UNIT-2
DISTRIBUTED SYSTEMS

Communication: Layered Protocols, Lower-Level Protocols, Transport Protocols, Higher-Level Protocols, Remote Procedure Call: Basic RPC Operation, Parameter Passing. Extended RPC Models, Remote Object Invocation: Distributed Objects, Binding a Client to an Object; Static verses Dynamic Remote Method Invocations, Parameter Passing, Message Oriented. Communication: Persistence and synchronicity in Communication, Message Oriented Transient Communication, and Message-Oriented' Persistent Communication, Stream Oriented Communication: Support for Continuous Media, Streams and Quality of Service, Stream Synchronization.

PROTOCOLS

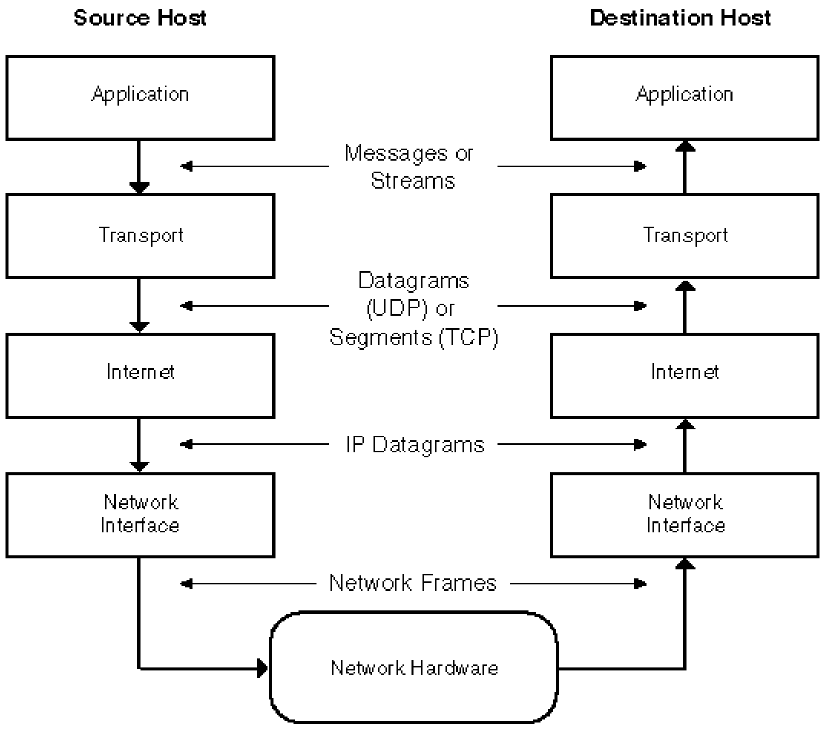
Protocol= Set of Rules for how computers communicate with each other.

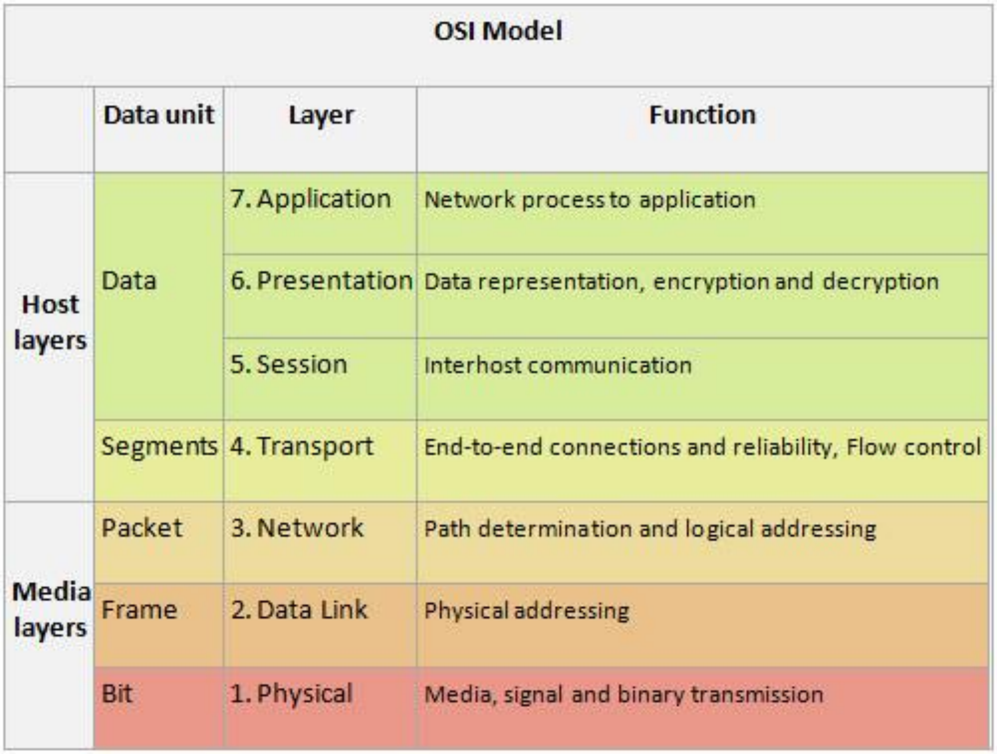
PROTOCOLS TYPES	
1. Lower level Protocols (Device to Device). The lowest protocol always deals with "low-level", physical interaction of the hardware. Every higher layer adds more features. **a. IP (Internet Protocol)** (The Address of the Machine)). **b. TCP (Transmission Control Protocol) (**Proof of Delivery, rules or reassembling partitioned messages)). c. Implementation in physical layer and data link layer of the stack. Group data bits into frames and adds a pattern called checksums at either end of frame.	**2. Higher Level Protocols (Program to Program)** a. FTP: File Transfer protocol. b. SMTP: simple mail transfer protocol. c. HTTP: Hyper Text Transfer Protocol. d. Network Layer chooses best path from sender to receiver by routing.

Layers in the OSI model, they can be grouped into three areas:

- **High-level Protocols (layers 5, 6 and 7 - Session, Presentation, and Application)** - how the data is presented, displayed, and summarized for the user - and in the reverse direction, how the user prepared data is assembled into meaningful data structures (high-level protocols).
- **Medium-level Protocols (Layers 3 and 4 - Network and Transport)** - how the data is assembled into packets and frames and how error checking and flow control is implemented - and in the reverse direction, how the received packets and frames are assembled into structures such as files and databases (medium-level protocols)
- **Low-level Protocols (Layers 1 and 2 - Physical and Data Link)** - how the data is converted into electrical pulses of one's and zero's (bits) and sent across cables or the physical medium, and in the reverse direction, how the electrical pulses are taken off the cable and converted to ones and zeros.

http://www.infocellar.com/networks/osi-model.htm
https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)



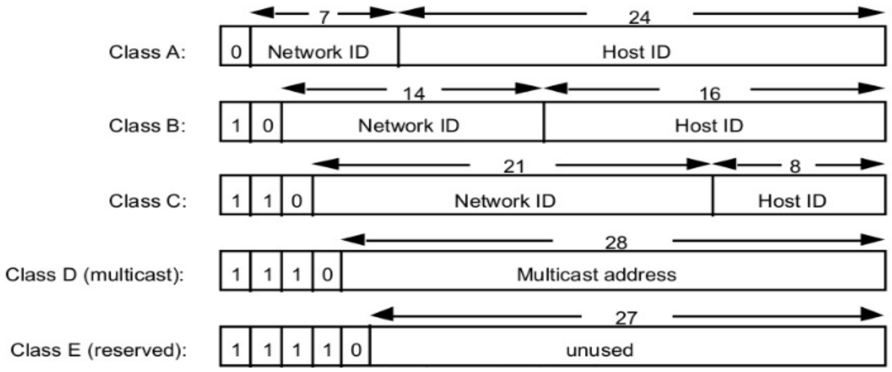


OSI Model

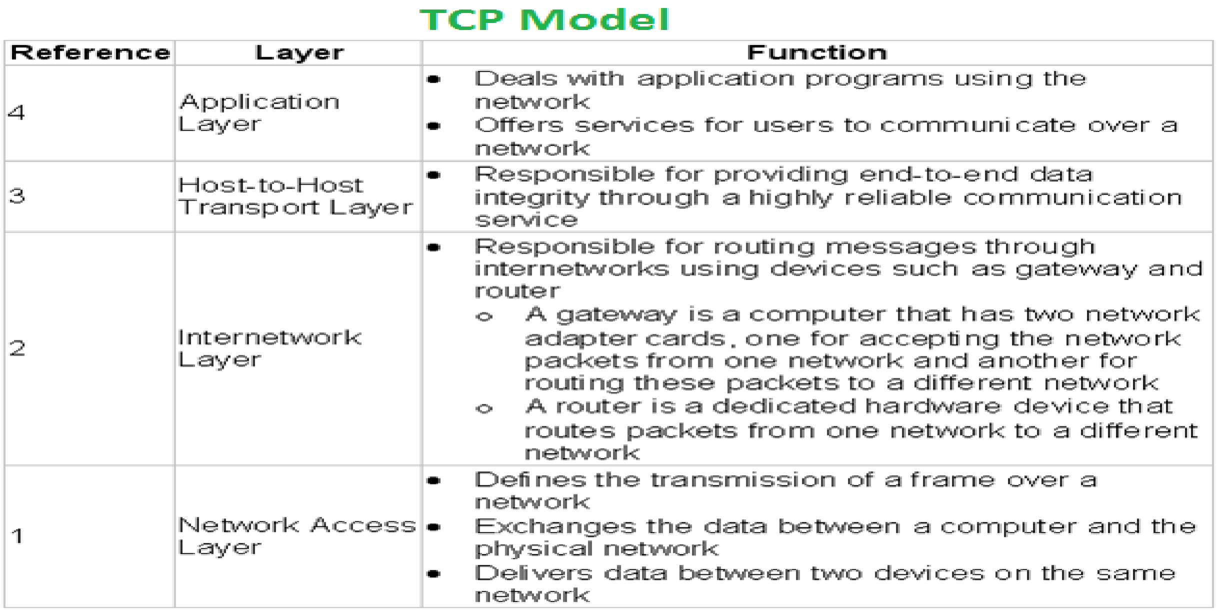
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, Flow control
Media layers	Packet	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Table 1: OSI Reference Model for Protocol Stacks

Internet address structure, showing field sizes in bits

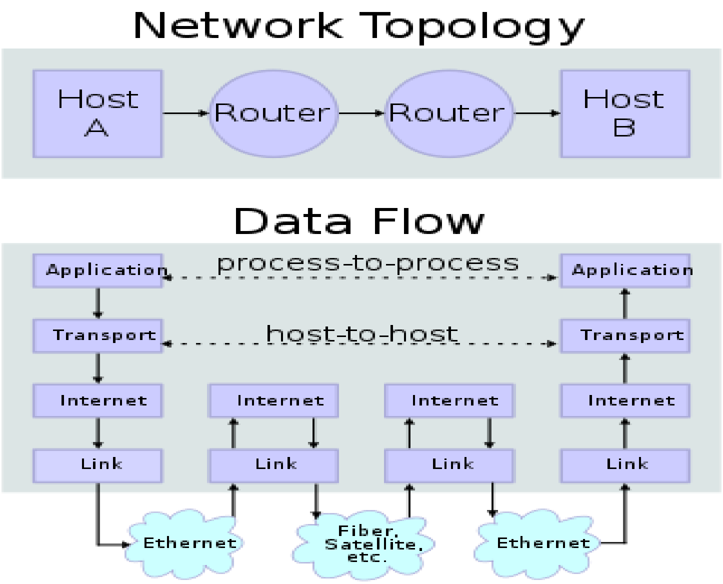


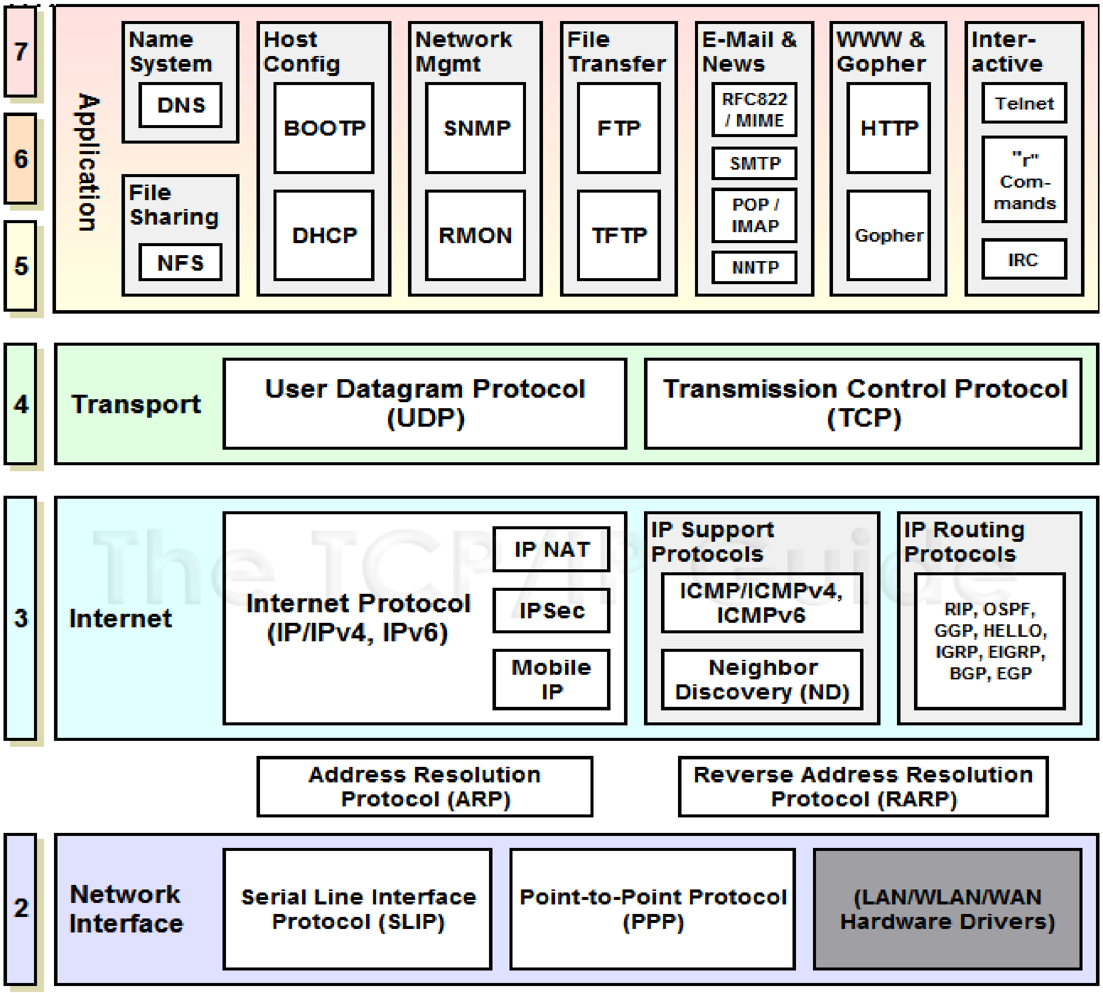
OSI	TCP
Application Layer	**Application Layer**
Presentation Layer	
Session Layer	
Transport Layer	**Transport Layer**
Network Layer	**Internet Layer**
Data link Layer	**Network Access Layer**
Physical Layer	

