Indian Institute of Information Technology Allahabad

Mid-semester Exam (February 2019)

Sixth semester B.Tech (IT& ECE)

Course Name	Course Code	Date of Exam	MM	Time
Distributed Systems	IDSS630E	Feb. 27, 2019	30	2 Hrs

***Important Instructions**: All questions are compulsory and preferably to be answered in strict order as is given in the question paper.*

1. **(5 marks) (*Multiple choice*)**:

   (a) An RPC (remote procedure call) is initiated by the:

   ```
   (i)    server     (ii) client
   (iii)   both     (iv)  neither
   ```

   Solution : (ii)

   (b) Location transparency allows: I.Users to treat the data as if it is done at one location. II. Programmers to treat the data as if it is at one location. III. Managers to treat the data as if it is at one location. Which one of the following is correct?

   ```
   (i)    I,II and III     (ii) I and II only
   (iii)  II and III only  (iv) II only
   ```

   Solution : (i)

   ```
   Explanation : In distrubuted computer networks, location transparency
   is the use of names to identify network resources,
   rather than their actual location. For example, files are accessed
   by a unique file name, but the actual data is stored in
   physical sectors scattered around a disk in either the
   local computer or in a network
   ```

   (c) Which event is concurrent with the vector timestamp (2, 4, 6) ?

   ```
   (i)    (3,5,7)     (ii) (1,3,5)
   (iii)  (1,4,6)     (iv) (1,4,7)
   ```

   Solution : (iv)

   (d) A client has a time of 5:05 and a server has a time of 5:25. Using the Berkeley algorithm, the client's clock will be set to:

   ```
   (i)    5:15     (ii) 5:20
   (iii)  5:25     (iv) 5:30
   ```

   Solution : (i)
   Explanation : (5:05+5:25)/2

   (e) The global state recording part of a single instance of the Chandy-Lamport algorithm requires —– messages and — time, where e is the number of edges in the network and d is the diameter of the network

   ```
   (i)   O(loge), O(d)   (ii) O(e), O(logd)
   (iii) O(e), O(2d)     (iv) O(e), O(d)
   ```

   Solution : (iv)

2. **(2+2+2+2= 8 marks) (*Clinet-Server Architecture*)**:

(a) Consider a chain of processes P_1,P_2.....P_n implementing a multitiered client-server architecture. Process P_i is client of process P_{i+1} and P_i will return a reply to P_{i-1} only after receiving a reply from P_{i+1}. What are the main problems with this organization when taking a look at the request-reply performance at process P_1

Solution :

Performance can be expected to be bad for large n. The problem is that each communication between two successive layers is, in principle, between two different machines. Consequently, the performance between P1 and P2 may also be determined by n - 2 request-reply interactions between the other layers. Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the performance at the highest level}

(b) What are the roles of the client and the server in a client/server application model? How does this differ from a peer-to-peer application model?
Solution :

In a client-server model :
Servers are processes that offer services;
Clients are processes that request for services
Clients follow request/reply model with respect to using services

Peer-to-peer model :
All nodes equally participate in data sharing.
All tasks are equally divided among all nodes.
Nodes are organized using a distributed data structure or have randomly selected neighbours
None of the nodes in the peer to peer network are dependent on the others for their functioning.
Each computer in the peer to peer network manages itself.
So, the network is quite easy to set up and maintain

(c) Where would you place the components in a three-tiered client-server architecture if you choose (i) the vertical distribution style, (ii) the horizontal distribution style
Solution :

Vertical distribution refers to the distribution of the different layers in a multitiered architectures across multiple machines. In principle, each layer is implemented on a different machine. Horizontal distribution deals with the distribution of a single layer across multiple machines, such as distributing a single database.

(d) You are asked to explain the following architecture in which a service using several application/compute servers serves client requests
First tier : Client requests; A logical switch (possibly multiple) dispatches the request to multiple compute servers
Second Tier : Application / Compute servers
Third tier : Set of distributed file systems or database. Each compute server in the second tier is attached to a *unique* distributed file or database of the third tier
How can this architecture improve the performance of the service?

3. **(1+2+2 = 5 marks) (*Middleware : Remote Procedure Call*)**:

(a) Why do we need RPC?
Solution :

Remote Procedure Call (RPC) is a protocol that one program can use
to request a service from a program located in another computer in a
network without having to understand network details. (A procedure call
is also sometimes known as a function call or a subroutine call.)
RPC uses the client/server model. The requesting program is a client
and the service-providing program is the server. Like a regular or
local procedure call, an RPC is a synchronous operation requiring
the requesting program to be suspended until the results of the
remote procedure are returned. However, the use of lightweight processes
or threads that share the same address space allows multiple RPCs
to be performed concurrently.

(b) What is the purpose of stub?
Solution :

When program statements that use RPC are compiled into an executable program, a stub is

There are several RPC models and implementations. A popular model
and implementation is the Open Software Foundation's Distributed Computing
Environment (DCE). The Institute of Electrical and Electronics Engineers defines RPC in
ISO/IEC, November 1991.

RPC spans the Transport layer and the Application layer in the Open Systems Interconne
to develop an application that includes multiple programs distributed in a network.

Alternative methods for client/server communication include message
queueing and IBM's Advanced Program-to-Program Communication (APPC).

(c) Assume a client calls an asynchronous RPC to a server, and subsequently waits until the server returns a result using another asynchronous RPC. Is this approach the same as letting the client execute a normal RPC?

