Midterm, questions and answers

Distributed Systems (University of Waterloo)

Question 1: Distributed System Models (20 pts)

(a) [6 pts] Explain two benefits of middleware to distributed systems 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] 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 servers being available at the same time is $0.9^4 = 0.6561$.

(c) [6 pts] Why is it not always a good idea to aim at implementing the highest degree of distribution transparency possible? Provide an example.

Solution: This is question 1.6 from Tanenbaum.

Aiming at the highest degree of transparency may lead to a considerable loss of performance that users are not willing to accept. For example, a distributed file system can cache files at the cost of relaxing the consistency model and thus losing some transparency.

(d) [4 pts] Explain two differences between active standby and passive standby systems.

Solution: First, in active standby systems the backup is executing the same requests as the primary, possibly in lock step. In the passive standby, the backup is inactive until the primary fails (except for receiving periodic updates from the primary).

Second, in an active standby system, the backup can take over immediately when the primary fails. The backup in a passive standby system may have to catch up on any state changes since its last update before it can take over for a failed primary.

Question 2: Network Basics (10 pts)

(a) [5 pts] If a client and a server are placed far apart, we may see network latency dominating overall performance. Explain why increasing the network bandwidth between the client and the server does not address this problem.

Solution: Network latency is the time that it takes a network packet to travel from the client to the server, which depends on their physical distance (and the routing), but is independent of the network bandwidth.

(b) [5 pts] Describe a possible solution to work around the latency problem. Relocating the client or the server is not an option.

Solution: This is Question 2.1 from Tanenbaum.

It really depends on how the client is organized. It may be possible to divide the client-side code into smaller parts that can run separately. In that case, when one part is waiting for the server to respond, we can schedule another part. Alternatively, we may be able to rearrange the client so that it can do other work after having sent a request to the server. This last solution effectively replaces the synchronous client-server communication with asynchronous one-way communication.

Question 3: Distributed Invocation (25 pts)

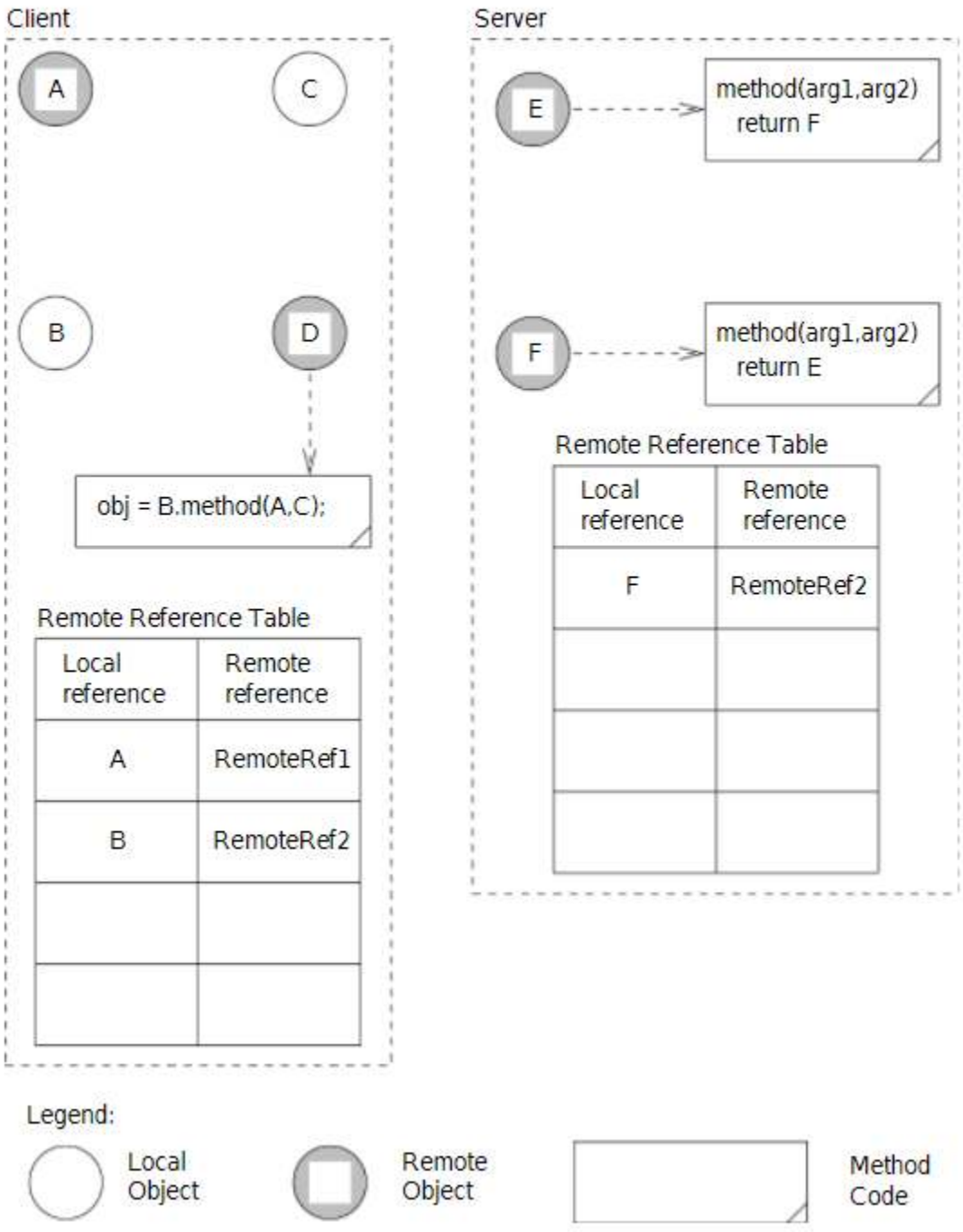
(a) [6 pts] What are the issues that you need to solve to marshal/unmarshall the following data types between different platforms (be brief)?

