CSCI/ECEN 5673: Distributed Systems
Spring 2022

Homework 1

Due Date: 02/17/2022 (11 AM)

- Please submit a PDF copy of your answers via the submission link on Canvas by 11 AM, February 17, 2022.
- Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Honor Code Pledge: On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance on this work.

Name: M S Akhil Teja

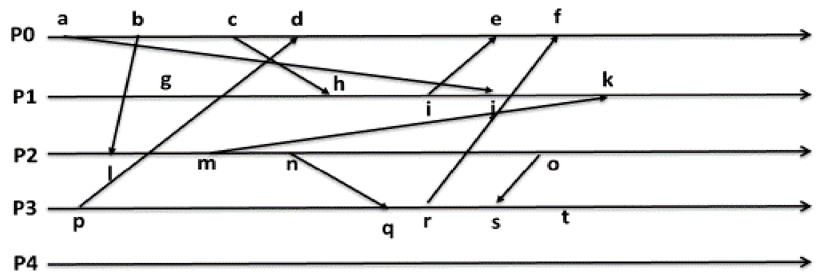
1. Read the paper titled *The Rise and Fall of CORBA* by Michi Henning. *https://cacm.acm.org/magazines/2008/8/5336-the-rise-and-fall-of-corba/fulltext*

   Suppose you are tasked with designing and implementing a middleware service that would be used by a large number of distributed systems developers. What lessons from CORBA development effort would you incorporate in developing this middleware service?

These are the lessons I would incorporate in the new middleware service.

- **Less Ramp up time**: The design of the middleware service has to be easy to understand and the rampup time should be less. CORBA has a steep learning curve and was complex to understand which I shall avoid for the new service.
- **Backward Compatibility:** The new middleware service shall be backward compatible and early versions shall be bug free backed with a good documentation. CORBA had backward compatibility issues in many releases.
- **Documentation**: Documentation is essential for faster rampup and future references. It helps newbies learn faster and help developers use the platform better. Good documentation helps developers refer and debug bugs soon.
- **Simplicity and Consistency**: Making APIs simple and consistent to use shall help developers in their development process. Keeping the system design simple shall give the developers and early learners an easy holistic understanding. Interoperable object references made using CORBA were complex.
- **Security Risks**: Security is really important for enterprise applications. I shall ensure the middleware service is secure using encryption and other security measures.
- **Multi-Language Support:** The middleware services should be implementable in different languages and cross language support should also be maintained. This shall help developers by giving them a flexibility to use any language and help early adoption. CORBA had no language mappings for C#, VB and .NET.
- **High performance and concurrency**: Design the system for a high-performance over large networks and support concurrency shall be kept in mind.
- **Portability**: The middleware should be easily portable. CORBA's had many portability problems thanks to it's complex IDL which had a large set of types.
- **Open-Source Support:** Using crowd pooling and open-source support shall help develop the platform fast and easy.
- **Interoperability:** It's important for the middleware to be inter-operable and should have a high performance and easy to understand for it to be widely used and adopted.

2. Consider the following figure that shows five processes ***(P0, P1, P2, P3, P4)*** with events ***a, b, c,*** ... and messages communicating between them. Assume that initial logical clock values are all initialized to ***0***.



a) **[15 Points]** Provide logical clock (C) values of each event shown.

C(a) = C(p) = C(g) = 0
C(b) = 1
C(l) = C(b)+1 = 2
C(c) = 2
C(m) = 3
C(d) = 3
C(h) = 3
C(i) = 4
C(n) = 4
C(j) = 5
C(e) = 5
C(o) = 5
C(q) = 5
C(k) = 6
C(r) = 6
C(s) = 7
C(f) = 7
C(t) = 8

b) **[15 Points]** Provide vector clock (V) values of each event shown.
5 processes so vector size of 5
V(a) = 10000
V(b) = 20000
V(c) = 30000
V(d) = 40010
V(e) = 53010
V(f) = 63330
V(g) = 01000
V(h) = 32000
V(i) = 33000
V(j) = 34000
V(k) = 35200
V(l) = 20100
V(m) = 20200
V(n) = 20300
V(o) = 20400
V(p) = 00010
V(q) = 20320
V(r) = 20330
V(s) = 20440
V(t) = 20450

c) **[10 Points]** Identify two events *ai* and *aj* to show that C(*ai*) < C(*aj*) does not necessarily imply *ai* $\rightarrow$ *aj*.
Consider the events p and l their logical clock values are C(l)=2 and C(p)=0. They occur on different process and there are no messages received from one event to another. So, we can't imply the happened before relationship here. Hence, both of them are concurrent events.

