**Assignment 1**

CS5352 Advanced Operating Systems Design Spring 2016

**Q1. (Chapter 1, Problem 2)** What is the role of middleware in a distributed system?

Middleware is a distributed system in the form of a layer of software, it is logically located between a higher level layer consisting of Users and Applications and a lower level layer consisting of operating system and basic communications. Middleware provides an abstraction that allows users at the higher level to view the lower level as a single-system view whilst still providing support to heterogeneous computers and networks.

**Q2. (Chapter 1, Problem 4)** Explain what is meant by (distribution) transparency, and give examples of different types of transparency.

Transparency is the practice of abstracting a distributed systems processes and resources to give the appearance of a single computer system. The different types of transparency in distributed systems are as follows:

* Access
  + An example of Access transparency is the act of taking systems which may be running different operating systems and making them appear to be a single functioning system.
* Location
  + An example of Location Transparency is the URL we use every day in our web browsing. This url takes us to a website that appears to have everything in a single location, but the site could be spread across multiple servers in multiple countries if there was a need.
* Migration
  + An example of Migration Transparency would be something like the index.html page of a website. This is a resource that can be moved without affecting how the resource is being accessed.
* Relocation
  + Relocation Transparency is a specialized form of Migration Transparency in which the resources can be relocated while in use. An example of this would be things like laptops that can be moved from place to place without ever being disconnected from their network.
* Replication
  + Replication transparency deals with abstracting the fact that several copies of a resource may exist an example of this would be mirroring web pages.
* Concurrency
  + Concurrency transparency deals with users who are simultaneously accessing a resource and the ability to hide from one user what another user is doing. Examples of this can be found in banking applications or in db management systems.
* Failure
  + Failure transparency is a way of making users do not notice that a resource they were accessing fails to work properly, an example of this is a web browser timing out when a web server is down and cannot deliver the page requested.

**Q3. (Chapter 1, Problem 7)** What is an open distributed system and what benefits does openness provide?

It is a system that offers services according to standard rules that describe the syntax and semantics of those services. It is able to easily interface with other open systems, and allows applications to easily be ported between different implementations of the same system.

**Q4.** Please briefly explain what cluster computing systems, high performance computing systems, grid computing systems, and cloud computing systems are.

Cluster computing consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.

High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to systems that function above a teraflop or 1012 floating-point operations per second.

Grid computing is a distributed architecture of large numbers of computers connected to solve a complex problem. In the grid computing model, servers or personal computers run independent tasks and are loosely linked by the Internet or low-speed networks. Computers may connect directly or via scheduling systems.

Cloud computing, also known as 'on-demand computing', is a kind of Internet-based computing, where shared resources, data and information are provided to computers and other devices on-demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources. It is quite similar to Grid Computing.

**Q5. (Chapter 1, Problem 9)** Scalability can be achieved by applying different techniques. Can you give three examples?

It can be achieved through distribution, caching, and replication.

**Q6. (Chapter 2, Problem 2)** What is a three-level client-server architecture?

It is an architecture in which there are three logical layers. Layer 1, which is the highest layer generally consists of a client user interface, the middle layer or Layer 2 contains applications; finally Layer 3 the lowest layer generally implements the data.

**Q7. (Chapter 2, Problem 4)** Consider a chain of processes P1, P2, ..., Pn implementing a multitiered client-server architecture. Process Pi is client of process Pi+1, and Pi will return a reply to Pi-1 only after receiving a reply from Pi+1. What are the main problems with this organization when taking a look at the request-reply performance at process P1?

The problem here is when we look at Pn process and n gets very large it can begin to have poor performance. Furthermore with a chain setup like this if a system in the middle of the chain were to crash or get bogged down for some reason it would cause major issues in which the performance of the entire chain is greatly reduced if not stopped all together.

**Q8. (Chapter 2, Problem 5)** In a structured overlay network, messages are routed according to the topology of the overlay. What is an important disadvantage of this approach?

Physical distances are not always well represented in logical paths. What may seem like a short distance for two nodes that are logically close to one another, even neighbors may in fact be a very large physical distance and therefore it can cause a very long time for messages that are routed on this path to be exchanged.

**Q9. (Chapter 2, Problem 8)** Consider an unstructured overlay network with N nodes in which each node randomly chooses *c* neighbors. Given two nodes, P and Q, what is the probability that they are neighbors of each other (either P is a neighbor of Q, or Q is a neighbor of P)?

**Q10. (Chapter 2, Problem 17)** Sketch a solution to automatically determine the best trace length for predicting replication policies in Globule.

The server of origin would need to trace Ti to Ti+1 to check its prediction of policy for that period. It can then compare this result against the results in the period Ti-1 to Ti . This will allow us to see if there are any access patterns between the two and allows the server to compute its prediction error. If we then use varying trace lengths we look for a prediction that is minimal and get an automatic detection of the best trace length. This method would contribute to the natural self-managing of Globule.