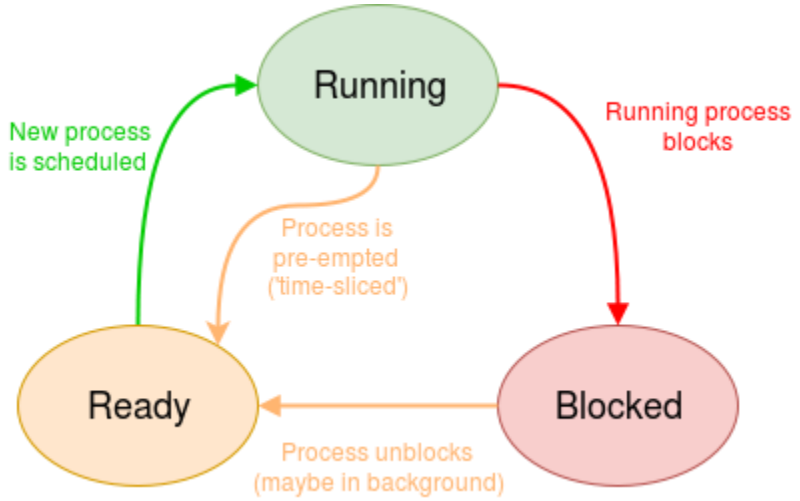
Midsem Answer Key

[CS3.301 Operating Systems and Networks]

Question 1 [Anirudh]

Answer:



- A process which is **ready** may:
  - be scheduled by the OS and set running
  - be terminated by some outside influence (e.g. 'kill')
- A process which is **running** may:
  - deliberately yield execution (becoming ready)
  - terminate itself (job complete)
  - be terminated by some outside influence (e.g. 'kill')
- A process which is **blocked** may:
  - become unblocked and become ready (e.g. because ...)
    - I/O has become ready
    - time has advanced
  - be terminated by some outside influence (e.g. 'kill')

See https://wiki.cs.manchester.ac.uk/COMP15212/index.php/Process_States for more info.

Question 2 [Sathvika & Divijh]

(a) Scenario A (2 points) [Sathvika]

Scenario: Three identical processes (P1, P2, P3), each 20ms, arrive simultaneously.

FIFO (1 pt):

- All processes are identical (same 20 ms), so the order doesn't matter.
- Turnaround time (TAT): (20 + 40 + 60) / 3 = 120 / 3 = 40ms
- Response time (RT): (0 + 20 + 40) / 3 = 60 / 3 = 20ms
- Benefit: Simple and fair since all jobs are of equal length.

SJF (1 pt):

- All jobs are identical, so it behaves the same as FIFO.
- Metrics are the same as FIFO.
- Benefit: No advantage here, as all jobs have the same burst time.

Note: Usually, when execution times differ, SJF performs better in terms of response time and turnaround time.

(b) Scenario B (3 points) [Divijh]

Scenario:

- P1 = 10ms
- P2 = 20ms
- P3 = 30ms
  All arrive simultaneously.

FIFO (1 pt):

Order: P1 → P2 → P3 or P2 → P1 → P3 (any order is fine since they arrive simultaneously).

- Completion times: P1 = 10, P2 = 30, P3 = 60
- Turnaround times (TAT): P1 = 10, P2 = 30, P3 = 60

- Average TAT: (10 + 30 + 60) / 3 = 33.3ms
- Response times (RT): P1 = 0, P2 = 10, P3 = 30
- Average RT: (0 + 10 + 30) / 3 = 13.3ms

SJF (1 pt):

- Order: P1 (10) → P2 (20) → P3 (30)
- Completion times: Same as FIFO → 10, 30, 60
- TAT: Same as FIFO = 33.3ms
- RT: Same as FIFO = 13.3ms

Round Robin (q = 1ms) (1 pt):

- Processes share the CPU in cycles of 1ms.
- Execution order cycles: P1 → P2 → P3 → P1 → ...
- Completion times:
    - P1 (10ms): finishes at 28
    - P2 (20ms): finishes at 49
    - P3 (30ms): finishes at 60
- Turnaround times (TAT):
    - P1 = 28
    - P2 = 49
    - P3 = 60
    - Average TAT: (28 + 49 + 60) / 3 = 45.66ms
- Response times (RT):
    - P1 = 0
    - P2 = 1
    - P3 = 2
    - Average RT: (0 + 1 + 2) / 3 = 1ms

(c) SJF vs FIFO Equivalence (2 points) [Divijh]

Two points were expected for 2 marks, 1 mark has been given per point for writing any of the following points.

