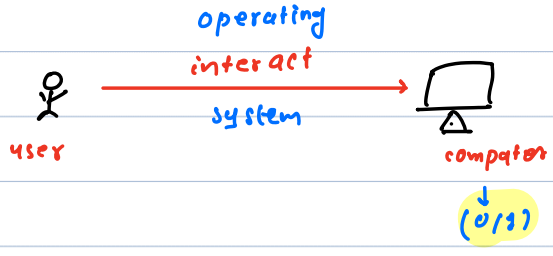
Operating System

How the application runs behind the scene.



it acts as an intermediary between computer hardware and user application

eg :-
- windows
- linux
- mac
- android
- ubunty

manager

↳ manage a project or team

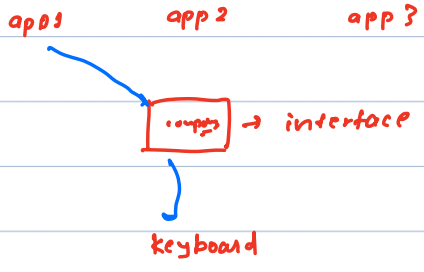
completion of project

time line.

can a single user used only one application?

chrome, hotstar, ... etc we have so many application running in the computer.

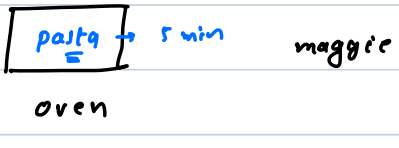
2) Resource management.



it help to shared the resources.

- interface between users and hardware.
- you just type google.com
- Abstraction of hardware complexition.

- uni programming
  - single
  - program
  - it can execute only a single program at a time.



we can't make two dishes at same time.

- once a program is starts executing. it continues till the program is completed or terminated.

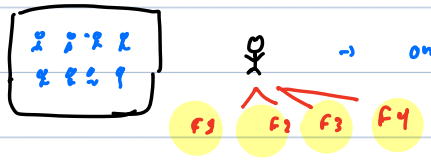
- multi programming it can execute multiple programs.

P1

P2

P3

P4



→ one food

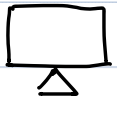
chef is multiple tasking.

- save time
- resource utilization.

sub categories of multiple prog.

- single user and multiple user
- single user

it supports only single user at a time



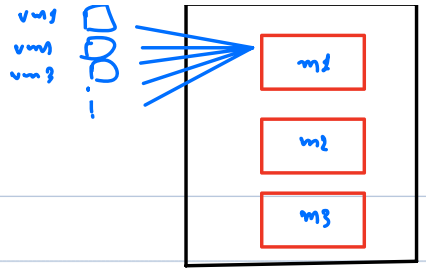
single person

multiple user

if supports multiple users simultaneously

eg: games.

AWS data centre



multiple users

1000 users

How we will run multiple program?

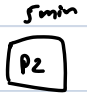
How OS is handle it?.

first P1 is execute and all are waiting after completion of P1 P2 will be execute and so on.

preemptive OS

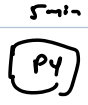
run program concurrency

→ t times 2 min (Partical part in each program)



→ execcute x times 1 min

→ execute y time 30 sec



Preemptive

Pause / switching a programme even if that not completed.

start P1

↓

execute P1 till t time (2min)

↓

switch to P2

↓

execute P2 till x time (3min)

↓

switch to P1

optimise utilization of CPU.

non-preemptive

- first we complete the program then we move to other Program.
- we have to wait till the program is execute.
- CPU is idle.

break till 10:15

Process

Chrome → download

Lines of codes in your app → Program → store in disk

↓

open / run

OS load the program in ram

↓

execution (when OS sends the program to execute)

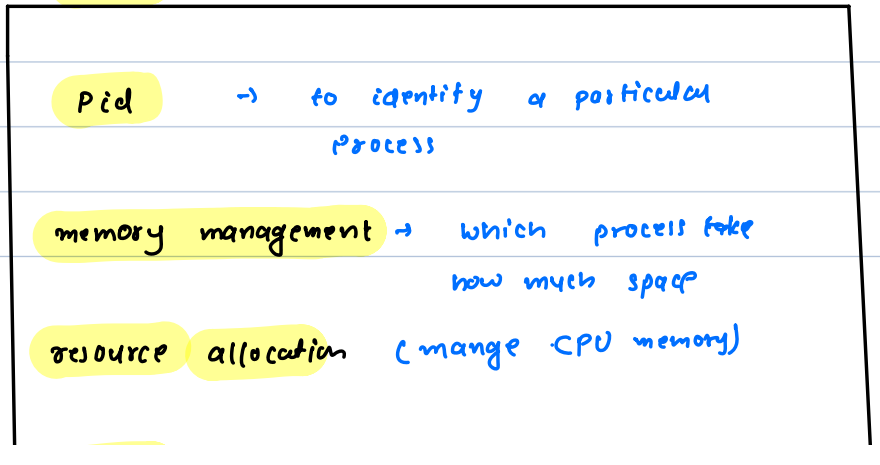
it is a process

- a process refers to an instance of program that is being executed by CPU

In a computer we have multiple process.

what information it contains.

PCB (process control block)



state → indicate process is running or completed.

priority :

implemet the process

```
class Process {
    int pid;
    string state;
    int priority
}
```

Process can be multiple types:-

1) I/O Bound process:

this type of process spend time on execution of I/O operations

eg: printer

2) CPU Bound process:

it primarily utilises CPU.

it spends minimum time waiting for I/O devices.

calculation

mining crypto currency

3) Mixed (I/O and CPU Bound) process

balance between CPU computation and I/O operations.

P1

CPU is idle → P2 → I/O operation

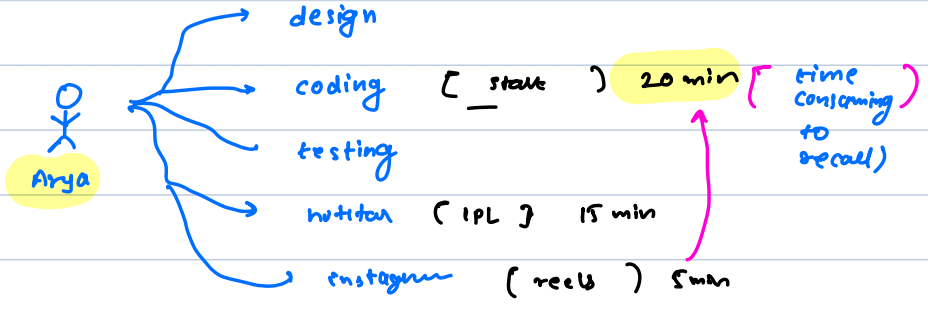
P3

CPU scheduling

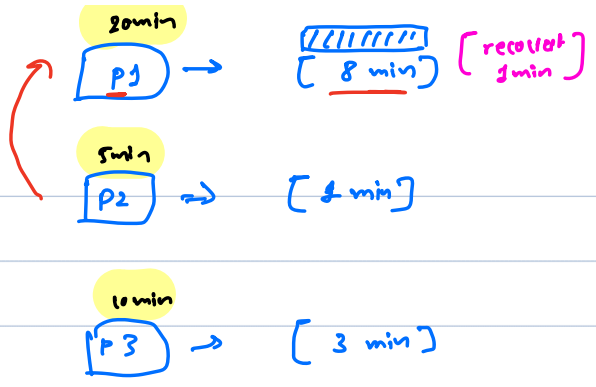
to improve efficiency / performance.

to increase throughput.

it is possible into preemptive program



- it recollecting the information that what he does in 20 min.



there is some delay during the switching

Context switching

it reads the state of the process

it execute the process

it has to save the proces.

it switch to other process

- if you have lot of process.

Good evening everyone

class start at 9:10

CPU scheduling algorithms

it is responsible for determining the order in which processes are executed.

P1

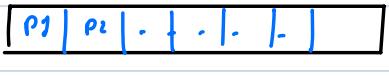
P2

P3

P4

1) **Arriving time (AT):**

the arrival time of a process is the time at which it enters a ready queue.



P1 → 3:05

2) **Burst time (BT):-** the amount of CPU time required by the process to finish its execution.

P1 takes 20 min to execute

3) **Completion time (CT)** : it is the time at which the process finishes its execution and exits the CPU.

P1 exit from CPU at 3:40

4) **Turnaround time (TAT)** it is difference between completion time and arrival time.

TAT of P1 is 3:40 − 3:05 = 35 min

5) **waiting time**. it is difference between TAT and burst time.

waiting time = (TAT − burst time)

P1 = 35 min − 20 min

= 15 min

FCFS (first come, first serve)

this is the simplest CPU algorithm.

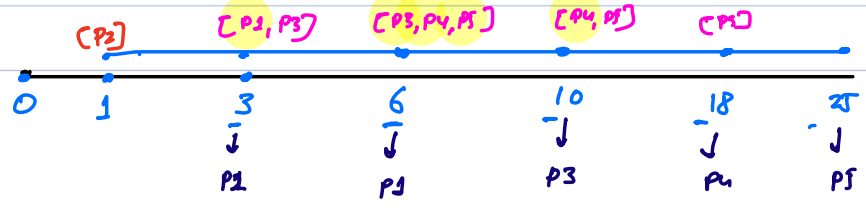
processes are executed in the order they arrive in ready queue.

- this algo is **non-preemptive** in nature.

queue in booking counter

P1	P2	P3	P4	P5

Pid	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P1	2	3	6	4	1
P2	1	2	3	2	0
P3	3	4	10	7	3
P4	4	8	18	14	6
P5	5	7	25	20	13

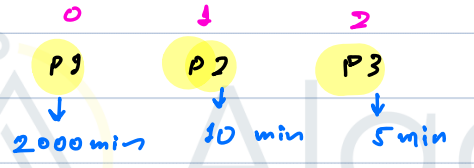


- FCFS is easy to implement.
- it has minimal overhead in terms of content switching.

if there tie breaking in arrival time.

A → 3 ms , B → 3 ms

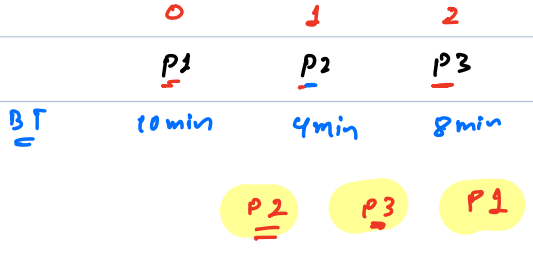
we pick minimum pid process first.



convoy effects where long processes ahead of short ones. lead to significant waiting time.

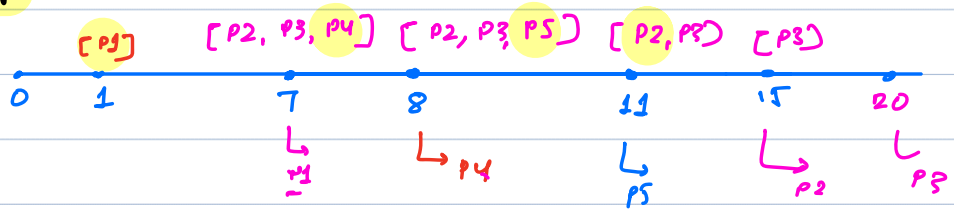
SJF (shortest job first) (non. preemptive).

SRTF (shortest remaining time first) (Preemptive)

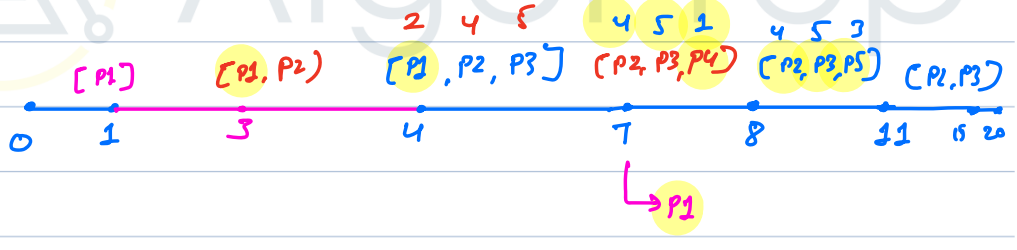


pid	AT	BT	CT
P1	1	6 3	7
P2	3	4	15
P3	4	5	20
P4	7	1	8
P5	8	3	11

SJF (non preemptive)

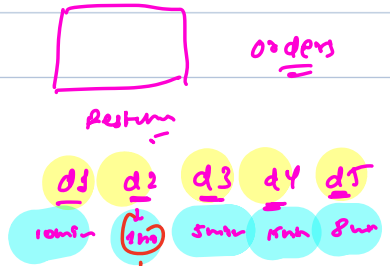


SRTF (preemptive)



- minimum average waiting time
- faster response time

0	1	2	3	4
P1	P2	P3	P4	P5
10 min	10,000	5 min	5 min	1 min



↓
indefinite time

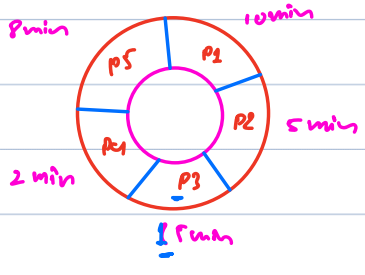
starvation of long process.

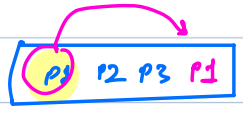
long processes may suffer from starvation if shorter processes continously arrive.

break till 10:30

Round robin (RR)

- it work in cyclic manner.
- each process receives a small unit of time. called a time quantum
- each process execute for a limited time. before being switch it placed back in ready queue.





4 min

Process

remaining BT ≥ qt

↓

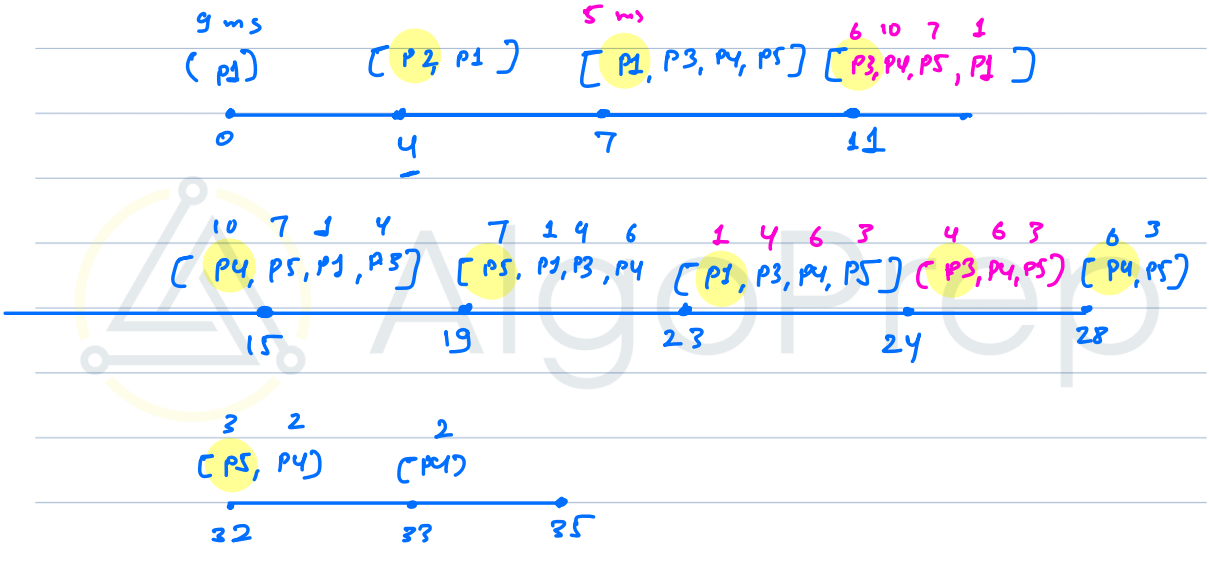
we execute the proces till qt and switch to next process

remaining BT < Quantum time

after completetion of the process we switch to next process

quantum time 4ms

Pid	AT	BT
P1	1	9 ms
P2	3	3ms
P3	5	6 ms
P4	6	10 ms
P5	7	7 ms

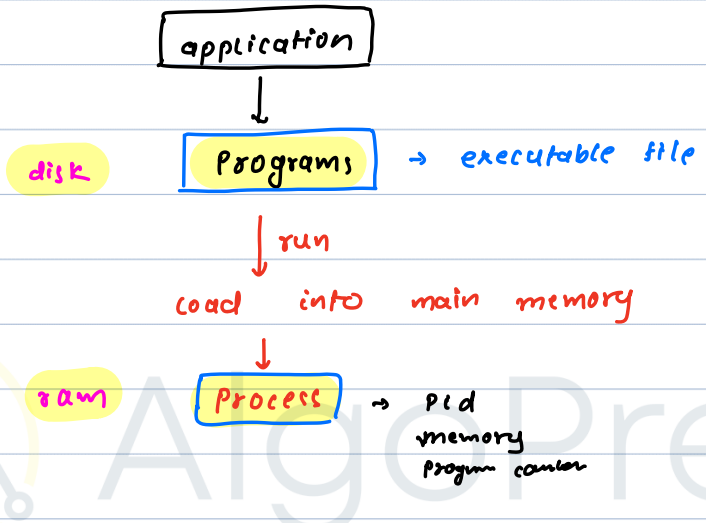


- if the quantum time is minimum.
  frequent preemtion or switching may occur.
- it leads to the high overhead due to content switching.

- what happen if quantum time is too large.
  limited context switching.
  it is now become similar FCFS algo.

Thread

how process will be created ?



we want this one by one or at the same time:

How we get all the features at same time.

Threads

we achieve this with multi threading.

- A thread is the smallest unit execution within a process.

- In each process we have one or many threads.

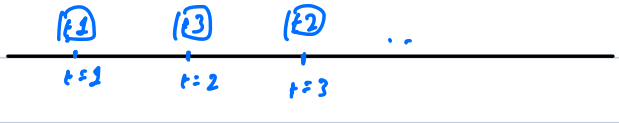
- How we actually run all the functionalities at same time.

single core vs multiple core

single core processor

if consist only single core.

tasks are executed one a time.



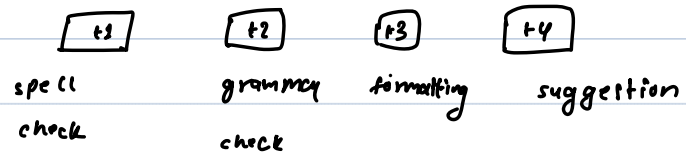
execute t1 then switch to t2.

we are not get parallelism.

we can't executed two or more thread at some time.

multiple core processor

it contains multiple cores.



it is allowing for true parallel execution.

GPU

↳ 100 or 1000 cores.

we have many more functionalities in game

executes many task simultaneously.

4 cores → 50 k.

1000 cores → 250 x 50k

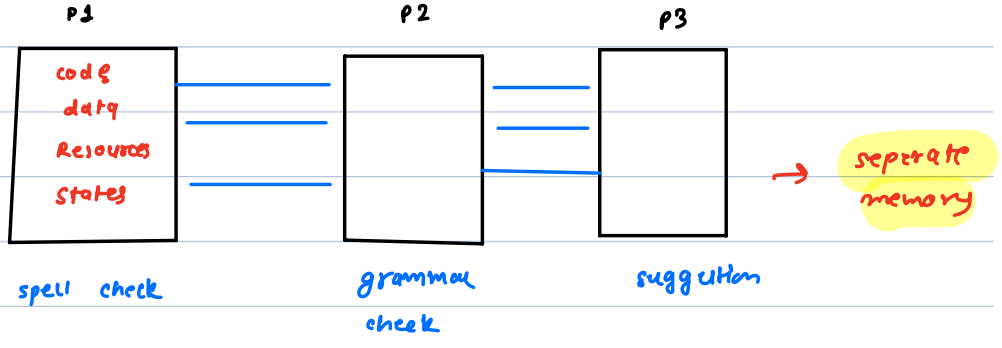
CPU

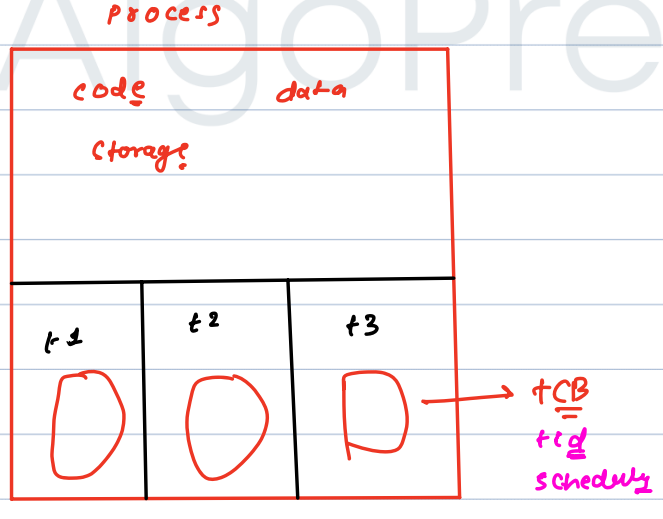
2 GHz - 3 GHz

cpu is more powerful than GPU cores.

0.5 GHz - 2 GHz

Process vs thread





- process take more storage as compared to thread
- all the threads within the process using same memory.
- higher overhead due to seperate memory space and content switching.

- In threads, we have efficient context switching due to share memory.

data sharing

inter. communication process (expensive)

to communicate or sharing data IPC is required in process.

- thread communicate directly with other thread due to shared memory.

Concurrency and parallelism

Single core process

T1 T2 T3 T4

T1 T3 T2 T4

we can execute multiple thread.
we run the thread in concurrent.

multiple core

c1

c2

c3

In this case threads are executing in parallel.

break till 10:13

In each process we have atleast one thread

to get the name of current thread.

```
Thread.currentThread().getName()
```

there are two ways to create thread in java.

1) to extend Thread class.
2) to implement the runnable interface.

Step 1:- make a class for new thread.

```
class Thread1 extends Thread {

    public void run() {
        Syso( thread 1 is running);
    }

}
```

step 2 call the thread from main class

```
main {

    Thread1  t1 = new Thread1();

    t1.start();

}
```

we can create thread using runnable interface.

```
class Thread1 implements Runnable {

}
```


step 2. call the thread from main class.

```
Thread 1  t1 = new Thread1();

Thread At1 = new Thread( t1 );

At1.start();
```

thread are executing parallel.

thread is not executing sequentially.

```
class Thread1 extends Thread {
    private int printnum;
    //constructor
     public Thread1(int i){
       this.printnum = i;
     }
 public void run(){
    System.out.println("thread is running hii " + printnum);
    System.out.println("curr thread :" + Thread.currentThread().getName());
}
}

// class Thread2 extends Thread {
//  public void run(){
//    int i = 100;
//    while(i-->0){
//     System.out.println("thread 2 is running hey");
//     System.out.println("curr thread Thread1:" + Thread.currentThread().getName());
//   }
// }
// }

// class Thread1 implements Runnable {
//  public void run(){
//     int i = 100;
//   while(i-->0){
//     System.out.println("thread 1 is running hii");
//     System.out.println("curr thread Thread1:" + Thread.currentThread().getName());
//   }
// }
// }

// class Thread2  implements Runnable {
//  public void run(){
//    int i = 100;
//    while(i-->0){
//     System.out.println("thread 2 is running hey");
//     System.out.println("curr thread Thread1:" + Thread.currentThread().getName());
//   }
// }
// }
```

```java
public class threadcl{

    Run | Debug
    public static void main(String[] args) {
    // System.out.println("hey everyone");
    // System.out.println("curr thread main func:" + Thread.currentThread().getName());
    //   Thread1 obj1 = new Thread1();
    //   Thread t1 = new Thread(obj1);
    //   t1.start();

    //   Thread2 obj2 = new Thread2();
    //  Thread t2 = new Thread(obj2);
    //   t2.start();
    //     // fun1();
    // }
// how we can print 1 to 100 with different threads
    for(int i = 1; i<=100; i++){
      Thread1 t1 = new Thread1(i);
      t1.start();
    }


    }
    // public static void fun1(){
    //     System.out.println("hey everyone from fun1");
    //     System.out.println("curr thread of fun1:" + Thread.currentThread().getName());
    // }

}
```

print 1 to 100 with seperate threads:

the order of output is not correct.

sleep() this method is used to pause the execution of the currently running thread.

sleep(long milliseconds)

↳ How much time you want to pause that thread.

- this method may be give some exception.

Interrupted execution.

try catch method to handle exception.

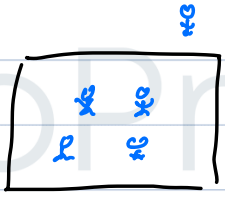
throws the exception.

Q timer print 1 then after 1sec 2 .. so on.

there is always a thread (main thread)

```
public main () {        ( main thread )
int sec = 10;
    Sys() ( " times started");
        while ( sec ≥0 )
        ( point the number
          decrease the number
          wait for 1 sec )
        syso ( " time up");
}
```

join() this method is used to wait for a **thread** to completely execute.



you are waiting to leave the guard.

thread → print (1 to 100)

```
main ( ) {
    Syso ( "  started");
    thread t1 = new thread();
    t1. start();
      t1. join();      ( waiting to complete this thread)
    Syso ( " completed");
}
```

main thread is wait till thread t1 is not

completely executed.

Q) 1 to 100 with seperate threads.

creating 100 threads.

Q) 1 to 10,00,000 with seperate.

create 10,00,000.

- it is not efficient way.
- it is responsibility of user to manages all the threads.

to solve this issue.

executor framework (event manager)

it take all the responsibility.

How many thread will be created.

if manages all the threads.

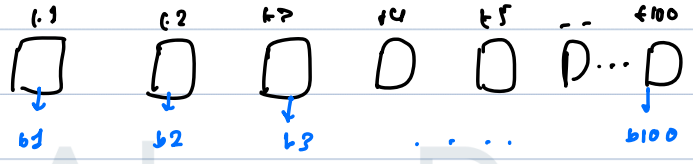
if improve your system performance.

user → give the task to the executor.

create the task and give to executor.

How if manages all the threads.

you have mautacture 100 bikes ?



make 100 -production unit.

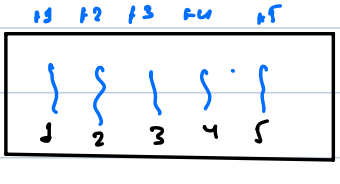
get orders of 1000 unit.

we are waiting lot of resources.

↓

to deal with this issue

Thread pool → manages all the threads



1 to 100.

task 6

after give task to t5 t1 is available

assign the new task to the t1.

we can reuse the thread.

break till 10:23

thread pool

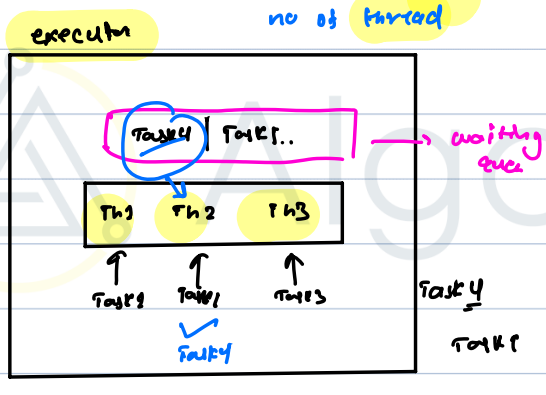
fixed

predefine no. of threads

- if you know workload.
- if you have limited resources.

new FixedThreadPool(x):

↓ no of thread



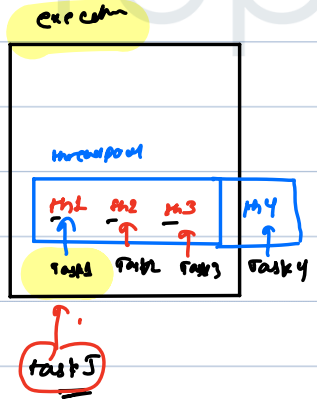
dynamic

it is automatically created no of threads as per requirements.

if workload is vary then we can use this approach.

- optimise resources utilities.

newCachedThreadPool();



- when your work is done still want to continue your service or terminate the service.

executer.shutdown(); → to terminate the service.

Q) is thread returning something?

to get return some data from thread

callable → it is similar to runnable.

↳ it creates the task.

MS word

↓

spell check

clrea clear

class myCallableThread implements Callable<Integer> {

int num;

make constructor

type of data we want.

public Integer call() throws exception {

return num*num;

}

}

main() {

ExecutorService es = Executors.newCachedThreadPool();

Callable<Integer> task1 = new myCallableThread(10);

result is not in order

Future<Integer> f1 = es.submit(task1);

↓

future object.

sysO (f1.get());

```java
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class thread5 implements Runnable{

  //create constructor
  private int number;   //1 , 2, 3 ,..100
  public thread5(int num){
      this.number = num;
  }

  public void run(){
      System.out.println("my task is :" + number + Thread.currentThread().getName()
  }

}

public class task1 {
    Run | Debug
    public static void main(String[] args) throws InterruptedException  {
        //print 1 to 100 using seperate threads
    //ExecutorService executorFix = Executors.newFixedThreadPool(8);
    ExecutorService executorVar = Executors.newCachedThreadPool();


    for(int i = 1; i<=100; i++){
        thread5 tk = new thread5(i); //create the task
         executorVar.submit(tk);
         // Thread t5 = new Thread(tk);
         // t5.start();
         // Thread.sleep(100);
    }

 System.out.println(x:"this is end");
  executorVar.shutdown();
    }
}
```

Good evening everyone 

class start at 9:10

```
main ( ) {
    ==========
Thread t = new Thread( new Runnable() {

    public void run() {

        for( int i=1; i<1000; i++) {
            sysO ( " I am inner thread " + i);
        }
    }
});
t.start();
```

(10) lines

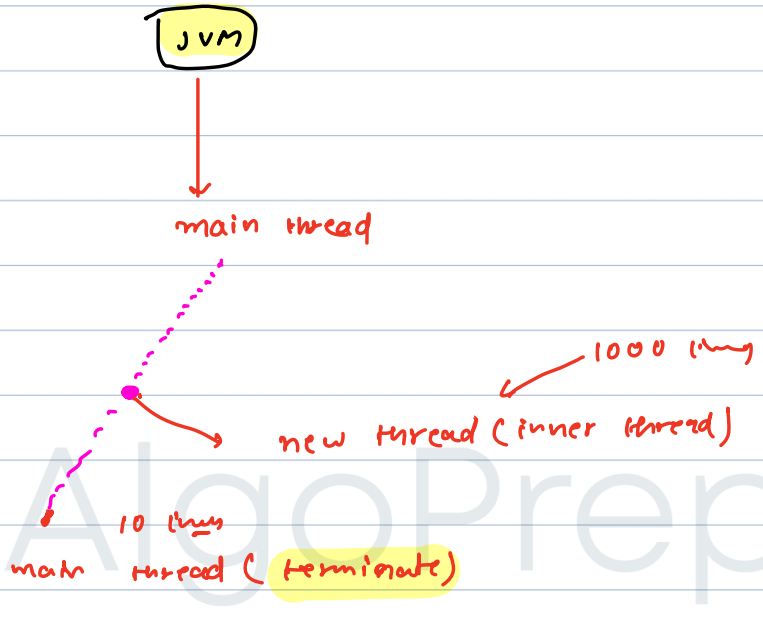
← last line of main thread.

```
}
```

1) when main thread is exit. then inner thread also terminated. or till it is

not exit it print few lines of inner thread then terminate after the main thread is exit.

2) inner thread is completely execute. even if the main thread is exit.



when i exit from the main thread then all the thread is terminated.

we set the inner thread to daemon thread.

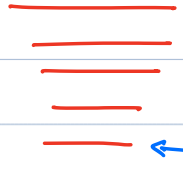
default thread is non daemon.

```
main ( ) {

Thread t = new Thread( new Runnable() {

public void run() {
```

```
public void run() {
    for (int i=1; i<1000; i++) {
        System.out.println("I am innerthread " + i);
    }
}
};
t.start();
t.setDaemon(true);
```



(10) line

← last line of main thread.

if we set the thread to daemon so when all the non daemon thread is executed the daemon thread will be killed.

Synchronization

```
totalProfitLoss
Amount(
  value =
)
```

Profit Addition

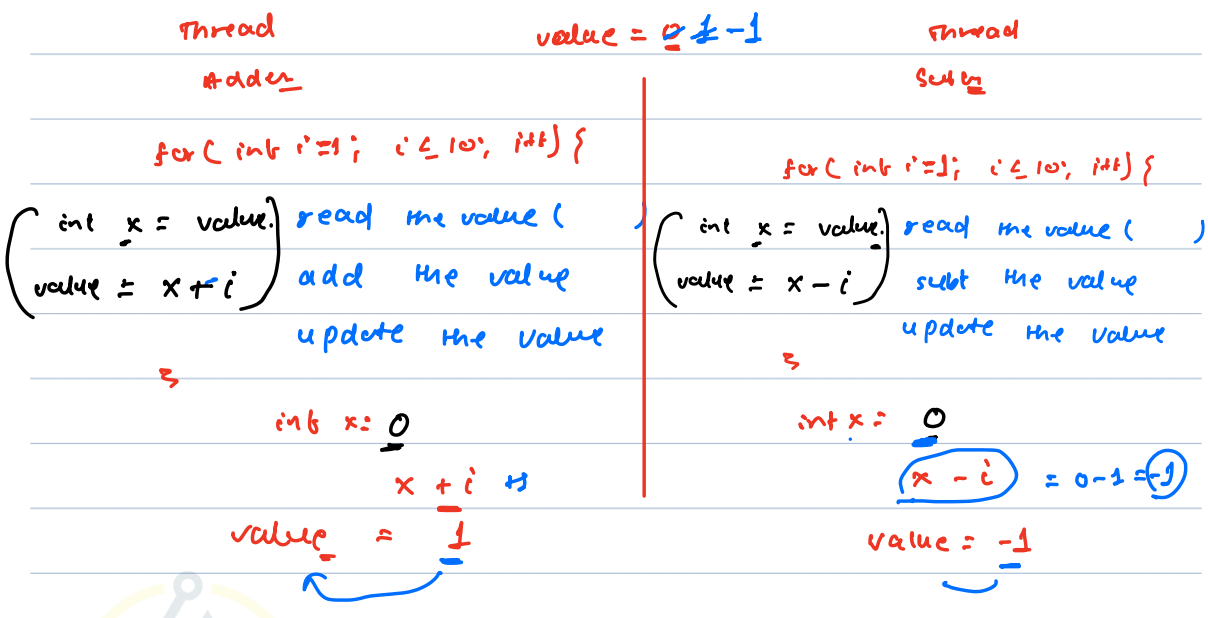
```
for(1 to 100){
amt.value += i;
}
```

Loss Subtraction

```
for (1 to 100){
amt.value -= i;
}
```

overall we get zero no profit no loss.

we get different value every time.



it happen when two or more thread is accessing the same variable or shared variable.

it is called synchronization.

1) outcome of the program becomes unpredictable and may vary depending on the timing. This is called race condition.

you get wrong values.

2) critical section

A critical section is a segment of code in program that accesses shared resources

such as variables.

3) Preemptive → pause and switch to another program.

Amount ((value) 50 42 56)

Thread 1

print A .

print B

dor(int i=0; i<10; i++) { 6

int x = amt.value; 50

x = x + i; 56

amt.value = x; 56

}

Thread 2

print A .

print B

dor(int i=0; i<10; i++) { 8

int x = amt.value; 50

x = x - i; 50-8=42

amt.value = x; 42

}

switch

answer should be 48

if the critical part of a program behave preemptively.

thread can be interrupted or suspended while it is executing the critical section.

↓ to deal this we use synchronization mechanism

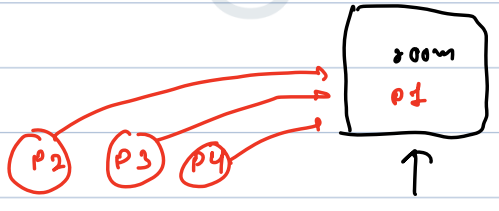
break till 10:35

what are the property for a good mechanism?

mutual exclusion: it ensure that at a time one thread can access a shared resources. we can prevent race condition.

liveness or progress :. it refers to a property that a system should continue to make progress even in the presence of various concurrency related issues.

no busy waiting



allowed one person at a time

we are not doing any progress

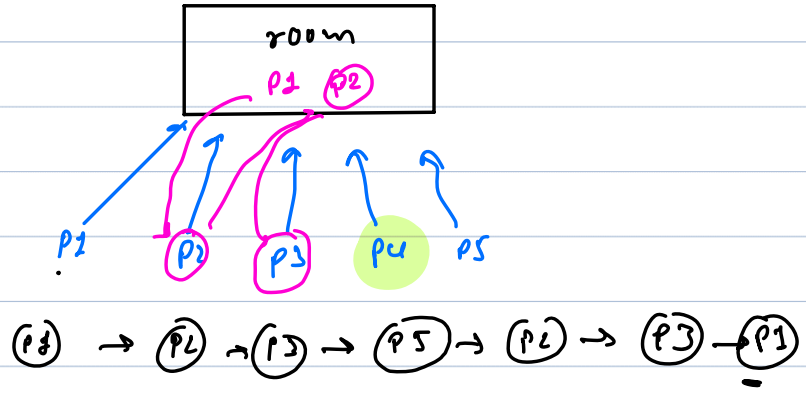
```
while( room not empty ) {
    knock the door
    waiting. to open the door.
}
```

way 2

we get signal or notify other room is empty.

Instead of waiting we can do other work.

bounded wait or fairness



no thread is unfairly kept waiting indefinitely to enter a critical part.

synchronisation mechanism

1) mutex → (lock)

mutual exclusion

- to achieve this mechanism we use locks on the critical part.
- A thread must take a lock before entering in critical section.

- before exit from the CS we should unlock it.
- it acts as a gatekeeper

Thread 1

print A

print B

for(int i=0; i<10; i++) {

int x = amt.value; 50

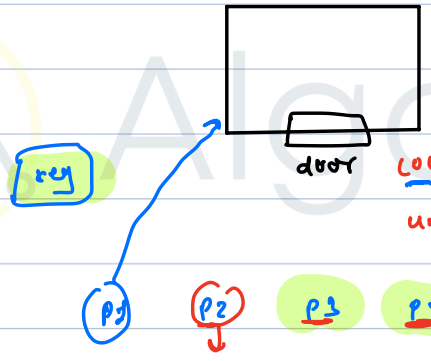
x = x + i; 56

amt.value = x; 56

}

apply lock

switch



not able to access

mass starvation

we have only one key and if we lost then there is scenario of mass starvation.

if two thread it lock the critical part

p1 lock the door

p2 lock the door

it create deadlock situation.

create a lock and send to thread

apply lock before critical section and unlock after critical.

synchronised (Account acc) {

critical section.

3

```
public class task3 {
    Run | Debug
    public static void main(String[] args) {
     System.out.println(x:"enter in main thread");

     Thread t = new Thread(
     new Runnable(){
      public void run(){
        //create task
        for(int i = 1; i<=100; i++){
            System.out.println("i am inner thread "+i);
        }
      }
     });
      t.setDaemon(on:true);
      t.start();


     for(int i = 1; i<=10; i++){
        System.out.println("exit from main thread "+i);
     }
    }
}
```

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class Addt implements Runnable{
    private Amount amt;
    private Lock lk;
    public Addt(Amount a , Lock lk){
        this.amt = a;
        this.lk = lk;
    }

    public void run(){
       for(int i = 1; i<=100000; i++){
          lk.lock();
          this.amt.val += i;
          lk.unlock();

       }
    }

}

class Subbt implements Runnable{
    private Amount amt;
    private Lock lk;
    public Subbt(Amount a ,Lock lk){
        this.amt = a;
        this.lk = lk;
    }

    public void run(){
       for(int i = 1; i<=100000; i++){
          lk.lock();
          this.amt.val -= i;
          lk.unlock();
       }
    }

}

```

```java
class Amount{
 int val;
}


public class task4 {
    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        Amount amt = new Amount();
        Lock lk = new ReentrantLock();
        Addt tk1 = new Addt(amt,lk);
        Subbt tk2 = new Subbt(amt , lk);
        Thread t1 = new Thread(tk1 );
        Thread t2 = new Thread(tk2 );

        t1.start();
        t2.start();

        Thread.sleep(millis:1000); //pause the main thread

        System.out.println(amt.val); // what is the value

    }
}

```

Good evening everyone

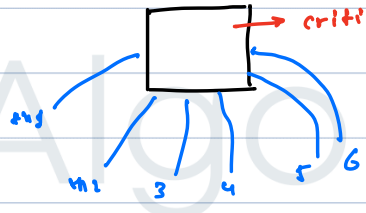
class start at 9:10

Synchronization problem

mutex → its use locks.

How many threads are allowed in critical section of one time ?.

→ 1

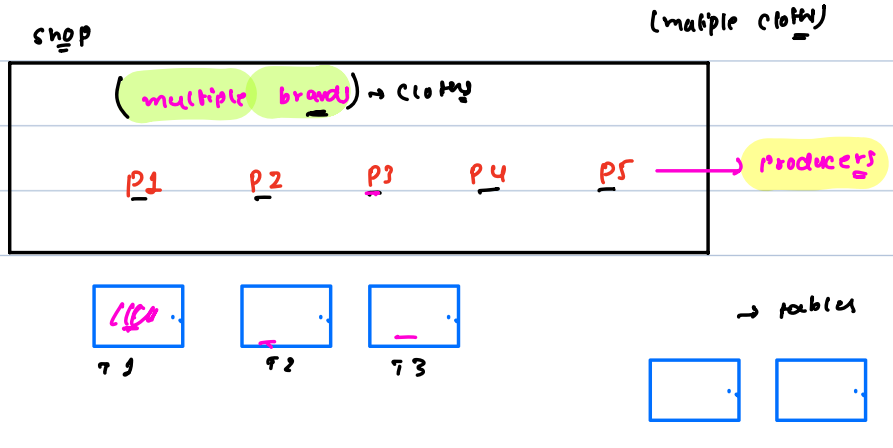


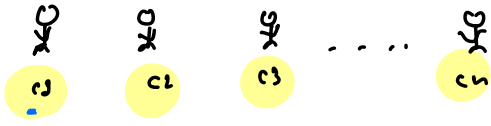
we apply lock before enter in CS and unlock when exit.

Semaphores

dijkstra 1965

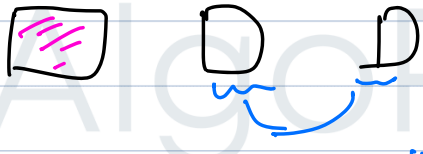
Producer - consumer problem





→ consumers

- A producer can only represent one cloth on a table.
- to represent the cloth, table must be empty.
- they want to buy some cloths. it is only possible when the table is not empty(cloths)
- How many producers can represent the cloth

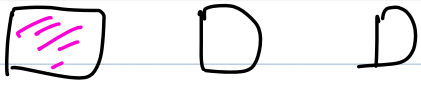


2

two available table.

number of empty tables.

- How many cutomers can buy the cloth.



1 → number of occupied table.

{ total no of available cloth).

now we have to make program for that.

Queue < cloths> shop;

maxsize = 3; → capacity of the shop (numbers of tables)

gnitially queue it empty.

shop :

Producer

```
if( shop.size() < maxsize){
    shop.add( new cloth());
}
```

we can't add more than the maxSize.

Consumer

```
if(shop.size() >0){
    shop.remove();   // purchase.
}
```

we can only purchase the cloth when it is available.

P5

```
if( shop.size() < maxsize){
    shop.add( new cloth());
}
```

P6

```
if( shop.size() < maxsize){
    shop.add( new cloth());
}
```

at the same time both want to represent the cloth.

soulution would be lock.

we have apply lock on the entire.
we can apply lock on each cell.

it can be stuck in deadlock situation.

is it fast enough?

mutex mechanism is slow.

- is there any way that multiple thread could safely access the resources.

break till 10:34

semaphore

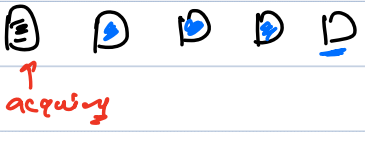
it is also control access to a shared resource.

it can be initialized with an intial number of permits.


```
Semaphore sempore = new Semaphore( 5 );
```

s.acquire(); // similar to lock.

available limit = 1



acquire

when you acquire the available limit is decrease by 1.

s.release(); available limit is increase by 1.

prod semapore → available resp → no. of table
max limit

cow semapore → (0) → available resource.

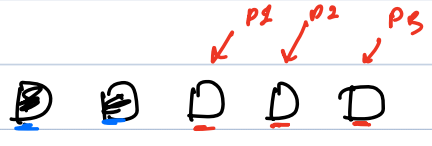
initially all the table is empty

Producer

```
run() {
    semaProd.acquire();
    shop.add(new Object());
    semaCons.release();   → increase resource limit.
}
```

Consumer

```
while(true) {
    semaCons.acquire()
    shop.remove();
    semaProd.release();
}
```



no. of producer enter = 3

no of consume enter = 2

= 5

```java
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedDeque;
import java.util.concurrent.Semaphore;

class Producer implements Runnable{
    private Queue<Object> shop;
    private int maxLimit;
    private String name;
    private Semaphore semaProd;
    private Semaphore semaCons;
    public Producer(Queue<Object> shop , int maxLimit , String name , Semaphore sp , Semaphore sc){
        this.shop = shop;
        this.maxLimit = maxLimit;
        this.name = name;
        this.semaProd = sp;
        this.semaCons = sc;
    }
    public void run(){
      int i = 10000;
      while(true){
        try {
            semaProd.acquire();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
            System.out.println("occupied table -"+shop.size() + " prod name -" +name);
           shop.add(new Object());
           semaCons.release();

      }

    }
}

class Consumer implements Runnable{
    private Queue<Object> shop;
    private String name;
    private Semaphore semaProd;
    private Semaphore semaCons;
    public Consumer(Queue<Object> shop  , String name ,  Semaphore sp , Semaphore sc){
        this.shop = shop;
        this.name = name;
        this.semaProd = sp;
        this.semaCons = sc;
    }
```

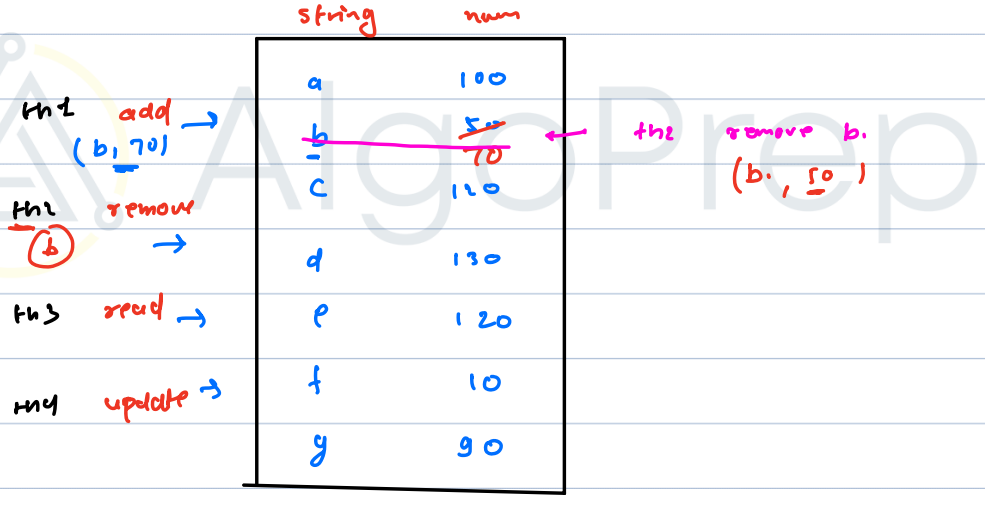
```
46      public void run(){
47        int i = 10000;
48        while(true){
49          try {
50              semaCons.acquire();
51          } catch (InterruptedException e) {
52              e.printStackTrace();
53          }
54              System.out.println("occupied table -"+shop.size() + " cons name -" +name);
55              shop.remove();
56             semaProd.release();
57        }
58    }
59  }
60  public class task5 {
        Run | Debug
61      public static void main(String[] args) {
62         Queue<Object> shop = new ConcurrentLinkedDeque<>();
63         int maxLimit = 5;
64
65        Semaphore semaProd = new Semaphore(maxLimit);
66        Semaphore semaCons = new Semaphore(permits:0);
67
68         Producer prod1 = new Producer(shop , maxLimit , name:"prod1" , semaProd , semaCons);
69         Producer prod2 = new Producer(shop , maxLimit , name:"prod2", semaProd , semaCons);
70         Producer prod3 = new Producer(shop , maxLimit , name:"prod3", semaProd , semaCons);
71
72        Consumer cons1 = new Consumer(shop,  name:"cons1", semaProd , semaCons);
73        Consumer cons2 = new Consumer(shop,  name:"cons2", semaProd , semaCons);
74        Consumer cons3 = new Consumer(shop, name:"cons3", semaProd , semaCons);
75        Consumer cons4 = new Consumer(shop,  name:"cons4", semaProd , semaCons);
76        Consumer cons5 = new Consumer(shop,  name:"cons5", semaProd , semaCons);
77
78          Thread tprd1 = new Thread(prod1);
79          tprd1.start();
80          Thread tcons1 = new Thread(cons1);
81          tcons1.start();
82          Thread tcons2 = new Thread(cons2);
83          tcons2.start();
84          Thread tprd2 = new Thread(prod2);
85          tprd2.start();
86          Thread tcons3 = new Thread(cons3);
87          tcons3.start();
88          Thread tcons4 = new Thread(cons4);
89          tcons4.start();
90          Thread tprd3 = new Thread(prod3);
91          tprd3.start();
92          Thread tcons5 = new Thread(cons5);
93          tcons5.start();
94
95      }
96  }
97
```

Good evening everyone

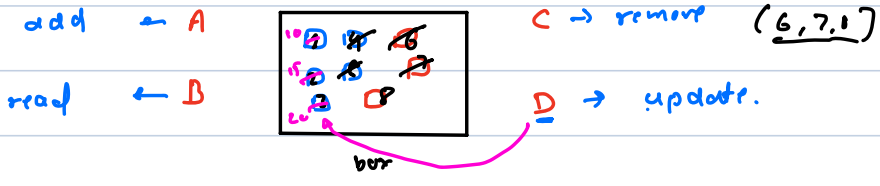
class start at 9:10

Semaphores

Hashmap → synchronize and concurrent Hashmaps.



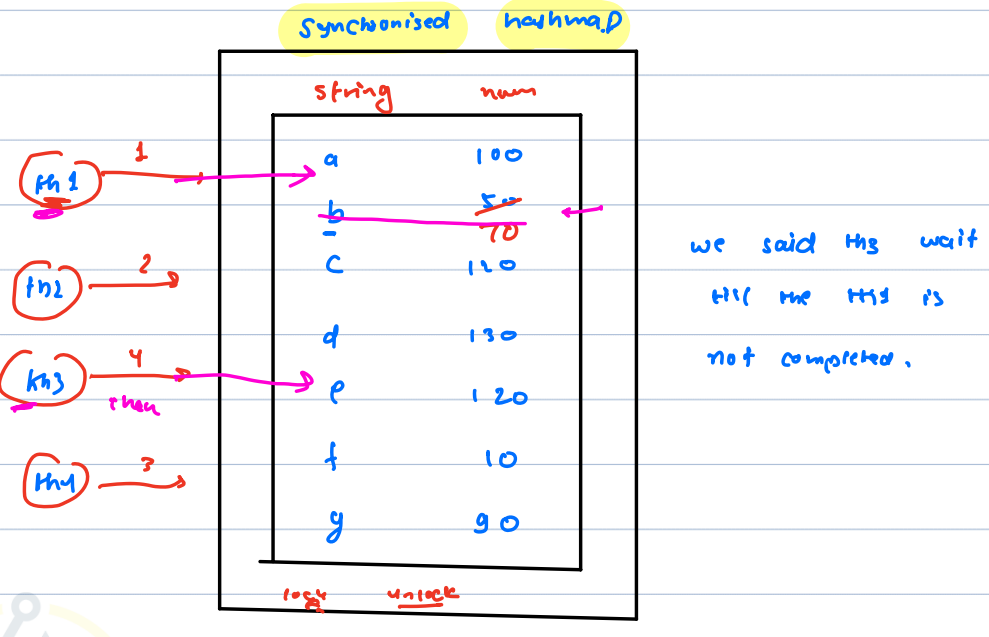
all the threads are try to access the map at same time.



A think that he add box 6 ?!

B told read data is 1, 2, 3

inconsistency happen in this.
if occurs race condition.



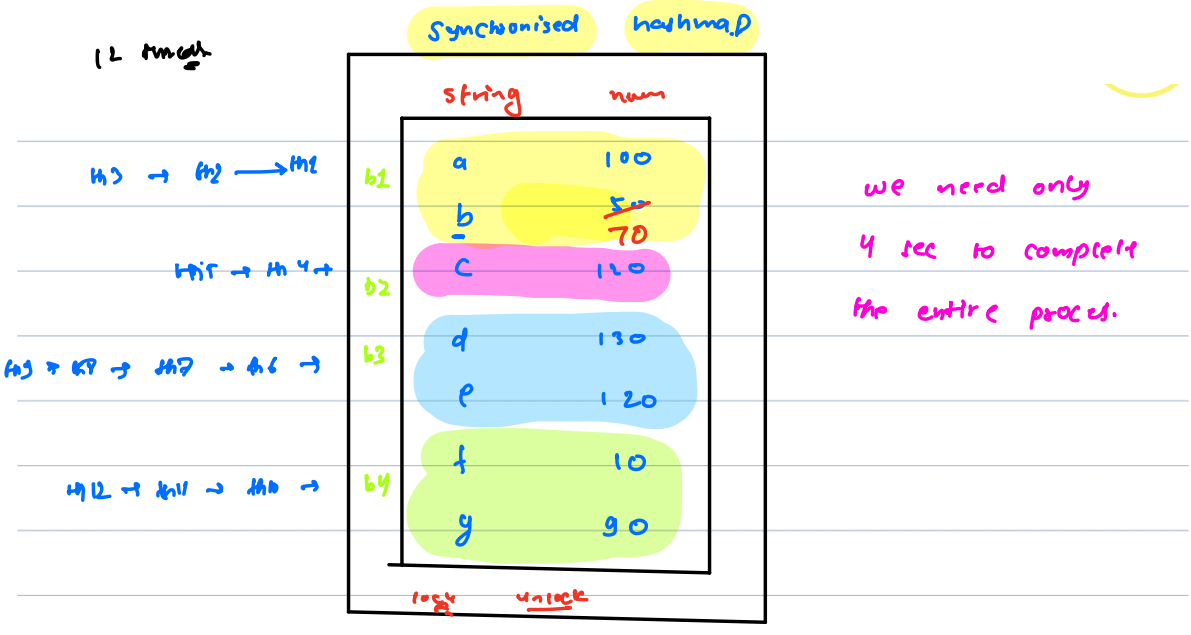
we said thg wait till the thg is not completed.

. when a person enter in this map then we lock the entire map.
. now all threads are going one by one.

if we have 10 thread at same time
1 thread can take 2 ses.
total time it takes 20 sec

Concurrent Hashmap

if allow multiple threads to operate on the map but. synchronised bucket inside map not the entire map.



we need only 4 sec to complete the entire proces.

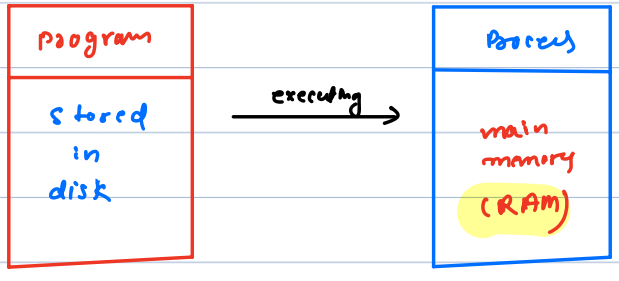
now we are synchronised buckets only.

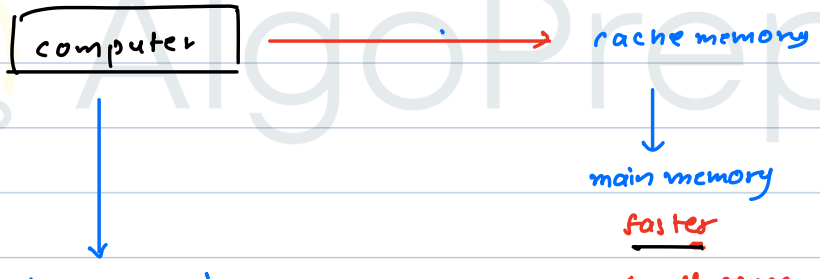
if is now support multi threading environment.

we improve the overall performance.

Memory management

how data is stored in the process.





cache memory and main memory is closed to the computer.

In the disk we have huge storage and in the ram we have less storage

need 1 tB / 4 GB ram

searching space is larger in disk and

smaller in the ram.

access the data in ram is much faster than the disk.

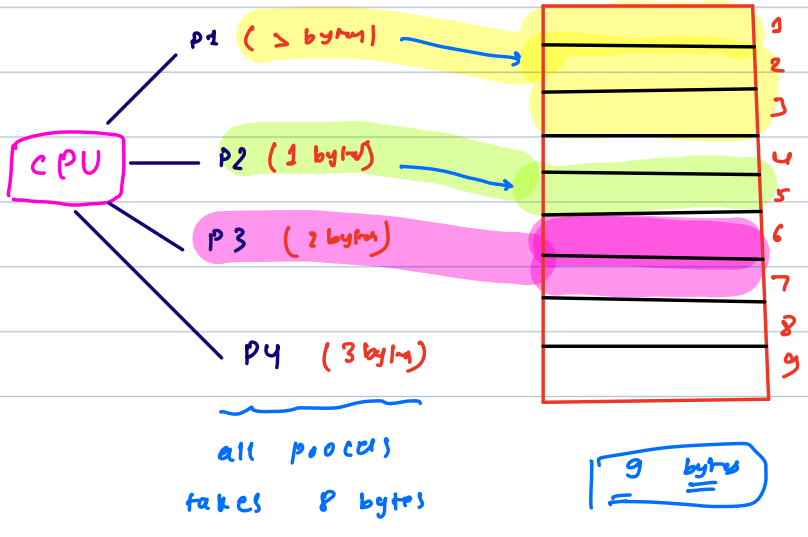
space	speed
Bigger	→ slower

due to speed issue that disk is very slow CPU can't talk with the disk.

if a game required 8 gb. and the ram of the phone is only 6 gb. still game is run in your phone.

How it is possible?.

How the data is stored in the memory.

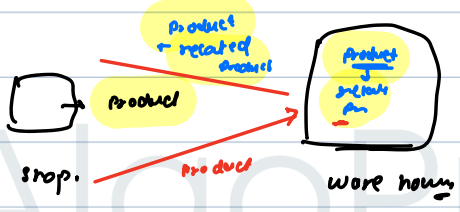


1) Contiguous memory Allocation

- each process is stored in a single contiguous block.

- it is easy to implement

- when we ask CPU to get some data then it take some nearby data as well.



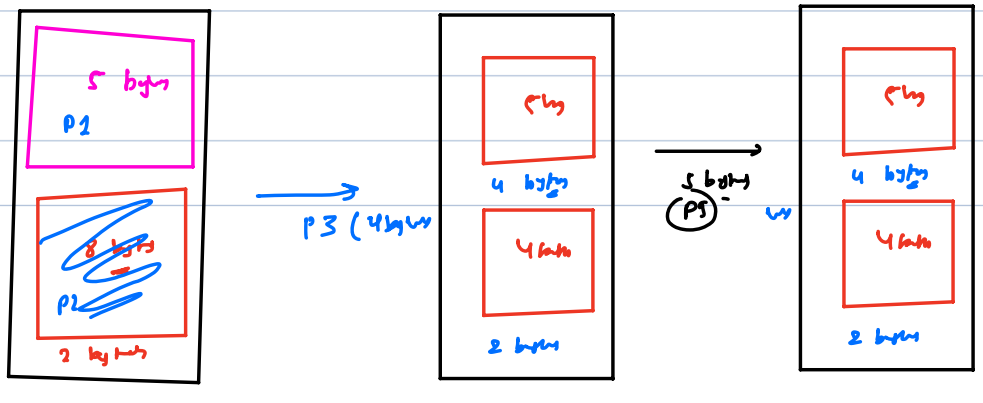
we save some time to not going everytime.

cons

fragmentation → it leading to inefficient use of memory.

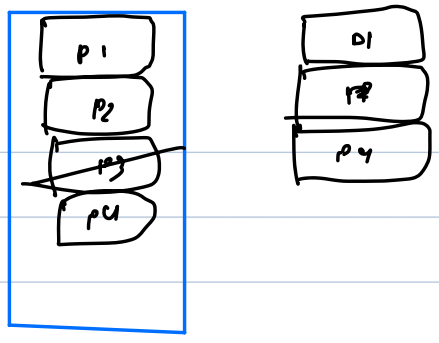
↳ divides memory into smaller parts

→ wastage of memory.



15

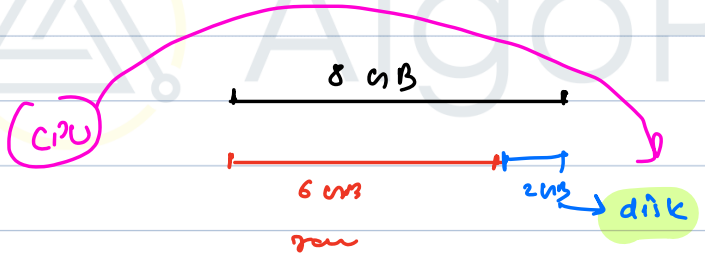
Defragmentation tools



break till 10:25

if a game required 8 gb. and the ram of the phone is only 6 gb.

Paging → it is a memory management used by operating system to efficiently manage memory in system.



whatever the space is available in your ram use this and store as much as you can and everything is stored in the disk.

if CPU tried to access some data which is present in the disk then first it brought to the main memory then give access to the CPU..

everything is handle by MMU (memory management unit)

Logical address (conceptual address)

Physical address (real address) (location, actual)

when a program is running a logical address is generated. and we reach to the physical address using this address.

MMU using page table to execute this

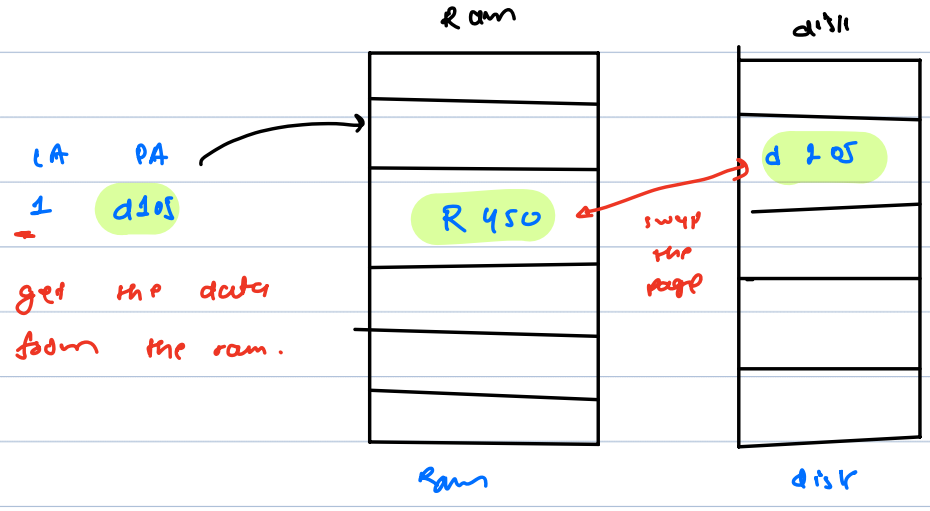
it map the logical address to physical address

- paging divides memory into fixed size of block known as pages → logical address of space

similarly it divides ram into fixed size of block called frames.

Page table

LA	PA
1	M 3FY)
2	D 857
3	D F66
4	M 32410
5	
6	



In the most case ram is full

Page swapping Algo → that how data it swap and which one.

FIFO

1 → 22:05
2 → 12:30
3 → 09:40
4 → 10:20
5 → 8:50
6 → 10:40

Page 5 come first in the memory. so we swap Page 5 to get the data.

LRU

↓

least recent used

1 → 5 min ago
2 → 2 min ago
3 → 9 min ago → replace page ?
4 → 7 min
5 → 1 min

no space limit and no memory waste

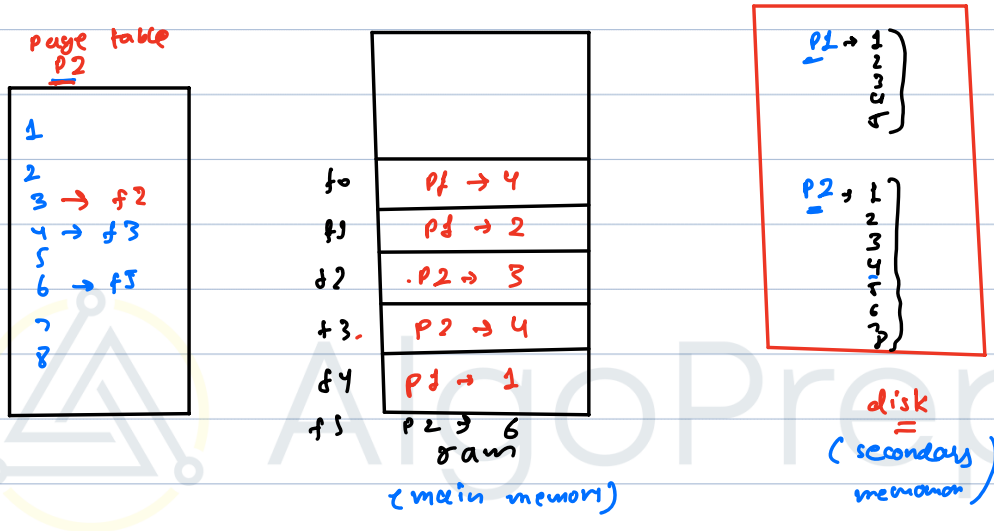
complex to implement.

page fault: it occurs when CPU asks data from ram but it is present on disk.

Good evening everyone

class start at 9:10

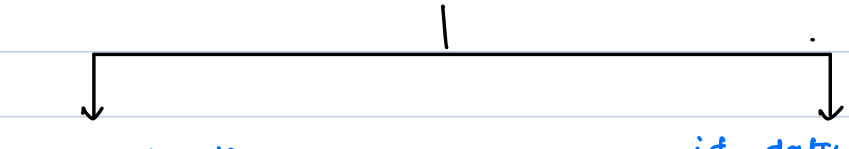
Paging



we need data of page 5 of process 2.

we create different page table for each process.

. if is stored in main memory.



if data is already present in your ram (main memory) the CPU can directly

if data is not present in ram. it trigges page fault.

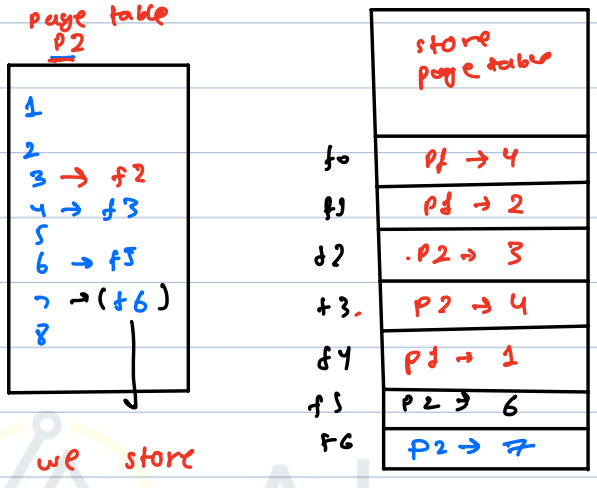
retrive the data from the main memory without any additional overhead.

it is referred as a hit.

to manage and resolve them we execute page fault handler routine.

- it retrieves data from disk

we need page 7 of Process 7.



we store additional information about data which present in disk.

ram (main memory)

the data is not yet loaded in the ram.

transfers the page from disk to ram.

update the table.

what if there is no free space in your ram?

we have to use page replacement algorithms.

if we have higher number of page fault.

it leads to degrade the performance.

LRU page replacement Algorithm

L

least recently used

we replace the page that has not been

used for longest period of time.

we can store only few page of a process.

page reference

1	5	3	4	6	5	3	6	4	6	7	4	5	6
1	1	~~1~~	4	4	~~4~~	3	3	3	~~3~~	7	7	~~7~~	6
3	5	5	~~5~~	6	6	6	6	6	6	6	~~6~~	5	5
		3	3	~~3~~	5	5	~~5~~	4	4	4	4	4	4
		✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓

How many page fault is occur?

total page fault is 10

total hit = 4.

higher no. of page fault can lead to a condition known as **thrashing**.

when system spends a significant amount of time swapping pages between disk and ram.

- when you have insufficient amout of space in your ram.

if we increas the number of page frames so page fault is decrease or increase?.

Belady's Anomaly.

In some page replacement Algo particullary FIFO (first in first out) when we **increase** the number of **page frame** it may

be increase the number of page fault.

FIFO

sequence of page reference.

1 4 5 2 1 4 3 1 4 5 2 3 1

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✗ ✗ ✓ ✓ ✗ ✓

3 frames

[1 4 5 2 1 4 5 2 1] ram

Total page fault = 10

hit = 3

4 frames

1 4 5 2 1 4 3 1 4 5 2 3 1

✓ ✓ ✓ ✓ ✗ ✗ ✓ ✓ ✓ ✓ ✓ ✓ ✓

hit hit

[1 4 5 2 3 1 4 5 2 3 1]

total no. of page fault = 11

hit = 2

optimise page replacemet algorithm.

remove the data which come last in the future.

computer networks

Basics of computer networks

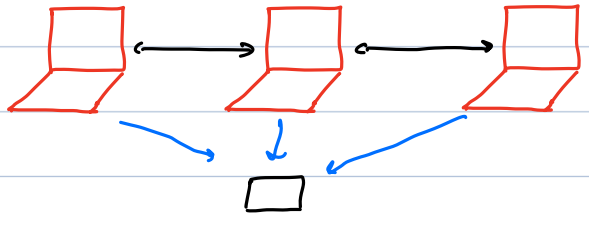
group of connected entities

social	network	peoples
railway	network	railway
computer	networks	computer

- how data is sent, travel from one place another place

- Computer network is a system of interconnected devices.

- we can communicate to each other within the network.
- we can exchange data.
- we can share resources.

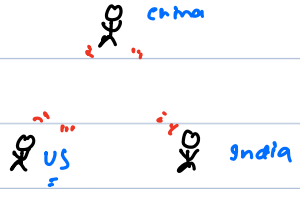


Internet defense

↳ U.S department of defense (1960)

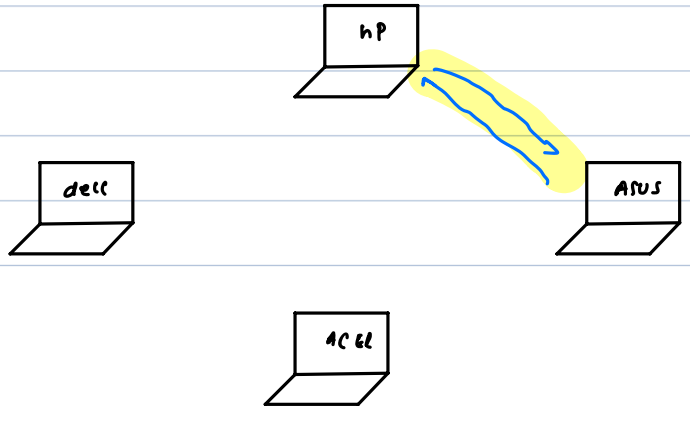
ARPA → advanced research project agency.

ARPANET (now it is called as internet) :



to communicate to each other we need common

binary language



to communicate to each other we have to follow some rules (Protocols)

- http
- ip
- tcp
- ftp

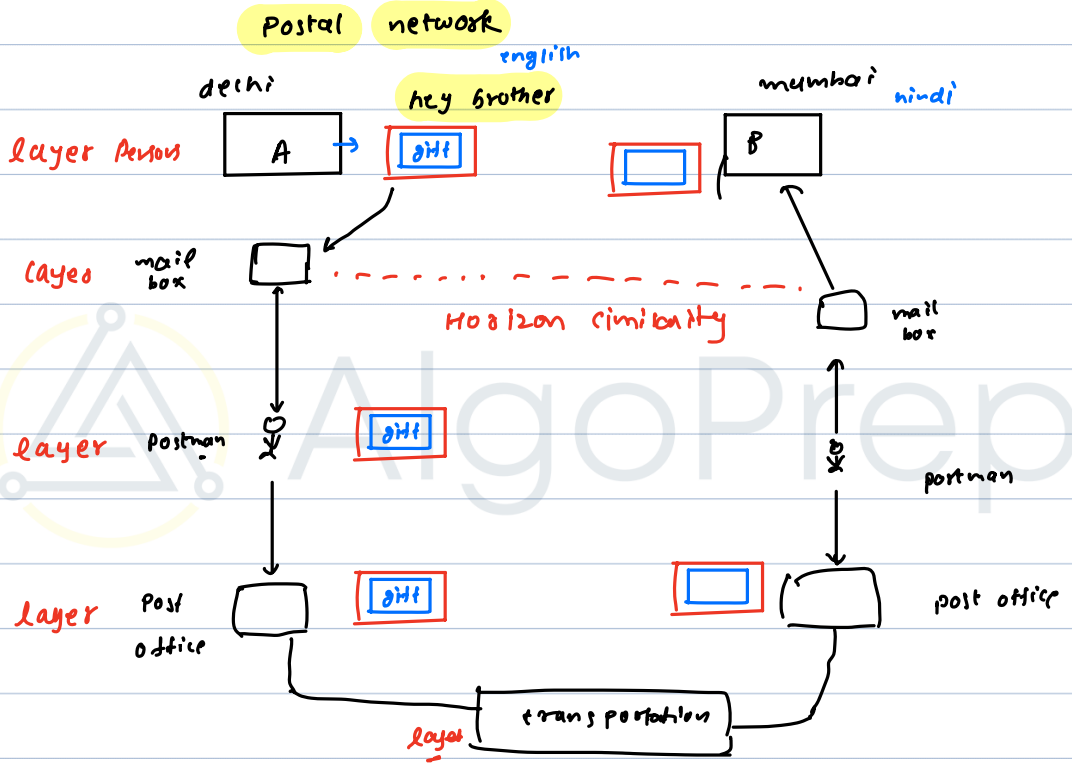
IETF — Internet engineering task force

open to everyone where anyone can share Protocols.

it is published on RFCs. (Request for comments)

Layering architecture

it is used to design the protocol.



- each layer responsible for their own work
- we divide the task into some manageable layers.

we hides the internal data

OSI Model (open system interconnection)

How the data is flow

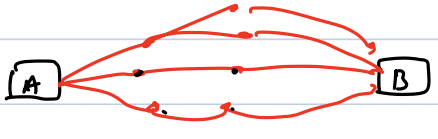
hey brother

Application layer — type the message and send to the person

Presentation layer — we format it, encrypt, compress this is understood by the receiver.

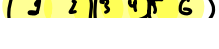
session layer — we are established the session.

transport layer — its ensure end to ends communication. tracks the record.

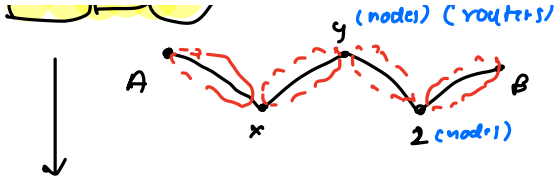


so many ways

Network layer — its decide the route that from which path packet is travelling.



data link layer

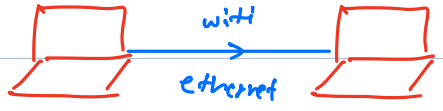


it manages nodes to nodes communication.

Physical layer

physical layer send the data in signal

electric pulse



first i have to travel from top to bottom in these layer then we travel from bottom to top.

break till 10:36

addresses

humans :- abc, H.N-101, xy2 pin code

H.N-101, xy2 pin code

computer

IP address

IPv4 IPv6

ipv4 32 bits

(00000000 . 00000000 . 00000000 . 00000000)

8 bits . 8 bits . 8 bits . 8 bits

192 . 168 . 1 . 3

(0-255) (0-255) (0-255) (0-255)

total 2^{32} combinations.

↳ 4 billion

196 . 231 . 259 . 3 /

more than 4 billion devices.

ip address is unique identifier.

ipv6 128 bits

we can generate 2^{128}

it is very huge number

we still mainly use ipv4

it is more popular.

A

101	102	103
201	202	203
301	302	303

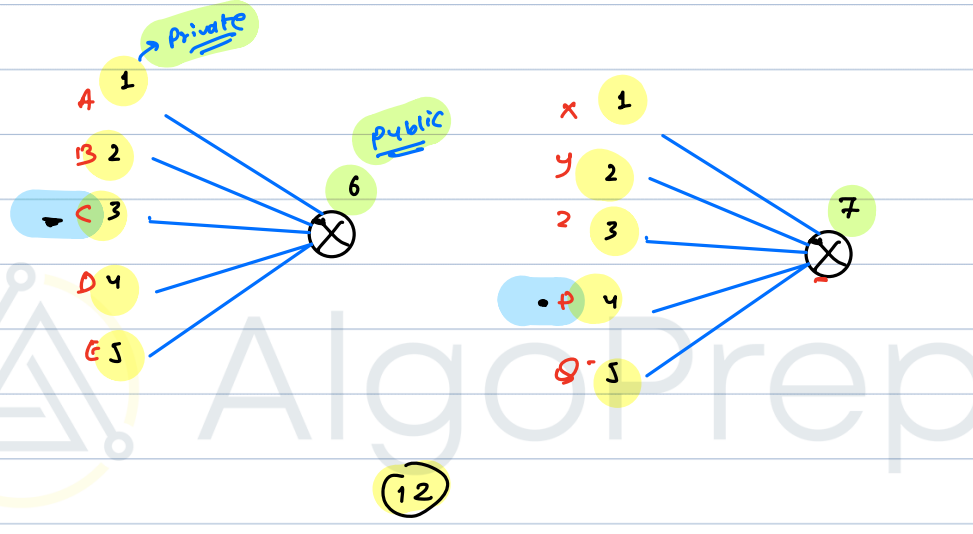
B

101	102	103
201	202	203
301	302	303

C

101	102	103
201	202	203
301	302	303

we have same room numbers in different hotels.



private ip → public address → Public addres → destination address

NAT (Network addrels translation)

Routers → public address

mobile hotspot → private address

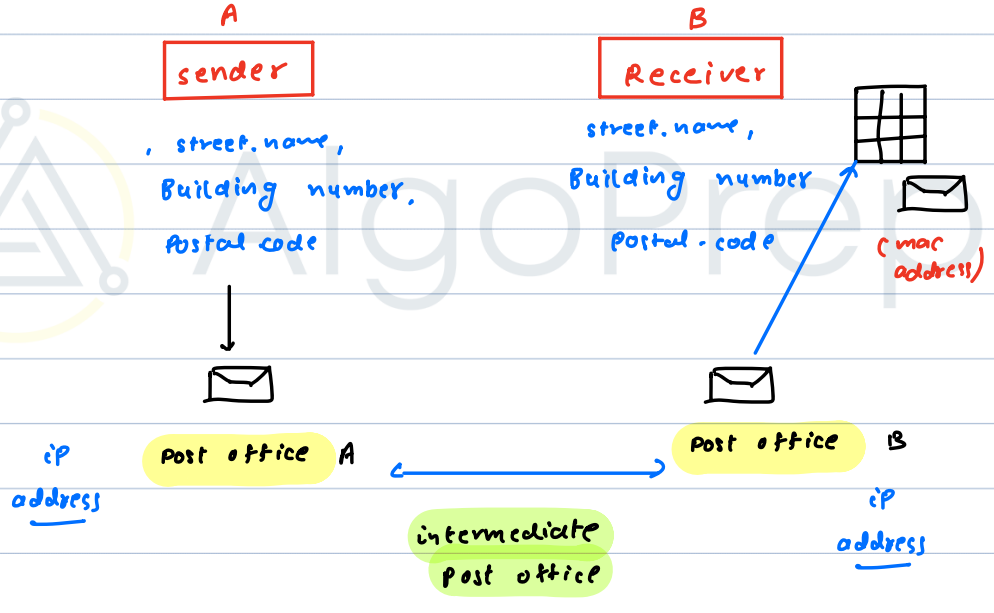
- if mapping the private and public addressess.
- its manages all the traffic in the network.

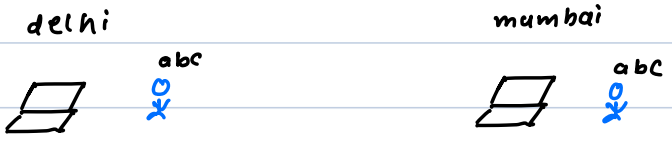
OSI model

IP address

Mac address

it is also known as physical address or hardware address.





any person can change the location

ip address → location

mac address → name.

it's represented as six pair of hexadecimal digit

48 bits

2B : 1C : 3D : E1 : F2 : 3A

8bits 8bits

(01)$_2$

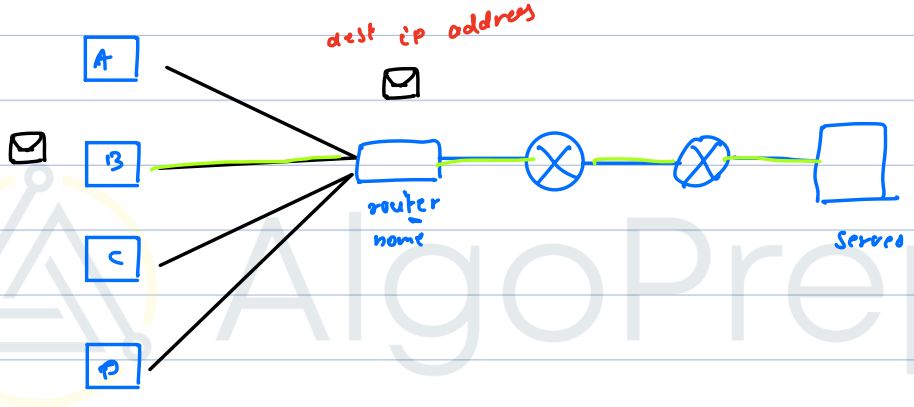
(0-7)$_8$

(0-9)$_{10}$

(0-9 A-F)$_{16}$

router is used mostly ip address.

switch is used mostly mac address.

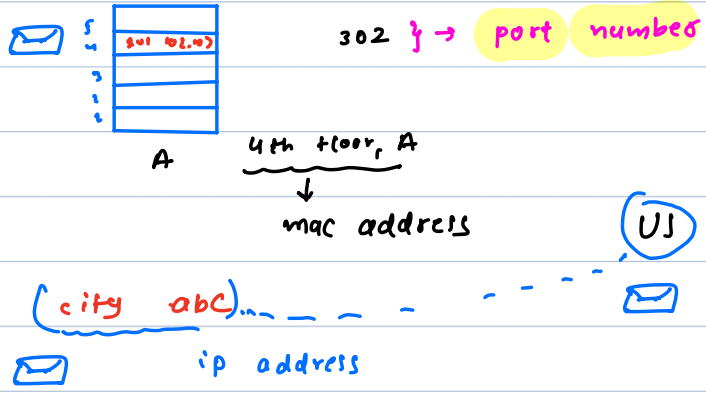


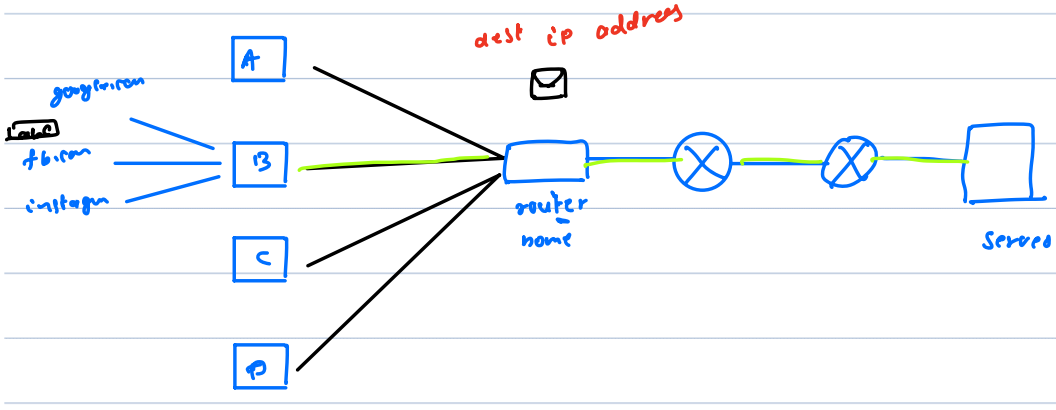
mac address is used to find device in local network.

ip address is operates on network layer

mac address is operates on data link layer.

post number





range of port number is between

0 - 65535

http :→ 80

https :→ 443

port number is assigned by your operating

system.

break till 10:12

DNS (domain name server)

phone book of internet

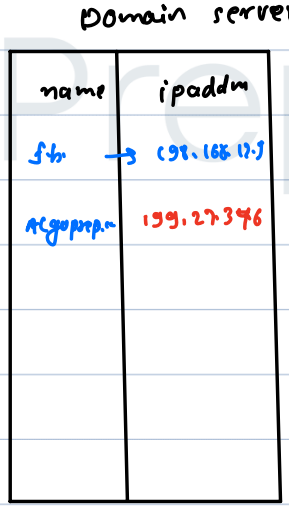
google.com (human readable name)

we need ip address to locate anything

192.168.17.13

facebook.com

Algoprep.in

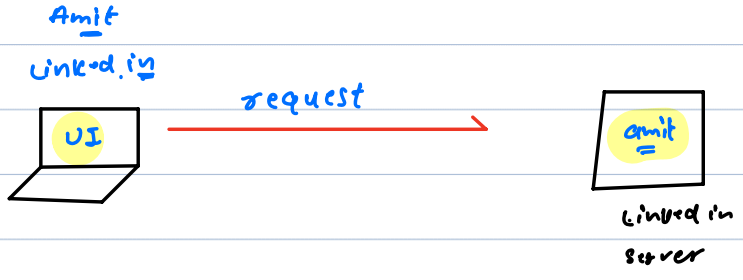


this feature is given by your ISP

(Internet service provider)

Application layer

generate the data to be sent.

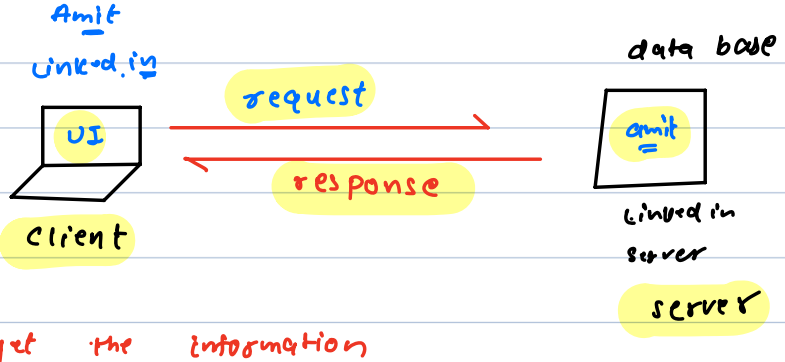


get the information
about amit

1) client server architecture

2) peer to peer architecture.

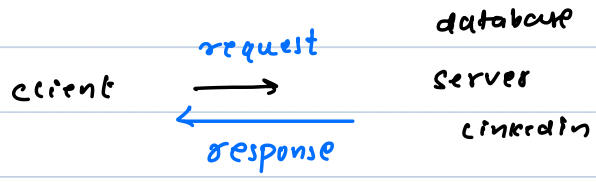
client server architecture



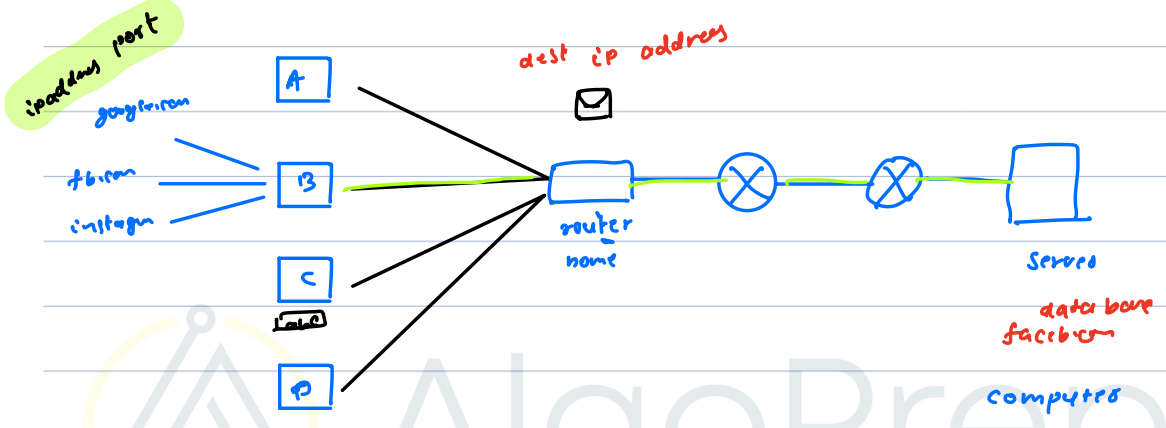
get the information
about amit

client is not have own data

server is owner of the data

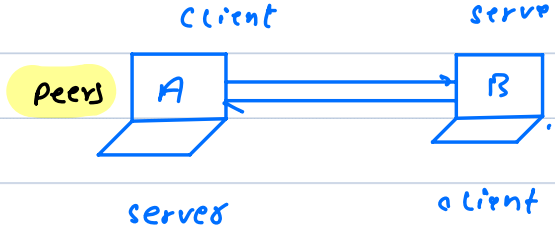


centralised system



2) peer to peer architecture.

peers is act like client as well as server.



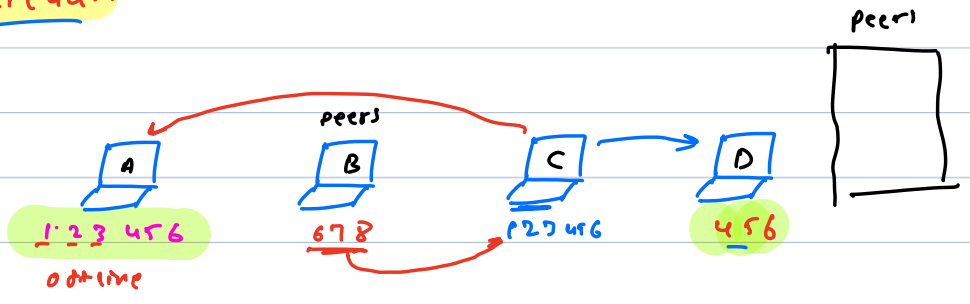
there is no external database.

torrent Peer to peer architecture.

shaitaan

torrent

downloading / uploading



it has no external database

10 movies

100 users

↓ download same

3 movies → are completely download → uploading

first time you leacher → you can't upload only download.

it is not reliable.

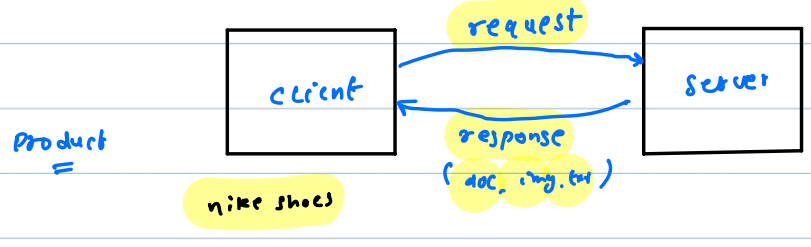
http

H -> hyper
T -> text
T -> transfer
P -> protocol

HTML
↓
hyper text markup lang.

it is responsible for web communication between web server and client.

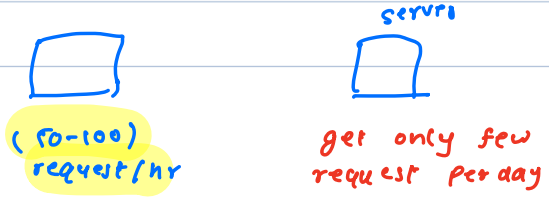
- it is operates on application layer.
- it is based on client server architecture.



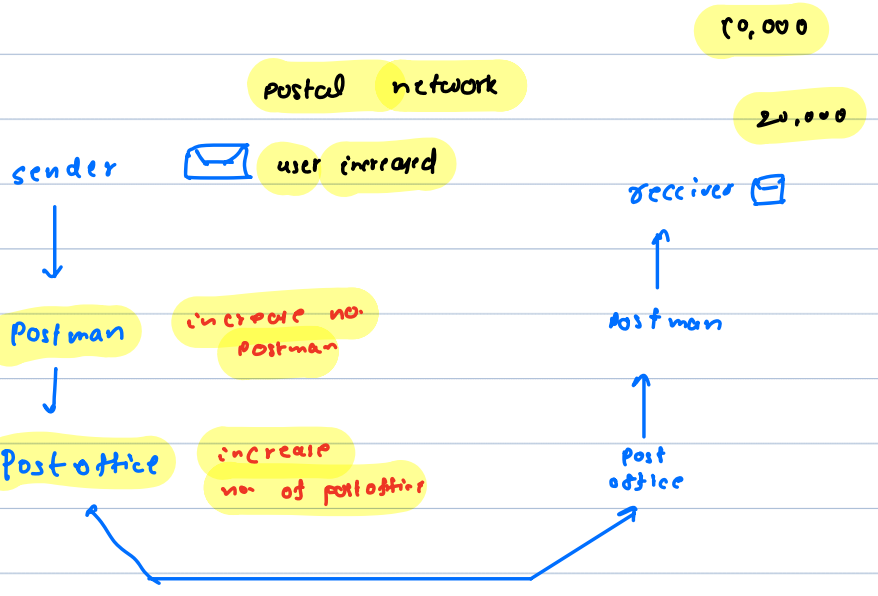
Product =
nike shoes

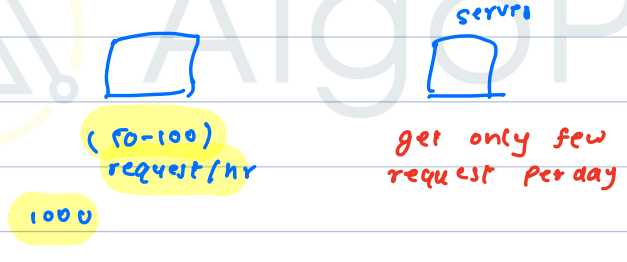
request give me info about this shoes

jio cinema

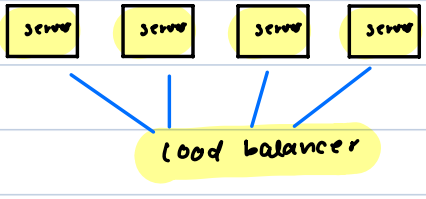


10,000 / hr





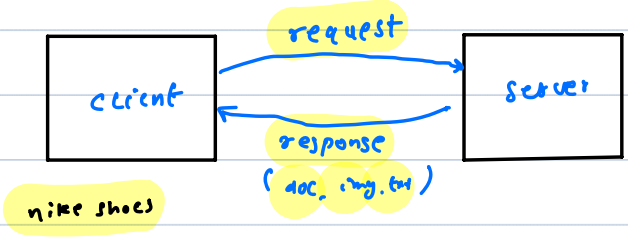
100000 / hr



suddenly get lot of traffic

we add more machines to the servers.

- http is stateless protocol?

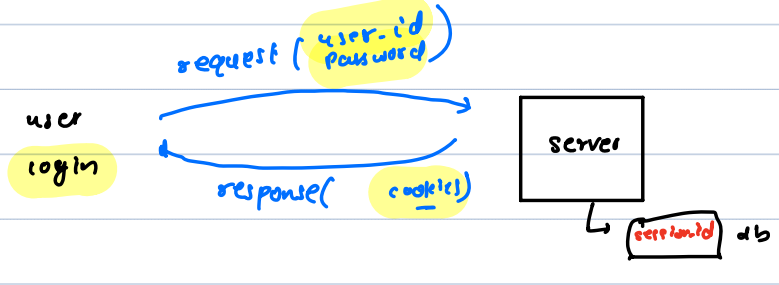


home / Product / shoes / nike

you navigate to the new page.

- each request is independent from each other
- it is doen't know about previous request.
- we need some more data about previous past

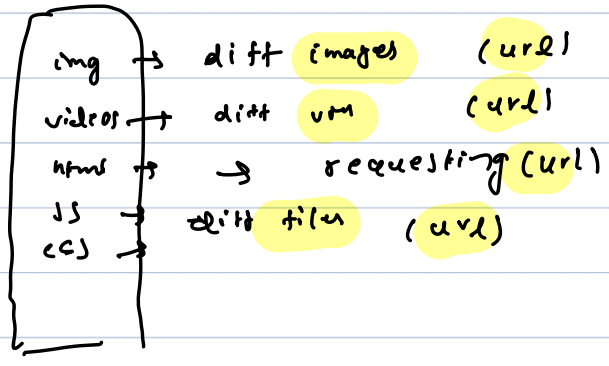
amazon



- first time it check that the user is valid or not. authenticate the user.
- it is send some cookies in response
- when we request every time we send that cookies.
- check the session is valid or not then server send the response

irctc → it creates short session

flipkart / amazon → it creates long session.



we have to create lots of request.

each time we request url from the server

URL (uniform resource locater)

1) https // : www.flipkart.com/ ------

http → https
↳ it is secured
↳ encrypted

2) domain name / ip address

3) port number → some domain name have default port

4) path

break till 10:10

request methods :-

it is used to indicate which type of action is performed on resources.

1) GET :- to fetch data from server.
there is no side effects on server.

2) Post :- it is used to add data in server.
- it may side effect on server.

sign up in any amazon

3) PUT :- it is used to update the data in server.

update your profile.

4) delete : it is used to delete data from the server.

get / post

lot of information

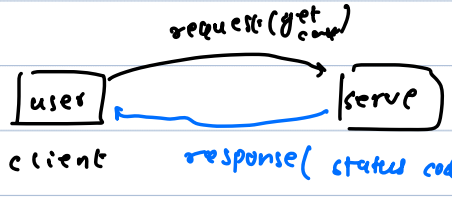
get command has limitation that they can only send small amounts of data.

when we have lot of information we can use post instead of get command.

In post command we don't have limitation.

response code / status code:

it signifies the outcome of the request



- 1xx codes

100 → informational.

101

102 → it is indicates that your request is processing.

- 2xx these types of codes represent successfull of the task.

200 OK

201

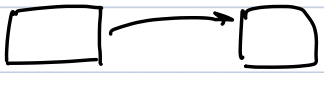
- 3 ** redirect.

300
301 → permanently moved
202
.
.
.

fb.com facebook.com

- 4 ** codes client error

400
401
402
403
.
.
.



400 → bad request.

404 → it is not found

- 5xx — server error

  500 → internal server error

  501

  502

  503 → service is not available

Http connection

1) non persistant connection
2) Persistant connection

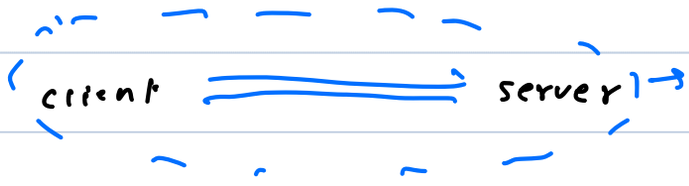
- non persistant connection is used in http/1.0.
- a new connection is established for each http request and response pair.
- after receiving the response from the server connection is closed.

open a web page

data is transmitted over tcp connection.

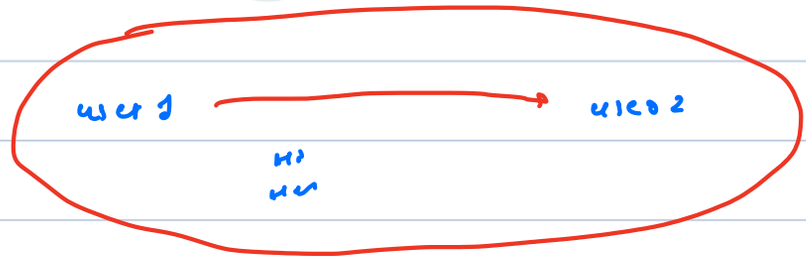
Persistant connection:

it is also known as keep-alive connection.
the client and server keep the connection
open after the initial request and response.



it is default in http/1.1 and later versions.

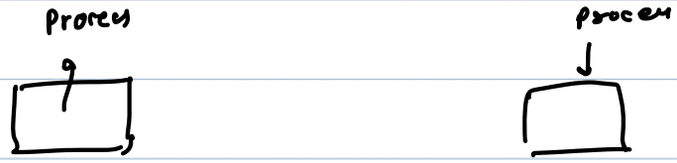
eg: social media chat.



it also minimize the delay.

Port number

it is used to uniquely identify a specific process or service running on a machine in a network.



Socket

it is combination of ip address and port number.

it is used to established connection between two processes.

TCP and UDP

transport layer

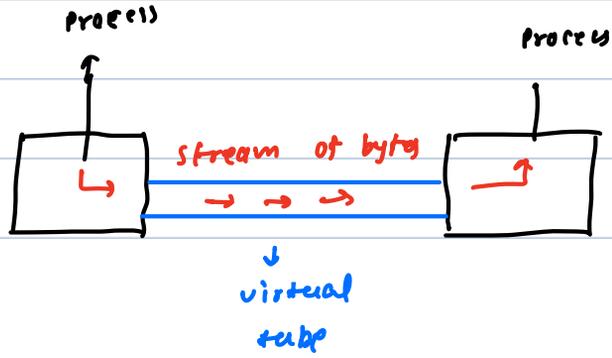
end to end communicaw.

transmission control protocol

it is used in transport layer.

tracking

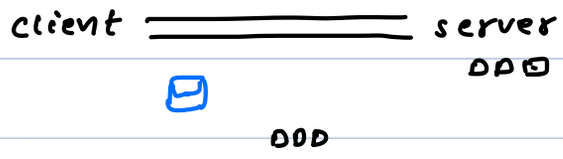
it is established a virtual communication channel between two proces.



it is called as segment.

1) it is reliable

it is ensure that data sent by the senders is received exactly as transmitted.



if any segment is loss or corrupted then it retransmitted the segment.

it is ensure there is no error.

TCP is uses acknowledgment mechanism to confirm that data has been successfully reached

(1 2 3 4 ..... 1000)

(1000 bytes)

(1 to 800)

client → server (800 byte)

200 bytes is loss.

tcp is used sequencing.

(20-60 byte)

TH | data it is called as segment.

1) source port number

2) destination port number

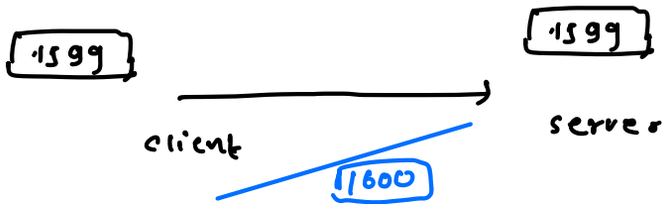
3) sequence number

1 2 3 4 5 6 — sender

2 3 4 1 6 — receiver

At receiver end we have to rearrange the packet before sending to the server.

acknowledgement number it indicate the next byte of the data to be send.



Header length (4 bit) (20-60) bytes

(0-15)

6 * 4 (24) 20

↳ scaling 24

28

8 * 4 = 32 32

multiples of 4.

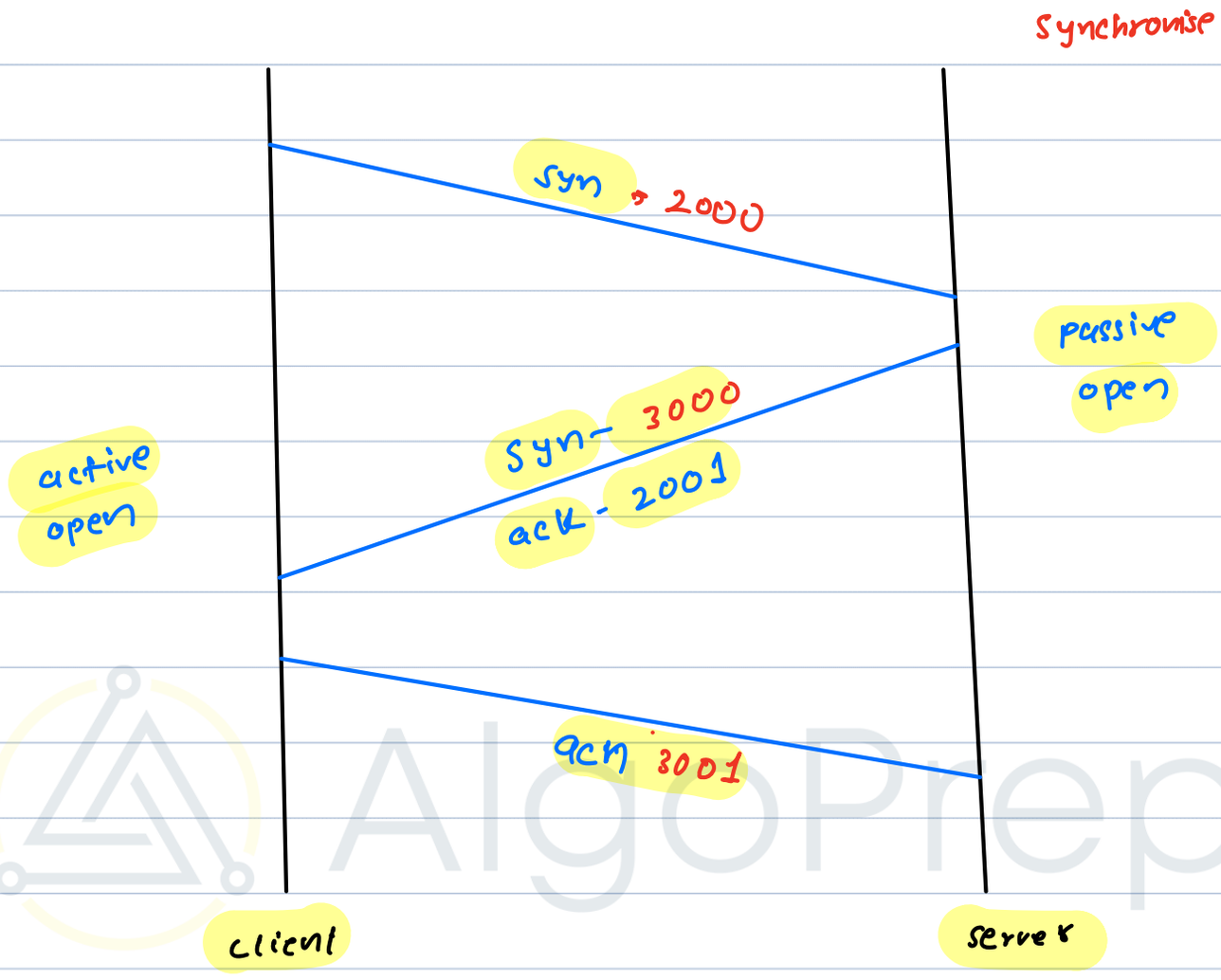
break till 10:05

- tcp is connection oriented communication

tcp is established a connection between the client and server before transmitting the data.

TCP connection establishment

it follows the three way handshake process.



step-1 the client initiate the connection by sending SYN (synchronise) to the server. it indicates that clients want to make a connection

step-2 the server responds SYN-ACK to the client

it is receipt of the client SYN segments

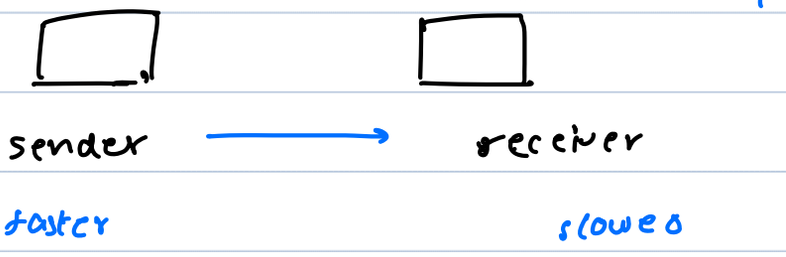
step-3

the client sent the ack to the server.

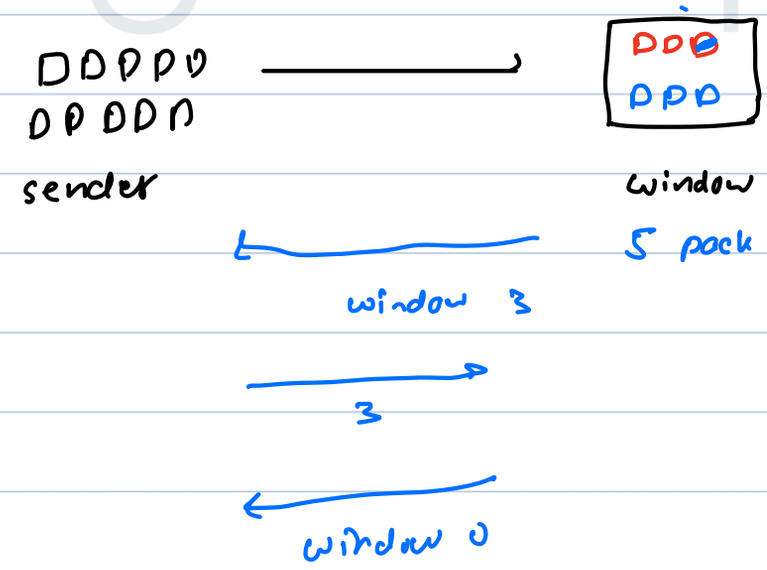
it confioms that client is read to transmit data.

flow control

it prevent overwhelming the receiving host with more data that it can proces.

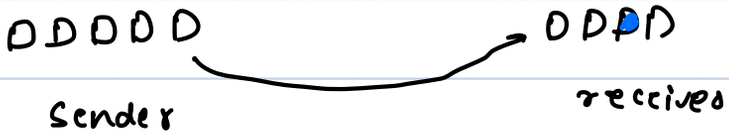


window size



error control

check sum (8 bit) is used for error handling



eg:-

8 bit | 8 bit | 8 bit

1 1 1 0 0 1 1 0 | 1 1 0 0 1 0 1 0 | 1 0 1 1 0 1 0 1

