BMCS3003 E-Test (Week 8)

202305 Semester

This midterm test (E-Test) will be assessing student's in terms of theses two learning outcome:

CLO2: Analyse a given scenario with parallel and distributed computing techniques. (C4, PLO2) 50%

CLO3: Discuss the variety of parallel and distributed computing techniques. (C2, PLO1) 50%

tohwj-pm20@student.tarc.edu.my Switch account

Draft saved

* Indicates required question

Email *

☑ Record **tohwj-pm20@student.tarc.edu.my** as the email to be included with my response

What are **Safe State** and **Unsafe State** in the Banker's Algorithm? * 4 points

- ○ Safe state means it is safe from race condition because there exists a safe space to race, whereas unsafe state means it may run into race condition
- ○ Safe state means the bank is not going to go bankrupt, whereas unsafe state means the bank may go bankrupt anytime
- ◉ Safe state means there exist a safe sequence of processes that when executed in the given order there will be no deadlock, whereas unsafe state means there is a potential deadlock
- ○ Safe state means the state of the resource is safe from corruption because there exists a protection system, whereas unsafe state means it is vulnerable to corruption
- ○ Safe state means the state can be saved in a secured storage because there exists a secured space, whereas unsafe state means it cannot be securely stored

The following are four functions showing various ways of using the binary semaphore to protect their respective critical sections by novice programmers. Which of the following answers best describe what happen after a set of threads concurrently run each function. * 4 points

```
std::binary_semaphore sem1{1};
std::binary_semaphore sem2{1};
std::binary_semaphore sem3{1};
std::binary_semaphore sem4{1};

void func1() {
  sem1.acquire();
  // critical section
  // ...
  sem1.release();
}

void func2() {
  sem2.acquire();
  sem2.acquire();
  // critical section
  // ...
  sem2.release();
}

void func3() {
  sem3.release();
  // critical section
  // ...
  sem3.acquire();
}

void func4() {
  sem4.acquire();
  // critical section
  // ...
}
```

- ○ No race condition for all functions
- ○ No deadlock for all functions
- ○ No race condition for func1, but all the threads encounter race condition for the rest of the functions
- ○ func1: no race condition, func2: all threads encounter deadlock, func3: all threads encounter race condition, func4: all threads encounter deadlock except for the first thread
- ◉ func1: all threads encounter deadlock, func2: no race condition, func3: all threads encounter race condition, func4: all threads encounter deadlock except for the first thread
- ○ No race condition for func1, but all the threads encounter deadlock for the rest of the functions

The following snapshot of the current resource allocations and the maximum needed resources for 5 processes (P0 to P4). Suppose P1 requests for **<0, 1, 1>** instances of resources. Given that the total instances of resources are **<5, 10, 6>**, determine if the OS should grant the resources to P1 using the Banker's algorithm for deadlock avoidance. If yes, what is the **safe sequence** to warrant such permission. * 4 points

Reference:
[Banker's Algorithm]
[The accompanied video]

Process	Allocation			Max		
	A	B	C	A	B	C
P0	1	0	0	5	7	3
P1	0	2	0	2	4	3
P2	0	3	1	0	9	2
P3	1	2	0	2	3	2
P4	0	0	2	3	4	3

- (selected) Yes, the OS should grant the resources because there is a safe sequence of P2-P1-P4-P0-P3
- No, the OS should reject the request because there is no safe sequence
- Yes, the OS should grant the resources because there is a safe sequence of P0-P1-P2-P3-P4
- Yes, the OS should grant the resources even though there is no safe sequence
- Yes, the OS should grant the resources because there is a safe sequence of P1-P4-P3-P0-P2
- Yes, the OS should grant the resources because there is a safe sequence of P1-P3-P4-P0-P2

Arrange the sequence of events taken by the Berkeley's algorithm in the distributed systems clock synchronization process. * 4 points

	Step 1	Step 2	Step 3	Step 4
For each follower's clock, it compute the time difference with its own time, discard all outliers, then sum all valid time differences and divide the result with the number of valid followers	○	○	◉	○
The leader sends to each follower the amount of time to adjust to the latter's clock	○	○	○	◉
The leader polls the clock from the followers using Cristian's algorithm	○	◉	○	○
A leader is chosen to be the timekeeper through some leader election process	◉	○	○	○

