Licence Technique (LT)

Client / Server Architectures

By: Dr. Jaber M. Jaber

27 December 2012

Dedication

To

My mother
Mahassen Ghazal Jaber
for her love and hard work

My father
Mouhammad M. Jaber
for his love and encouragement

My wife
Ramziah Rashwani Jaber
for her love, caring, and infinite patience

My children
Mouhammad, Moustafa Al Ameen, Mahassen, and Khalidah
for their love and innocence

Feed-Back

Any kind of constructive remark, suggestion, question, or information is always welcomed and is highly appreciated. Please, use the following contact information to express your feed-back:

Dr. Jaber M. Jaber

+961.3.60.31.80
jjaber@ul.edu.lb
jaber2002@hotmail.com
http://www.jaber2002.tk/

Beirut, Lebanon

Contents

List of Abbreviations

2PC:	Two-Phase Commit
ACID:	Atomicity – Consistency – Isolation - Durability
AN:	Area Network
AP:	Application Program
API:	Application Programming Interface
BL:	Business Logic
BR:	Business Rule
C/S:	Client / Server
CORBA:	Common ORB Architecture
DA:	Data Access
DB:	Database
DBMS:	DB Management System
DBS:	DB System / DB Server
E-:	Electronic
FIFO:	First In First Out
FTP:	File Transfer Protocol
GUI:	Graphical UI
HTTP:	HyperText Transfer Protocol
IDL:	Interface Definition Language
IIOP:	Internet Inter-ORB Protocol
IOR:	Interoperable Object Reference
LAN:	Local AN
LUW:	Logical Unit of Work
MAN:	Metropolitan AN
MOM:	Message-Oriented Middleware
NOS:	Network OS
OMG:	Object Management Group
OO:	Object Orientation / Object-Oriented
OOUI:	Object-Oriented UI
ORB:	Object Request Broker
OS:	Operating System
P2P:	Peer-To-Peer
PC:	Personal Computer
RPC:	Remote Procedure Call
SAN:	Storage AN
SFTP:	Secure FTP
SHTTP:	Secure HTTP
SQL:	Structured Query Language
TP:	Transaction Processing
UI:	User Interface
XML:	eXtensible Markup Language
WAN:	Wide AN
WWW:	World Wide Web

Chapter I: Fundamental Concepts of C/S Technologies

I.1 Client / Server (C/S) Components

A client/server system is composed of three components: client, server, and the communication network. The basic communication protocol between the client and the server is called Request-Response.

I.1.1 Client

A client is a logical entity that requests services from another logical entity called server. A client is often called a service requestor or service consumer.

I.1.2 Server

A server is a logical entity that provides services to different clients. A server is often called a service provider.

I.1.3 Network

An interconnected collection of autonomous electronic devices (such as computers, mobile phones, printers, routers, switches, hubs, iPads, iPods, etc.) that are capable of exchanging information.

I.1.4 Service

A service is simply a set of one or more functions / resources that are provided by a server upon client's request or server's request.

I.2 C/S Connection

The C/S architecture is based on a server machine that sends the responses to the clients via a communication network.

I.2.1 Connection Functioning

1. The end-user sends a connection request to the server by providing his or her user name and user password.
2. The user's request is translated to SQL query at the client level.
3. The query is intercepted and sent to the server along the network card information via the communication network.
4. The DB server receives and then accepts the SQL query. It processes it and then sends the response back to the client.
5. The SQL response circulates on the communication network.
6. The client receives the server's response and then presents it to the end-user.

I.2.2 Secure C/S Communication

A typical secure C/S communication follows the following steps:

1. Open the secure connection:
    a. Open the secure connection of the underlying protocol.
    b. Reciprocal authentication of communicating entities.
    c. Negotiation of the transfer security mode.
    d. Exchange of session key.
2. Transfer data:

a. The sender encrypts the message then constructs the secure packets.
b. The receiver decrypts the message and detects the eventual attacks.
c. The session key should be changed periodically.

3. Close the secure connection:
   a. Close the secure connection of the underlying protocol.
   b. Destruct the context of the secure connection.