SJF and FIFO would have same turnaround time when:

- All jobs have equal burst times.

[only 0.5 has been given if mentioned “equal burst times **and** same arrival time” as the former is sufficient i.e the latter isn’t necessary/needed]

- All processes arrive simultaneously and are executed in non-decreasing burst time order(executed in the order of SJF when arrived simultaneously would lead to same turnaround time).
- All processes arrive increasing order of burst times
- No overlapping processes
- Or any other valid (generalized) scenario

(d) SJF vs RR Response Time Equivalence (1 point) [Sathvika]

Key idea: Response time is the time until a process is first scheduled.

SJF and RR have the same response time when:

- All processes arrive at time 0 (same time), and
- The Round Robin quantum is greater than or equal to the maximum job execution time, and
- The SJF job order matches the order of the first CPU allocation in Round Robin.

Question 3 [Akhila]

For the LangOS memory management module, the **LRU (Least Recently Used)** policy is the most suitable and practical choice. AI and language model workloads often exhibit strong **locality of reference**, meaning they frequently re-access the same data and instructions. LRU is designed to capitalize on this behavior by keeping recently used pages in memory, which directly boosts performance by minimizing slow disk I/O from the swap space.

Other policies are less ideal for a production system like LangOS:

- **FIFO** is too simplistic and can mistakenly evict a frequently used page just because it was loaded early.
- The **Optimal** policy is impossible to implement in a real system because it requires knowing the future, which isn't practical.

- A **Random** policy would lead to unpredictable performance, which is unacceptable for a stable OS.

Therefore, LRU provides the best balance of performance and practicality for the team's needs. The following analysis validates this choice.

LRU Policy Analysis (Cache Size = 3)

Trace 1: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

Access	Hit/Miss	Evict	Resulting Cache State (Least to Most Recent)
0	Miss	-	[0]
1	Miss	-	[0, 1]
2	Miss	-	[0, 1, 2]
0	Hit	-	[1, 2, 0]
1	Hit	-	[2, 0, 1]
3	Miss	2	[0, 1, 3]
0	Hit	-	[1, 3, 0]
3	Hit	-	[1, 0, 3]
1	Hit	-	[0, 3, 1]
2	Miss	0	[3, 1, 2]
1	Hit	-	[3, 2, 1]

- **Total Accesses:** 11
- **Hits:** 6
- **Misses:** 5
- **Hit Rate:** (6 / 11) * 100% = **54.5%**
- **Miss Rate:** (5 / 11) * 100% = **45.5%**

Trace 2: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Access	Hit/Miss	Evict	Resulting Cache State (Least to Most Recent)
1	Miss	-	[1]
2	Miss	-	[1, 2]
3	Miss	-	[1, 2, 3]
4	Miss	1	[2, 3, 4]

1	Miss	2	[3, 4, 1]
2	Miss	3	[4, 1, 2]
5	Miss	4	[1, 2, 5]
1	Hit	-	[2, 5, 1]
2	Hit	-	[5, 1, 2]
3	Miss	5	[1, 2, 3]
4	Miss	1	[2, 3, 4]
5	Miss	2	[3, 4, 5]

Export to Sheets

- **Total Accesses:** 12
- **Hits:** 2
- **Misses:** 10
- **Hit Rate:** (2 / 12) * 100% = **16.7%**
- **Miss Rate:** (10 / 12) * 100% = **83.3%**

FIFO Policy Analysis (Cache Size = 3)

Trace 1: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

Access	Hit/Miss	Evict	Resulting Cache State
0	Miss	-	[0]
1	Miss	-	[0, 1]
2	Miss	-	[0, 1, 2]
0	Hit	-	[0, 1, 2]
1	Hit	-	[0, 1, 2]
3	Miss	0	[1, 2, 3]
0	Miss	1	[2, 3, 0]
3	Hit	-	[2, 3, 0]
1	Miss	2	[3, 0, 1]
2	Miss	3	[0, 1, 2]
1	Hit	-	[0, 1, 2]