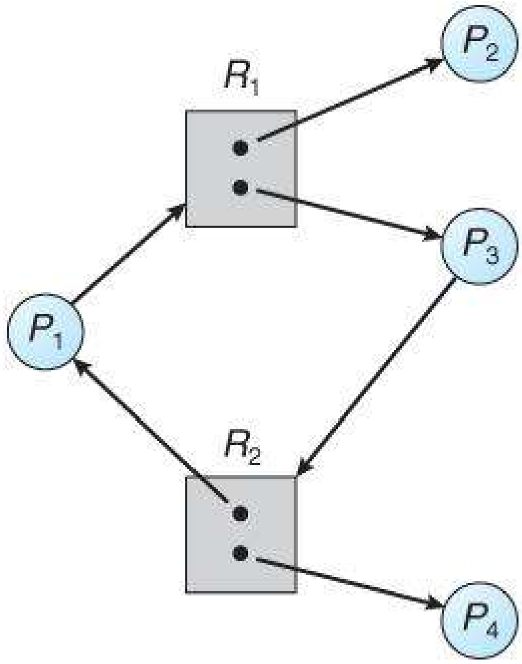
What are the challenges with distributed computing? * 4 points

- [ ] Transparency: Unlike on a single computing node, due to the nature of distributed systems whereby the functionalities are disaggregated and spread over different computing nodes interconnected by the network may appear complex to the users. To hide the complexity by creating functional transparency such that overall appearance is a simpler single system may not be easy.
- [x] Heterogeneity: Differences in programming languages, data format, hardware, OS, networks, in a system make interoperation among nodes difficult
- [x] Failure Handling: Failures in distributed systems are more challenging to rectify or recover because of complex interactions among computing nodes.
- [ ] Security: This is to ensure data integrity and confidentiality when the data is shared in the public network. It is also to ensure entities in the systems must be verifiable and privileges are correctly assigned through authentication and authorization, respectively.
- [x] Concurrency: Problems may arise when multiple processes attempt to access the same resources at the same time, thus steps needed to ensure any manipulation in the system remains in the correct state at all time requires sophisticated and slow distributed concurrency control.
- [x] Scalability: Some problems are tightly dependent and difficult to break into multiple independent parts for solving by multiple computing nodes. Also, poorly distributed workload may incur high communication overhead requiring higher network bandwidth.

What is the advantage of a Monitor compared to Binary Semaphore in implementing mutual exclusion protocol? * 4 points

- (•) To prevent unintentional mistakes done by programmers that can lead to deadlock or race condition, which is possible under binary semaphore
- ( ) It is able to monitor the thread that is currently in the critical section and kills it if it is misbehaving
- ( ) There is no advantage
- ( ) It can do everything a binary semaphore can and also able to view all critical sections
- ( ) It is more complex than binary semaphore

The following Resource-Allocation graph contains a cycle. But there is no possible deadlock. Why? * 4 points



- ◯ Because P1 can exchange resources with P3
- ◉ If either P2 or P4 release their respective resource, then this allows P1 and P3 to acquire the resources without encountering deadlock
- ◯ Because R1 and R3 have enough resources for 3 processes to acquire at the same time
- ◯ Because P2 and P4 don't need the resources they are holding
- ◯ If either P1 or P3 release their respective resource, then this allows P2 and P4 to acquire the resources without encountering deadlock
- ◯ Because P1 can request P3 to release its resource

One way to prevent a deadlock is to allow the resource held by a process to be **preempted** by the OS. This will require: * 4 points

- [x] The capability to rollback the state of the resource held by the preempted process before passing it to another competing process
- [ ] No rollback of the resource held by the preempted process, because the competing process can continue processing it
- [x] Restarting the preempted process to run the critical code again when the resource is made available
- [ ] Killing the preempted process for good
- [ ] The OS to purge the resource's state and then move it from the memory to storage drive
- [x] No process restarting because it can continue from where it was last preempted