I.3 Client User Interface Types

I.3.1 Text-Based / Non-GUI User Interface

Text-based (also called non-GUI) user interfaces give out the requests to the server with a minimum human interaction.
Text-based interfaces are classified into two categories:

- Multi-Task Non-GUI applications like robots and testers.
- Non-Multi-Task Non-GUI (also called Single-Task Non-GUI) applications like ATMs and barcode readers.

I.3.2 Graphical User Interface (GUI)

- GUI interfaces involve more interactions between the end-user and the graphical interface.
- GUI dialogs implement the event-driven model and object/action model.
- Most dialogs are sequential by nature.

I.3.3 Object-Oriented User Interface (OOUI)

- OOUI interfaces are used by people who do multiple variable tasks without a predefined order.
- OOUI interfaces give multiple views over objects.
- Objects are manipulated directly on the screen (in general via drag & drop operations).

I.4 Server Types

I.4.1 Operating System (OS) / Network OS (NOS)

- Resource Manager.
- System software responsible for managing resources such as CPU time, primary & secondary memories, I/O devices, files, DBs, process & thread management, scheduling, etc.

I.4.2 File Server

A file server is a software product that is responsible for managing files and serving files and directories/folders to different clients in a networking environment (FTP, Netware). An OS usually includes a file server.

I.4.3 Print Server

A print server is a software product that is responsible for managing and scheduling printing jobs.

I.4.4 DB Server

- DB Manager.
- Software product responsible for data management: data storage, organization, structure, retrieval, manipulation, control, sorting, optimization, etc.

I.4.5 Application Server

An application server is a software product that acts as a middleware between the client and the server in a distributed C/S architecture. The application server is responsible for holding business logic.

I.4.6 Transaction / Transactional Server

A transaction server (also called transactional server) is a software product that is responsible for managing business transactions in a distributed C/S environment. A transaction server may be used to maintain high-performance availability and integrity of data. It's best suited to secure high-speed applications.

I.4.7 HTTP Web Server

An HTTP Web server is a software product that is responsible for hosting Web sites. It communicates with Web browsers through HTTP or SHTTP/HTTPS protocols.

I.4.8 E-Mail Server

An E-mail server is a software product that is responsible for hosting mails and distributing mails to appropriate mailboxes relative to specific accounts.

I.4.9 FTP Server

An FTP server is a software product that is responsible for sending and receiving files efficiently and securely between different clients and/or servers.

I.4.10 Forum / Newsgroup Server

A forum/newsgroup server is a software product that is responsible for managing news & forums.

I.4.11 Component / Object Server

A component/object server is a software product that is responsible for managing components (i.e., executable objects) and objects in a distributed object architecture like CORBA.

I.4.12 Groupware Server

1. A software product that supports the creation, flow, and tracking of non-structuring information in direct support of collaborative group activity.

2. A collection of technologies that allows us to represent complex processes that are centered on collaborative human activities.

I.4.13 Firewall Server

A firewall is a software product that protects an organization from external intruders and threats.

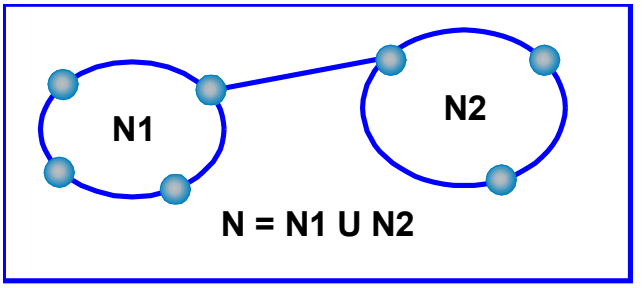
I.4.14 Proxy Server

A proxy server is a software product that is responsible for managing proxies. A proxy is an object that controls the access to other objects.

I.5 Network Types

I.5.1 internet

internet = inter | net = interconnected networks = network of networks



I.5.2 Internet

The Internet is the biggest public WAN network worldwide.

I.5.3 Intranet

An intranet is a private network within an organization.

I.5.4 Extranet

An extranet is a private network that relates two or more organizations together.

