CS454/654 Midterm Exam
Fall 2004

(3 November 2004)

Question 1: Distributed System Models (18 pts)

(a) [4 pts] Explain two benefits of middleware to distributed system programmers, providing an example for each benefit.

Solution: Some benefits are:

- Middleware can provide high-level abstractions that make it easier to develop distributed systems. These abstractions hide some of the details of the implementation of the system. For example, RPC hides marshaling and communication code behind a procedural interface to remote procedures.
- Middleware isolates the programmer from the operating system. Programs can be written to the middleware layer and can be (more easily) ported to other machines that support the same middleware (i.e., CORBA implementations are available on a variety of machines).
- Middleware can provide some forms of transparency to the programmer automatically. For example, middleware can handle the data representation problem, converting data in messages so they are appropriate for the architecture on which the receiving process is running.

(b) [4 pts] What is the difference between a distributed operating system and a network operating system?

Solution: (This is question 1.11 in your book.) A distributed operating system manages multiprocessors and homogeneous multicomputers. A network operating system connects different independent computers that each have their own operating system so that users can easily use the services available on each computer.

(c) [4 pts] Consider a distributed system consisting of four replicated servers. Each of the servers is available at any instant with a probability of 90%. If the system is designed so that the system can be operational if any one of the four servers is operational, what is the overall system availability? What if the system is designed such that all four servers have to be available for the entire system to be available?

Solution: The probability of one server being down is 0.1. The probability of all 4 servers being down simultaneously is $0.1^4 = 0.0001$ So the probability of at least one being available is $1 - 0.0001 = 0.9999$. On the other hand, the probability of all four serves being available at the same time is $0.9^4 = 0.6561$.

(d) [6 pts] What are three advantages of client-server systems? What are three advantages of peer-to-peer systems?

Solution: The advantages of client-server systems are the following:

1. More efficient division of labor between the client machines and the servers.
2. Horizontal and vertical scaling of resources are possible.
3. Better price/performance on client machines possible since they don't need to be as powerful.
4. Ability to use familiar tools on client machines since users can use their own machines as clients.
5. Clients can access remote data and resources.
6. Overall better system price/performance.

The advantages of peer-to-peer systems are the following:

1. Uniform functionality among machines since there is no separation of clients and servers.
2. Better scalability since servers would not be bottlenecks.
3. Less initial expense, since there is no need for a dedicated server.
4. Possibly better system availability since there are no centralized servers.

Question 2: Network Basics (15 pts)

(a) [6 pts] The two types of transport services that the Internet provides to its applications are TCP and UDP. Discuss the four main characteristics that separate these services.

Solution: The following are the more important principles characteristics that separate TCP from UDP:

1. In TCP, two end-systems first establish a connection (i.e., "handshake") before either starts to send application data to the other. In UDP, there is no "handshaking" (TCP is connection-oriented while UDP is connectionless).
2. TCP provides reliable data transfer, i.e., all application data sent by one side of the connection arrives at the other side of the connection in order and without any gaps. In UDP, there are no guarantees of reliable data transfer.
3. TCP provides flow control, i.e., it makes sure that neither end of a connection overwhelms the buffers in the other end of the connection by sending to many packets to fast. There is no flow control in UDP.
4. TCP provides congestion control, i.e., regulates the amount of data that an application can send into the network, helping to prevent the Internet from entering a state of grid lock. There is no congestion control in UDP.

(b) [3 pts] Suppose there is exactly one packet switch between a sending host and a receiving host. The transmission rates between the sending host and the switch is R_1 and the transmission rate between the switch and the receiving host is R_2. Assuming that the switch uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L? (Ignore queuing, propagation delay, and processing delay.)

