CS 250 Long Examination

Name: Albert Dale Palacio

Date: April 11, 2017

*Explain why Remote Procedure Calls (RPCs) are not appropriate for group communication. Illustrate by giving a sample scenario.*

In an application standpoint, more specifically in Remote Method Invocation (RMI), Remote Procedure Calls for messaging as opposed to sockets although feasible are not appropriate. This is because that for a RPC-driven communication system there must be lots of channels to listen to and much more handlers. Given, a chat system that only one server will handle the messages then assuming there is an influx multiple messages there might be a blocking method that will prevent the server from accommodating the other messages.

*If your system is dynamic that it can use either broadcast or multicast depending on the situation, under what circumstances should you use broadcast over multicast or vice versa?*

Broadcast is commonly used if there is a message that must be received by all listening ports. Multicast only sends messages to chosen ‘listening’ ports in a system. Ideally, for alarm sending or announcement sending broadcast is preferred. However, if we say in a real-world scenario, we prefer multicasting on radio sets or television sets.

*Explain under what circumstances do layered communication protocols work and not work. Do you think the existing layered communication protocols need further improvements? If yes, what are they?*

The layered communication protocols work such that it is modular and standardized already. However, in my experience working on these layered communication protocols, some protocols are rarely used and often considered as one with another. In the advent of a telecommunications boom, I have developed for a Software-defined Networking environment powerful enough to power a country’s internet infrastructure. In this project, drawbacks of the standard OSI model were very apparent such that it was not feasible to think about data attenuation or speed anymore simply because the problems of the 80’s are nowhere near the 2010’s

*Give a simple discussion with illustration of the evolution of memory management, form uniprogramming to multiprogramming then to virtual memory.*

Memory management in whittled down definition is a computer’s capability of segregation and utilization. A ‘uniprogramming’ system is a centralized number cruncher in which the data sent to and from the computer would pass only one processor. This presented a problem of doing unreasonable tasks such as repetition of computation and big data processing a problem since the programmers or their grandchildren won’t be alive to witness the tasks end. Enter multiprogramming, with multiprogramming, tasks are divided evenly among the processors (e.g. two mathematicians solve problem A and problem B at the same time) and were submitted to housekeeping tasks as the programmers deem needed. This was supposed to be enough but with the array of good housekeeping and memory management algorithms at bay virtualization was the next step. The fast-paced IT industry that we know of cannot be alive if we didn’t have virtualization because the cheapest decent computer would be so pricey that it must not encounter any processing problems because it will surely falter and would be sequential.

*Differentiate a loosely-coupled machine from tightly-coupled machine. What are the distinguishing marks between the two?*

A loosely-coupled machine has different memory allocations for each individual processor such that it is independent enough to handle its own tasks and relay its results to some server. A tightly-coupled machine has a single memory for each individual processor in which a queueing system with a time quantum might be employed. Both have some drawbacks, like loosely-coupling means that we must mind the communication between processes and tightly-coupling means that we must manage what processes tinker with what part of the memory always.

*Discuss how blocking and non-blocking communication primitives operate both for sending and receiving operations.*

Sending through blocking communication (callbacks in javascript) busies the server receiving the message because it must reply to the sender. Sending through non-blocking communication busies the client and have the server receive all tasks even though it is still processing the most recent request. The non-blocking server can then use processing algorithm like FIFO or round robin to fulfill each tasks.

*Explain the difference between internal and external fragmentation. Give a situation where an internal fragmentation happens and another situation where external fragmentation happens.*

Internal fragmentation happens when the memory allocations are larger than the process its catering. External fragmentation happens when a memory could cater a process but there is not a contiguous strip of allocation for it to be placed on.

*In your opinion, what are really the compelling reasons for moving into distributed systems from centralized systems.*

In a world where everything seems so fast-paced already we have computers that already think like humans. In my opinion, a distributed system is more of a hive-mind mentality like political parties and their supporters such that given a problem with the power of a million computers, that all have the same opinion and goal would achieve it faster. That said, a more centralized system is always more secure than a distributed system and with technology blurring this assurance there won’t be any more reasons as to why not go for distributed systems.

*Describe and illustrate the Belady's Anomaly in one of the Page Replacement algorithms.*

When the number of frames increases, page fault rate increases as well. Belady’s anomaly in a FIFO queue occurs when the number of frames allocated to a virtual memory is increased because it was thought to be executed faster and more fault-safe. Execution times might experience a slowing down almost if the frames are increased by volume and the process overflows therefore an anomaly.

*Discuss what is meant by paging and by segmentation. Discuss also the merits of using one over the other.*

In paging, the physical memory is divided to equal sized chunks and a process is given several pages and a little bit more (wasting memory a little bit). In segmentation, the processes are broken up into address spaces and to have sharing. Paging can be used to save on physical memory and segmentation is commonly used in programming since there is a specific size to what segments can handle.