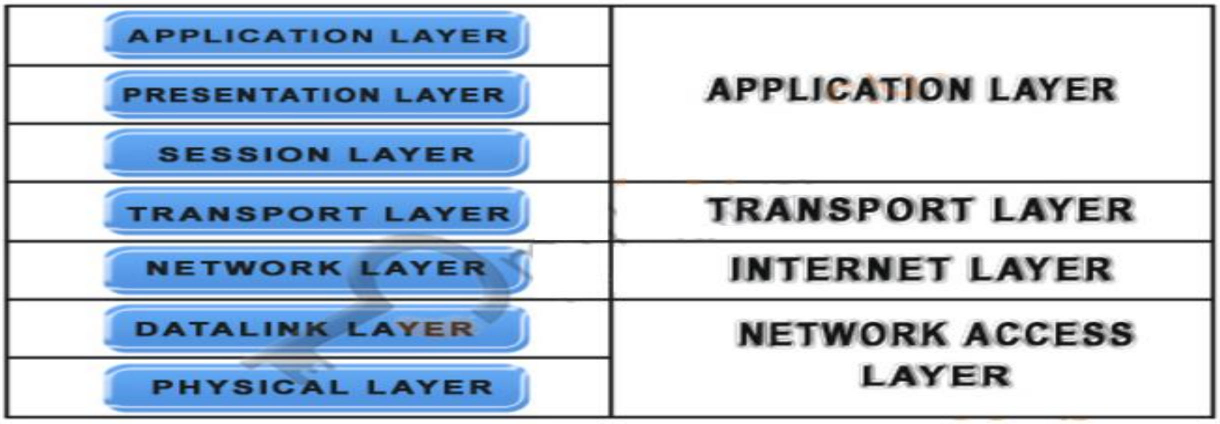


TCP Model

Reference	Layer	Function
4	Application Layer	• Deals with application programs using the network • Offers services for users to communicate over a network
3	Host-to-Host Transport Layer	• Responsible for providing end-to-end data integrity through a highly reliable communication service
2	Internetwork Layer	• Responsible for routing messages through internetworks using devices such as gateway and router ○ A gateway is a computer that has two network adapter cards, one for accepting the network packets from one network and another for routing these packets to a different network ○ A router is a dedicated hardware device that routes packets from one network to a different network
1	Network Access Layer	• Defines the transmission of a frame over a network • Exchanges the data between a computer and the physical network • Delivers data between two devices on the same network

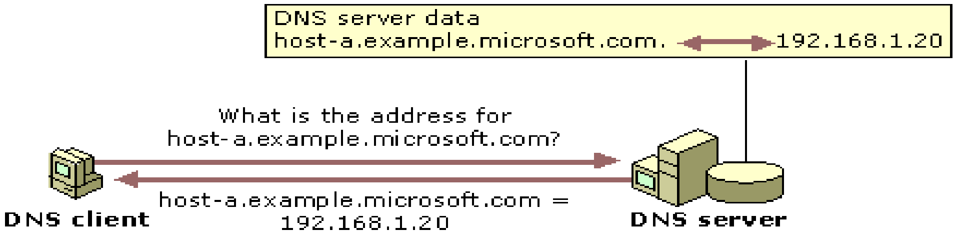

Network Topology
Host A
Router
Router
Host B
Data Flow
Application
process-to-process
Application
Transport
host-to-host
Transport
Internet
Internet
Internet
Internet
Link
Link
Link
Link
Ethernet
Fiber, Satellite, etc.
Ethernet


7
6
5
Application
Name System
DNS
File Sharing
NFS
Host Config
BOOTP
DHCP
Network Mgmt
SNMP
RMON
File Transfer
FTP
TFTP
E-Mail & News
RFC822 / MIME
SMTP
POP / IMAP
NNTP
WWW & Gopher
HTTP
Gopher
Inter-active
Telnet
"r" Com-mands
IRC
4
Transport
User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
3
Internet
Internet Protocol (IP/IPv4, IPv6)
IP NAT
IPSec
Mobile IP
IP Support Protocols
ICMP/ICMPv4, ICMPv6
Neighbor Discovery (ND)
IP Routing Protocols
RIP, OSPF, GGP, HELLO, IGRP, EIGRP, BGP, EGP
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)
2
Network Interface
Serial Line Interface Protocol (SLIP)
Point-to-Point Protocol (PPP)
(LAN/WLAN/WAN Hardware Drivers)



APPLICATION LAYER	APPLICATION LAYER
PRESENTATION LAYER	
SESSION LAYER	
TRANSPORT LAYER	TRANSPORT LAYER
NETWORK LAYER	INTERNET LAYER
DATALINK LAYER	NETWORK ACCESS LAYER
PHYSICAL LAYER	

Domain Name System (DNS)
It is used for naming computers and network services that is organized into a hierarchy of **domains**. **DNS** naming is used in TCP/IP networks, such as the Internet, to locate computers and services through user-friendly **names**.


DNS server data
host-a.example.microsoft.com. ↔ 192.168.1.20
What is the address for host-a.example.microsoft.com?
host-a.example.microsoft.com = 192.168.1.20
DNS client
DNS server

Network File System (NFS)
It **allows a user on a client computer to access files over a computer network** much like local storage is accessed.

Bootstrap Protocol (BOOTP)
It **automatically assigns an IP address to network devices from a configuration server.** The **BOOTP** was originally defined in RFC 951.

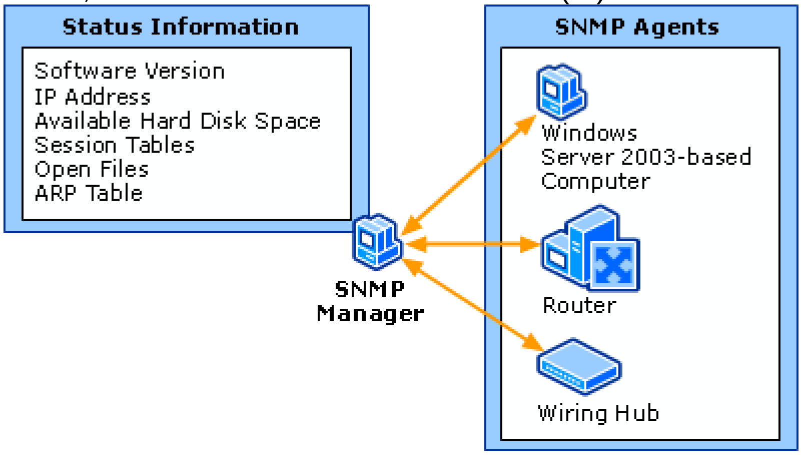
Dynamic Host Configuration Protocol (DHCP)
It automatically assign an IP address to a computer from a defined range of numbers (i.e., a scope) configured for a given network.

Differences between BOOTP and DHCP

- BOOTP supports a **limited number of client configuration parameters called vendor extensions,** while DHCP supports a larger and extensible set of client configuration parameters called options.
- BOOTP uses a two-phase bootstrap configuration process in which clients contact BOOTP servers to perform address determination and boot file name selection, and clients contact Trivial File Transfer Protocol (TFTP) servers to perform file transfer of their boot image. DHCP uses a single-phase boot configuration process whereby a DHCP client negotiates with a DHCP server to determine its IP address and obtain any other initial configuration details it needs for network operation.

- BOOTP clients do not rebind or renew configuration with the BOOTP server except when the system restarts, while DHCP clients do not require a system restart to rebind or renew configuration with the DHCP server. Instead, clients automatically enter the Rebinding state at set timed intervals to renew their leased address allocation with the DHCP server. This process occurs in the background and is transparent to the user.

Simple Network Management Protocol (SNMP)
It is used for **collecting** information from, and **configuring**, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP) network.



Remote Monitoring (RMON)
It is a standard specification that **facilitates the monitoring of network operational activities** through the **use of remote devices known as monitors or probes. RMON** assists network administrators (NA) with efficient network infrastructure control and management.

File Transfer Protocol (FTP)
It is used to transfer computer files between a client and server on a computer network.

Trivial File Transfer Protocol (TFTP)
It is used for transferring files that is simpler to use than the File Transfer Protocol (FTP) but less capable. It is used where user authentication and directory visibility are not required.

Request for Comments (RFC)
It is a type of publication from the Internet Engineering Task Force (IETF) and the Internet Society (ISOC), the principal technical development and standards-setting bodies for the Internet.

Simple Mail Transfer Protocol (SMTP)
It is used for electronic mail (email) transmission.

Post Office Protocol (**POP**)
It is used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

IMAP (Internet Message Access Protocol)

It is a standard email protocol that stores email messages on a mail server, but allows the end user to view and manipulate the messages as though they were stored locally on the end user's computing device(s).

Network News Transfer Protocol (NNTP)
It is an application protocol used for transporting Usenet news articles (netnews) between news servers and for reading and posting articles by end user client applications.

Hypertext Transfer Protocol (HTTP)
It is an application protocol for distributed, collaborative, hypermedia information systems. **HTTP** is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

Gopher protocol (TCP/IP application layer)
It is designed for distributing, searching, and retrieving documents over the Internet.

Telnet (TCP/IP protocol)
It is a user command for accessing remote computers. Through **Telnet**, an administrator or another user can access someone else's computer remotely.

Internet Relay Chat Protocol (IRCP)
It is an application layer protocol that facilitates communication in the form of text. The chat process works on a client/server networking model. IRC clients are computer programs that a user can install on their system.

User Datagram Protocol (**UDP**)
It is simplest Transport Layer communication protocol available of the TCP/IP **protocol** suite. It involves minimum amount of communication mechanism. **UDP** is said to be an unreliable transport **protocol** but it uses IP services which provides best effort delivery mechanism.

Transmission Control Protocol (**TCP**)
It is a core **protocol** of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as **TCP**/IP.

IP Nat
It enables private **IP** networks that use unregistered **IP** addresses to connect to the Internet. **NAT** operates on a router, usually connecting two networks together, and translates the private (not globally unique) addresses in the internal network into legal addresses, before packets are forwarded to another network.

Private Network--→Use unregistered IP Address---→Internet.

Routing Information Protocol (RIP)

It is one of the oldest distance-vector routing **protocols** which employ the hop count as a routing metric. **RIP** prevents routing loops by implementing limit on the number of hops allowed in a path from source to destination.

Each rip router maintains a routing table which is a list of all the destinations networks it knows how to reach along with the distance to that destination.

Open Shortest Path First (OSPF)

It uses a link state routing (LSR) algorithm and falls into the group of interior routing **protocols**, operating within a single autonomous system (AS). It is defined as **OSPF** Version 2 in RFC 2328 (1998) for IPv4.

Address resolution protocol (ARP(IP----Hardware Address))

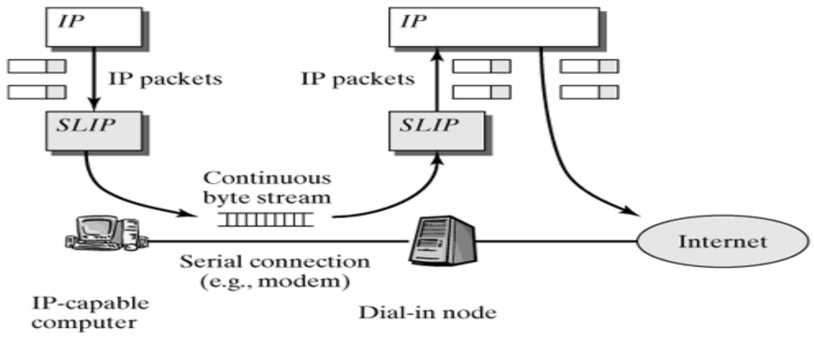
It is a **protocol** used to map IP network addresses to the hardware addresses used by a data link **protocol**. It below the network layer as a part of the interface between the OSI network and OSI link layer.

Reverse Address Resolution Protocol (RARP (Hardware Address ----IP))

It is an obsolete computer networking **protocol** used by a client computer to request its Internet **Protocol**(IPv4) address from a computer network, when all it has available is its link layer or hardware address, such as a MAC address.

