CLIENT-SERVER GAME TASK

Version 2.1
BSc Computing for Games
COMP260

Gareth Lewis

"Essentially everyone, when they first build a distributed application, makes the following eight assumptions:

- The network is reliable;

- Latency is zero;

- Bandwidth is infinite;

- The network is secure;

- Topology doesn't change;

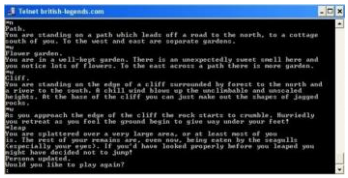
- There is one administrator;

- Transport cost is zero;

- The network is homogeneous.

All prove to be false in the long run and all cause big trouble and painful learning experiences."

— Peter Deutsch



MUD1, written by Roy Trubshaw and Richard Bartle—now, Prof. Bartle—was the first virtual world to run multiplayer over a network. It was written in 1978, pre-dating the Internet.

Introduction

For this assignment, you will extend your MUD from assignment 1 to create a remote, robust and secure MUD service. Your MUD service will feature secure player accounts and secure client server communications as well as a robust server game data model that will manage players in a persistent manner. The server application will be hosted on a remote server.

Knowledge of computer networking is essential. You will need to work with highly sophisticated network technologies to satisfy increasing demand for social experiences in games. Experience working with client-server architectures, in particular, is highly desired by employers even outside games.

This assignment is formed of several parts:

(A) Extend your client-server MUD such that:

i. The server operates from a remote Digital Ocean Ubuntu Droplet
ii. The server supports multiple simultaneous thin clients, with no significant local processing / data storage
iii. The system incorporates network communications between a local client and remote server over the Internet using TCP
iv. Communications between client and server are encrypted
v. The server manages all major game content (e.g., world topology and description);
vi. Player access to the server is controlled with accounts and salted passwords
vii. Sensitive user data is encrypted and no plain text passwords are stored on client or server
viii. Server data is managed within appropriate databases
ix. Databases discriminate between session and persistent data.
x. The MUD is implemented as a persistent game world
xi. Your MUD makes some interesting use of persistent game data

cont…

(B) Implement a final prototype of the game that will:
   i. Respond to security concerns raised in the security wiki;
   ii. Revise any other issues raised by your tutor and/or your peers.

(C) Present a live demonstration of the MUD that will:
   i. Show your academic integrity and technical communication skills.

Please Note: Falmouth University Games Academy have partnered with DitigalOcean to provide **EACH** of you with your own remote server on their cloud platform. You will receive login details after completing the first coursework task. You **MUST** act in an ethical and professional manner when using this platform. If you encounter technical difficulties, please contact `games.techs@falmouth.ac.uk` in the first instance.

Assignment Setup

This assignment is a programming task. Fork the GitHub repositories at:

```
https://github.com/Falmouth-Games-Academy/comp260-client
https://github.com/Falmouth-Games-Academy/comp260-server
```

Ensure that you maintain the readme.md file. Modify the .gitignore to the defaults for the programming languages and integrated development environments you will be using, as appropriate.

Part A

Part A is a **single formative submission**. This work is **individual** and will be assessed on a threshold basis. The following criteria are used to determine a pass or fail:

(a) Submission is timely.
(b) Enough work is available to conduct a meaningful review.
(c) A broadly appropriate review of a peer's work is submitted.

To complete Part A, prepare draft versions of the server and client programs. Deploy these to the DigitalOcean server. Also ensure that the source code and related assets for both client and server are pushed to GitHub. These should be made available for review prior to the scheduled peer-review session. Then, attend the scheduled peer-review session.

Important: It is your individual responsibility to ensure the server code has been deployed and is running ahead of the peer review. Tinkering and last minute bug fixing should be done **before** the workshop. Though do not worry if, at this stage, your work is not feature complete and the game still suffers from a few bugs. This is

cont…

an opportunity to get feedback and advice on your work-in-progress.

You will receive immediate **informal feedback** from your **peers**.

Part B

Part B is a **single summative submission**. This work is **individual** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of this document for further detail.

To complete Part B, revise both the client and server programs based on the feedback you have received. Then, upload both to the LearningSpace. Please note, the LearningSpace will only accept a single .zip file.

You will receive **formal feedback** from your **tutor** three weeks after the final submission deadline.

Part C

Part C is a **single summative submission**. This work is **individual** and will be assessed on a threshold basis. The following criteria are used to determine a pass or fail:

(a) Enough work is available to hold a meaningful discussion.
(b) Clear evidence of programming knowledge **and** communication skills.
(c) No breaches of academic integrity.

To complete Part C, prepare a practical demonstration of the MUD. Ensure that the source code for both client and source code, in addition to any related assets, are pushed to GitHub and made available for review prior to the scheduled viva session. Then, attend the scheduled viva session.