1. integers
2. characters
3. pointers or references to objects

Solution:

1. The big endian/little endian issue. There must be an agreement on the specific format.
2. How to encode characters, e.g., do you use ASCII or EBCDIC?
3. You cannot transfer pointers across machines. Since this comes up mainly during call- by-reference, you can emulate this call method by the copy/restore approach.

(b) [9 marks] Consider the following state of a client and server in a remote method invocation system. Shaded circles represent remote objects (those that implement a remote interface), unshaded circles are local objects, and the boxes represent code associated with the class.



For the method invocation in object D at the client side, show the following:

1. The contents of the marshaled request message, labeling each part.
2. The contents of the marshaled reply message.
3. Update the above figure, adding any additional objects and table entries that would have to be present after the completion of the method call. Clearly label the roles of any new objects (copies of existing objects, proxies, *etc.*).

Solution: The request message has to include three fields: the receiver object, the method identifier, and the method arguments. For this request, the message thus contains

RemoteRef2, identifier for `method()`, RemoteRef1, Copy of C

where RemoteRef2 is the remote object reference for proxy B (or implementation object F on the server) and RemoteRef1 is the remote object reference for remote object A. At the server end, a proxy for RemoteRef1 must be created (object G) since this is the first time the server has seen this remote reference. In addition, C' (a copy of object C) is created at the server for the parameter C from the client, since it is local to the client.

This call invokes the code in object F at the server (since the remote object reference for B on the client is identical to that for F on the server). This means that a reference to E is returned. E is a remote object but does not yet have a remote object reference, so one is created (RemoteRef3) and that remote object reference is returned in the reply message. At the client side, a proxy for this returned reference, object H, must be created since the reference does not exist in the remote reference table.

Figure 1 shows the new state of the system. Additions are indicated with italicized font. The role of each object is indicated above.