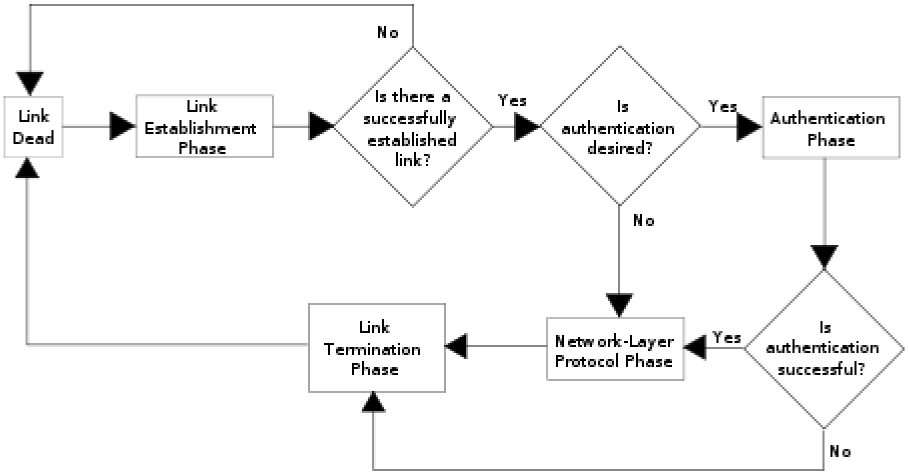
Serial Line Internet Protocol (SLIP)

It is an encapsulation of the Internet Protocol designed to work over serial ports and modem connections. It is documented in RFC 1055.

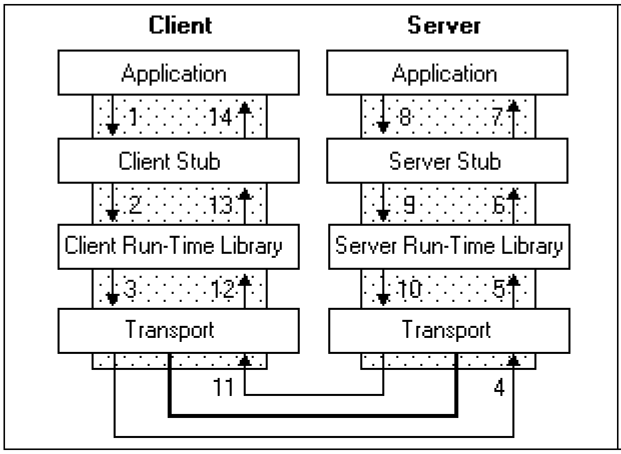


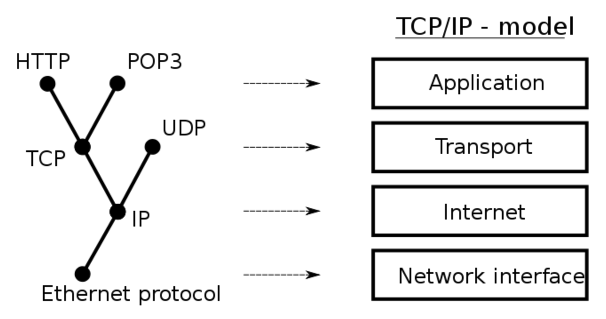
Point-to-Point Protocol (PPP)

It is used to establish a direct connection between two nodes. It can provide connection authentication, transmission encryption (using ECP, RFC 1968), and compression.

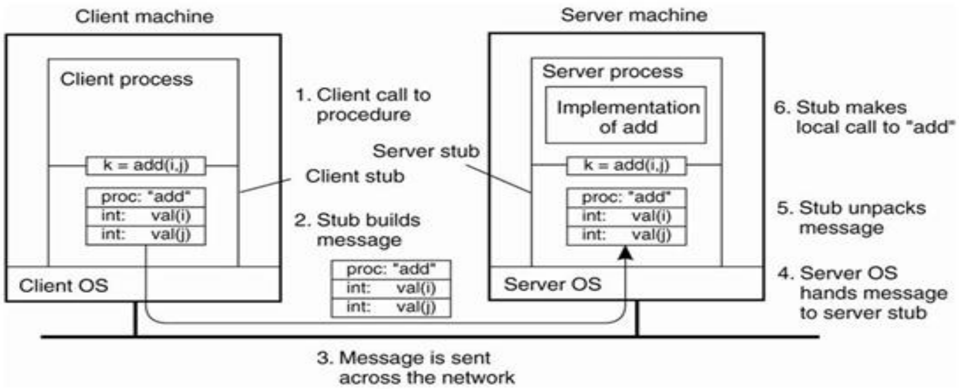

No
Link Dead
Link Establishment Phase
Is there a successfully established link?
Yes
Is authentication desired?
Yes
Authentication Phase
No
Link Termination Phase
Network-Layer Protocol Phase
Yes
Is authentication successful?
No

Remote Procedure Call (**RPC**)
It is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. (A procedure call or function call or a subroutine call.) **RPC** uses the client/server model.



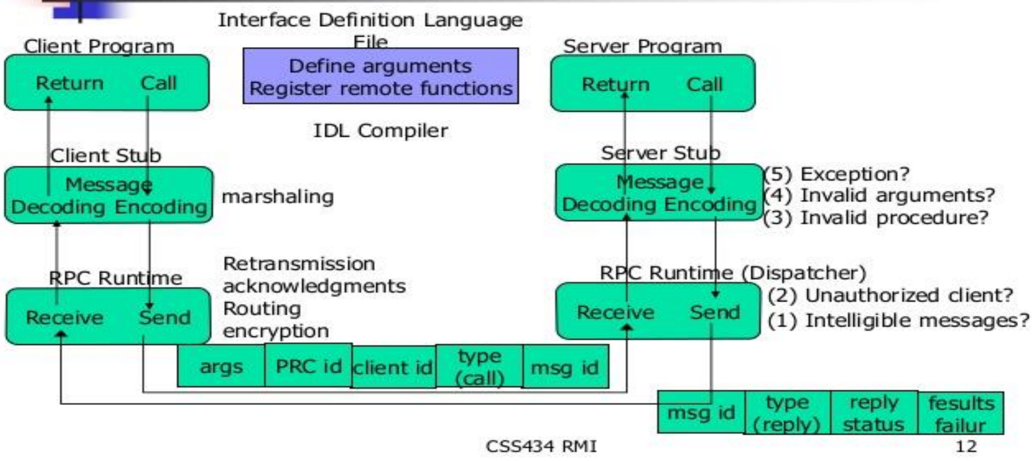


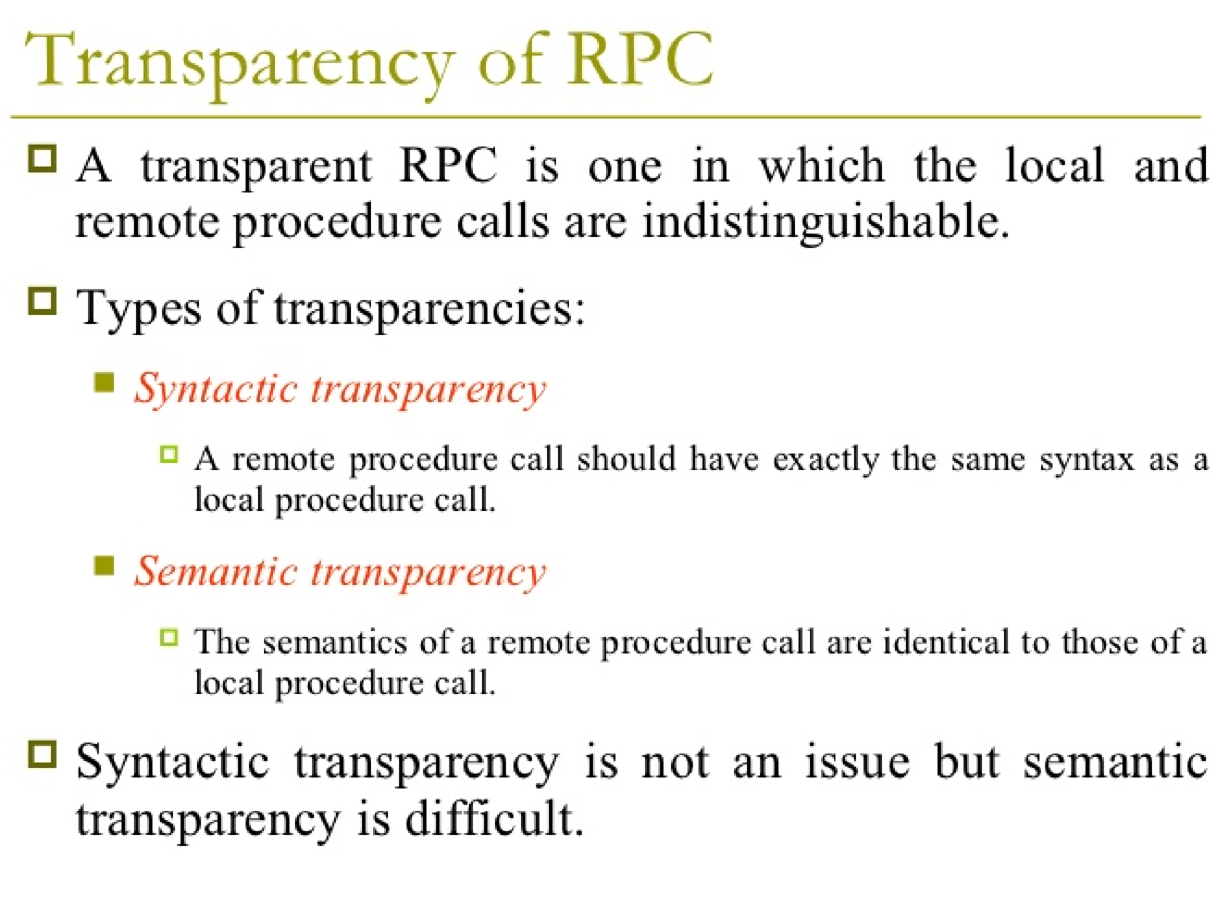
Basic RPC Operation



The steps involved in a doing a remote computation through RPC.

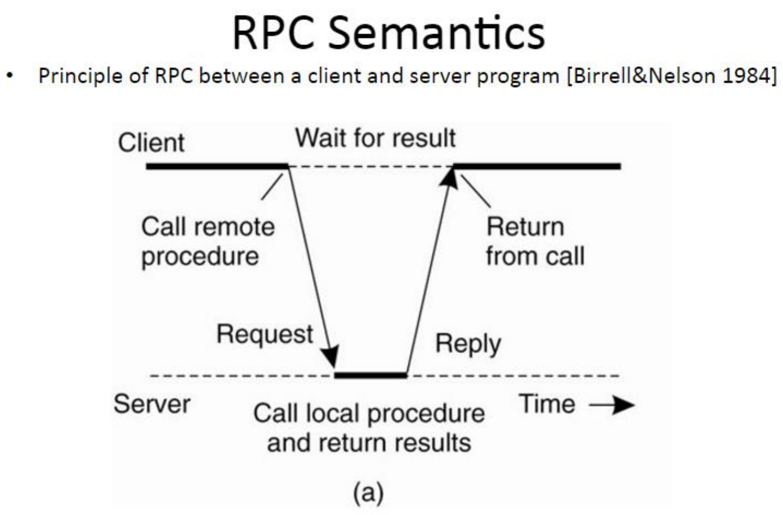
RPC Mechanism





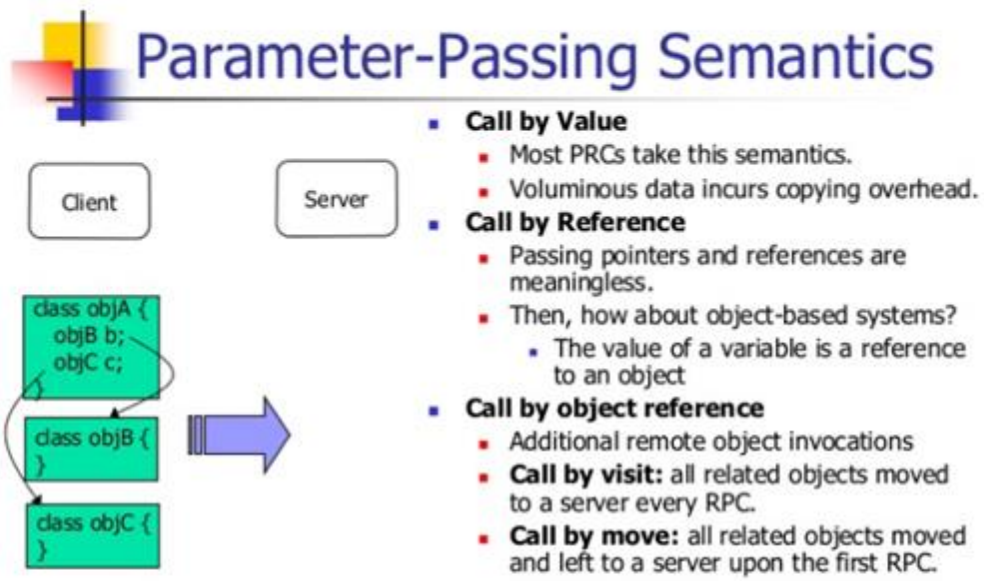
Transparency of RPC

- A transparent RPC is one in which the local and remote procedure calls are indistinguishable.
- Types of transparencies:
  - *Syntactic transparency*
    - A remote procedure call should have exactly the same syntax as a local procedure call.
  - *Semantic transparency*
    - The semantics of a remote procedure call are identical to those of a local procedure call.
- Syntactic transparency is not an issue but semantic transparency is difficult.