- **Total Accesses:** 11
- **Hits:** 4

- **Misses:** 7
- **Hit Rate:** (4 / 11) * 100% = **36.4%**
- **Miss Rate:** (7 / 11) * 100% = **63.6%**

Trace 2: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Access	Hit/Miss	Evict	Resulting Cache State
1	Miss	-	[1]
2	Miss	-	[1, 2]
3	Miss	-	[1, 2, 3]
4	Miss	1	[2, 3, 4]
1	Miss	2	[3, 4, 1]
2	Miss	3	[4, 1, 2]
5	Miss	4	[1, 2, 5]
1	Hit	-	[1, 2, 5]
2	Hit	-	[1, 2, 5]
3	Miss	1	[2, 5, 3]
4	Miss	2	[5, 3, 4]
5	Hit	5	[3, 4, 5]

- **Total Accesses:** 12
- **Hits:** 3
- **Misses:** 9
- **Hit Rate:** (3 / 12) * 100% = 25**%**
- **Miss Rate:** (9 / 12) * 100% = 75**%**

Comparison Summary (Cache Size = 3)

For **Trace 1**, LRU is the clear winner with a **hit rate of 54.5%** compared to FIFO's 36.4%. This highlights how LRU's logic is better for traces with locality.

For **Trace 2**, FIFO (25.0% hit rate) actually performs slightly better than LRU (16.7% hit rate). However, the performance of both is poor, and the next section reveals why FIFO is still the wrong choice.

Effect of Increasing Cache Size to 4 Pages

Increasing memory capacity reveals a critical difference between the policies.

LRU (Cache Size = 4)

Trace	Hits	Misses	Hit Rate	Miss Rate	Improvement (from 3 frames)
Trace 1	7	4	63.60%	36.40%	+9.1 percentage points
Trace 2	4	8	33.30%	66.70%	+16.6 percentage points

- **Trace 1:** Hit rate improves to **63.6%** (7 hits). An improvement of 9.1 percentage points.
- **Trace 2:** Hit rate improves significantly to **33.3%** (4 hits). An improvement of 16.6 percentage points.

FIFO (Cache Size = 4)

Trace	Hits	Misses	Hit Rate	Miss Rate	Improvement (from 3 frames)
Trace 1	7	4	63.60%	36.40%	+27.2 percentage points
Trace 2	2	10	16.70%	83.30%	No improvement

- **Trace 1:** Hit rate improves to **63.6%** (7 hits).
- **Trace 2:** Hit rate **shows no improvement**, remaining at **16.7%** (2 hits).

This demonstrates **Belady's Anomaly** in FIFO for Trace 2: adding more memory did not improve its performance. LRU, in contrast, effectively used the extra space to increase its hit rate. This makes LRU a much more reliable and scalable choice.

Marking:

1. **Recommendation and Justification**
    a. **1 Mark:** Correctly recommending the **LRU policy**.
    b. **1 Mark:** Providing a clear and valid reason (e.g., based on the principle of locality, or by stating it achieves a higher overall hit rate in the analysis).

2. **Analysis for Cache size = 3**
    a. **1 Mark:** Correctly showing the trace table for **LRU on Trace 1** and calculating the hit/miss rate.
    b. **1 Mark:** Correctly showing the trace table for **LRU on Trace 2** and calculating the hit/miss rate.
    c. **1 Mark:** Correctly showing the trace table for **FIFO on Trace 1** and calculating the hit/miss rate.
    d. **1 Mark:** Correctly showing the trace table for **FIFO on Trace 2** and calculating the hit/miss rate.
3. **Analysis for Cache size = 4**
    a. **1 Mark:** Correctly calculating the performance (hit/miss rates) for both LRU and FIFO with the larger cache size.
    b. **1 Mark:** Correctly identifying that LRU's performance improves while FIFO's does not for Trace 2, and properly explaining this observation (e.g., by mentioning Belady's Anomaly).

