CDS-yz709-sup3

Q1 Definitions
Q2 NFS
Q3 Time
Q4 NTP
Q5 Vector clocks

Q1 Definitions

Briefly define each of these terms, and give an example of its use.

- (a) idempotent: operations that be applied multiple times without changing the result beyond the initial application
    - Used with retry to ensure exactly-one semantics when sending request to a database, so that we don't have to do deduplication
- (b) location transparency: the ability to access objects without the knowledge of their location
    - Client contacts directory service to locate server, so the servers are identified by 128 bit unique identifiers (UUIDs) rather than their actual location, hence a better security
- (c) marshalling: encoding some message so that third party will not be able to understand the message contents
    - Remote procedure call for online payments. The RPC client marshals the function arguments and transfers the message to the server. The server unmarshals the arguments and calls the exact function implementation to provide payment service.
- (d) pure names: one that contains no information about the object or where that object may be found
    - A user ID is a pure name
- (e) impure names: one that contains information about the underlying object encoded within the name
    - The IP domain address, https://www.cl.cam.ac.uk/
    - The email address ending as @cam.ac.uk

- (f) batching: jobs with similar requirements are grouped together and run through the computer as a set, the computer would process groups of jobs in a sequential order
    - Processing of input and output in the operating system, because there might be a group of operations that would like to do inputs and a group of other operations that would like to do outputs
- (g) middleware: a conduit between clients and servers, it enables communication and data management for distributed applications
    - Middleware helps application authors write software intended to run on more than one machine at a time, because middleware provides an interface that abstracts and hides the different implementations of different machines
- (h) clock skew: the difference between two clocks at a point in time
    - Clock skew is used in network time protocol to estimate the difference between clocks over a network (i.e., the difference between the clock on the client and the clock on a server)
- (i) clock drift: the relative difference in clock frequency or rate, when one clock de-sync with another one
    - Quartz clocks can be used to measure the impact of temperature

> 💡 Comments:

Q2 NFS

Write a 300-word description of NFS, how it works, when you might use it and why. Compare and contrast NFS versions 2 and 3.

- NFS is a network file system on a distributed system, enables local users to access remote data and files in the same way they accessed locally.
- All versions of NFS use TCP/IP stack and use User Datagram Protocol (UDP) running over an IP network to provide a stateless network connection between the client and server. The NFS uses a remote procedure call layer (RPC), where the NFS client makes RPC calls to an NFS server.

- The NFS is implemented in a client/server computing model, where an NFS server manages the authentication, authorisation, and management of clients and all the data shared within a specific file system. Once authorised, clients can view and access the data through their local systems like they would access it from an internal disk drive.
- NFS version 3 has more features than NFS version 2, including 64-bit file handles, safe async writes and more robust error handling. When mounting a file system via NFS, Linus uses NFS version 3 by default if the server supports it.

> 💡 Comments: I didn't find anything about NFS in the lecture notes, but when I am browsing 2019-2020 lecture notes, the concept of NFS is mentioned there.

Q3 Time

For each of the following potential uses of time, for which is a local oscillator sufficient, for which is synchronised real-time across a distributed system sufficient (e.g., UTC as distributed using NTP), and for which is a logical or vector clock mechanism most appropriate? Explain your answers.

- (a) local process scheduling:
  - A local oscillator is sufficient for local process scheduling.
  - Process scheduling happens in the operating system using the process scheduler, so it has no interactions with peripherals; we only need to use a system clock, which reports the time elapsed since a start point from the system boot up.
- (b) local I/O:
  - Local oscillator is sufficient for local I/O.
  - I/O devices are pieces of hardware used by a human or another system to communicate with a computer; they receive data packets or send data packets (i.e., keyboards, printers, headphones and touch screens).
  - We could use the local oscillator system clock to manage the data packets' order and use FIFO buffers to store data packets. There is no need to

synchronise the time between computers and the I/O devices because the order of events and data packets matters.

- (c) distributed filesystem consistency:
  - A logical or vector clock mechanism is sufficient.
  - We need to update, store and delete data on a distributed filesystem so that the order of events is important to ensure the consistency of the data stored. The only logical clock can help ensure causality and the correct order of events.
- (d) cryptographic certificate/ticket validity checking:
  - Synchronised real-time across a distributed system (UTC as distributed using NTP) is more appropriate.
  - Each ticket will have a valid period, and we usually have distributed servers that check the validity of the tickets; hence need to make sure the server clocks are synchronised. The order of the tickets doesn't matter because they are independent of each other.
- (e) tracing causal links in a distributed application over many nodes:
  - A logical or vector clock mechanism is more appropriate.
  - For a distributed application over many nodes, the local oscillator cannot synchronise between nodes. Logical clocks are consistent with causal dependencies while physical clocks are not; we have to maintain the causal dependency order since we want to trace causal links.

Comments:

Q4 NTP

NTP is a simple protocol that is foundational to other distributed systems protocols (e.g. the Kerberos authentication protocol will not work if two machines' clocks are too far apart, so they must be synchronised first).

