on/off and change color, and to provide current status. Implement the client code per the protocol specification. If you are working as a team, use the server code written by the other team member to run all test cases.

Test Cases should demonstrate the following operations:

1. *Turn the light bulb on and set the color (if the light bulb is already on, the operation should make no change)*
2. *Change the light color*
3. *Determine the status of the light: on/off, color*
4. *Turn the light off (If the light bulb is already off, the operation should make no change)*

Negative test cases should include:

5. *The client sends a message to a server that does not respond*
6. *The client receives a message with an incorrect message format*

Upload the following files:

7. *iot-client.py*
8. *Screenshots of the client and server output demonstrating all test cases(.pdf)*
9. *Wireshark trace demonstrating all test cases (.pcap file)*

NOTE: Before you submit your files, make sure you name them correctly and remove any personal identifying information.

Grade the Protocol Implementation

Please go to the following tasks **"Implement the Protocol Design for Server"** and **"Implement the Protocol Design for Client"** task sections above. The due date for this task is December 3rd by midnight. Download the following files:

- *protocol_design_implemented [from the server implementation task]*
- *Iot-server.py*
- *Iot-client.py*
- *Screenshots (.pdf) [from both server and client implementation tasks]*
- *Wireshark trace [from both server and client implementation tasks]*

Review the implementation against the instructions carefully. Examine whether the implementation of the protocol was completed according to the proposed design. The protocol design that the students have implemented will be uploaded along with the rest of the files implemented.

Also follow the test cases and instructions provided here.

Test cases and Instructions

Run all test cases on the client and server files. Capture a wireshark trace while the programs are running. Use the trace output on the screen and the wireshark.pcap file to grade the program.

Note: The grader MUST run the programs and capture their own wireshark trace for grading purposes. The screenshots and wireshark packet capture files submitted by the implementors are used to compare your results in the event there are errors.

Grade Breakdown:
30 points for protocol implementation; each test case is worth 6 points

Test Case 1 (6):

- Turn the light bulb on with a supported color.
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 2 (6):

- Change the color of the light bulb
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 3 (6):

- Get the status of the light bulb (on/color; off; broken)
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 4 (6):

- Turn the light off
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Negative Tests (6):

- Does the client operation retry according to the specification requirements when the server does not respond? (2)
- If the message format is incorrect, does the server/client respond with the appropriate error? (2)
- If the color is not supported, does the server respond as expected? (2)

Consolidate Grade

Please go to the following tasks **"Implement the Protocol Design for Server"** and **"Implement the Protocol Design for Client"** task sections above. The due date for this task is December 7th by midnight. Download the following files:

- *protocol_design_implemented [from the server implementation task]*

- *Iot-server.py*
- *Iot-client.py*
- *Screenshots (.pdf) [from both server and client implementation tasks]*
- *Wireshark trace [from both server and client implementation tasks]*

Review the implementation against the instructions carefully. Examine whether the implementation of the protocol was completed according to the proposed design. The protocol design that the students have implemented will be uploaded along with the rest of the files implemented.

Also follow the test cases and instructions provided here.

Test cases and Instructions
Run all test cases on the client and server files. Capture a wireshark trace while the programs are running. Use the trace output on the screen and the wireshark.pcap file to grade the program.

Note: The grader MUST run the programs and capture their own wireshark trace for grading purposes. The screenshots and wireshark packet capture files submitted by the implementors are used to compare your results in the event there are errors.

Grade Breakdown:
30 points for protocol implementation; each test case is worth 6 points

Test Case 1 (6):
- Turn the light bulb on with a supported color.
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 2 (6):
- Change the color of the light bulb
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 3 (6):
- Get the status of the light bulb (on/color; off; broken)
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Test Case 4 (6):
- Turn the light off
    - Successful operation (3 points)
    - Correct Message Format per wireshark trace (3 points; -1 for each error)

Negative Tests (6):
- Does the client operation retry according to the specification requirements when the server does not respond? (2)

- If the message format is incorrect, does the server/client respond with the appropriate error? (2)
- If the color is not supported, does the server respond as expected? (2)

(Optionally) Dispute Grade

Optionally, you can review and dispute your grades found at **"Grade the Protocol Implementation"** and **"Consolidate Grade"** tasks (please note that not all assignments will have a consolidate grade task as it only triggers under special circumstances). When there are only two grades provided, the highest total grade of the two will be your assignment grade. To dispute your grade with the Professor, you must regrade your own implementation and justify your proposed grade.

To regrade yourself, please review your implementation against the proposed protocol design and the test cases provided by the instructor.

The due date for disputing your grade is December 9th by midnight. There will be no grace period for this task.