Which of the following are true for the implementation of Producer-Consumer synchronization problem? * 4 points

- [ ] It can be solved using two counting semaphores and one mutex
- [x] One counting semaphore is used to protect the accesses to the queue/buffer and counter variable from race condition
- [ ] One mutex is used to protect the accesses to the queue/buffer and counter variable from race condition
- [ ] It can be solved using two mutexes and one counting semaphore
- [x] One counting semaphore is used to count how many slots are full in the buffer. For as long as its count value is not 0, a consumer is able to dequeue a message from the buffer. Otherwise it will be blocked.
- [x] One counting semaphore is used to count how many slots are empty in the buffer. For as long as its count value is not 0, a producer is able to enqueue the generated message. Otherwise it will be blocked.

What are the benefits of distributed systems? * 4 points

- [x] Scalability: Ability to scale beyond the computing power of a single node. If the problem size increases, the number of nodes can be increased to match the computing needs.
- [x] Resiliency: With many computing nodes implementing redundancies, it ensures a single failure doesn't equate to systems-wide failure
- [x] Resource/Data sharing: Resources are available to multiple users without needing large storage space to store copies.
- [ ] Cost: As the number of nodes increases, it drives down the cost.
- [ ] Bandwidth: The more computing nodes, the more data transferred required due to data migrations
- [ ] Energy: Multiple computing nodes can improved energy consumption. This reduces utility bills.
- [x] Speed/Performance: Multiple computing nodes cooperatively solving different parts of a problem concurrently and then combining the result at the end will improve the performance

Which is true for Peterson’s Algorithm in process synchronization? * 4 points

- ○ None of the statements is true
- ○ It is one of the ways to implement mutual exclusion protocol for 2 threads without needing a hardware level atomic access instruction
- ○ It can be used to replace counting semaphore
- ○ It does not implement mutual exclusion protocol
- ◉ It is a synchronization algorithm for 2 or more threads and relies on the hardware level atomic access instruction to operate precisely and correctly
- ○ It is the only way to implement mutual exclusion in software

There are four conditions that are necessary to achieve deadlock, match the following descriptions to their respective conditions: * 4 points

	Mutual Exclusion	Hold and Wait	No Preemption	Circular Wait
The protected resource must be accessible only by one thread at a time. That is, if a thread holds on to the resource, the other threads wanting the same resource have to wait for it to be released.	◉	○	○	○
Suppose there are n threads {t1, t2, t3, ..., tn}, then t1 waits for the resource held by t2, t2 waits for the resource held by t3, and so on; and the last thread tn waits for the resource held by t1.	○	○	○	◉
A thread must be simultaneously holding at least one resource and waiting for at least one resource that is currently being held by some other thread	○	◉	○	○
Once a thread is holding a resource, then that resource cannot be taken away from that thread until it	○	○	◉	○

voluntarily
releases the
resource

What is the outcome of a race condition by concurrent threads entering into a critical section? * 4 points

- (selected) The outcome depends on the order of concurrent threads executions and the outcome can be different for each case.
- Always a normal desirable outcome
- All threads race with each other and no thread wins. The outcome is that all concurrent threads stall in the critical section.
- No outcome
- The outcome is that the concurrent threads run in a specific sequence

Which of the following statements are true for token-passing mutual exclusion algorithms for distributed systems? * 4 points

- [ ] None of the statements are true
- [ ] All sites not holding the token can access the critical section and need not wait
- [ ] A site can use the token only once in its lifetime
- [ ] The site holding the token has no right to access the critical section
- [ ] Each site can create a token at will and pass them around freely
- [ ] There can be more than one token in the distributed system, but only when all tokens are collected can a site enters the critical section
- [x] There cannot be more than one token to protect a resource
- [x] All sites not holding the token cannot access the critical section and must wait
- [x] Only the site holding the token has the right to access the critical section

If 5 threads wanted to **only read** a shared file at the same time. Does the shared file needs to be read in mutual exclusive fashion? * 4 points

