

DISTRIBUTED SYSTEMS:

COMPUTING ARTEFACT

Version 1.0

BSc(Hons) Computing for Games

COMP260

Gareth Lewis

|  |  |
| --- | --- |
| */var/folders/v7/bjz4cys124x93w_t3bjp7dgw0000gn/T/com.microsoft.Word/Content.MSO/CC0A6944.tmp*  *“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*        *Multi-threading is commonly used to improve performance in games.*  Image result for unreal engine  *A computer lets you make more mistakes faster than any invention in human history - with the possible exceptions of handguns and tequila.*  *-Mitch Ratcliffe* | Introduction This assignment is in two parts (Parts I & Part II) and should provide material for the Technical Report assignment. Part I is concerned with building a turn-based game that can be remotely hosted on a Digital Ocean droplet whilst Part II is concerned with building a real-time game using appropriate middleware to handle object replication across nodes. For both parts of the assignment, you have a free choice of technology stacks, though the module will concentrate on Python, C# and Unity. Part I Turn-based Service Provision This part of the assignment is concerned with designing, implementing and hosting a simple turn-based game on a remotely hosted server using Digital Ocean as a service provider. By *turn-based*, we are referring to a class of games where players take it in turns to play, like board games, card games and classical multi-user dungeons.  To complete this part of the assignment:   1. Use UML to **Analyse** existing turn-based client/server applications to determine how clients and servers work together to create a network-based service. 2. Use UML to **Design** an architecture that will support your game of choice using an appropriate application technology stack. 3. **Implement** your application and **host** your server on a Digital Ocean droplet.  Part A **Use UML to Analyse existing turn-based client/server applications**  To complete this part of the assignment, we will have in-class sessions that use UML to analyse the structure, function and data communications of several client/server applications.  You will receive **informal feedback** from your **tutor** during the tutorial sessions and you will receive **formal feedback** from your **tutor** three weeks after the final submission deadline. Part B **Use UML to Design an architecture that will support your game service**  To complete this part of the assignment, create an appropriate set of UML documentation that captures the structure, function and data communications aspects of the service that you wish to create.  You will receive **informal feedback** from your **tutor** during the tutorial sessions and you will receive **formal feedback** from your **tutor** three weeks after the final submission deadline. Part C **Implement your application and host your server on a Digital Ocean droplet**  To complete this part of the assignment, implement your client/server solution using an appropriate technology stack and host it on a provided Digital Ocean droplet.  Care should be taken to remember that the Digital Ocean droplet may not have exactly the same functionality as your development Windows machine, so some consideration should be given to how you will undertake development and testing.  You will receive **informal feedback** from your **tutor** during the tutorial sessions and you will receive **formal feedback** from your **tutor** three weeks after the final submission deadline. Part II Real-time gaming provision  * Make a real-time game using Unity & a networking framework of your choice (photon or gamesparks) * Some kind of design * Appropriate hosting  Additional Guidance Creating reliable client/server services sounds like a big task that is both hard and complex. In reality it is actually comprised of several hard and complex tasks that are fairly small, and we will discover that by breaking large tasks down into smaller tasks they become soluble.  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 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 Digital Ocean instance, your server stability, and the network protocols you are using.  **Part I Turn-based Service Provision**  The Digital Ocean droplet provides a *headerless* server which can only be accessed remotely. This can make development and debugging difficult. In addition, cross-platform languages (C# & Python) may not be as compatible across different operating systems as we would expect. A common solution to these issues is to develop in Linux, either on a standalone box or a virtual machine running on a Windows host.  Material on UNIX-based servers and commands will not be formally covered again. Here is a resource to remind yourself: http://mally.stanford.edu/~sr/computing/basic-unix.html  **Part II Real-time gaming provision** 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. |

# Marking Rubric – Part I: Turn-based Service Provision

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Learning Outcome Name** | **Learning Outcome Description** | **Criteria** | **Weighting** | **Refer for**  **Resubmission** | **Adequate** | **Competent** | **Very Good** | **Excellent** | **Outstanding** |
| Architect | Integrate appropriate data structures and interoperating components into computing systems, with reference to their merits and flaws. | Analysis | 20% | No analysis presented | Some analysis of client & server components  Incomplete class hierarchy, state model and/or data communications models | Competent analysis of client & server components  Fairly complete class hierarchy, state model and/or data communications models with only small aspects missing / wrong | Good analysis of client & server components  Complete class hierarchy, state model and/or data communications models | Excellent analysis of client & server components  Complete class hierarchy, state model and/or data communications models  Some consideration of issues in source materials | Outstanding analysis of client & server components  Complete class hierarchy, state model and/or data communications models  Significant consideration of issues in source materials and solutions presented |
| Design | 30% | No design presented | Some design of planned client & server components  Incomplete class hierarchy, state model and/or data communications models | Competent design of planned client & server components  Fairly complete class hierarchy, state model and/or data communications models with only small aspects missing / wrong | Good design of planned client & server components  Complete class hierarchy, state model and/or data communications models  Implementation appears possible from design specs | Excellent design of planned client & server components  Complete class hierarchy, state model and/or data communications models.  Implementation appears straightforward from design specs | Outstanding design of planned client & server components  Complete class hierarchy, state model and/or data communications models  Implementation appears simple from design specs |
| Remote Hosting | 50% | No working software presented | Client & server only work locally | Client & server work faultlessly locally  Client & server work sporadically remotely | Client & server work faultlessly locally  Client & server work fairly well remotely | Client & server work faultlessly locally  Client & server work faultlessly remotely | Client & server work faultlessly locally  Client & server work faultlessly remotely  Working support for sharding |