Solution: This is a simplified version of the question in assignment 1 (question 3) since it involves only one switch rather than two as in the assignment. At time t_0 the sending host begins to transmit. At time $t_1 = L/R_1$, the sending host completes transmission and the entire packet is received at the router (no propagation delay). Because the router has the entire packet at time t_1, it can begin to transmit the packet to the receiving host at time t_1. At time $t_2 = t_1 + L/R_2$, the router completes transmission and the entire packet is received at the receiving host (again, no propagation delay). Thus, the end-to-end delay is $L/R_1 + L/R_2$

(c) [6 pts] Within a router, what are the sources of delay in a packet-switched network? Which of these delays are reduced or eliminated in a circuit-switched network, and why?

Solution: There are three sources of delay within a router:

Nodal processing. The router must check error bits and select an outgoing network link.

Queuing delay. The packet must wait for the outgoing link to be free.

Transmission delay. This is the time to get the bits onto the network link.

(Note: the fourth source of delay, *propagation delay*, is not within the router. It is the delay experienced by for the bits to reach the next router once the packet is on the network link). In a circuit-switched network, queuing delay is eliminated because the router has allocated bandwidth for the connection. Circuit-switching also reduces nodal processing since the outgoing link was selected when the circuit was set up.

Question 3: Distributed Invocation (25 pts)

(a) [4 pts] In message-based communication systems, what are the differences between persistent messaging and transient messaging?

Solution: There are two differences: (1) In transient messaging, both the sender and the receiver processes have to be running for the message to succeed while in persistent messaging the receiver does not have to be running when the sender sends the message and the sender does not need to be running when the receiver gets the message, and (2) in transient messaging, the sender puts the message on the net and if it cannot be delivered to the sender or to the next communication host, it is lost, while in persistent messaging the message is stored in the communication system as long as it takes to deliver the message to the receiver.

(b) [4 pts] Describe how connectionless communication between a client and a server proceeds when using sockets.

Solution: This is question 2-13 in your book. Both the client and the server create a socket, but only the server binds the socket to a local endpoint. The server can then subsequently do a blocking/non-blocking read call in which it waits for incoming data from any client. Likewise, after creating the socket, the client simply does a blocking/non-blocking call to write data to the server. There is no need to close connection.

(c) [8 pts] Describe the role of the interface definition language and stub compiler in the creation of stubs and skeletons in an RMI/RPC system.

Solution: The interface definition language provides a means of specifying the remote interface offered by the server or remote object. This includes method signature (name and arguments). This single definition is used to create both stubs and skeletons, so the programmer doesn't have to maintain two versions of the interface.
The signatures in the interface are used to construct the format of the messages that will be exchanged between the client and server. With the signature and a data representation scheme, the message layout can be determined.
For the purposes of dealing with heterogeneity, it is also possible for both client and server to determine which bytes in the message are associated with each parameter in the message. Thus, it is possible to determine which parts of the message may need to be converted. For example, we can

pick out the integers and convert them between big and little endian, without affecting other data. In multi-lingual systems, such as CORBA, the interface definition language is language-neutral, so it is possible to construct stubs and skeletons in various programming languages.

(d) [6 pts] In a system where the client communicates with the server over an RPC, the client keeps sending a request (say operation_x()) to the server until the server responds with the result of the request. What RPC failure semantics is being implemented in this case? Explain how the client and server has to be implemented to achieve this semantics.

Solution: For this to work, the RPC semantics is at-most-once – the client keeps trying until the request is executed at least once. For this semantics, the server has to do one of two things: (1) operation_x() may have an idempotent implementation so that multiple executions leave the server in the same state as a single executions, or (2) the server has to keep track of repeats of the request that can be implemented by the client attaching a sequence number to each request indicating whether they are repeats of earlier requests or new ones, and the server has to check these sequence numbers and not execute the request if it has already executed it before, instead returning the result from its buffer.

(e) [3 pts] Describe the differences between a client making one regular (synchronous) RPC call to a server and a client making one asynchronous RPC call to the server and the server responding (when it is done) with another asynchronous RPC call.