```
  10  1 1 1 1 1 1
      1 1 1 0 0 1 1 0
      1 1 0 0 1 0 1 0
+     1 0 1 1 0 1 0 1
-------------------------
  10  0 1 1 0 0 1 0 1
                    1 0
-------------------------
      0 1 1 0 0 1 1 1
-------------------------
      1 0 0 1 1 0 0 0
```

check sum

```
  10
   1
```

```
  0 0 1 +1
  0 1 0 +1
  0 1 1 +1
  1 0 0
  1 0 |
  1 1 1
```

if the sum is exceed 8 bit. wrap the numbers

At receives end

```
  1 0 1 1 1 1 1 0
    1 1 1 0 0 1 0
  1 1 0 0 1 0 1 0
  1 0 1 1 0 1 0 1
  1 0 0 1 1 0 0 0
-------------------
1 0 1 1 1 1 1 0 1
                1 0
-------------------
  1 1 1 1 1 1 1 1
-------------------
```

wrap up

after adding all the bytes with checksum we check that if all bit are 1 then it means there is no error.

```
    1 1 1 1 1 1
    1 1 1 0 0 1 0
  0 1 0 0 1 0 1 0
  0 0 1 1 0 1 0 1
  1 0 0 1 1 0 0 0
-------------------
1 1 1 1 1 1 1 0 1
                1
-------------------
  1 1 1 1 1 1 1 0
-------------------
```

there is an error
it retransmit the data.

it make it slow

UDP user datagram protocol.

1) connectionless communication

there is no establishment of a connection
before data transmitted.
each datagram is independent.

online games (BGMs) live streaming

delay between the data transmission
wait

2) unreliable delivery

udp is not guarantee delivery of data
or ensure it arrives in the correct
order.

3) low overhead.

src, dst port number
datagram length checksum

4) fast transmission

UDP does not wait for acknowledgement.