Distributed Programming Notes

Exam Questions

- Explain how a web browser communicates with a web server (Which protocols are involved, how they handle information, talk about NIC and different layers)
- 32-bit network addresses (A – E)
- Describe types of distributed computing and provide an example
- What is the difference between stateful and stateless server
- Object server
- What is RMI registry
- What is the purpose of logs in multi-threaded applications (Demonstrate with code snippets)
- Explain four challenges of distributed computing and suggest how to fix these challenges
- What are the key aspects of transparency in distributed computing (Access should be hidden, user shouldn't know they're in Dublin for example, Scaling, Performance)
- What are the key components in inter-process communication
- What are the two types of threading synchronization
- 3 roles of a connection socket
- How does the remote object get resgistered in rmi (Describe how an RMI server registers a remote object, and provide pseudocode)
- What is the difference between connection-less and connection-based communication
- Multi-cast communication
- Software layering, how do layers communicate with each other
- What are the benefits and drawbacks of connection-based communication and connection-less communication
- Describe with an appropriate diagram the different states a java thread can be in
- Describe the architecture of RMI using a diagram to show the different layers and how they relate to each other
- What is the purpose of the remote interface in RMI
- Discuss the differences between an iterative and concurrent server
- Fallacies of communication
- Wait and notify
- How does a the java socket api provide event synchronization

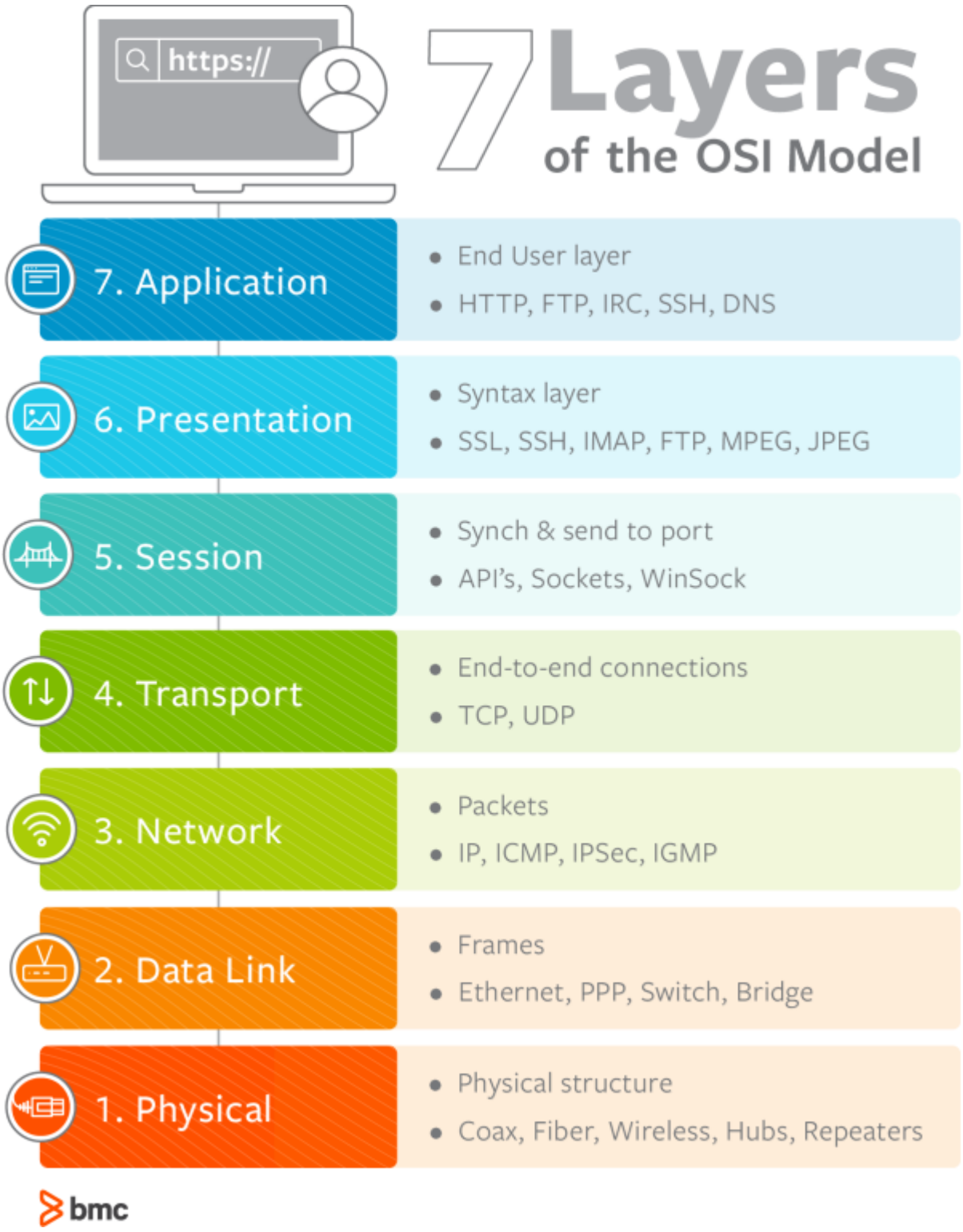
- Describe the challenges in implementing distributed systems (compared to a standard system)
- Describe how you would create a thread in java (Pseudocode written)
- What is a difference between a thread and a process (Indicate this through code)
- What is the role of the connection socket and the data socket
- There are 8 fallacies, name any two of these fallacies and discuss the reasons why they cause problems (Provide REAL WORLD examples)
- Two methods of improving scalability and performance in threaded programs (Provide sample code)
- Compare and contrast RMI and socket based servers (Operation, and transmissional)
- Define the race condition