Solution:

- In synchronous RPC, there is one message from the client to the server and one message returned for a total of 2 messages. In asynchronous RPC, there are a total of 4 messages: client request, server ack, server response, client ack.
- In synchronous RPC, the client remains blocked until the server responds or the time out period expires. In asynchronous RPC, the client continues operation.

Question 4: Distributed Name Servers (22 pts)

(a) [4 pts] Two requirements of the Domain Name System (DNS) are (1) to scale to large numbers of computers, and (2) to allow local organizations to administer their own naming systems. How does DNS achieve these two goals?

Solution:

1. Hierarchical partitioning of the name database increases scalability. Each name server holds part of the naming database, exploiting the organization into zones. Queries regarding computers in the local zone are answered by servers within that zone without requiring other servers. This localizes searchers. Secondly, by caching at local DNS servers, a lot of searches for computers outside the local zone can also be satisfied without going outside.
2. The hierarchical partitioning of the name database is based on organization and geography. This allows local organizations to administer their own naming systems.

(b) [18 pts] Give the records that need to be in the DNS database for the zone *cs.uwaterloo.ca* according to the following description of the environment:

- The *cs.uwaterloo.ca* node represents the domain as well as the zone.

- There are three name servers in this : *goedel.cs.uwaterloo.ca* and *babbage.cs.uwaterloo.ca* as well as two mail servers: *turing.cs.uwaterloo.ca* (as primary) and *neumann.cs.uwaterloo.ca* (as secondary).
- *goedel.cs.uwaterloo.ca* is a Sun Unix machine. It runs its own mail server (i.e., mail sent to userid@goedel.cs.uwaterloo.ca is resolved by *goedel.cs.uwaterloo.ca*) as primary and uses *turing* as secondary mail server, and has two IP addresses 128.97.79.100 and 128.97.79.101.
- *babbage.cs.uwaterloo.ca* is also a Sun Unix machine and it uses two mail servers: *turing.cs.uwaterloo.ca* as the primary and *neumann.cs.uwaterloo.ca* as secondary (i.e., mail addressed to userid@babbage.cs.uwaterloo.ca can be resolved either by *turing* or *neumann*). Its IP address is 129.97.79.55.
- *turing* and *neumann* are both Sun Unix machines and they can receive mail addressed directly to them (i.e., *userid@turing.cs.uwaterloo.ca* and userid@neumann.cs.uwaterloo.ca are valid and need to be resolved). They have IP addresses 128.97.79.65 and 128.97.79.90, respectively.
- There is a Web server www.cs.uwaterloo.ca) and an ftp server (ftp.cs.uwaterloo.ca) that run on *turing*.

Fill in the following table with your answer. (Hint: for a full answer, you need 24 entries.)

Solution: Table 1 (on the next page) shows the records that are needed.

Question 5: Distributed File Systems (20 pts)

(a) [6 pts] What are three major differences between NFS Version 3 and Version 4?

Solution: The differences are the following:

- NFS version 3 servers are stateless, while NFS version 4 servers are stateful.
- NFS V3 has special operators to create and destroy non-regular files (e.g., symbolic links, directories) while NFS V4 considers them uniformly.
- NFS V3 performs name resolution iteratively while NFS V4 can perform it recursively.
- In NFS V3, name resolution cannot cross mount points, while this is possible in NFS V4.

(b) [3 pts] Suppose the current denial state of a file in NFS is WRITE. Is it possible that another client can first successfully open that file and then request a write lock? Explain your answer.

Solution: (This is question 10-6 in your book.) Yes. If the second client requires read/write access (i.e., BOTH) but no denial (i.e., NONE), it will have been granted access to the file. However, although a write lock may actually be granted, performing an actual write operation will fail. Remember that share reservation is completely independent from locking.

(c) [6 pts] Describe the callback promise mechanism that is used to ensure cache coherence in AFS.