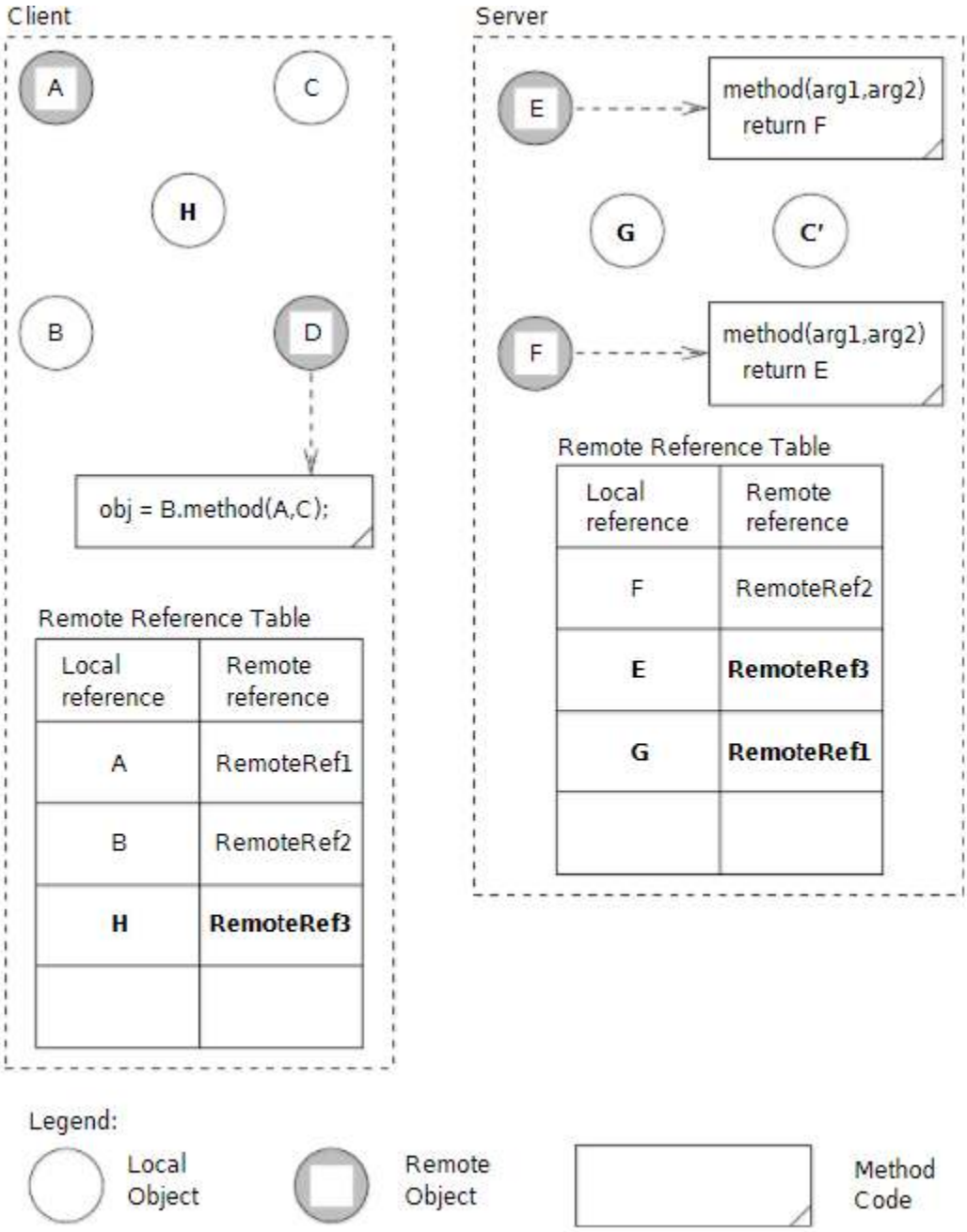


Figure 1: The new state of the RMI system.

(c) [6 pts] Java and other languages support exceptions, which are raised when an error occurs. How would you implement exceptions in remote calls?

Solution: This is Question 2.10 from Tanenbaum.

Because exceptions are initially raised at the server side, the server stub can do nothing else but catch the exception and marshal it as a special error response back to the client. The client stub, on the other hand, will have to unmarshal the message and raise the same exception if it wants to keep access to the server transparent.

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

Solution:

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
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.

Question 4: Distributed Naming (25 pts)

(a) [9 pts] The following shows the (shortened) response of the DNS root name server 198.41.0.4 when queried for the *A* resource record for `www.cs.uwaterloo.ca`, as output by the "dig @198.41.0.4 www.cs.uwaterloo.ca" command.

```
[..]
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 7, ADDITIONAL: 8

;; QUESTION SECTION:
;www.cs.uwaterloo.ca.           IN      A

;; AUTHORITY SECTION:
ca.                     172800  IN      NS      CA05.CIRA.ca.
ca.                     172800  IN      NS      NS-EXT.ISC.ORG.
ca.                     172800  IN      NS      CA03.CIRA.ca.
[..]

;; ADDITIONAL SECTION:
CA05.CIRA.ca.           172800  IN      A       129.33.164.84
NS-EXT.ISC.ORG.         172800  IN      A       204.152.184.64
CA03.CIRA.ca.           172800  IN      A       192.228.25.45
[..]
```

For each of the sections that can appear in a DNS response, discuss **what kind** of information and **why** the root name server includes this information in the response given above.

Solution:

A DNS response can include an answer section, an authority section, and an additional information section. It does not include a question section, which is given only as a comment by the dig program.

Answer section. This section is empty. The root name server doesn't have the *A* record for `www.cs.uwaterloo.ca`.