Write a Java program which acts as an NTP client and prints the current time, as estimated from the NTP server, to the console whenever it is run. For example:

```
bash$ java -jar current-time.jar
Fri Feb 21 11:01:23 GMT 2014
bash$
```

You will need to send UDP packets in an appropriate format to an NTP server. The Computer Lab has a set of NTP servers which you may wish to use:

```
ntp0.cl.cam.ac.uk
ntp1[bc].cl.cam.ac.uk
```

Please make sure your packets conform to NTP version 3 or later (see RFC 1305 or later) and that you don't send more than two packets to the server each time you run your program.

```
package Concurrent_n_Distributed_Systems;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;

public class CurrentTime {
    public static void main(String[] args) throws UnknownHostException, IOException {
        if (args.length != 1) {
            System.out.println("usage: java CurrentTime <systemname>");
            System.exit(1);
        }

        String machine = args[0];
        final int port = 123;
        Socket s = new Socket(machine, port);
        BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream
()));
        String timestamp = br.readLine();
        System.out.println(machine + ": current time is: " + timestamp);
    }
}

// output: connection is refused
```

Comments:

Q5 Vector clocks

Vector clocks are a technique for determining a causal ordering of events in a distributed system. They can also help to prevent certain (undesirable) orderings from occurring by holding back events. This question asks you to apply the general algorithm and also compare vector clocks with Lamport clocks. Note that this question refers to the form of vector clocks presented in lecture, in which timestamps are updated on both message send and message receive.

(a) Given an initial state and the sequence of messages below, show the value of the Lamport and vector clocks at each node (A, B, C, D) at each send or receive event it participates in.

Event	Lamport				Vector			
	A	B	C	D	A	B	C	D
initial								
A → B			-	-			-	-
A → C		-		-		-		-
A → D		-	-			-	-	
B → D	-		-		-		-	
B → C	-			-	-			-
C → D	-	-			-	-		
C → A		-		-		-		-
A → B			-	-			-	-
C → B	-			-	-			-
final								

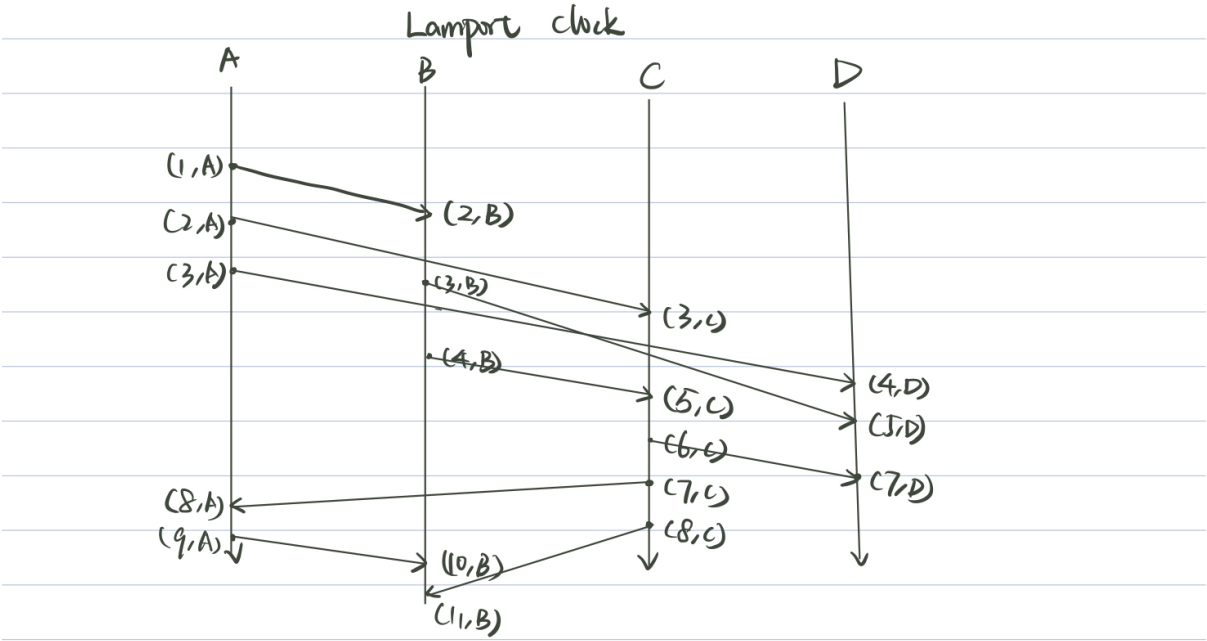
Event	Lamport				Vector			
	A	B	C	D	A	B	C	D
initial	0	0	0	0	<0,0,0,0>	<0,0,0,0>	<0,0,0,0>	<0,0,0,0>
A → B	1	2	-	-	<1,0,0,0>	<1,1,0,0>	-	-
A → C	2	-	3	-	<2,0,0,0>	-	<2,0,1,0>	-
A → D	3	-	-	4	<3,0,0,0>	-	-	<3,0,0,1>
B → D	-	3	-	5	-	<1,2,0,0>	-	<3,2,0,2>
B → C	-	4	5	-	-	<1,3,0,0>	<2,3,2,0>	-
C → D	-	-	6	7	-	-	<2,3,3,0>	<3,3,3,3>
C → A	8	-	7	-	<4,3,4,0>	-	<2,3,4,0>	-
A → B	9	10	-	-	<5,3,4,0>	<5,4,4,0>	-	-
C → B	-	11	8	-	-	<5,5,5,0>	<2,3,5,0>	-
final	9	11	8	7	<5,3,4,0>	<5,5,5,0>	<2,3,5,0>	<3,3,3,3>