Solution:

1. Each server, for each of its files, keeps a list of the clients that hold a copy of that file.
2. The server *promises* to call back all of these clients when their copies become invalid.
3. When a client modifies a copy that it has cached, it notifies the server, which, in turn, notifies all the clients on the list.

Name	Record type	Record Value
cs.uwaterloo.ca	SOA	(whatever you want to define a zone start)
cs.uwaterloo.ca	NS	goedel.cs.uwaterloo.ca
cs.uwaterloo.ca	NS	babbage.cs.uwaterloo.ca
cs.uwaterloo.ca	MX	1 turing.cs.uwaterloo.ca
cs.uwaterloo.ca	MX	2 neumann.cs.uwaterloo.ca
goedel.cs.uwaterloo.ca	HINFO	Sun Unix
goedel.cs.uwaterloo.ca	MX	1 goedel.cs.uwaterloo.ca
goedel.cs.uwaterloo.ca	MX	2 turing.cs.uwaterloo.ca
goedel.cs.uwaterloo.ca	A	128.97.79.100
goedel.cs.uwaterloo.ca	A	128.97.79.101
babbage.cs.uwaterloo.ca	HINFO	Sun Unix
babbage.cs.uwaterloo.ca	MX	1 turing.cs.uwaterloo.ca
babbage.cs.uwaterloo.ca	MX	2 neumann.cs.uwaterloo.ca
babbage.cs.uwaterloo.ca	A	128.97.79.55
turing.cs.uwaterloo.ca	HINFO	Sun Unix
turing.cs.uwaterloo.ca	MX	1 turing.cs.uwaterloo.ca
turing.cs.uwaterloo.ca	MX	2 neumann.cs.uwaterloo.ca
turing.cs.uwaterloo.ca	A	128.97.79.65
neumann.cs.uwaterloo.ca	HINFO	Sun Unix
neumann.cs.uwaterloo.ca	MX	1 turing.cs.uwaterloo.ca
neumann.cs.uwaterloo.ca	MX	2 neumann.cs.uwaterloo.ca
neumann.cs.uwaterloo.ca	A	128.97.79.90
www.cs.uwaterloo.ca	CNAME	turing.cs.uwaterloo.ca
ftp.cs.uwaterloo.ca	CNAME	turing.cs.uwaterloo.ca

Table 1: Answer to question 4c - DNS Records

4. After such an invalidation message (callback message) recipient clients are removed from the server's list.
5. When a client opens a file, it checks with the server to see if the file is still valid; if it is valid, there is no need to transfer the file again.

(d) [5 pts] Two students of mine with user id's joe and jill want to mount my shared papers directory that is on an NFS server so we can all have access to the same text as we work on them. joe wants to mount it at ˜/shared/pap and jill is going to mount it at ˜/mywork/publications/profs/papers. The students are under /usr/grads/ and my directory is under /fsys2/export/ozsu. Show the mount trees schematically.

Solution: See Figure 1.

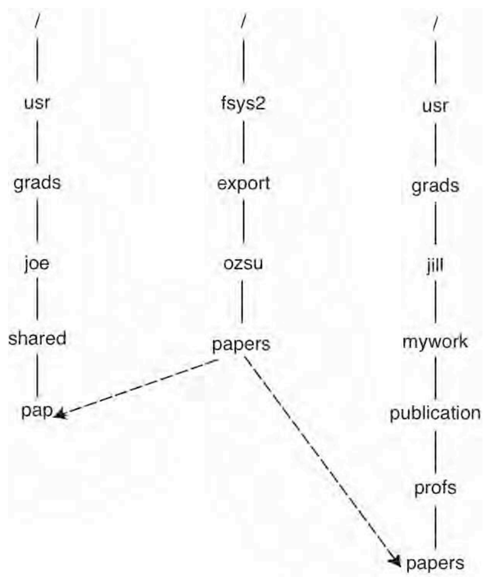

Figure 1: Answer to Question 5(d)