Keywords

- UDP - User Datagram Protocol
- TCP - Transmission Control Protocol
- Flow control – (Used in TCP to ensure that too much data isn't sent)
- For internet the best route is usually the shortest route (But it might not always be the best because it might be congested)
- Important that transmission rate must be specified, so that there is no loss of data.

Network Architecture

- Organised into separate layers
- Each layer provides services to the layer above it
- Each layer can communicate with the same type of layer on another machine. E.G. (Layer 1 on machine 1 can communicate with Layer 1 on machine 2)
- How devices send data from 1 machine to another
- Abstracted (User doesn't have to know how the entire system works)
- A layer needs to know which process to service the layer above it



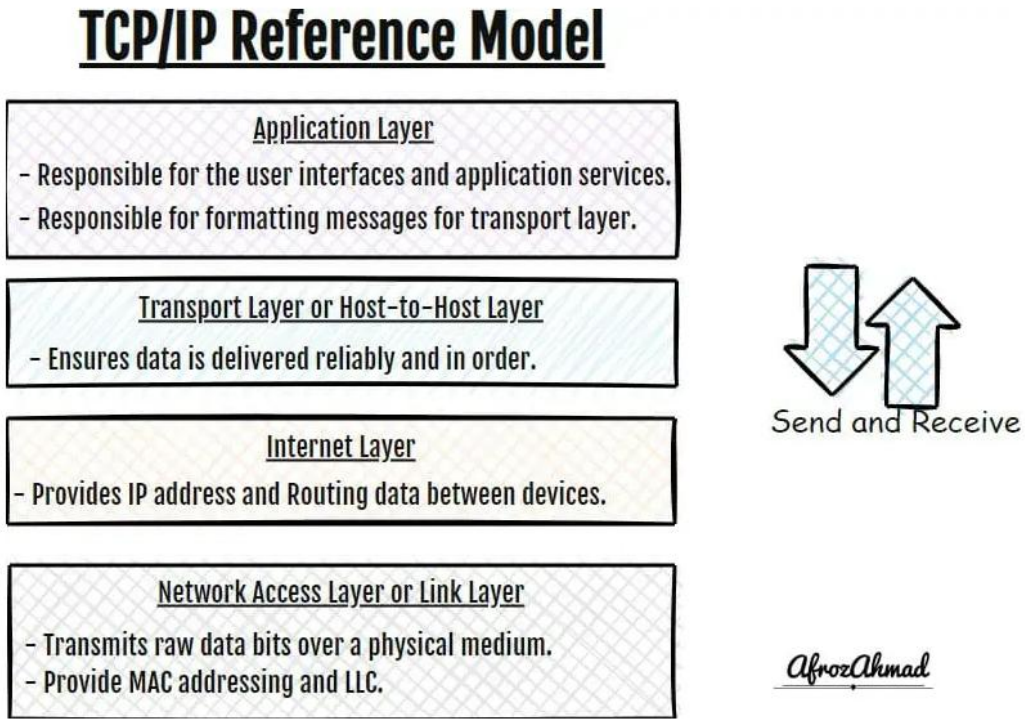
(Data link layer contains headers)

- In application layer, there is either a datagram or a segment

Network Architecture for Internet

- More simplified

- 4 layers: physical, Internet, transport and application
- Identified through port numbers or sockets



- The **Network Access Layer** (Layer 1) manages the physical hardware and media for data transmission, including hardware (MAC) addressing and the media (Fiber, Copper wire) itself.
- The **Internet Layer** (Layer 2) is responsible for logical IP addressing, routing, and packet forwarding, essentially determining how data packets navigate through the network.
- The **Transport Layer** (Layer 3) ensures reliable data transfer, managing data flow control, error checking, and segmentation.
- Lastly, the **Application Layer** (Layer 4) provides protocols for specific data communications services to applications, enabling users to interact with the network.

Protocols

- A set of rules for formatting and processing data
- Protocol suite (TCP/IP)
- The idea of setting up connections and terminating connections
- Use IDs to identify hosts (E.G. IP addresses)

Application layer – HTTP (Formats the URL into the actual page)

TCP Ensures data is sent at an ok rate

Operating Systems

OS – A software layer used to abstract away and manage details of hardware resources

Software Program – Inactive most of the time

Software Process – Active, but usually idle

A process must be run sequentially

A process consists of multiple parts:

1. Text (Program Code)
2. Program Counter (Next instruction to be executed)
3. Stack

Process States

- New
- Running
- Ready
- Waiting*
- Terminated

CPU Process Management

Context Switch (One process is executed, and based on interruptions the same task it passed to a different process)

(Add image here)

Threads

A lightweight process

Created by extending the thread class, or using the “Runnable” interface

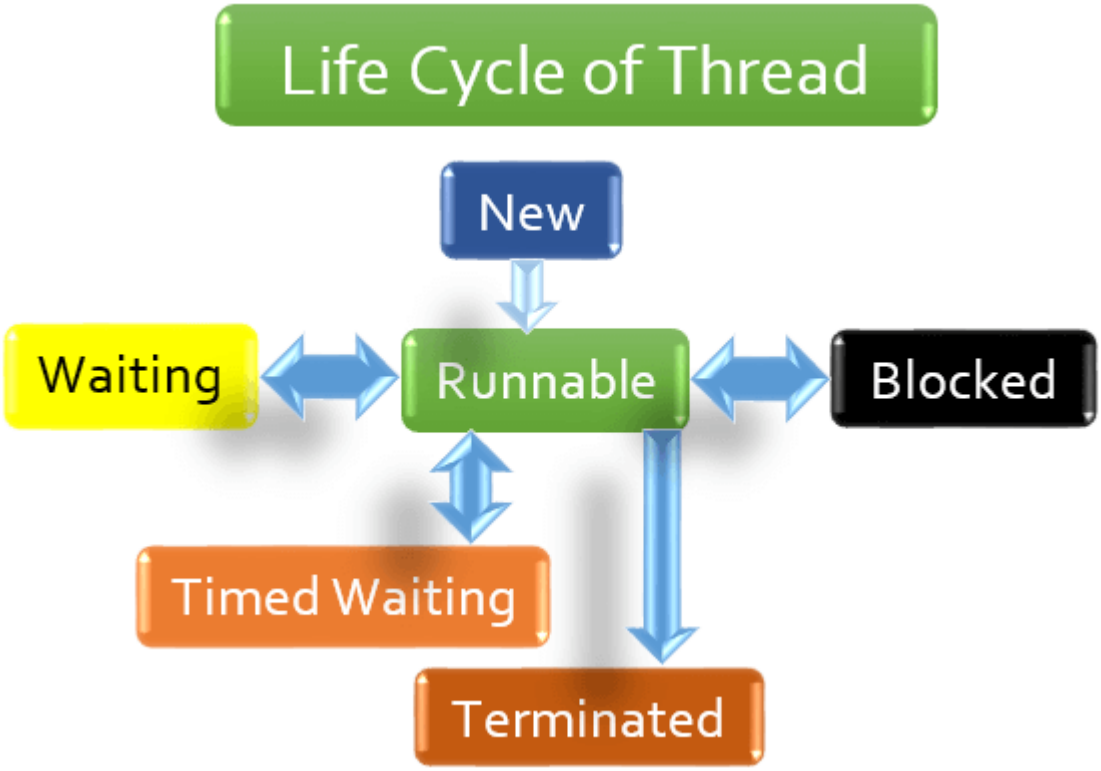
Thread lifecycle – On state, Running state, Dead state

Flags indicate whether or not a thread is running

Volatile if the flag shouldn’t be flagged

Race – When a bunch of commands run parallel

Put threads to sleep if they’re not doing anything for time (Store it in memory)



Program counters which help manage the threads within the process (All kept in the PCB Program Control Blob)

Many program counters in the same space.

Any java program must be written into bytecode and then translated into binary for the machine to understand.

JVM – Java Virtual Machine

Applet must be downloaded, and run on the host machine (It needs the main method)

REST Conditions are involved.

Threading Synchronization

Two types – Method level , Block level

Method level – tell the methods to be synchronized (public synchronized)

Whatever calls the method gets a “lock” on the one it called, lock is released when the call is complete.

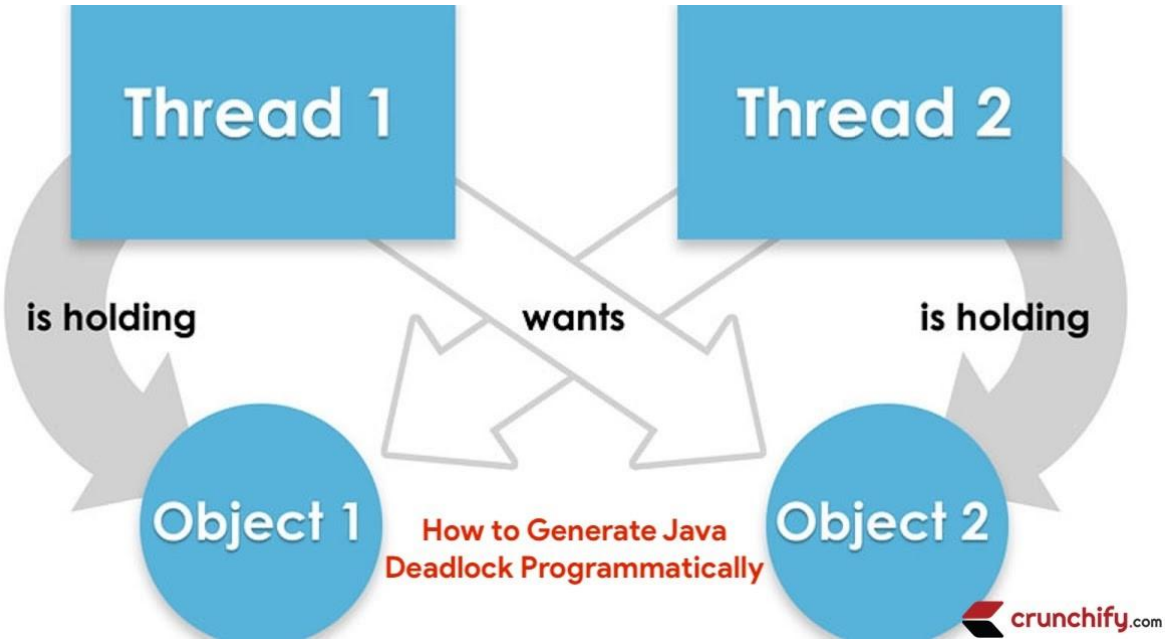
Stops other processes from accessing the object

Block level – Aka synchronized

Locks a section of code against a certain object

(Gets access to a specific “Block” of code and locks it down)

Deadlock when two or more threads are trying to access and lock each other but are waiting forever and locked against each other.