Question 4 [Prasoon]

Answer : (a) The ACK field is always present in the TCP header, but the ACK bit in the flags determines if the value is valid.

Explanation : The **TCP header format is fixed**, meaning the acknowledgment number field is always present. However, whether it should be interpreted depends on the **ACK control flag**.

- If the ACK flag is **set** (which is the case in almost all packets after the first SYN), then the acknowledgment number is **valid** and tells the sender which byte is expected next.
- If the ACK flag is **not set**, the field is simply ignored by the receiver.

Why Not Others :

(b) Incorrect - the ACK field is **always part of the header**; it doesn't "appear" or "disappear." Only its validity changes.

(c) Incorrect - the acknowledgment number is used throughout the connection for reliable delivery, not just in the three-way handshake.

(d) Incorrect - acknowledgment is a **core part of TCP reliability** and not something optional left to the application.

Question 5 [Aviral]

Answer : (b) A context switch may happen if the kernel's TCP send buffer is full, causing the process to block until space is available.

Explanation : The **send() system call is not a blocking call**. The TCP send buffer temporarily **holds unacknowledged data** until the receiver can accept it. A full buffer blocks the sending process because it cannot proceed until the buffer has space, and the OS may choose to schedule another process in the meantime.

Why Not Others :
(a) A context switch does not always occur on every system call. System calls may complete without blocking, and the process can remain scheduled.
(c) TCP send involves kernel space and kernel memory buffers.
(d) Context switches depend on buffer states and scheduling policies, not just inter-process communication location.

Marking : 1.5 Marks for correct answer (0 if any other option marked) and 1.5 for reasoning.

Question 6 [Shlok]

Answer: Option (a)

(a) MAC source/destination addresses are rewritten at every hop.

- **Layer 4 (Transport Layer):** The source (`5000`) and destination (`6000`) **port numbers remain unchanged** throughout the entire journey. They are used for end-to-end communication between the specific processes on Machine A and Machine B.
- **Layer 3 (Network Layer):** The source (`192.168.1.10`) and destination (`10.0.0.20`) **IP addresses also remain unchanged**. These addresses identify the original sender and final recipient of the packet across different networks.
- **Layer 2 (Data Link Layer):** The **MAC addresses change at each hop** (i.e., at the router). The MAC address is used for communication on a specific local network segment.
    - **Hop 1 (Machine A to Router R):** The Ethernet frame has:
        - Source MAC: `AA:BB:CC:DD:EE:01` (Machine A)
        - Destination MAC: `AA:BB:CC:DD:EE:11` (Router's LAN interface)
    - **Hop 2 (Router R to Machine B):** The router strips the old Layer 2 header and creates a new one for the next network segment:
        - Source MAC: `AA:BB:CC:DD:EE:22` (Router's WAN interface)
        - Destination MAC: `AA:BB:CC:DD:EE:02` (Machine B)

Because the MAC addresses are updated by the router to facilitate delivery on the next local network link, option (a) is the only correct description of the process.

Bonus Question 1 [Prakhar]

Hop Limit identification (0.5 point) Student correctly identifies that

the Hop Limit field is used. If incorrect → 0 points.

• Explanation (1.5 points total)

– 0.5 point: Explains that the hop limit decreases by 1 at every hop.

– 0.5 point: Mentions that when hop limit reaches 0, an ICMP Time

Exceeded message is sent back.

– 0.5 point: Explains that by varying the hop limit, intermediate nodes

can be revealed, thereby tracking the path.

Bonus Question 2 [Prakhar]

Correct Option Selection (1 point) Student must select option (c):

Interactive tasks could wait for unpredictable amounts of time to be scheduled.

– If any other option is chosen → 0 points (entire question).

– If multiple options are chosen → 0 points (entire question).

– If correct option not chosen → justification will not be considered.

• Explanation / Justification (2 points) Only graded if option (c) is

correctly chosen.

– 2 points: Explanation highlights that O(1) scheduler penalized interactive tasks due to misclassification, causing poor responsiveness.

Mentions that CFS was designed to improve fairness and responsive-

ness.

– 0.5 point: Mentions starvation but without linking to responsive-

ness/interactive tasks.

– 0 points: No explanation or irrelevant justification.