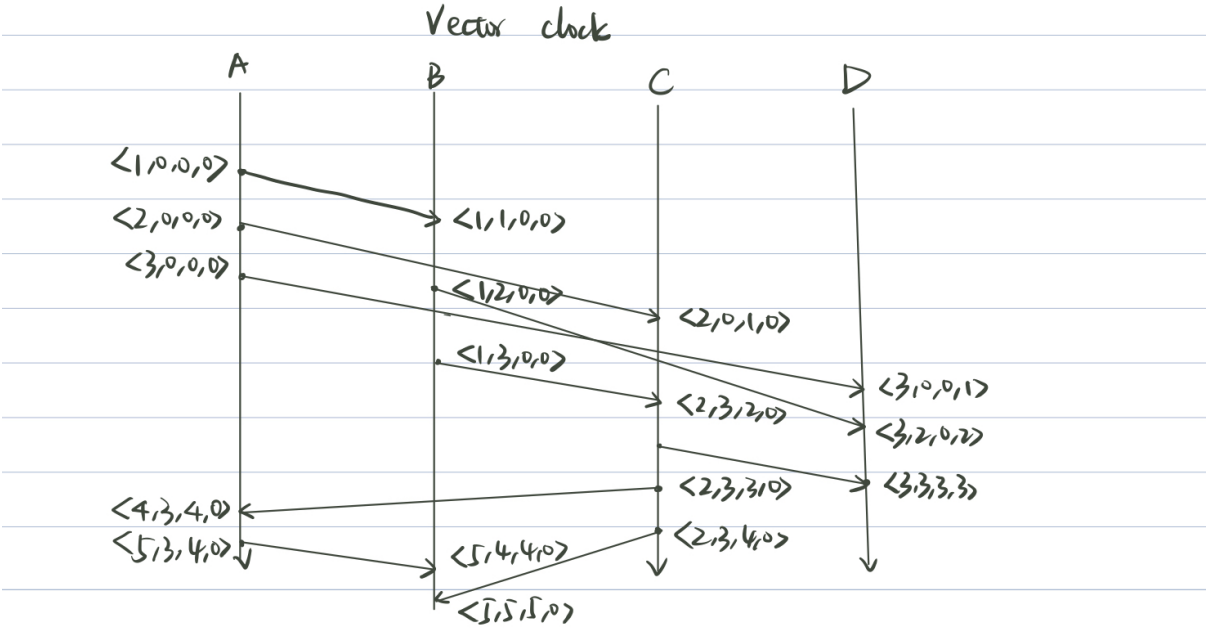
(b) Using the Lamport and vector clocks calculated above, state whether or not the following

events can be determined to have a *happens-before* relationship.

Events		Lamport	Vector
A → C sent	A → D sent		
A → D sent	B → C sent		
B → C sent	C → B sent		

- (A → C) sent and (A → D) sent:
  - Lamport: Yes, and events have a happens-before relationship because the event $(3, A) > (2, A)$ on the same process A
  - Vector: Yes, and events have a happens-before relationship because $< 2, 0, 0, 0 >< < 1, 0, 0, 0 >$
- (A → D) sent and (B → C) sent:
  - Lamport: No, because these two events might happen concurrently $(4, B)||(3, A)$ or might happen as $(4, B) > (3, A)$, we cannot decide based on the time $4 > 3$.
  - Vector: Yes, and we can deduce these events happen concurrently because their timestamps are comparable $T||T' \iff V(a)||V(b) \iff (a||b)$
- (B → C) sent and (C → B) sent:
  - Lamport: Yes, because events $(8, C) > (5, C)$, according to the transition relationship, we can deduce that two events have a happens-before relationship.
  - Vector: Yes, and events have a happens-before relationship because $< 2, 3, 4, 0 >< < 2, 3, 2, 0 >$ and according to the transition relationship, we can deduce happen-before relation.


Lamport clock
A
B
C
D
(1,A)
(2,B)
(2,A)
(3,C)
(3,A)
(3,B)
(4,D)
(4,B)
(5,C)
(5,D)
(6,C)
(7,D)
(7,C)
(8,A)
(8,C)
(9,A)
(10,B)
(11,B)


Vector clock
A
B
C
D
<1,0,0,0>
<2,0,0,0>
<3,0,0,0>
<1,1,0,0>
<1,2,0,0>
<2,0,1,0>
<1,3,0,0>
<3,0,0,1>
<2,3,2,0>
<3,2,0,2>
<2,3,3,0>
<3,3,3,3>
<4,3,4,0>
<5,3,4,0>
<5,4,4,0>
<2,3,4,0>
<5,5,5,0>

💡 Comment: