Ch-5

Remote calls invocation

Applications
Remote calls, Menage passing
Kernel OS
TCP & UDP

Q. Physical address space = 128 KB
Size of physical memory = ?

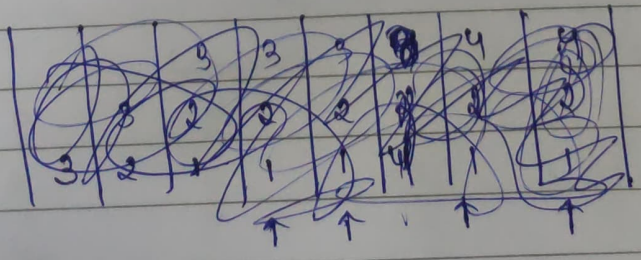
$128 \rightarrow 2^7$
$2^7 \times 2^{10}$
$= 2^{17}$
$\log_2 2^{17}$
$\Rightarrow 17$ B

Size of physical M/M = 24 B
physical address space = $2^{24} = 2^{20} \times 2^4 = 16$ MB

Size of physical M/M = 32 B
physical address space = $2^{32} = 2^2 \times 2^{30} = 4$ GB

Q. page T. size = 3
pages bit strings :-
3, 2, 1, 2, 1, 4, 2, 1, 7, 8
FIFO

		1	1	1	1	1	1	1	7
	2	2	2	2	2	2	2	7	7
3	3	3	3	3	4	4	4	4	4
			↑	↑	↑	↑	↑		



Hit = 4
page fault = 6

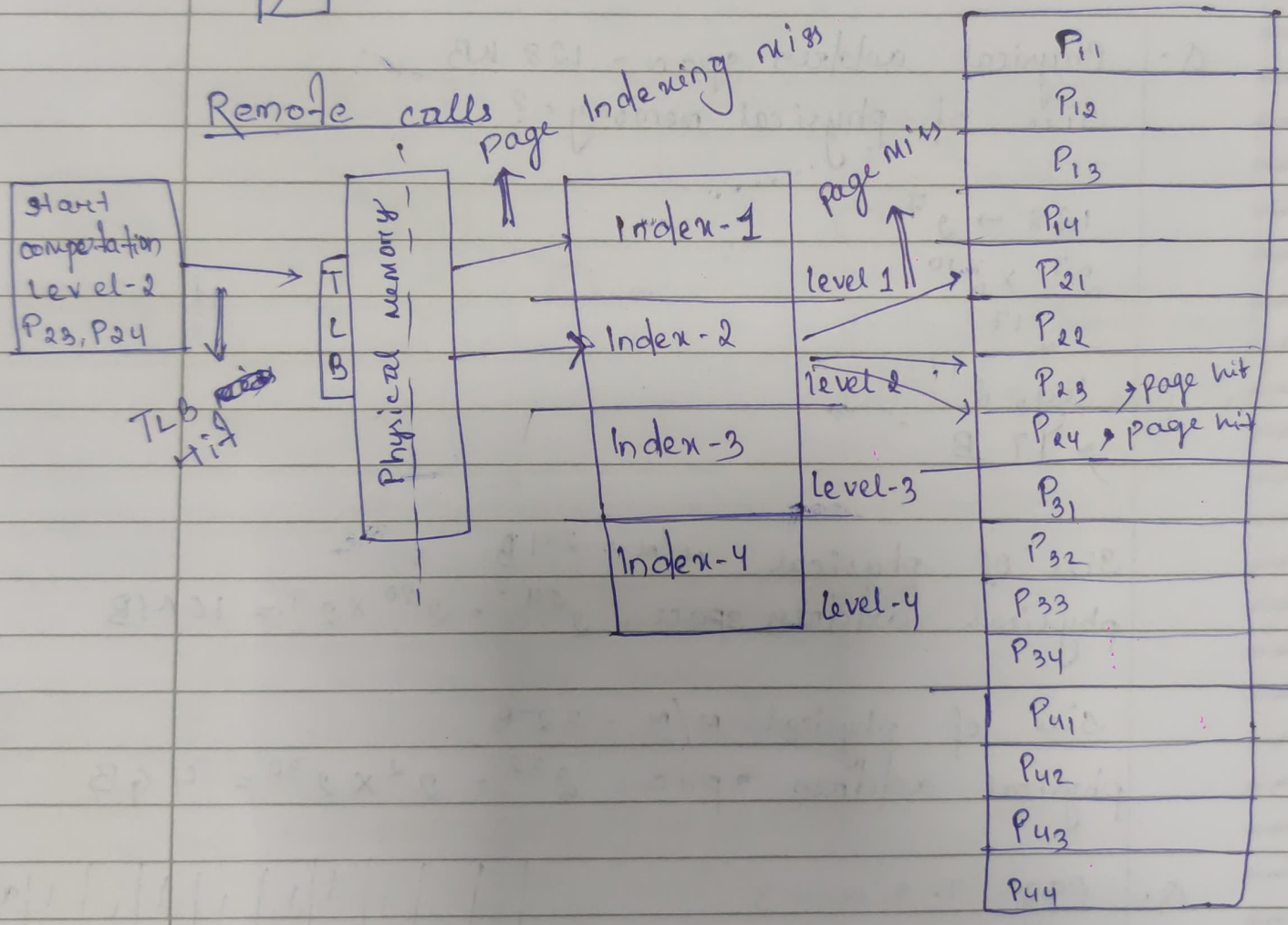
Page T. size = 4

bit string :- 3, 2, 1, 2, 7, 4, 2, 1, 7, 8, 1

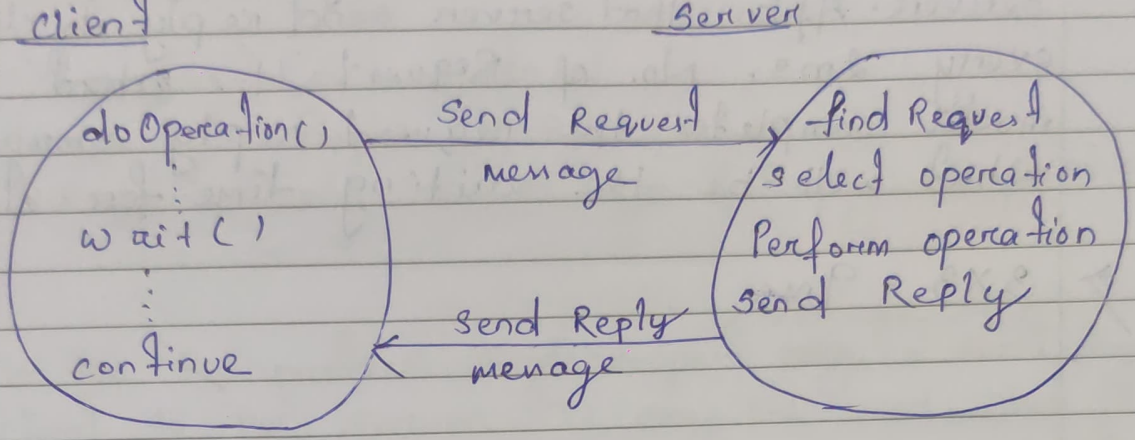
FIFO

7	
1	
~~2~~	8
~~3~~	4

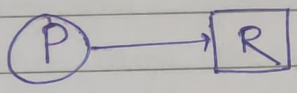
Page fault = 5



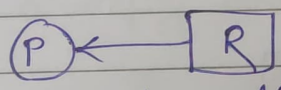
Request - Reply Communication



Note :-



i.e., Process is requesting a resource



Resource is allocated to process.

(i) Resource Available
(ii) Resource allocated
(iii) Resource Requested

Q. For process P_1, P_2, P_3
Required Resources $R_1\ R_2\ R_3$ [7, 5, 9]
Allocated Resources to $P_1\ P_2\ P_3$ [4, 3, 8]
At what condition of Resources system will complete the request-reply communication.

~~Available~~ ~~Total~~ ~~Requested~~

[3, 2, 1]

Q. There is an omission-failure model, which sends a request after every 3ms to the server. After that server send reply to client in every 2ms. No. of Request = 4. If this model completes the request reply communication. What will be the waiting time for client.