RPC Semantics

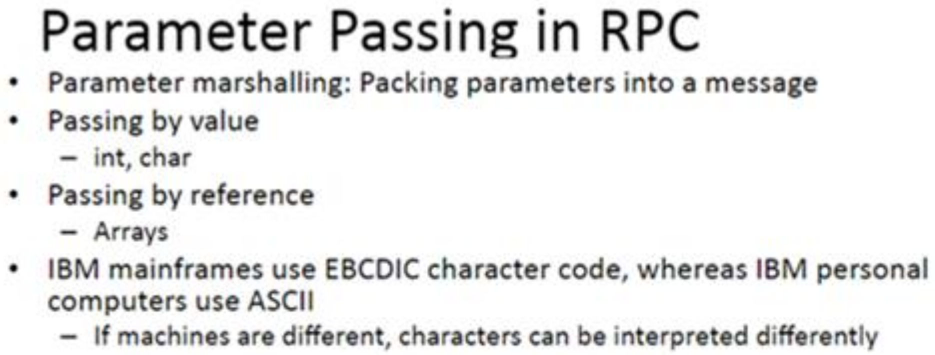
- Principle of RPC between a client and server program [Birrell&Nelson 1984]


Client
Server

```
class objA {
  objB b;
  objC c;
}
class objB {
}
class objC {
}
```

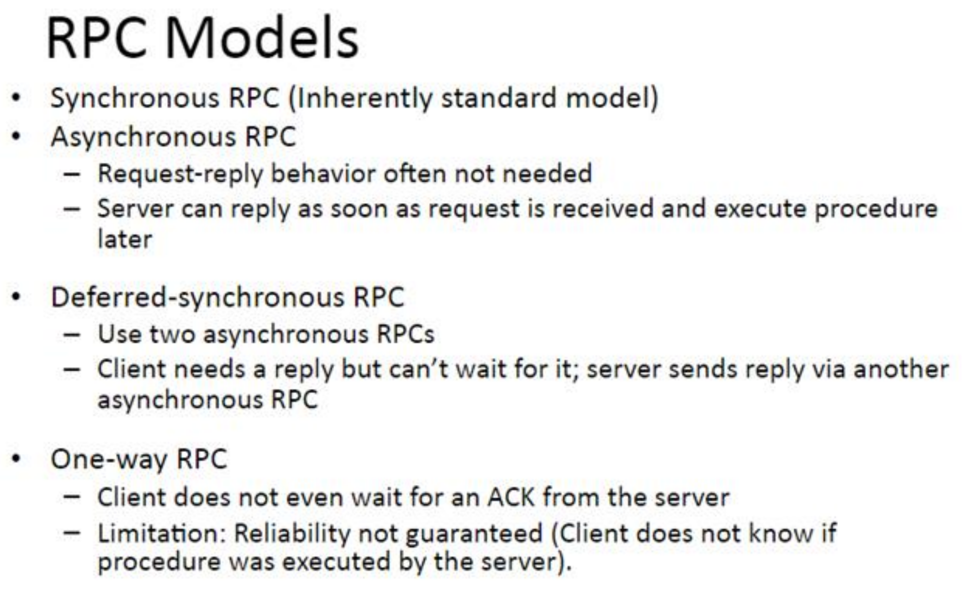
Parameter-Passing Semantics

- **Call by Value**
  - Most PRCs take this semantics.
  - Voluminous data incurs copying overhead.
- **Call by Reference**
  - Passing pointers and references are meaningless.
  - Then, how about object-based systems?
    - The value of a variable is a reference to an object
- **Call by object reference**
  - Additional remote object invocations
  - **Call by visit:** all related objects moved to a server every RPC.
  - **Call by move:** all related objects moved and left to a server upon the first RPC.



Parameter Passing in RPC

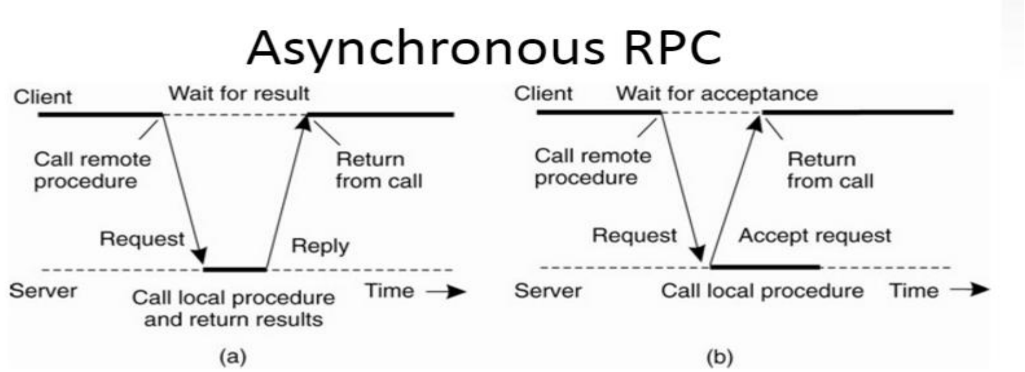
- Parameter marshalling: Packing parameters into a message
- Passing by value
  - int, char
- Passing by reference
  - Arrays
- IBM mainframes use EBCDIC character code, whereas IBM personal computers use ASCII
  - If machines are different, characters can be interpreted differently



RPC Models

- Synchronous RPC (Inherently standard model)
- Asynchronous RPC
  - Request-reply behavior often not needed
  - Server can reply as soon as request is received and execute procedure later
- Deferred-synchronous RPC
  - Use two asynchronous RPCs
  - Client needs a reply but can't wait for it; server sends reply via another asynchronous RPC
- One-way RPC
  - Client does not even wait for an ACK from the server
  - Limitation: Reliability not guaranteed (Client does not know if procedure was executed by the server).

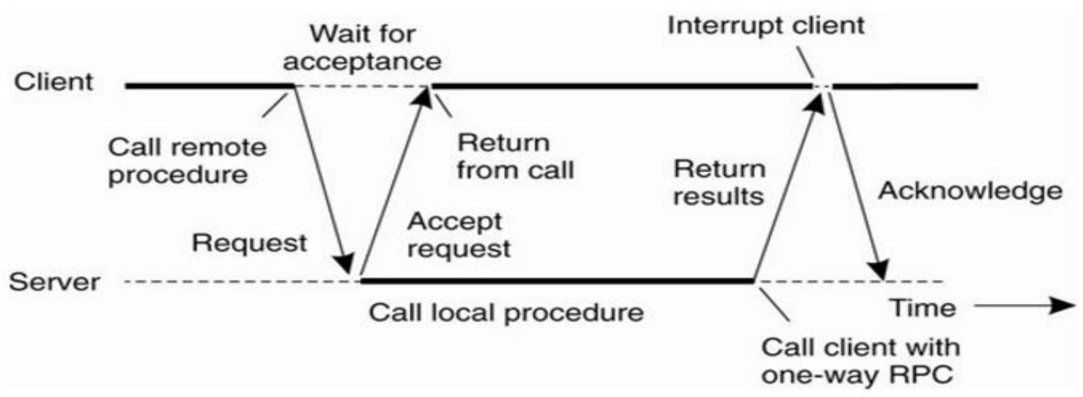
Asynchronous RPC



a) The interconnection between client and server in a traditional RPC
b) The interaction using asynchronous RPC

Deferred Synchronous RPC

- A client and server interacting through two asynchronous RPCs

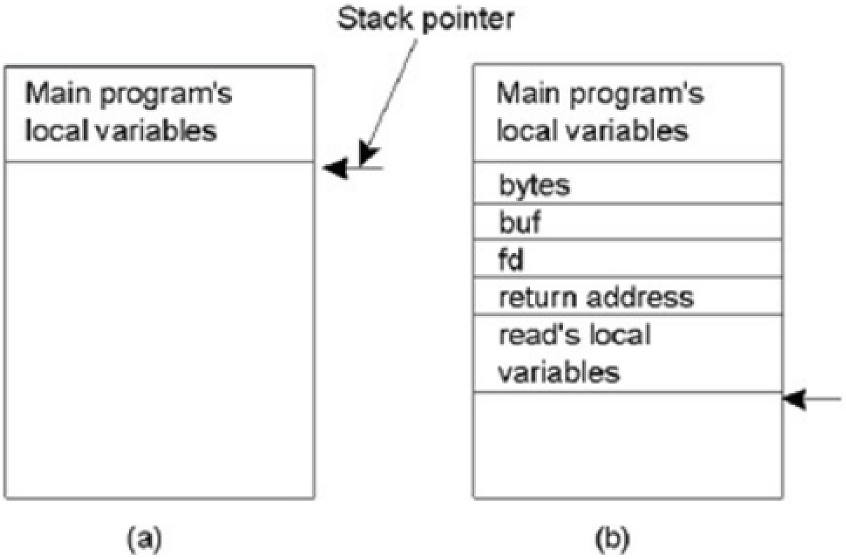


Conventional Procedure Call

a) Parameter passing in a local procedure call: The stack before the call to read

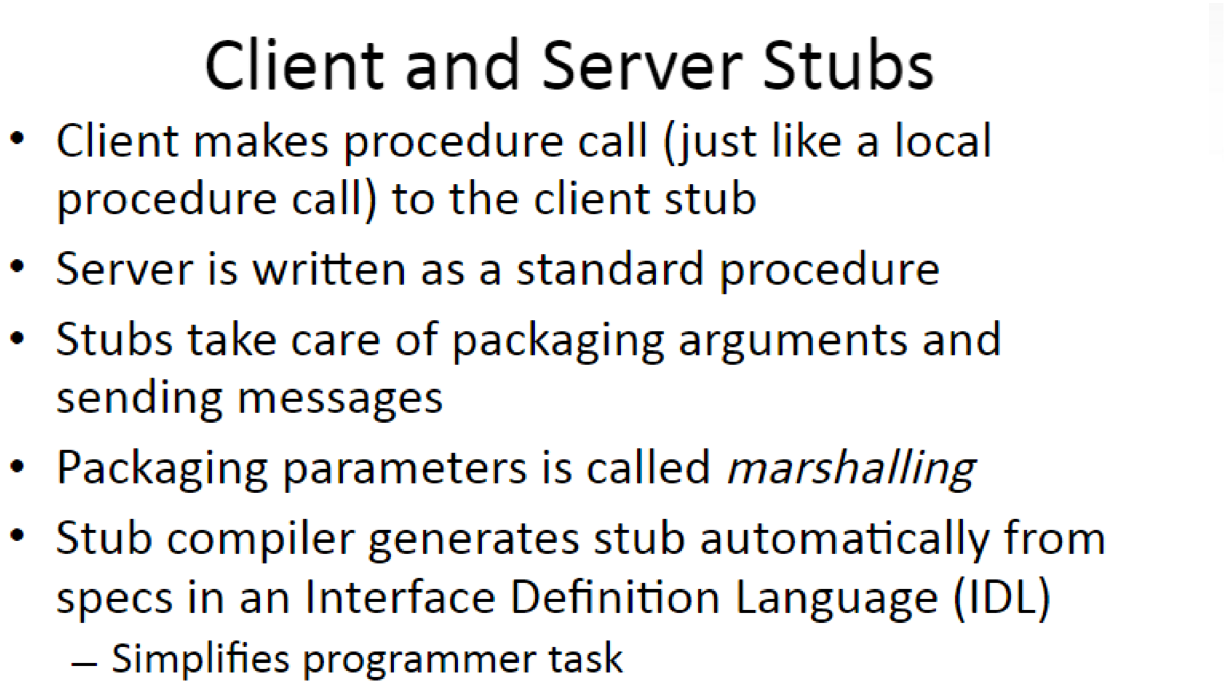
b) The stack while the called procedure is active

Count = read(fd, buf, nbytes)



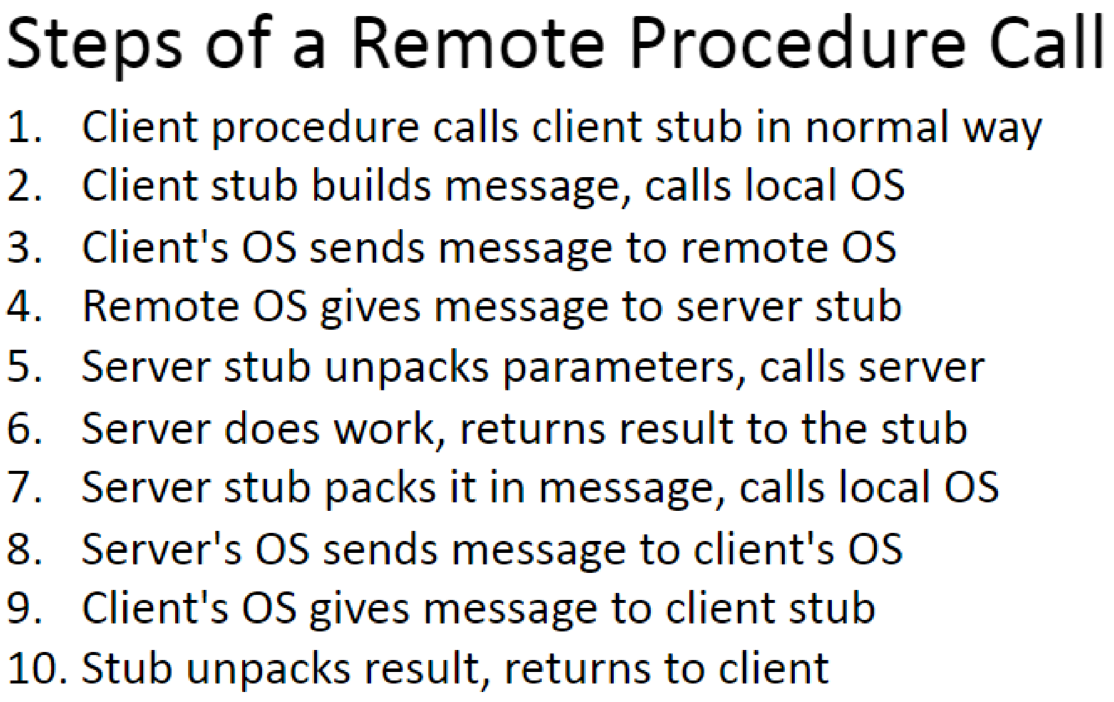
Possible Issues

- Calling and called procedures run on different machines
- They execute in different address spaces
- Parameters and results have to be passed, it can be complicated when the machines are not identical.
  - How do you represent integers – big-endian little-endian
- Either or both machines can crash and each of the possible failures causes different problems.



Client and Server Stubs

- Client makes procedure call (just like a local procedure call) to the client stub
- Server is written as a standard procedure
- Stubs take care of packaging arguments and sending messages
- Packaging parameters is called *marshalling*
- Stub compiler generates stub automatically from specs in an Interface Definition Language (IDL)
  - Simplifies programmer task

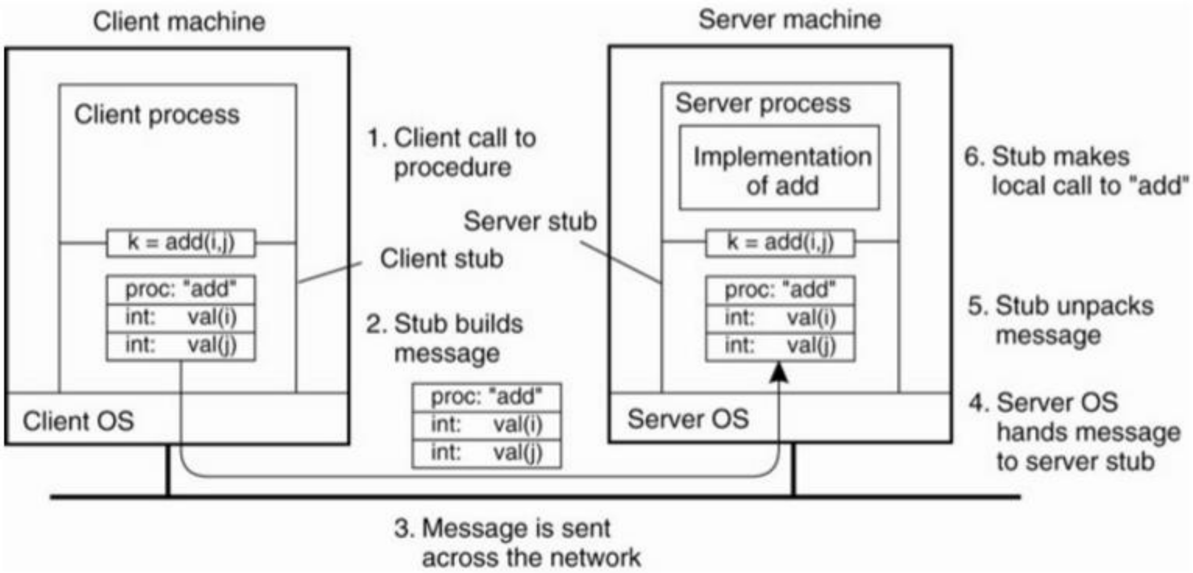


Steps of a Remote Procedure Call

1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Example of an RPC

No message passing at all is visible to the programmer.



Marshalling

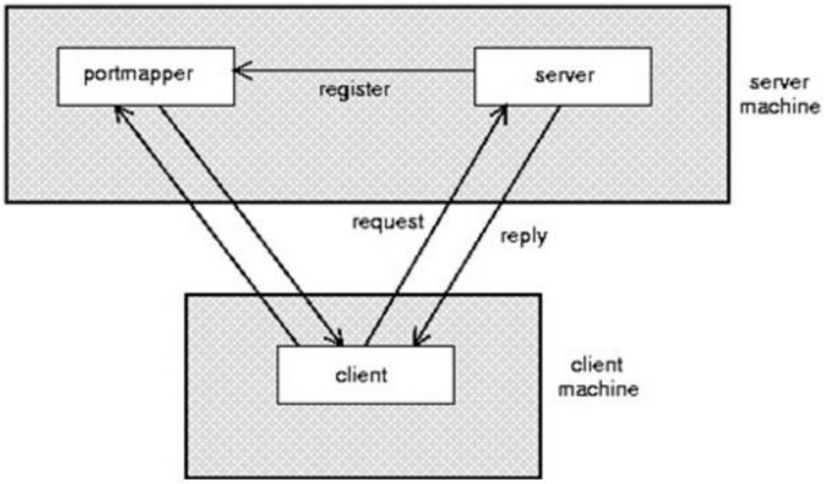
- Problem: different machines have different data formats
  - Intel: little endian, SPARC: big endian
- Solution: use a standard representation
  - Example: external data representation (XDR)
- Client stub marshals the parameters to the runtime library for transmission
- Server stub unmarshals the parameters and call the server
- The reply goes back by the reverse route

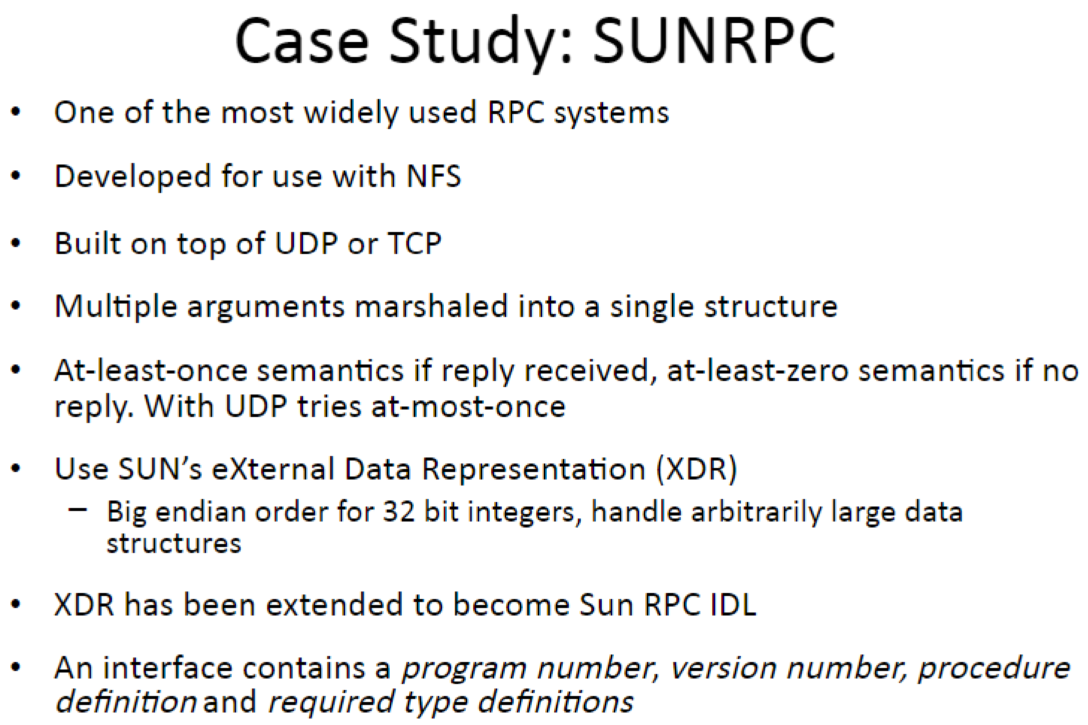
Binding

- Problem: How does a client locate a server?
  - Use Bindings
- Server
  - Export server interface during initialization
  - Send name, version no, unique identifier, handle (address) to binder
- Client
  - First RPC: send message to binder to import server interface
  - Binder: check to see if server has exported interface
    - Return handle and unique identifier to client

Binder: Port Mapper

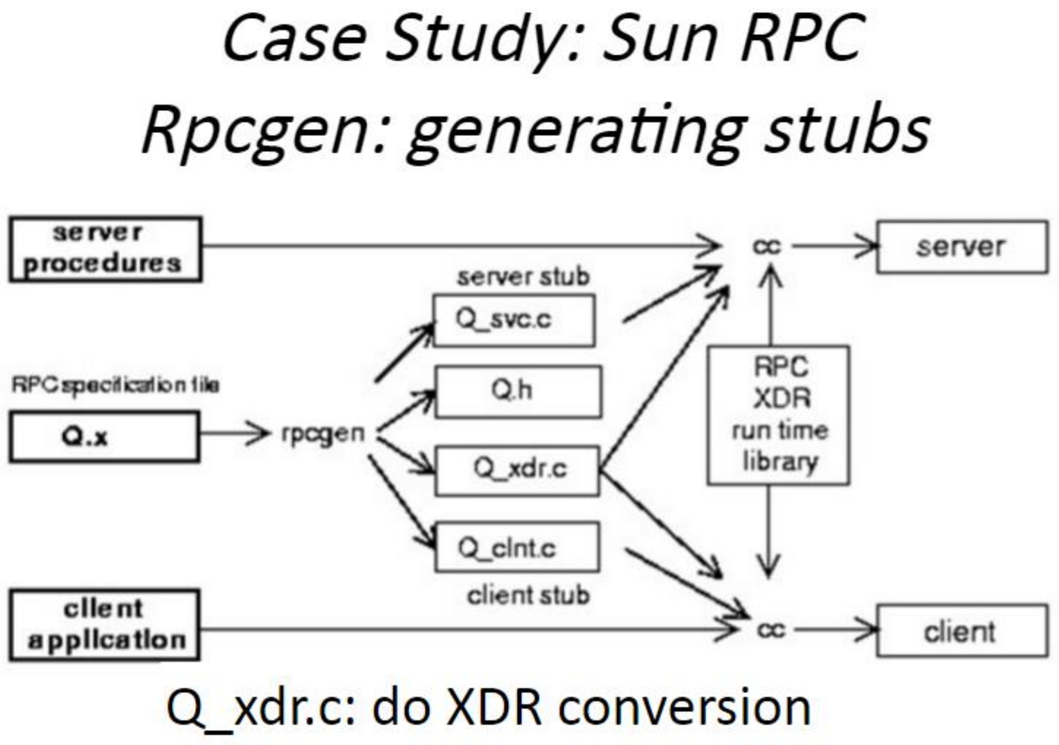
Server start-up: Create port
Server stub calls *svc_register to* register prog #, version # with local port mapper
Port mapper stores prog #, version #, and port
Client start-up: call *clnt_create* to locate server port
Upon return, client can call procedures at the server





Case Study: SUNRPC

- One of the most widely used RPC systems
- Developed for use with NFS
- Built on top of UDP or TCP
- Multiple arguments marshaled into a single structure
- At-least-once semantics if reply received, at-least-zero semantics if no reply. With UDP tries at-most-once
- Use SUN's eXternal Data Representation (XDR)
  - Big endian order for 32 bit integers, handle arbitrarily large data structures
- XDR has been extended to become Sun RPC IDL
- An interface contains a *program number, version number, procedure definition* and *required type definitions*



Case Study: Sun RPC

Rpcgen: generating stubs

Q_xdr.c: do XDR conversion

Remote Object Invocation

RMI (**Remote Method Invocation**) is a way that a programmer, using the Java programming language and development environment, can write **object**-oriented programming in which **object**s on different computers can interact in a distributed network.

UNIT-2
DISTRIBUTED SYSTEMS

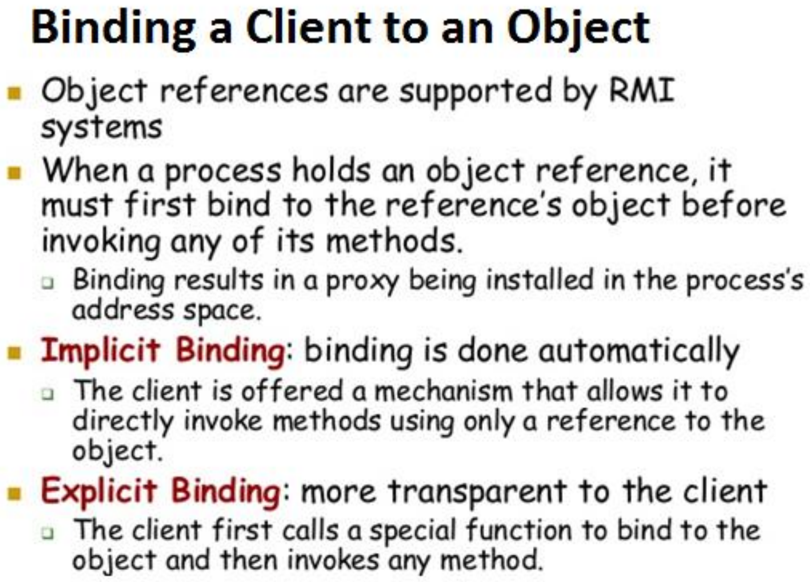
Distributed objects are objects (in the sense of object-oriented programming) that are distributed across different address spaces, either in multiple computers connected via a network or even in different processes on the same computer, but which work together by sharing data and invoking methods.

The main method of distributed object communication is with remote method invocation, generally by message-passing: one object sends a message to another object in a remote machine or process to perform some task. The results are sent back to the calling object.

Local vs. Distributed Objects

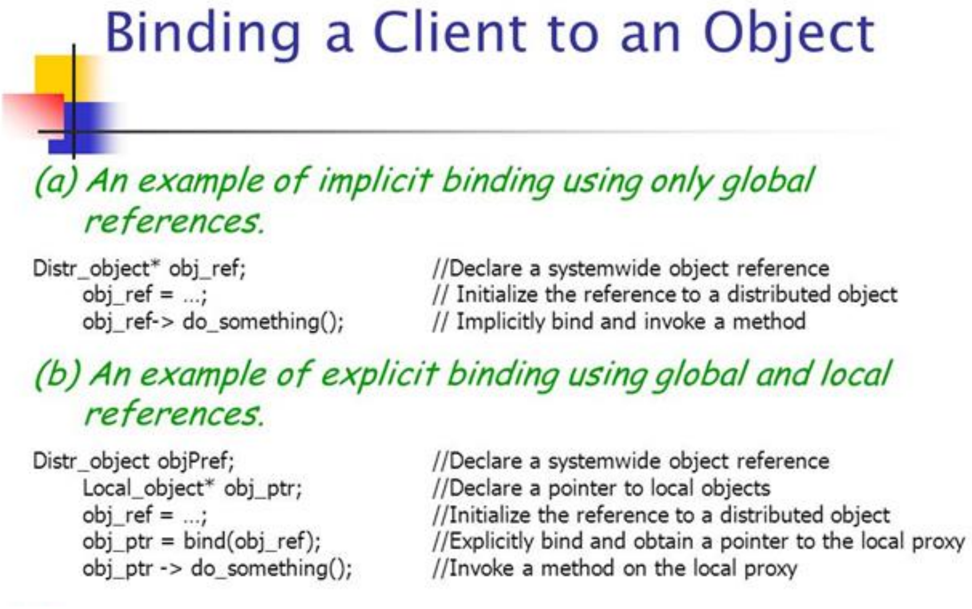
Local and distributed objects differ in many respects. Here are some of them:

1. **Life cycle** : Creation, migration and deletion of distributed objects is different from local objects
2. **Reference** : Remote references to distributed objects are more complex than simple pointers to memory addresses
3. **Request Latency** : A distributed object request is orders of magnitude slower than local method invocation
4. **Object Activation** : Distributed objects may not always be available to serve an object request at any point in time
5. **Parallelism**: Distributed objects may be executed in parallel.
6. **Communication** : There are different communication primitives available for distributed objects requests
7. **Failure**: Distributed objects have far more points of failure than typical local objects.
8. **Security**: Distribution makes them vulnerable to attack.



