Homework Assignment 2 Solutions
ECE 151 / CMPSC 171 - Spring 2013

MARKING SCHEME:
Problems 1,3,4,5 - 15 points
Problems 2 and 6 - 20 points

1. Compare the reading of a file using (a) a single-threaded file server and (b) a multithreaded file server given the following measurements. It takes 25 msecs for the server to get a request from a client, dispatch it, and do the rest of the processing. The data needed are in a cache in main memory. In one-half of the requests, a disk operation is needed, which requires an additional 85 msecs, during which time the thread sleeps.
In case (a) how many requests/sec can the server handle?
In case (b) how many requests/sec can the server handle?

For single threaded server,
When data is in cache: Time taken=25 msec
Probability of data being in cache=1/2
When data is in disk : Time taken=(25 msec + 85 msec)
Probability of data being in disk=1/2
Average time taken = (25 msec * 1/2) + (110 msec * 1/2)
= 67.5 msec
Now, Requests per sec = $(\text{Average time taken})^{-1}$
= **14.81 requests**
For multi-threaded server,
Even if one thread is waiting, some other thread is being executed. Hence, at any time at least one thread is executing independent of status of other threads.
Average time taken = 25 msec
Requests per sec = $(\text{Average time taken})^{-1}$
= **40 requests**

2. Read about Bit Torrent in the text or on the Web. (a) Summarize what you have learned about Bit Torrent in one or two paragraphs. (b) Explain why Bit Torrent uses a distributed server (as shown on Slide 24, Lecture 3).

(a) Bit Torrent is a protocol that supports the practice of peer-to-peer file sharing and is used for distributing large amounts of data over the Internet. It allows downloading from multiple servers using multiple TCP connections. This is the main difference between classical downloading and Bit Torrent protocol. In classical downloading, data is downloaded using single TCP connection to a single machine.

A user who wants to upload a file first creates a small torrent descriptor file that they distribute by conventional means (web, email, etc.). This file contains metadata about the files to be shared and about the tracker, the computer that coordinates the file distribution. Each piece is protected by a cryptographic hash contained in the torrent descriptor. This ensures that any modification of the piece can be reliably detected, and thus prevents both accidental and malicious modifications of any of the pieces received at

other nodes. The file itself is made available through a BitTorrent node acting as a seed. Those with the torrent descriptor file can give it to their own Bit Torrent nodes which, acting as peers or leechers, download it by connecting to the seed and/or other peers. As each peer receives a new piece of the file it becomes a source (of that piece) for other peers. Pieces are typically downloaded non-sequentially and are rearranged into the correct order by the BitTorrent Client, which monitors which pieces it needs, and which pieces it has and can upload to other peers. Pieces are of the same size throughout a single download (for example a 10 MB file may be transmitted as ten 1 MB Pieces or as forty 256 KB Pieces). Due to the nature of this approach, the download of any file can be halted at any time and be resumed at a later date, without the loss of previously downloaded information.

(b) Bit Torrent uses a distributed server for load balancing. Also as files are distributed over various servers spread over a large area geographically, the latency between peer-seed is reduced. Simultaneous downloading of different pieces of a file from various servers requires distributed server approach.

3. Explain why object (process) migration is so difficult in heterogeneous distributed systems.

The main problem with object (process) migration is that the target machine may not be suitable for executing migrated object due to different local hardware, OS and I/Os. An object may use certain local resources that may not be available at target machine. In a heterogeneous system, different machines may have different resources and different object-to-resource binding. Hence, while migrating, we must consider the compatibility of the two systems in these respects.
A way over this problem is to have an abstract machine that is implemented on different platforms but provides same interfacing. Eg. Java Virtual Machine.

4. In terms of transparency, openness and scalability, explain why services provided by transport-level protocols such as TCP and UDP are often inappropriate for implementing distributed applications?

Transport-level communication services such as services provided by TCP and UDP protocols hardly offer distribution transparency meaning that the application developers are required to pay significant attention to implementing communication, often leading to proprietary solutions. Setting a TCP connection between two end points requires the application developer to configure the connection with the exact IP addresses of the machines and the socket port numbers of the end points. These sort of applications are difficult to port and to interoperate with other applications.
The TCP and UDP protocols support a lot of error control and flow control mechanism. These involve the transmission of substantial amount of control traffic in the network along with the data traffic. There is also substantial traffic overhead involved in setting up the connection between two end points. Hence the traffic in the network grows exponentially as the number of the TCP/UDP connections in the network increases. Hence scalability becomes an issue in these kind of applications.