Important: It is your individual responsibility to ensure the server is running and stable ahead of the viva. Tinkering and last minute bug fixing should be done **before** the allotted time-slot. No extra time will be given. You must also ensure that a network connection can be obtained using your computer. Test the wired and/or wireless network connection at the venue ahead of the viva.

You will receive immediate **informal feedback** from your **tutor**.

Additional Guidance

A common pitfall is poor planning or time management. Often, students underestimate how much work is involved in first learning programming concepts and then actually applying them. Programming is quite unlike other subjects in that it cannot be crammed into a last minute deluge just before a deadline. It is, therefore, very important that you begin work early and sustain a consistent pace: little and often. The live

cont…

deployment, in this assignment, is an added dimension. Aim to complete your client and server a week early so you have sufficient time to troubleshoot your DigitalOcean instance, your server stability, and the network protocols you are using.

It is very easy for the deadline for this assignment to sneak up on students. Although a MUD looks quite simplistic in comparison to the games you have been making using game engines, it requires a considerable effort to get an application operating robustly over a computer network.
Be wary! Start early, and dive into research on these topics.

Do not underestimate matters of security. You **MUST** consider the security of your solution and incorporate appropriate safeguards in order to obtain a passing grade on the distributed infrastructure criterion. Start the research journal early, and incorporate both your own findings as well as the findings of your peers into your work. Act as a research community and develop a discourse that raises awareness of key issues.

On a related note, you **MUST** also incorporate a database into your solution. This is an absolute requirement. If you didn't master the material on SQLite and mySQL databases that were introduced to you in the first year, then you should revisit these topics. A great resource to do so is W3Schools:

`https://www.w3schools.com/sql/default.asp`

Likewise, the material on UNIX-based servers and commands will not be for- mally covered again. Here is a resource to remind yourself:

`http://mally.stanford.edu/~sr/computing/basic-unix.html`

Consider carefully how the MUD service should cope with session and persistent data. Persistent data will likely be user and player-centric, like username, password, experience, room, inventory, and so on. Session data is likely to be things like current chat log. Things that developers would expect to be discarded when the player finishes playing.

FAQ

- **What is the deadline for this assignment?**
  Falmouth University policy states that deadlines must only be specified on the MyFalmouth system.

- **What should I do to seek help?**
  You can email your tutor for informal clarifications. For informal feedback, make a pull request on GitHub.

- **Is this a mistake?**
  If you have discovered an issue with the brief itself, the source files are available at:

  https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs.

cont…

Please raise an issue and comment accordingly.

Additional Resources

Additional resources have been migrated to the Talis Aspire system, which is available at:

http://resourcelists.falmouth.ac.uk/.

cont…

Marking Rubric

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Threshold	40%	At least one part is missing or is unsatisfactory.	Submission is timely. Enough work is available to hold a meaningful discussion. Provided a meaningful review of a peer's work. Clear evidence of programming knowledge and communication skills. Clear evidence of use of appropriate version control techniques, including regular commits and some use of branching. No breaches of academic integrity.				
Remote Service	10%	Client and server cannot communicate Server is not hosted remotely on Digital Ocean droplet	Server is hosted on DO droplet and requires multiple restarts during the viva	Server is hosted on DO droplet and requires a single restart during the viva	Server is successfully hosted on DO droplet and runs without the need to reset it during the viva	Server is successfully hosted on DO droplet and runs without the need to reset it during the viva Server can support multiple clients from different IP addresses	Server is successfully hosted on DO droplet and runs without the need to reset it during the viva Server can support multiple clients from different domains
User Security	10%	Application has no user / account security Username + password is not required to play MUD	Users can log on during viva regardless of credentials	Users can successfully log in during viva Users cannot log in with incorrect credentials	Users can successfully log in during viva Users cannot log in with incorrect credentials A user cannot log on multiple times	Users can successfully log in during viva Users cannot log in with incorrect credentials A user cannot log on multiple times New accounts can be created from client	Users can successfully log in during viva Users cannot log in with incorrect credentials A user cannot log on multiple times New accounts can be created from client & validated
Communication Security	10%	Client server communication are implemented as unencrypted byte streams that can be cast to strings and read	MUD service uses a trivial approach to encrypt packet data	MUD service uses a weak approach to communication security	MUD service uses a strong approach to communication security, such as AES/ Rijndael	MUD service uses a strong approach to communication security, such as AES/ Rijndael, and adds packet sequencing to stop packet replay hacking	MUD service uses multiple strong approaches from security wiki that go beyond typical industry standards
Data Persistence	30%	Server does not use relational database to manage persistent game data or client data On returning to the game, players restart at the 'beginning' of the dungeon rather than where they last were	User data is stored in server-side SQL database Player data is stored in server-side SQL database	Implementation of developer-defined persistent features that work badly e.g.: -last log-on details -last attempted log-on -returning player information / MotD -message service for off-line players -ownership/transference of game objects	Implementation of developer-defined persistent features that work reasonably well	Implementation of developer-defined persistent features that work well	Implementation of developer-defined persistent features that work very well