Thread Scheduling

- Premptive & non Pre-emptive
- Premptive – pauses a running thread and executes all other running threads in a schedule
- Non-pre-emptive – Never interrupt another running thread
- Time-sliced: Little chunks of a thread runs, and then is passed to the next thread, for scheduling they are given a specific time to run
- Priority can go from 1 minimum – 10 maximum

Distributed Programming

Distributed system – a connection of independent computers that are networked.

Distributed Computing – A study in this field

Advantages

Great for saving money (Instead of using one computer with high power, use smaller processes to accomplish a task)

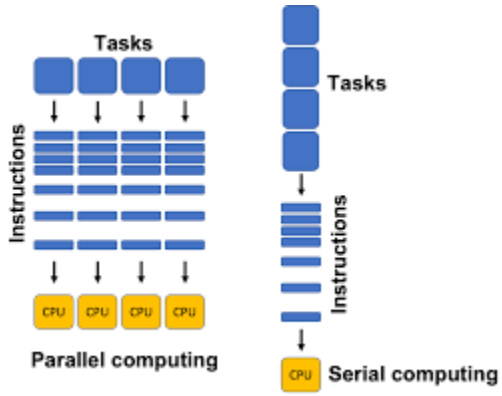
Higher ability / Redundancy (If one machine fails another can complete the task)

Types of Computing

- **Single machine** – Monolithic, Parallel (Doesn’t need networking)

- **Multi machine** – Grid, Cluster, Co-operative (Must use networking)

Parallel Computing – Multiprocessing



Grid Computing

Different locations

Combines resources over all these different locations

Cluster Computing

- Happens in **single location** – All computers in this location all work together on it (There can be many different locations)
- One machine serves as a "**master** machine" which hands off all the tasks to the "**slaves**"

Challenges

- Security
- Scalability – How do you grow
- Failure handling – Users should know if a machine is failed
- Concurrency – Allow resources to be accessed by many people at the same time (Making sure the system is always available)
- Transparency – Showing stuff to the user
- Heterogeneity – Create specifications, and documentation **(Develop the system in layers)**