Binding a Client to an Object

- Object references are supported by RMI systems
- When a process holds an object reference, it must first bind to the reference's object before invoking any of its methods.
  - Binding results in a proxy being installed in the process's address space.
- **Implicit Binding**: binding is done automatically
  - The client is offered a mechanism that allows it to directly invoke methods using only a reference to the object.
- **Explicit Binding**: more transparent to the client
  - The client first calls a special function to bind to the object and then invokes any method.



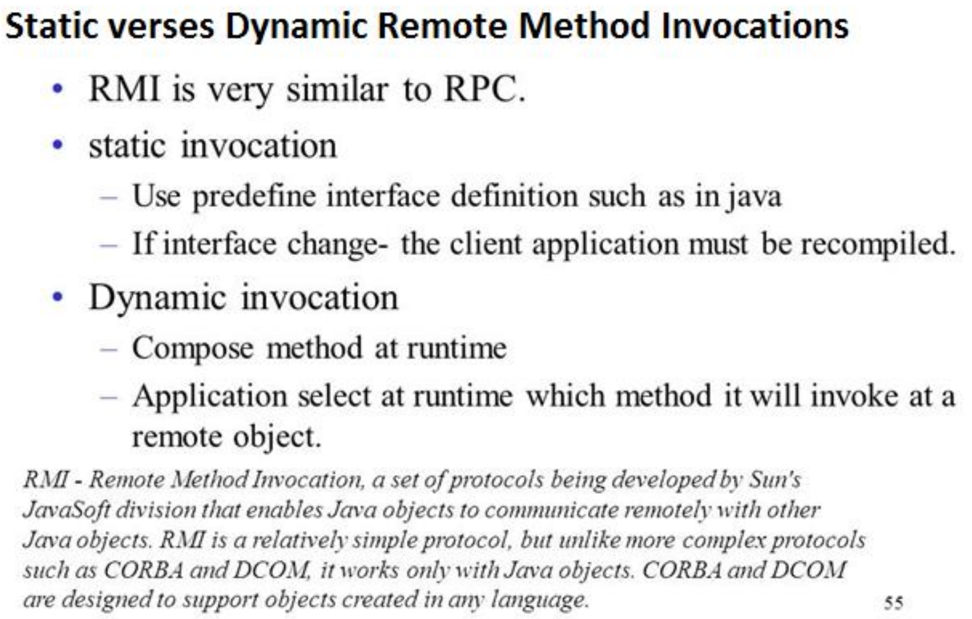
Binding a Client to an Object

(a) An example of implicit binding using only global references.

```
Distr_object* obj_ref;          //Declare a systemwide object reference
     obj_ref = ...;             // Initialize the reference to a distributed object
     obj_ref-> do_something();  // Implicitly bind and invoke a method
```

(b) An example of explicit binding using global and local references.

```
Distr_object objPref;            //Declare a systemwide object reference
     Local_object* obj_ptr;      //Declare a pointer to local objects
     obj_ref = ...;              //Initialize the reference to a distributed object
     obj_ptr = bind(obj_ref);    //Explicitly bind and obtain a pointer to the local proxy
     obj_ptr -> do_something();  //Invoke a method on the local proxy
```

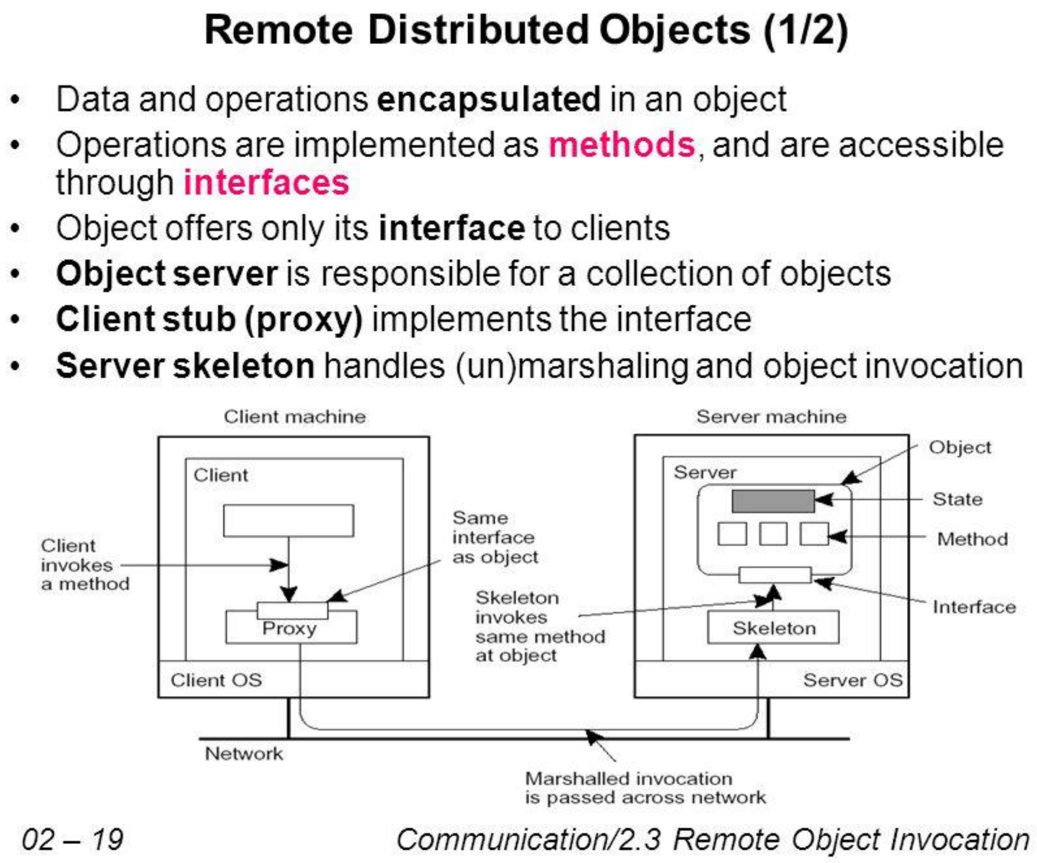


Static verses Dynamic Remote Method Invocations

- RMI is very similar to RPC.
- static invocation
  - Use predefine interface definition such as in java
  - If interface change- the client application must be recompiled.
- Dynamic invocation
  - Compose method at runtime
  - Application select at runtime which method it will invoke at a remote object.

RMI - Remote Method Invocation, a set of protocols being developed by Sun's JavaSoft division that enables Java objects to communicate remotely with other Java objects. RMI is a relatively simple protocol, but unlike more complex protocols such as CORBA and DCOM, it works only with Java objects. CORBA and DCOM are designed to support objects created in any language.

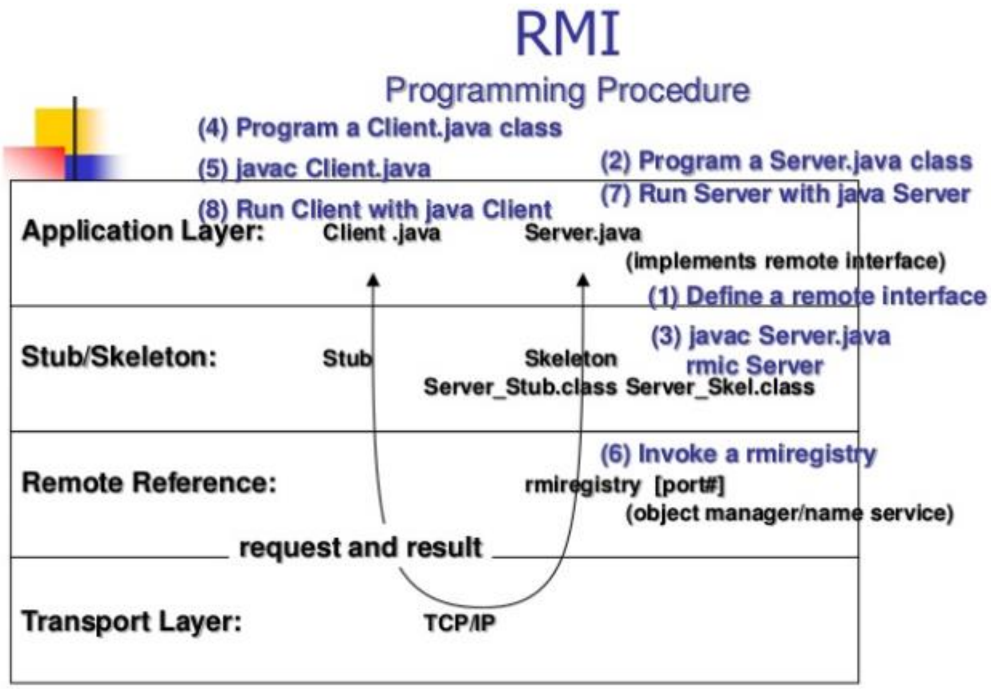
55



Remote Distributed Objects (1/2)

- Data and operations **encapsulated** in an object
- Operations are implemented as **methods**, and are accessible through **interfaces**
- Object offers only its **interface** to clients
- **Object server** is responsible for a collection of objects
- **Client stub (proxy)** implements the interface
- **Server skeleton** handles (un)marshaling and object invocation

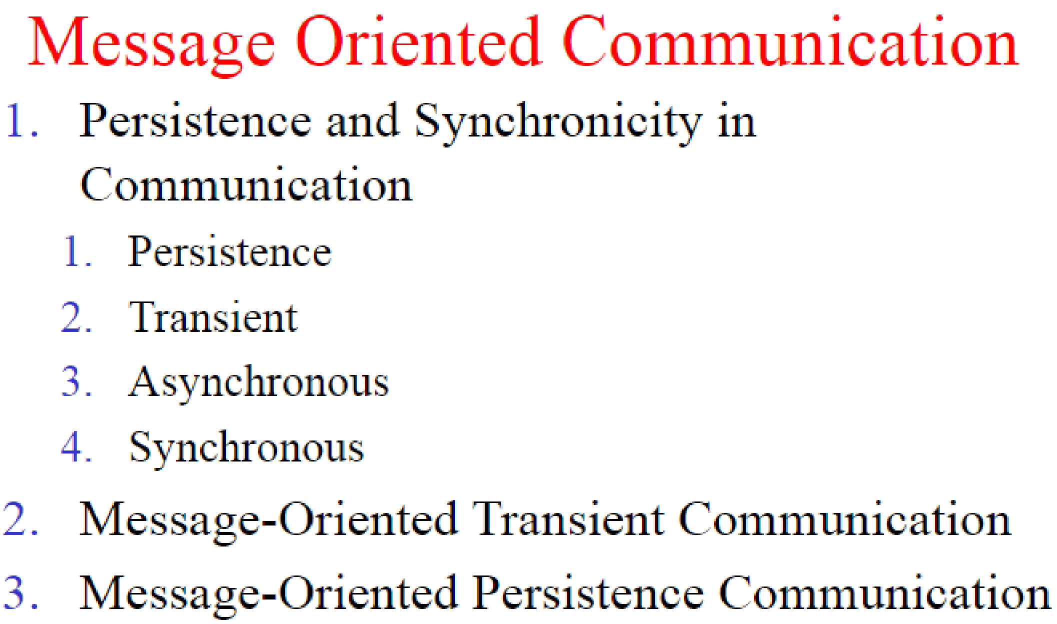
02 – 19 Communication/2.3 Remote Object Invocation



RMI

Programming Procedure

(4) Program a Client.java class
(5) javac Client.java
(8) Run Client with java Client
(2) Program a Server.java class
(7) Run Server with java Server
Application Layer:
Client .java
Server.java
(implements remote interface)
(1) Define a remote interface
Stub/Skeleton:
Stub
Skeleton
(3) javac Server.java
rmic Server
Server_Stub.class Server_Skel.class
Remote Reference:
(6) Invoke a rmiregistry
rmiregistry [port#]
(object manager/name service)
request and result
Transport Layer:
TCP/IP



Message Oriented Communication

1. Persistence and Synchronicity in Communication
    1. Persistence
    2. Transient
    3. Asynchronous
    4. Synchronous
2. Message-Oriented Transient Communication
3. Message-Oriented Persistence Communication