⇒ 3 X 3 = 9ms

Request Reply structure

Message Type
Operation ID
Remote Reference
Reference ID
Arguments / bite frequency

Styles of exchange protocols in Request-Reply communication

1) Request (R) protocol
2) Request-Reply (R-R) protocol
3) Request-Reply-Acknowledge Reply (R-R-A) protocol

In the R protocol single request message is sent by client to the server. It may be used when there is no value to be returned from the remote operation and the client requires no confirmation that the operation has been executed. The client may proceed immidiately after the request message is sent as there is no need to wait for a reply message.

R-R protocol is useful for most client-server exchange because it is based on Request-Reply protocol. A special acknowledgement message are not required because server's reply message is regarded as an acknowledgement message for a client's request message.

The R-R-A protocol is based on the exchange of 3 messages i.e., Request, Reply, acknowledge reply. The acknowledged reply message contains the request id from the reply message being acknowledged. This will enable the server to discard entries from its history.

Name	Menage sent by		
	client	Server	client
	~~Request~~		
R	Request		
R R	Request	Reply	
R R A	Request	Reply	Acknowledge

ch-1 → 1.1, 1.2, 1.3, 1.5
ch-2 → complete
ch-3 → 3.1, 3.2, 3.3, 3.4
ch-4 → 4.1, 4.2, 4.3
ch.5 → 5.1, 5.2, 5.3, 5.5

Q. find the network ID for a multicast switching if the class IP address is 187.24.47.172

187.24.0.0 (unicast)
187.24.255.255 (multicast)

96.124.11.118
96.0.0.0 (unicast)
96.255.255.255 (multicast)

Q. Length of datagram = 150 B
Latency = 3 ms
If the data transfer rate is 30 B per ms.
find the data transmission time.

Transmission time = latency + $\frac{\text{Length of datagram}}{\text{Data Transfer rate}}$

$= 3 + \frac{150}{30} = 3 + 5 = 8$ ms

Q. Length of datagram = 320 B
latency = 2 ms
data transmission time = 7 ms
find data transfer rate

transmission time = latency + $\frac{\text{Length of datagram}}{\text{data transfer rate}}$

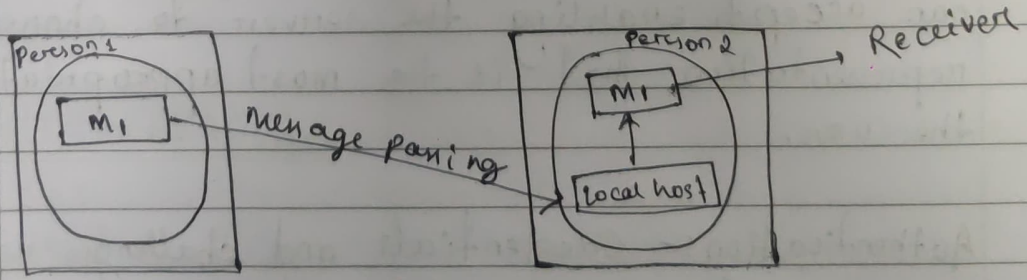
$\Rightarrow 7 = 2 + \frac{320}{\text{rate}}$

$\Rightarrow 5 = \frac{320}{\text{rate}}$ Data transfer rate = $\frac{320}{5} = 64$ B/ms

peer-peer
peer-pt
pt-peer

Point to Point communication in MPI

MPI :- Menage Paring Interface



Selected operations	Blocking	Non-blocking
Generic	MPI_send	MPI_Isend
Synchronyze	MPI_Ssend	MPI_Issend
Buffer	MPI_Bsend	MPI_Ibsend
Ready	MPI_Rsend	MPI_Irsend

HTTP

HTTP is a protocol that specifies the messages involved in a request-reply exchange, the methods, arguments and results, and the rules for representing them in a messages. It supports a fixed set of methods (GET, PUT, POST, etc). that are applicable to all of the server's resources. It is unlike the previously described protocols, where each service has its own set of operations. In addition to invoking methods on web resources, the protocol allows for content negotiation and password style authentication:

content negotiation: clients' requests can include information as to what data representations they can accept, enabling the servers to choose the representation that is the most appropriate for the user.

Authentication:- Credentials and challenges are used to support password-style authentication. On the first attempt to access a password-protected area, the server reply contains a challange applicable to the resource.

HTTP is implemented over TCP. In the original version of protocol, each client-server interaction consisted of the following steps :-

- The client requests and the server accepts a connection at the default server port or at a port specified in the URL.
- The client sends a request menage to the server.
- The server sends a reply menage to the client
- The connection is closed.