Solution :

No, this is not the same. An asynchronous RPC returns an acknowledgement
to the caller, meaning that after the first call by the client,
an additional message is sent across the network. Likewise, the server
is acknowledged that its response has been delivered to the client.
Two asynchronous RPCs may be the same, provided reliable communication
is guaranteed. This is generally not the case.

4. **(3+2+2 = 7 marks)** (***Clock Synchronization***):

(a) What problem of Lamport clock does vector clocks solve?
Solution :

Lamport clocks can guarantee that if a < b then C(a) < C(b).
However it can't guarantee, that if C(a) < C(b)
then event a happened before b

Vector Clock :
In a system with N processes, each process keeps a vector timestamp TS[N]

```
1. In Process i,

a. TS[j] is logical time of process j as process i knows about it.
b. TS[i] is the lamport clock of process i.

2. When a new event is generated, TS[i] = TS[i] + 1
3. When sending a message we copy the vector clock to the message
4. On recipient of message with vector timestamp MTS we set
the TS of process i as,
  TS[k] = max(TS[k], MTS[k]) for k = 1 to N

The happened before ordering in vector clocks is defined as follows.

 e1 < e2 iff TS(e1)[k] <= TS(e2)[k] for k = 1 to N
    * atleast one of the value of indices of e1 should be less than e2's value

The timestamp of an event would tell us what all events among
all processes would have influenced the generation of that
particular event. So if e1 < e2, then e2 would have witnessed
all the events that has been witnessed by e1.
With vector clocks we can ascertain if TS(e1) < TS(e2) then e1 < e2
```

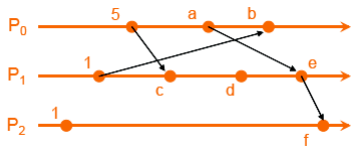
(b) t_i and t_j are time-stamps of event e_i and e_j. If $t_i < t_j$ when vector clocks are used, show that t_i must be $< t_j$ when logical clocks are used

Solution:

```
 When vector clocks are used, if ti and tj are time-stamps of
 events ei and ej respectively, then
ti < tj only if (i) ti and tj are time-stamps of the send and receive events of a
message, respectively, or (ii) by rules of transitiveness of precedence. Under these
conditions, ti < tj also when logical clocks are used.
```

(c) Assign Lamport's time stamp to the events shown in the figure below :

Event	Logical Clock value
a	
b	
c	
d	
e	
f	



Solution :

Event	Logical Clock value
a	6
b	7
c	6
d	7
e	8
f	9

5. **(2+3=5 marks) (*Global Snapshot Problem and Termination Detection*)**:

   (a) A transit-less state of a system is a state in which no messages are in transit. Give an example of a system in which all states recorded by the Chandy-Lamport's algorithm are necessarily transit-less.

   Solution :

   State of a channel Chij is recorded as empty if Pi records its state and sends a marker to Pj, and Pj records its state on receiving the marker. If Pj has already received a marker along some other channel, it might record the state of Chi j as being non-empty. Hence a system in which each process has exactly one incoming channel will have a transit-less state. A system with a star topology with edges pointing outwards is a good example

   (b) Provide an alogithm to determine whether a distributed computation has terminated.

   Solution :

   System Model :
   A process may either be in active or inactive state.
   An idle process becomes active upon receiving a computation message.
   If all process idle => computation terminated.

   Huang's Termination Detection Protocol:

   The goal of this protocol is to detect when a distributed computation terminates.
   n processes
   Pi process; without loss of generality, let P0 be the controlling agent
   Wi. weight of process Pi; initially, W0 = 1 and Wi = 0 for all other i.
   B(W) computation message with assigned weight W
   C(W) control message sent from process to controlling agent with assigned weight W

   Protocol :
   an active process Pi sends a computation message to Pj

```
    Set Wi' and Wij to values such that Wi' + Wij = Wi,
    Wi' > 0, Wij > 0. (Wi' is the new weight of Pi.)
    Send B(Wij) to Pj

Pj receives a computation message B(Wij) from Pi

    Wj = Wj + Wij
    If Pj is idle, Pj becomes active

Pi becomes idle by:

    Send C(Wi) to P0 ( or to another Process )
    Wi = 0
    Pi becomes idle

Pi receives a control message C(W):

    Wi = Wi + W
    If Wi = 1, the computation has completed.

Example : \\
Say a process P0 is the controlling agent, with W0 = 1. It asks P1 and P2
to do some computation. It sets

    W01 = 0.2
    W02 = 0.3
    W0 = 0.5

P2 in turn asks P3 and P4 to do some computations. It sets

    W23 = 0.1
    W24 = 0.1

When P3 terminates, it sends C(W3) = C(0.1) to P2, which changes W2 to 0.1 + 0.1 = 0.2.

When P2 terminates, it sends C(W2) = C(0.2) to P0, which changes W0 to 0.5 + 0.2 = 0.7.

When P4 terminates, it sends C(W4) = C(0.1) to P0, which changes W0 to 0.7 + 0.1 = 0.8.

When P1 terminates, it sends C(W1) = C(0.2) to P0, which changes W0 to 0.8 + 0.2 = 1.

P0 thereupon concludes that the computation is finished.

Total number of messages passed: 8 (one to start each computation,
one to return the weight).
```