Authority section. This section includes a list of authoritative name servers for the `.ca` domain. The client should contact one of them to resolve its query.

Additional information section. This section lists the IP addresses of the above name servers. The root name server includes this information since the client will require it to contact one of the name servers.

(b) [4 pts] Taking full benefit of the response given above, show how the "dig" program should be invoked next to continue the resolution of `www.cs.uwaterloo.ca`.

Solution: One of "dig @129.33.164.84 www.cs.uwaterloo.ca", "dig @204.152.184.64 www.cs.uwaterloo.ca", or "dig @192.228.25.45 www.cs.uwaterloo.ca".

(c) [6 pts] On February 6, 2007, a denial of service attack was launched against the 13 root DNS servers. The attack managed to disable at least three of the 13, yet there was relatively little impact on the Internet at large. Give and explain two reasons why DNS was able to withstand this assault.

Solution: The primary reasons are:

1. Replication, clearly. The remaining 10 DNS servers were able to handle the remaining load.
2. Caching, which can improve availability. The attack was launched against the root servers. Clients and local name servers can cache these DNS entries for a long amount of time before they expire. Many clients were likely reusing their cached entries rather than contacting the root servers.

(d) [6 pts] Describe the two benefits of recursive name resolution. Considering that DNS root name servers do not provide recursive name resolution, is it still possible to achieve some of these benefits?

Solution: First, it can cut the cost of communicating with name servers to resolve the name. Recursive resolution can take advantage of the fact that, in many cases, name servers within a subdomain of a given server are geographically close together. If the name is on a distant name server, the client may only have to send one long-distance message to a name server, and the recursive resolution of that name may take place over a much smaller geographical area. Second, it provides more opportunities for caching. Intermediate name servers can respond not just with the resolution of the name, but also with addresses of any other intermediate name server involved in the resolution process. Each name server along the resolution path can cache this information. As well, this caching takes place at the name servers. As a result, all clients benefit from this caching, not just the client issuing the resolution request.

Lower-level name servers might be willing to provide recursive name resolution so they might give us these benefits. Also, by directing all resolution requests to a local name server, we might still benefit from caching.

Question 5: Distributed File Systems (20 pts)

(a) [6 pts] In NFSv3, give one reason why name translations are done in an iterative manner for each part of the name. Explain.

Solution: Here are several reasons why pathnames are looked up one part at a time

1. pathnames may cross mount points,
2. pathnames may contain symbolic links, and
3. pathnames may contain '..'.

Case 1 requires reference to the client mount tables, and a change in the server to which subsequent lookups are dispatched. For case 2, symbolic links are interpreted in accordance with the client's view of the file tree. Case 3 requires a check in the client against the 'pseudo-root' of the client process making the request to make sure that the path doesn't go above the pseudo-root.

(b) [8 pts] Give and briefly explain the consistency semantics of UNIX, NFSv3, and AFSv2. Furthermore, for both NFSv3 and AFSv2, describe under what circumstances clients might notice that the semantics are different from the UNIX case.

Solution:

- UNIX: one-copy update semantics/UNIX semantics, read sees all previous writes;
- NFSv3: approximation to one-copy update semantics/UNIX semantics with a delay (about 3 seconds) in achieving consistency;
- AFSv2: session semantics, consistency is achieved only when closing a file.

For NFSv3, after a write operation, the corresponding data cached in clients other than the one performing the write is invalid (since it does not reflect the current contents of the file on the NFS server), but the other clients will not discover the discrepancy and may use the stale data for a period of up to 3 seconds (the time for which cached blocks are assumed to be fresh).

For AFSv2, the difference between AFS and UNIX is much more visible. Lost updates will occur whenever two processes at different clients have a file open for writing concurrently.

(c) [6 pts] Between AFSv2 and NFSv3, which one is more appropriate for a wide area network? State your reasons.

Solution: Of these two, AFSv2 would be more appropriate. The primary reason is that the number of messages between client and server in AFS is much less than the number for NFSv3 (for the same workload).

The number of messages from client to server is reduced in AFS. AFS clients only contact servers for `open` and `close` calls, where NFS clients constantly send messages to the server to verify the consistency of files and directories in their cache.

Unlike NFS, AFS does have messages from servers to client. In NFS, since the servers are stateless, these messages are not possible. In AFS, the callback messages are sent from servers to clients. However, these messages are small invalidation messages. Further, these invalidations only occur when something interesting has happened (*i.e.*, the file has changed).