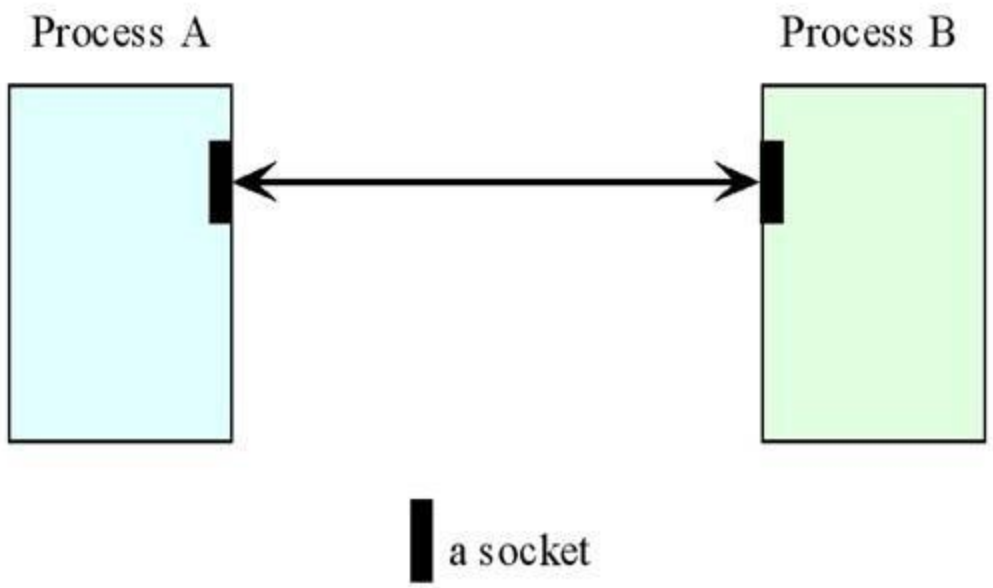
Computing Fallacies – Misconceptions

1. Network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. Network is secure
5. Topology doesn't change

6. There is one administrator
7. Transport cost is zero
8. Network is homogeneous (The same)

Socket API

- IPC programming interface
- The most widely used API type



Sockets – Like a phone number (Someone calls a certain number, the person with that number picks up)

Port 80 (Used for HTTP services)

Port 23 (Used for TelNet)

Using port numbers: Which identifies the specific process needed

(Layers provide services to each other through INTERFACES) Example of an interface is a socket

- Datagram – basic transfer unit associated with a packet

Client Server Paradigm

A model that tells the roles of processes - Server & Client

Provides **Network Services** – A service to allow network users to share resources

Telnet – Connecting to remote machines on the same network

Uses 3-layer architecture (Presentation, Application, Service)

Important

Surrogate Server – Acts as a intermediate for the main server

For connection based the server interlinks between multiple clients

Iterative – One client at a time (Any new clients are blocked)

Concurrent – Multiple clients in parallel (multi-threading)

1. Take in client
2. Give them the info
3. Close the connection

Testing a network , Develop a Client first, test backwards

State info (who has logged in etc…)

Stateless – doesn't have responsibility on the client

Stateful – has responsibility on the client

Data marshalling – A way of putting data in a way that can be understood easily

Paradigms – (Models, Patterns)

- **Abstraction** – hiding details
- Used in IDE
- Achieved using paradigms and models, (Know models on when to appropriately use them don't reinvent the wheel!!)
- Message passing abstraction – lowest level (Sends a message to receiver, may trigger further requests) SEND and RECEIVE
- Message passing paradigm based on the socket API
- **Client-server paradigm** - Server is only concerned with responding
- **Asymmetric roles -** Server doesn't need to manage events, focuses solely on requests
- **Peer-to-peer paradigm –** Processes communicate back and forth
- **Message System Paradigm –** An abstraction layer that handles all the messages between server and client
- Models in this system – Point to point(uses middleware), Publish-subscribe(subscription)
- Point to point – Messages are put in a queue, which the client consumes and then acknowledges
- Publish-subscribe – If someone has an interest in a particular subject they may **subscribe** to that event. All subscribers, connect to one central topic, which all subscribers receive

- RPC (Remote procedure call) Model – allows a client to execute functions on a remote server
- RMI (Remote method invocation) – Objects residing within remote hosts, these objects contain methods
- RMI callback – Call the client if an event takes place (Code needs to be both on the client and the server)

Stab on a client that will act as a proxy

Create a observer for when the event takes place

RMI setup

Naming is started

Remote object created

Server is bound

Client looks up object

Once an object is created it has to be **live** which uses a lot of cpu and memory

Rmi cost – A separate machine for separate servers

ROA Remote Object Activation – Register an object but don't have it activated, only run them if there is a request to do so

RMID – Remote method activation system daemon (Disk activation monitor)

The daemon tracks what can be activated, and where they are.
It checks if the object is already active, and then if not activates it

ROA – Tells RMID what needs to be activated and how to, done using a description of the object

Behind the scenes – A **stub** is created, stub has: reference to live object, activation id, reference to RMI daemon

A stub runs on the server that has a reference to the remote object

Server side is different, instead of created the object, a **stub** is created which the client can use to access to the object

To activate the object the method (stub) calls the faulting reference are forwarded to the new object and then returned to the client

Object Serialization

Convert a object into a stream of bytes, so a whole object can be sent over a socket (Marshalling)

3 different ways:

1. use default – object outputstream, inputstream (SDK checks if an object is serializable or not. Use serialver to check if something serializable)
2. Use custom class – Both must have a copy of the class file
3. Customizing the default protocol –

As a programmer you should write your own read and write objects