Message + Network

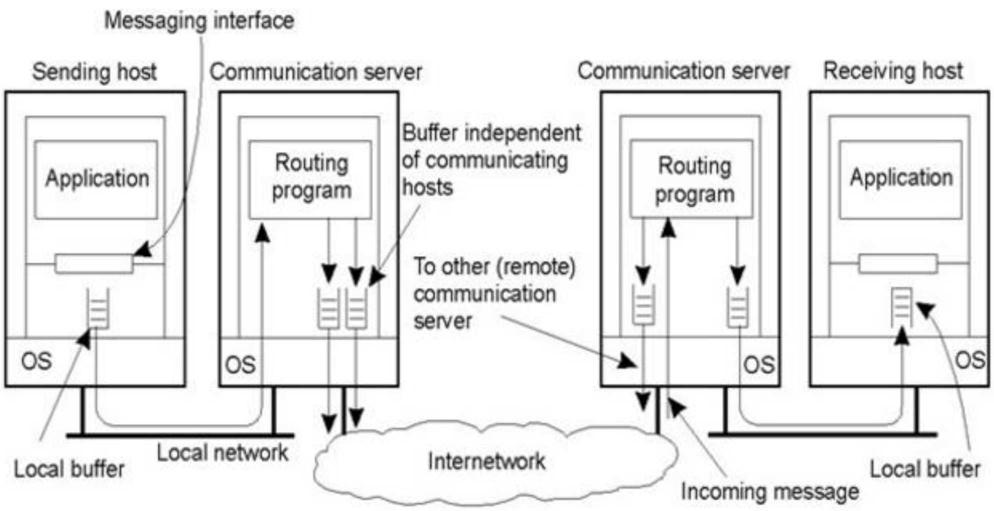
- Communication system is organized as a computer network
- Application are always executed on hosts,
- each host offer an interface to the communication system
- Each host are connected through a network of communication servers
- Which responsible for passing message between hosts

Persistence Communication

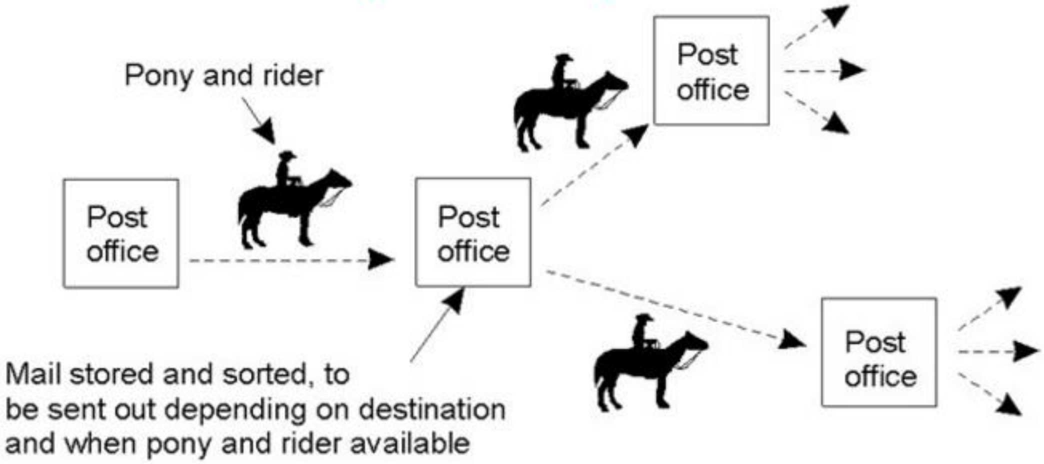
Message that has been submitted is stored by the communication system as long as it takes to deliver it to receiver.

Example: Mail

Persistence and Synchronicity in Communication



Persistence and Synchronicity in Communication


Pony and rider
Post office
Post office
Post office
Post office
Mail stored and sorted, to be sent out depending on destination and when pony and rider available

Transient Communication

A message is stored by the communication system as long as the sending and receiving application is running.

The message will be discarded if the communication server cannot delivery the message to destination server.

Example: Router.

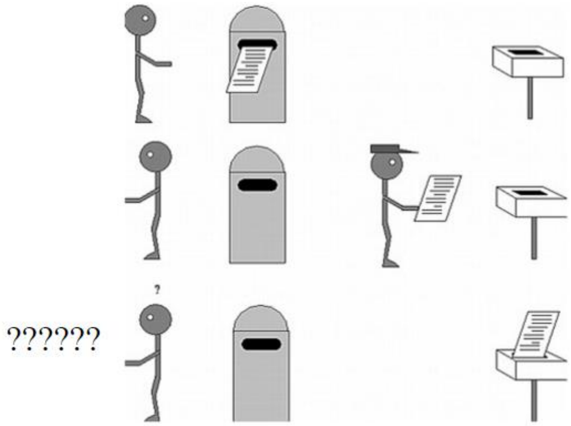
Asynchronous Communication

Sender continuous immediately after it has transmitted the message.

The message is either stored in local buffer or at the first communication server

Example: asynchronous - the answering machine

Asynchronous Send

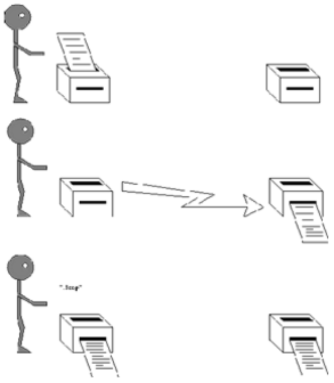


Synchronous

The sender is blocked until its message is stored in local buffer at the receiver end or the message has been delivered.

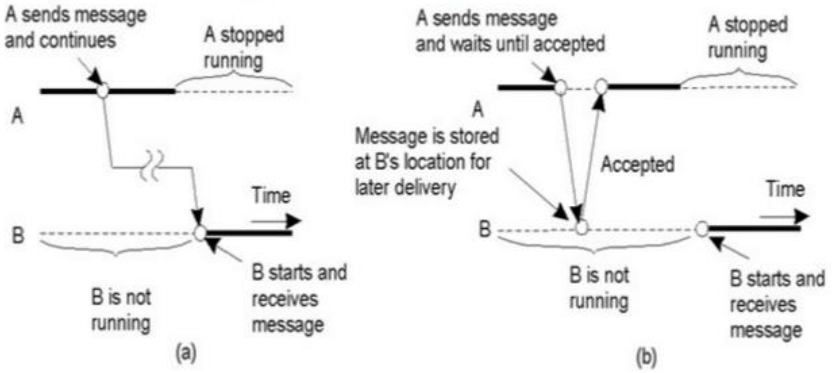
The strongest form of Syn Comm is when the sender can only continue executing after the receiver process the message.

Synchronous Send



Provide information about the relative execution points of sender and receiver - causes synchronization of the two.

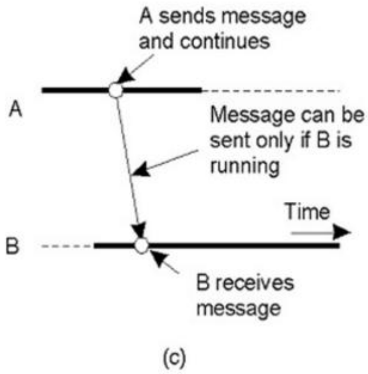
Persistence and Synchronicity in Communication (3)


A sends message and continues
A stopped running
A
B
Time
B is not running
B starts and receives message
(a)
A sends message and waits until accepted
A stopped running
A
Message is stored at B's location for later delivery
Accepted
Time
B
B is not running
B starts and receives message
(b)

Persistence Asynchronous Comm

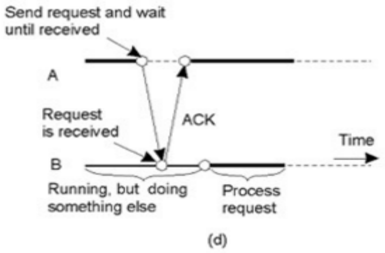
Persistence synchronous Comm

Persistence and Synchronicity in Communication


A sends message and continues
A
Message can be sent only if B is running
Time
B
B receives message
(c)

c) Transient asynchronous communication

Transient Synchronous Communications


Send request and wait until received
A
Request is received
ACK
Time
B
Running, but doing something else
Process request
(d)

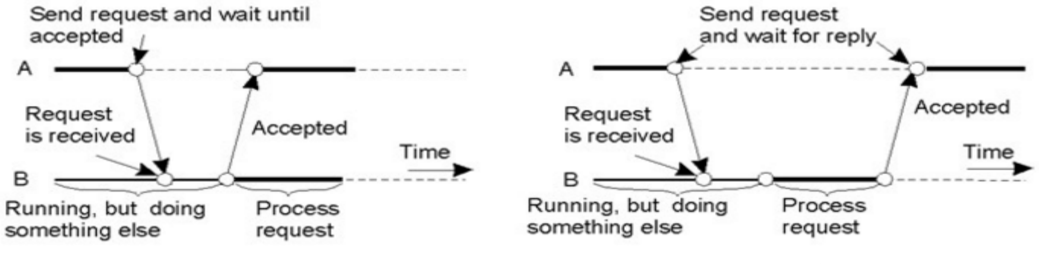
d) Receipt-based transient synchronous communication

Weakest form, based on message receipt.

The sender is blocked until the message is stored in receiver's local buffer.

The sender receive an acknowledge(receipt) and continue.

Persistence and Synchronicity in Communication (5)


Send request and wait until accepted
A
Request is received
Accepted
Time
B
Running, but doing something else
Process request
Send request and wait for reply
A
Request is received
Accepted
Time
B
Running, but doing something else
Process request

(e) (f)

e) Delivery-based transient synchronous communication at message delivery –client idle until its request has been accepted for further processing

f) Response-based transient synchronous communication-client waits until receives a reply from the server. Ie- client-server.

Berkeley Sockets

Socket primitives for TCP/IP.

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Message-Oriented Persistence Communication

Message queuing system/Message Oriented Middleware

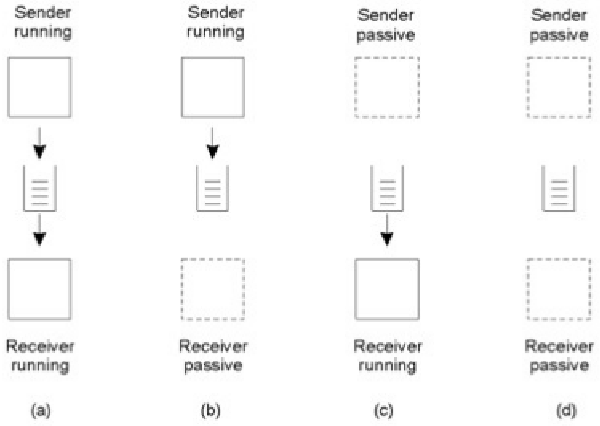
Offer intermediate-term storage capacity for message- without requiring sender & receiver to active

Support longer time message transfer

Message Queuing Model

- App'n communicate by inserting message in its own private queue
- Also possible the queue being shared by other App'n
- Message are guaranteed to be inserted in queue but not to receive by receiver
- Message is forwarded over a series of communication servers
- Receiver and sender is independent.

Message-Queuing Model



Four combinations for loosely-coupled communications using queues.

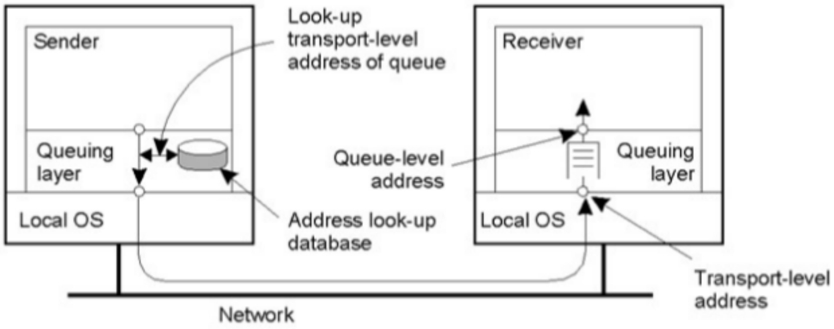
Message-Queuing Model

Basic interface to a queue in a message-queuing system.

Primitive	Meaning
Put	Append a message to a specified queue
Get	Block until the specified queue is nonempty, and remove the first message
Poll	Check a specified queue for messages, and remove the first. Never block.
Notify	Install a handler to be called when a message is put into the specified queue.

General Architecture of a Message-Queuing System (1)

The relationship between queue-level addressing and network-level addressing.

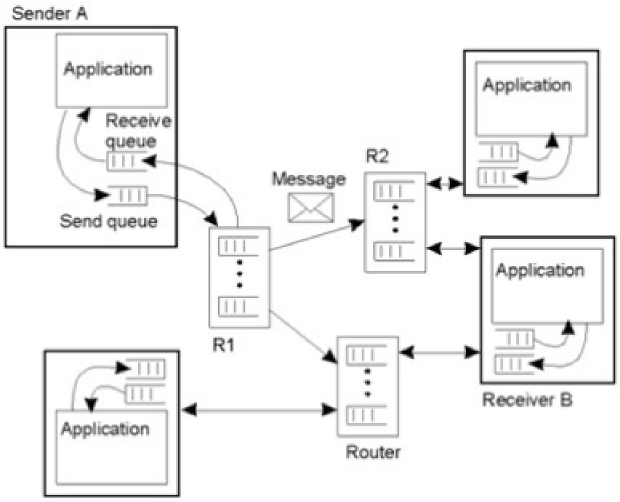


General Architecture of a Message Queuing System

- Message can only be put to local queues
- Called as source queue
- And also- message can be read from local queues
- Message put in the queue will contain the destination queues to which it should be transferred.
- Message queuing system maintain a database of queue names to network location(DNS)
- Queues are manage by queue managers
- Special queue managers that operate as routers or relay
  - Forward the incoming messages to other queue managers

General Architecture of a Message-Queuing System

Use few router with the knowledge of topology
Only the router need to be updated when queues are added/deleted
Queue manager has to know only the nearest router



The general organization of a message-queuing system with routers.

Message Broker

➢Each time the new application is added- different message format will introduce.

➢Require the sender and receiver have the same message format.- agree with common message format.