d) **[10 Points]** Assuming P0 < P1 < P2 < P3 < P4, provide a total ordering of all events constructed from the logical clock C. Is this total order unique, i.e., is it possible to construct a different total order than the one you constructed?
Considering the above ordering of processes, the total ordering would be:
p=>g=>a=>b=>l=>c=>m=>h=>d=>n=>i=>q=>o=>j=>e=>r=>k=>s=>f=>t

If the assumption about processes changes, the total ordering changes but it's unique if the assumption is maintained.

e) **[10 Points]** Suppose process P4 sends a message m (send event is *aa* and the corresponding receive event is *bb*). Show how $aa \rightarrow t$, $aa \rightarrow o$, and *aa* and *b* are concurrent. Identify an event a_i ($a_i \neq aa$) such that $a_i \rightarrow bb$.

A. Since nothing is mentioned about whom the process P4 sends the message to, I am assuming it to be P2 and that the receive event on P2 occurs after event 'l' and before event 'm'. Such an assumption doesn't contradict the given statements. Since aa is the send event and bb the corresponding receive event, aa->bb and since bb occurs of P2 before 'o', bb->o. From transitive property, we can say aa->o. There is a message from 'o' to 's', so we have o->s. s occurs before t on P3, so s->t and also, o->t from transitive property. Since, aa->o and o->t, we have aa->t, aa occurs on P4 and b occurs on P0 and since there are no send/receive between these events they are concurrent. Hence, we have b->l and l,bb occur on P2 in the same order, so l->bb. Thus there exists b such that b->bb and b!=aa

3. Browse the NTP project webpage (*http://www.ntp.org*).

   (a) [15 Points] Explain how NTP computes filter dispersion.

Source: Clock Filter Algorithm (udel.edu)

In the clock filter algorithm, the offset and delay samples from the on-wire protocol are inserted as the youngest stage of an eight-stage shift register, thus discarding the oldest stage. Each time an NTP packet is received from a source, a dispersion sample is initialized as the sum of the precisions of the server and client. Precision is defined by the latency to read the system clock and varies from 1000 ns to 100 ns in modern machines. The dispersion sample is inserted in the shift register along with the associated offset and delay samples. Subsequently, the dispersion sample in each stage is increased at a fixed rate of **15 µs/s**, representing the worst-case error due to skew between the server and client clock frequencies. In each peer process the clock filter algorithm selects the stage with the smallest delay, which generally represents the most accurate data, and it and the associated offset sample become the peer variables of the same name. The peer dispersion statistic is determined as a weighted sum of the dispersion samples in the shift register. Initially, the dispersion of all shift register stages is set to a large number "infinity" equal to 16 s. The weight factor for each stage, starting from the youngest numbered i = 1, is 2-i, which means the peer dispersion is approximately 16 s. As samples enter the register, the peer dispersion drops from 16 s to 8 s, 4 s, 2 s, and so forth. In practice, the synchronization distance, which is equal to one-half the delay plus the dispersion, falls below the select threshold of 1.5 s in about four updates. This gives some time for meaningful comparison between sources, if more than one is available. The dispersion continues to grow at the same rate as the sample dispersion.

   (b) [15 Points] Is NTP still at a security risk[yes/no]? If no, explain how NTP is being protected from vulnerabilities? If yes, what are some best practices that one can use while using NTP that might reduce the impact of security risks?

Source: NTP Security Analysis (udel.edu)
NTP is vulnerable to couple of attacks especially MITM and sometime DDOS too.
Hence, it still is a security risk. The countermeasures that can be used are:

- Respond to NTP control queries only from authorized parties.
- Encrypted communications and authentication can be used for securing the network.
- Symmetric mode should be used with a different crypto key every time and should be done between trusted peers only.
- Broadcast mode should be used only among trusted networks.
- To prevent bogus packets, we can monitor the origin timestamp. This would also prevent replay attack.
- Using a public NTP server for external hosts.

4. A clock is being synchronized with a time server using the Cristian's probabilistic clock synchronization algorithm. Request send time is 4:20:15.200, the server time is 4:19:10.800 and the time when the response is received is 4:20:15.300

a) **[5 Points]** What is the client's estimate of the server time?

   T0 = 4:20:15.200

   Tserver = 4:19:10.800

   T1 = 4:20:15.300

   T1-T0 = 100ms

   T1-T0/2 = 50ms

   Tnew=Tserver +50ms

   = 4:19:10.850

   Hence, client estimate would be 4:19:10:850

b) **[5 Points]** Is the client's clock running faster or slower than the server clock? What will the client do after estimating the server time?

Client's clock is running faster than server clock. Rather than resetting the client time, to attain the synchronized time, client OS shall slow its clock till it is in sync with the server time. This it does by updating Linear Compensating function using PIC (Programmable Interrupt Controller).