HTTP request menage

Method	URL	HTTP version	header	Menage body

HTTP Reply menage

HTTP version	status code	Reason	Header	Menage body

HTTP methods

GET
Head
Post
PUT
Delete
options
trace
Batch

HTTP Methods

1) GET → Request the resource whose URL is given as its argument. if the URL refers to data then the web server replies by returning the data identified by that URL. If the URL refers to a program then the web server runs the program and returns its output to the client.

2) HEAD → This request is identical to GET but it does not return any data however it does return all the information about the data (time of last modification, its type or its size)

3) POST → It specifies the URL of a resource that can deal with the data supplied in the body of request. This method is used when the action may change data on the server.

4) PUT → Request that the data supplied in a request is stored with the given URL as its identifier either as a modification of an existing resource or as a new resource.

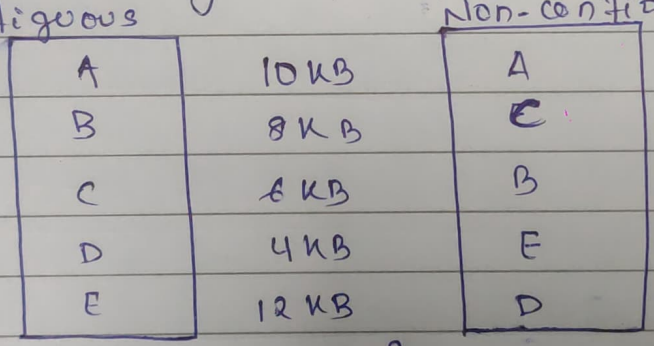
5) DELETE → The server deletes the resource identified by the given URL. Servers may not always allow this operation in which case the reply indicates failure.

6) OPTIONS → The server supplies the client with a list of methods. It allows to be applied to the given URL (for example → In the case of a special requirement or complex request, HTTP may use GET, HEAD, and PUT together)

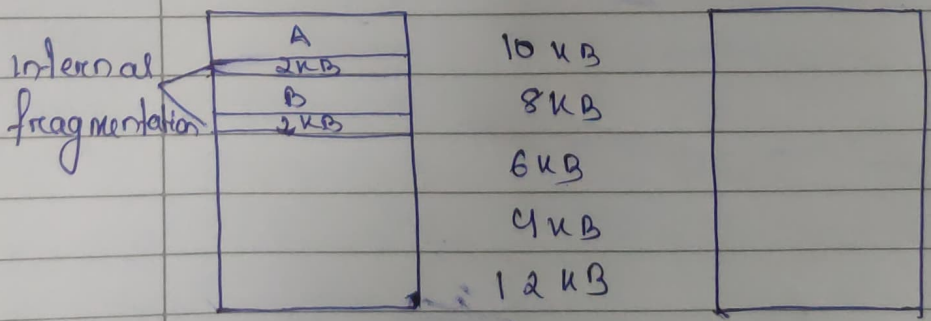
7) TRACE → The server sends back the request message.

Request based allocation and fragmentation in distributed systems

→ Contiguous allocation
→ Non-contiguous allocation



process Request = A, B, C, D, E



process Request = A, B, C, D, E
8KB 6KB 10KB → External fragmentation

Q. Memory block size :- 180 KB, 80 KB, 60 KB, 140 KB, 10 KB

(a) Process request size :- 180 KB, 80 KB, 59 KB, 140 KB

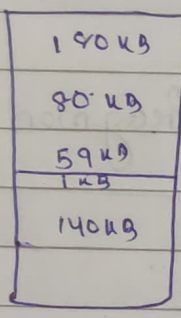
(b) Process request size :- 180 KB, 90 KB, 60 KB, 140 KB, 10 KB

(c) Process request size :- 160 KB, 60 KB, 40 KB, 250 KB, 10 KB

(d) Process request size :- 180 KB, 70 KB, 40 KB, 140 KB, 10 KB

classical example of fragmentation →

if allocation is contiguous then find the time of frag.



180 KB
80 KB
60 KB
140 KB
10 KB

(a) Internal fragmentation

(b) External fragmentation

(c) Internal & External fragmentation

(d) Internal fragmentation