- ○ Yes, because all threads can encounter deadlock
- ○ Yes, if each thread does not agree to share and no, if each thread agrees to share
- ○ No, because there is no possibility of deadlock
- ◉ Yes, because all threads are accessing the same file concurrently and there is a potential race condition
- ○ No, because they are all friendly threads
- ○ No, because even though all threads are accessing the same file concurrently, there can never be a race condition

One way to prevent a **circular wait** in *deadlock prevention scheme* for a situation where concurrent threads need to successively acquire multiple resources (without releasing any before successfully acquiring them all) is to follow certain steps. Choose the correct steps from the following list. * 4 points

Note:
1. Each resource is protected by a unique mutex
2. The steps below are not displayed in order

- [x] Enumerate each resource protected by a unique mutex with a unique number
- [ ] Just wait even the code is circular
- [ ] Label all resources with only one number
- [ ] Don't label the resources
- [ ] Each concurrent thread needing multiple resources acquires them in any order
- [x] Each concurrent thread needing multiple resources must acquire them in ascending order of their unique numbers

Which of the following are true statements? * 4 points

- [x] A mutex ensures mutual exclusion of a critical section it is protecting
- [ ] None of statements are true
- [ ] Mutual exclusion means all threads are able to access the race condition at the same time
- [ ] A counting semaphore does not ensure mutual exclusion
- [x] Binary semaphore is a special type of counting semaphore and has a maximum count value of 1
- [x] A critical section consists of data only

Spinlock can implement mutual exclusion protocol for threads running on a multicore processor. Which of the following is needed for the implementation? * 4 points

- ○ One semaphore object for each core
- ◉ A lock variable/object in the shared memory accessible by all cores
- ○ Nothing is needed
- ○ A spin counter by the master core
- ○ One mutex object by all cores

Content sharing can be implemented using P2P (peer-to-peer) network or client-server architecture. Why scaling a P2P to a larger network makes the sharing faster and more efficient than scaling the clients to a higher number in client-server architecture? * 4 points

Reference:
[What's the difference between peer-to-peer (P2P) networks and client-server?]

- [ ] In client-server architecture, a client helps the server to deliver to other clients wanting the same content
- [x] In P2P, a node receiving a content from a group of nodes can later participate with that group to share the content with other nodes wanting the same content, thus increases the delivery rate to a node.
- [ ] Increasing P2P nodes increases the network bandwidth and hence increases the transfer rate
- [x] In client-server architecture, a client does not help out the server to deliver to other clients wanting the same content
- [x] Increasing P2P nodes increases both clients and servers because a node can assume the 2 roles at the same time, whereas increasing the number of clients in the client-server architecture does not increase the number of server
- [ ] In client-server architecture, all clients assume the role of a server to deliver content
- [x] In client-server architecture, the server delivery rate to each client degrades as the number of clients increase

Which of the following statement is true for the Raymond's Tree algorithm? * 4 points

- ( ) The root is the only site that holds the token and has the right to access the critical section
- ( ) When site k sends the token to site j, site k retains its root status
- ( ) Raymond's Tree does not form a directed acyclic graph
- ( ) All edges in the tree are pointing away from the root site
- (•) The root site does not hold a token