❑Conversion are handled by special node in a queuing network

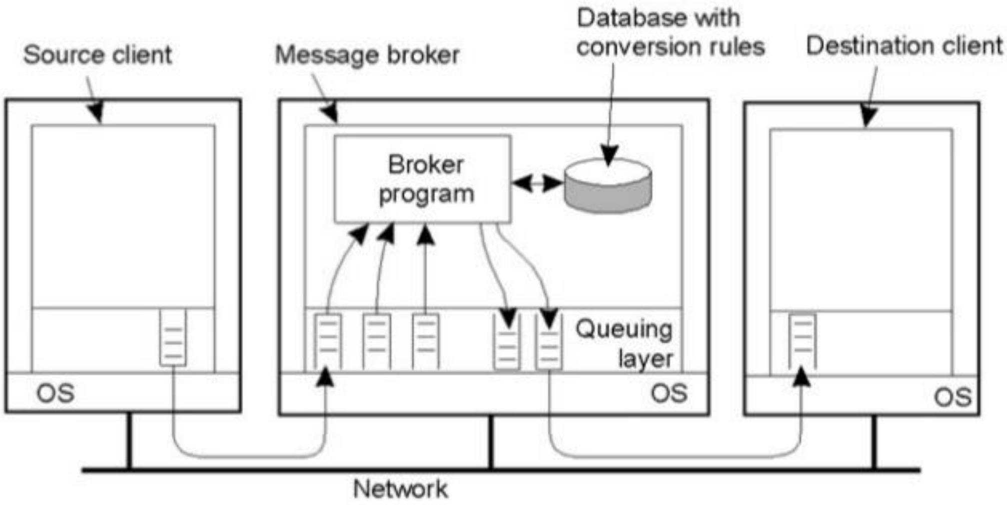
❑Known as MESSAGE BROKER.

❑Purpose – to convert the incoming message to a format that can be understood by destination application.

❑Message reformatter

Message broker is an intermediary program that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver in a telecommunication network where programs communicate by exchanging formally-defined messages.

Message Brokers



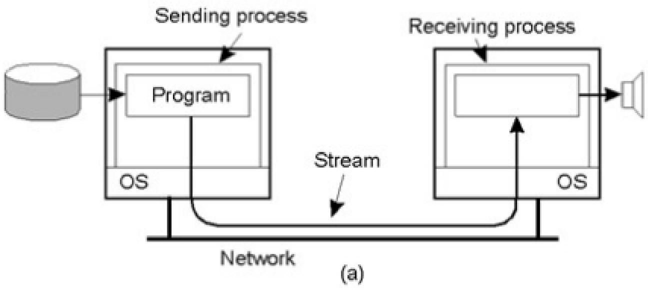
Data Stream

- Support for continuous media
- Sequence of data unit
- Isochronous Transmission mode
  - Distributed multimedia system
  - Refers as stream
- Stream /2
  - Simple stream – consist of only single sequence of data
  - Complex stream – several related simple streams(sub stream

*In **telecommunications and computing** , a **data stream** is a sequence of digitally encoded coherent signals (packets of data or datapackets) used to transmit or receive information that is in transmission.*

*In **electronics and computer architecture** , a **data stream** determines for which time which data item is scheduled to enter or leave which port of a systolic array, a Reconfigurable Data Path Array or similar pipe network, or other processing unit or block.*

Data Stream (1)



Setting up a stream between two processes across a network.

Source and sink

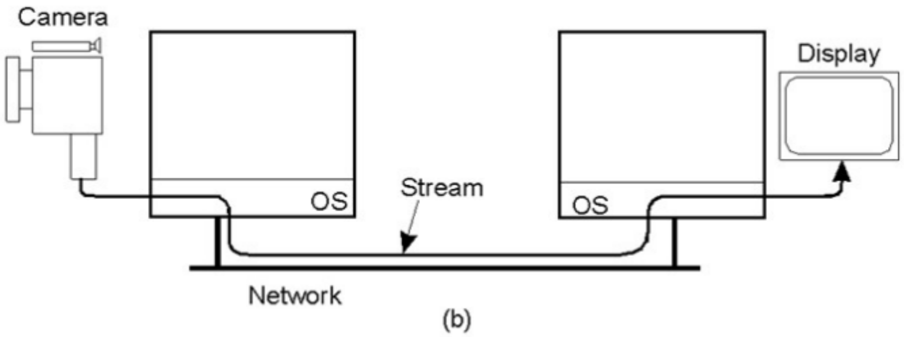
Source could be a process

Reading an audio file from a disk – transmit byte by byte

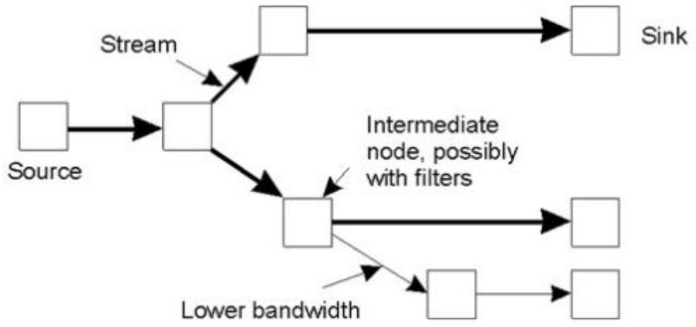
Sink – fetching the byte as they come in – passing them to local audio device

Data Stream

Setting up a stream directly between two devices.



Data Stream (3)



An example of multicasting a stream to several receivers.

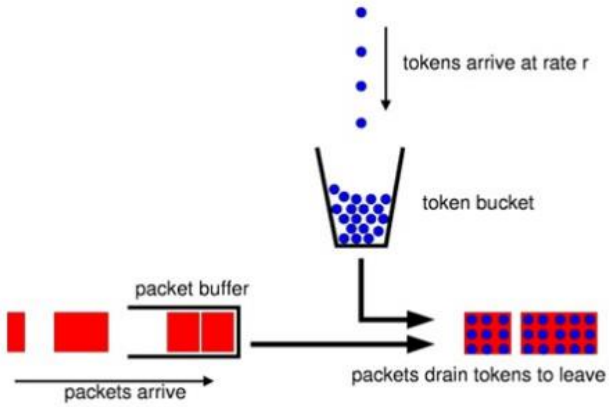
Data stream is multicast to many receivers

Filters are use to adjust the quality of incoming stream

Specifying QoS

Characteristics of the Input	Service Required
•maximum data unit size (bytes) •Token bucket rate (bytes/sec) •Toke bucket size (bytes) •Maximum **transmission** rate (bytes/sec)	•Loss sensitivity (bytes) •Loss interval (μsec) •Burst loss sensitivity (data units) •Minimum **delay** noticed (μsec) •Maximum delay variation (μsec) •Quality of guarantee

Specifying QoS (2)



The principle of a token bucket algorithm.

Flow Spec

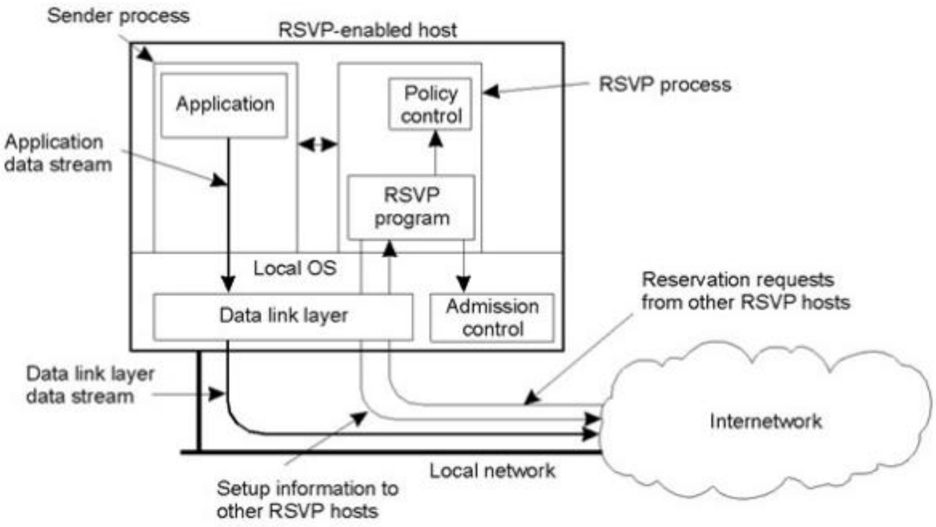
1. Loss sensitivity – acceptable loss rate

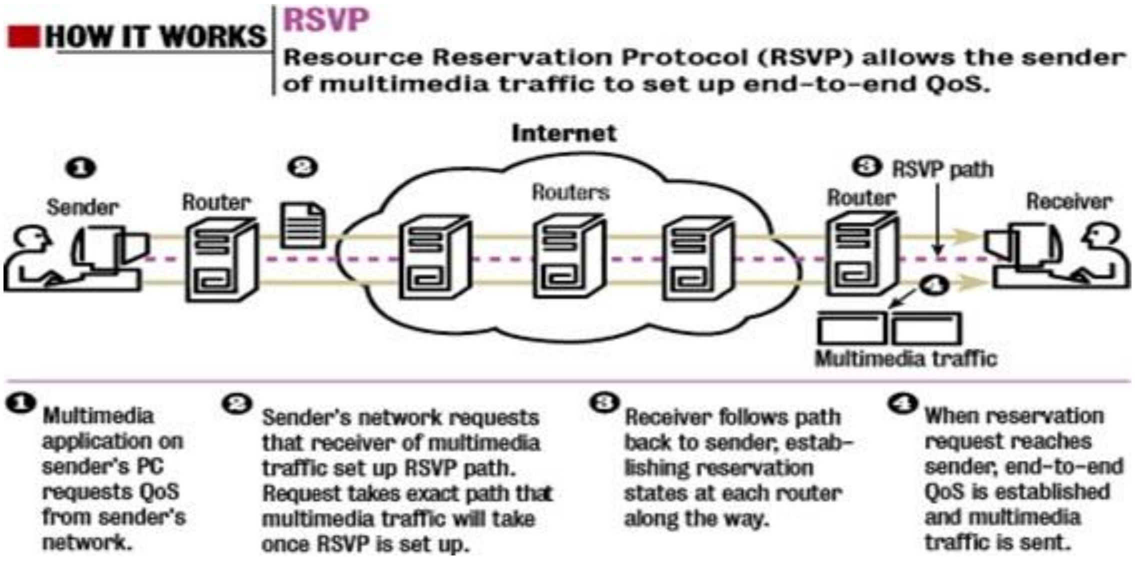
- Loss interval (µsec)
- Burst loss sensitivity (data units) – how many consecutive data unit may be lost
- Minimum **delay** noticed (µsec)- how long the tolerable delay before noticed by receiver
- Maximum delay variation (µsec)- maximum tolerate jitter for video and audio
- Quality of guarantee- how serious the service requirement should be taken

Setting up Stream

- Sender in RSVP provide flow specification
  - Bandwidth, delay, jitter etc.
- The specification is handed over to RSVP process that is colocated at the same machine as the sender
- RSVP is receiver-initiated QoS protocol(receiver are required to send reservation requests along the same path to the sender)
- Receiver may set a new parameter value(flow specification) to the sender.

Setting Up a Stream





HOW IT WORKS | **RSVP**

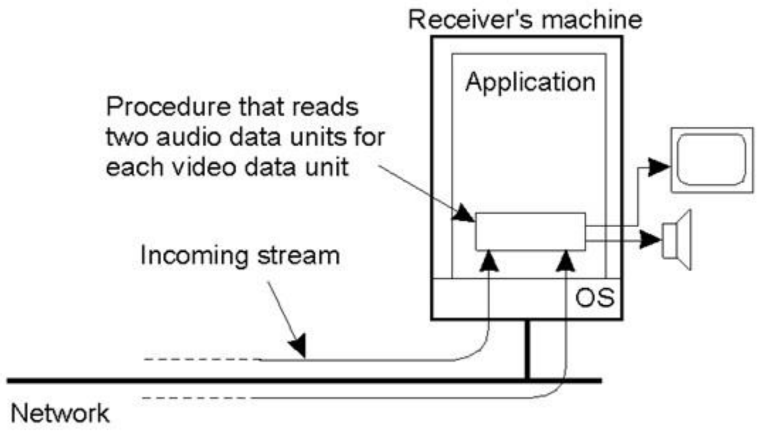
Resource Reservation Protocol (RSVP) allows the sender of multimedia traffic to set up end-to-end QoS.

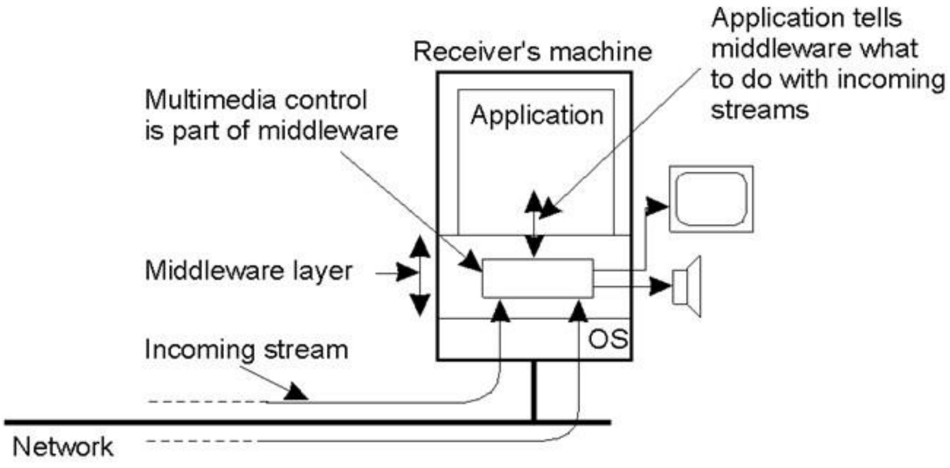
❶ Multimedia application on sender's PC requests QoS from sender's network.

❷ Sender's network requests that receiver of multimedia traffic set up RSVP path. Request takes exact path that multimedia traffic will take once RSVP is set up.

❸ Receiver follows path back to sender, establishing reservation states at each router along the way.

❹ When reservation request reaches sender, end-to-end QoS is established and multimedia traffic is sent.


Receiver's machine
Application
Procedure that reads two audio data units for each video data unit
Incoming stream
OS
Network


Receiver's machine
Application tells middleware what to do with incoming streams
Application
Multimedia control is part of middleware
Middleware layer
Incoming stream
OS
Network

Complete

UNIT-2
DISTRIBUTED SYSTEMS

http://slideplayer.com/slide/5179691/