5. A client calls an asynchronous RPC to a server, and subsequently waits until the server returns a result using another asynchronous RPC. Is this the same as having the client execute a conventional synchronous RPC? Explain why or why not.

No they are not the same.

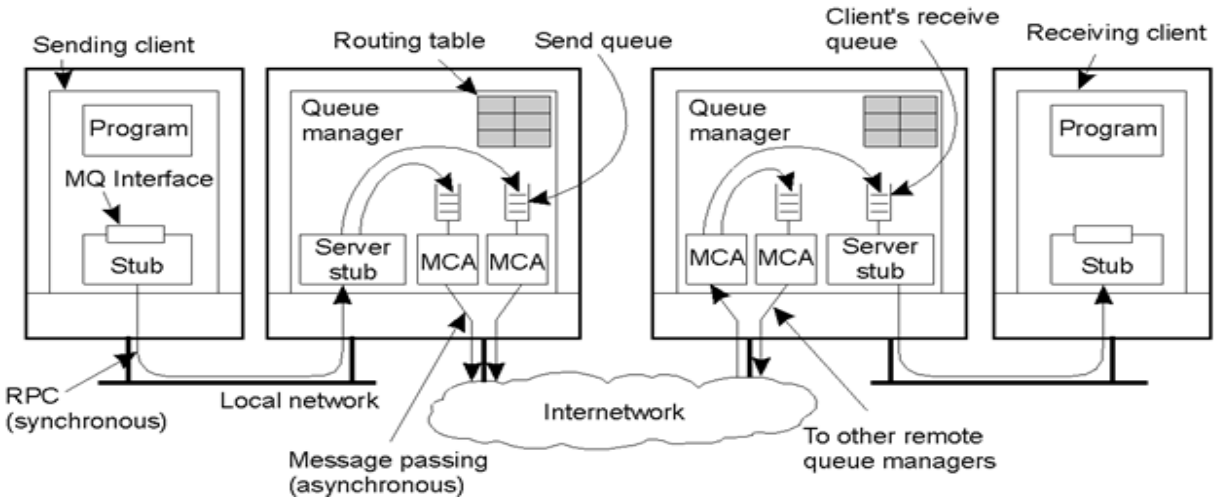
- In the case of a synchronous RPC the client sends the request to the server and blocks until the response is received from the server.

- In the case of the asynchronous RPC that has been explained in the problem statement, the client sends the request to the server and gets an acknowledgement from the server for receiving the request. This series of steps is part of the asynchronous RPC that has been initiated by the client and this RPC ends once the client processes the acknowledgement received from the server. The client now has the way of knowing that the server has received the request and started processing it. As per the situation explained in the problem statement the client blocks until it gets the response from the server. The server initiates another asynchronous RPC and sends the response to the client and blocks until it receives the acknowledgement for its response from the client.

Though both the procedures requires the client to block until it gets the response from the server, the difference is the transfer of the two acknowledgement messages, one for the request from the client and the other for the response from the server.

6. Explain how Message Oriented Middleware (MOM), such as IBM'sMQ-Series, uses message queues to provide high-level asynchronous communication.

IBM's MQ series is a message queueing system that provides the service of passing the messages from one system to the another in the network. The general architectural view of this message queueing system is shown in the figure below.



- Queue Manager is a middle-level message broker in this system that is responsible for routing the message sent by the client to the intended destination.
- The sending client initiates a synchronous RPC and sends the message to a queue manager in the local network. This message is loaded into the send queue in the queue manager. This message contains the information regarding the destination address and the actual data contents. The queue manager then processes the messages in the send queue and classifies them based upon the queue managers to which they need to be sent. This classification is done on the basis of the information in the routing table present in each queue manager.
- Message channels are unidirectional, reliable connections established between the queue managers. Each message is monitored by a message channel agent (MCA) present on the queue manager side. A queue manager initiates an asynchronous RPC and sends the message to the destination queue manager through the corresponding message channel.

- The receiving queue manager picks the received message from the receive queue and sends it to the intended destination. This last phase in the transmission of the message is initiated by a synchronous RPC.