What are the differences between **spinlock** and **mutex**? * 4 points

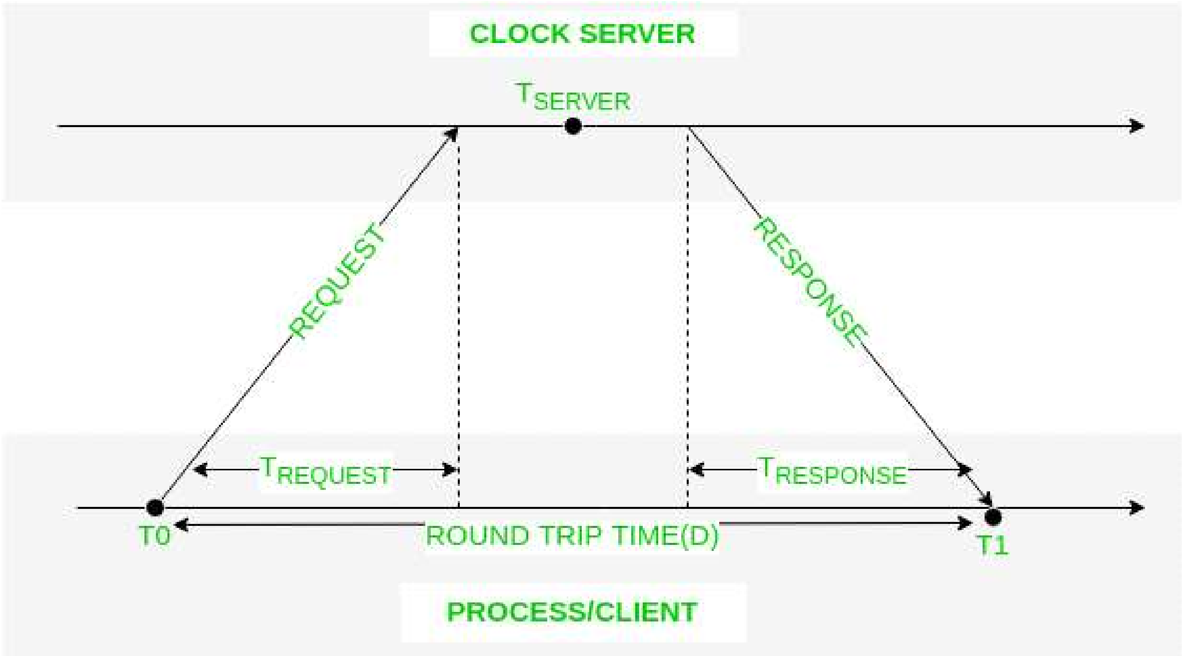
- [ ] A spinlock spins the key to lock, whereas a mutex mutates a thread to become a lock
- [ ] A spinlock is a lock, whereas a mutex is a critical section
- [x] In spinlock a waiting thread wastes CPU processing power by spinning. However, in mutex a waiting thread is enqueued in the blocking queue and the CPU is relinquished to another thread, thus wasting no CPU power.
- [ ] There is no difference between them
- [ ] A spinlock can lock only, whereas a mutex can unlock only
- [x] A spinlock spins in a loop checking if the lock is available, whereas a mutex will block the thread if it has been acquired by another thread

Which of the following statements are true for the Suzuki-Kasami's algorithm? * 4 points

- [x] Upon exiting from critical section, site k randomly sends the token to a site in hope it will use it
- [x] Upon receiving a REQUEST from site j, site k updates its request array as follow: Rk[j] = max(Rk[j], sn), where sn is the sequence number in the REQUEST message.
- [x] Upon receiving a REQUEST from site j, site k immediately sends the token to site j
- [ ] If a site wants to access the critical section, it sends REQUEST message only to the site holding the token
- [ ] Upon receiving a REQUEST from site j, site k immediate enqueues site j in its queue
- [ ] If a site wants to access the critical section, it sends REQUEST message to all sites participating in the distributed mutual exclusion

Refer to the diagram below. The server clock keeps an accurate current time, Tserver and the client make a time request at T0 and it receives back the response at time T1. Suppose 0.6788 second has passed since the client sent Trequest to the server, it receives the message response from the server containing a timestamp of 12:23:44.5245, what would be the approximated current time if the client use Cristian's algorithm to compute. * 4 points

Your answer MUST be in the form **XX:XX:XX.XXXX**, where X is a digit between 0-9. Please don't add any space in your answer, otherwise the system will mark it as wrong.



12:23:44:751

Match the following traits to the correct coupling types of distributed systems? * 8 points

	Loosely-Coupled Distributed Systems	Tightly-Coupled Distributed Systems
CPUs with dense multicores	○	◉
Uses off-the-shelf network for interconnections	◉	○
Uses specialized ultra-highspeed backplane for interconnections	○	◉
Independent memory, hard drive storage, and OSes	◉	○
Fast inter-process communication	○	◉
Slow inter-process communication	◉	○
General purpose hardware, which is abundant and cheap	◉	○
Shared clock and memory making synchronization easier and data sharing faster	○	◉

Page 1 of 1

Submit Clear form