I.6 Expectations from C/S Computing & Technologies

- High level of reliability.
- Centralized resources.
- Better security.
- Administration at the server level.
- Evolutionary network.
- High cost.
- Available at the desk.
- Open system adoption.
- Integration of emerging technologies.
- Workstation power.
- Work group empowerment.
- Preservation of existing investments.
- Possible use of open OLTP.
- Gateways / other mainframes access software
- Etc.

I.7 Official Exams

2012 – S2 – Q3

2012 – S1 – Q1

2011 – S1 – Q2, [Q5]

2010 – S2 – Q1, Q3

2010 – S1 – Q3

2009 – S2 – [Q2]

2009 edu – S2 – Q2

2008 – S1 – Q1

2008 edu – S1 – [Q3]

2006 – S1 – Q1

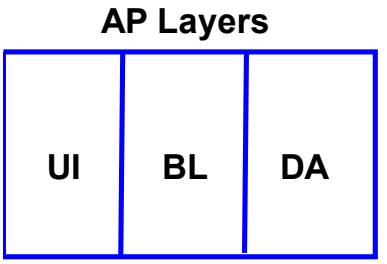
2005 – S2 – Q2

2005 – S1 – Q4

Chapter II: C/S System Architectures

II.1 Application Layers

The majority of application programs (APs) have three layers: the presentation (also called user interface) layer, the business logic (also called procedures) layer, and the data access layer.



II.1.1 User Interface (UI) / Presentation

The UI layer is responsible for user interface and presenting the received information/results to the end-users.

II.1.2 Business Logic (BL) / Procedures

The BL layer is responsible for holding all the procedures and business logic. The business logic is the set of all business rules in an organization.

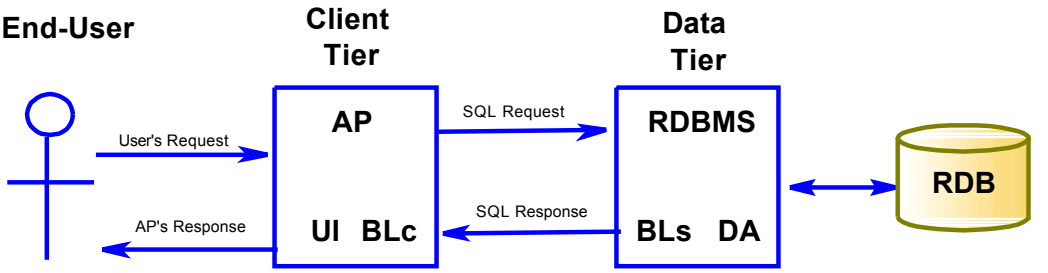
II.1.3 Data Access (DA)

The DA layer is responsible for accessing data usually stored in a database or in external files like text files, Excel files, or XML files.

II.2 2-Tier Architecture

2-Tier architecture is a kind of C/S system architectures that is composed of a client tier and a data / information tier. The client requests services from the server.

II.2.1 Diagram



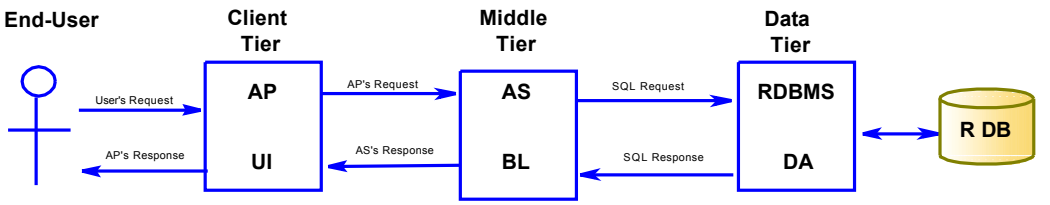
In 2-tier client/server system architecture, the business logic is split between the client (BLc) and the server (BLs). The BLc is the subset of the BL that is located at the client side, and the BLs is the subset of the BL that is located at the DB server side.

II.2.2 Limitations

- Not suitable for the Web.
- Not suitable for distributed applications.
- The majority of the load is on the server.
- The server is the weakest part.
- Not scalable.
- No support for server-server infrastructure.
- Great number of concurrent connections to DB servers.
- No access to mainframes.
- Non-transactional (conversational instead).
- Support for synchronous communication only ➔ no asynchronous communication.
- Homogeneous data management (one DB vendor), ➔ No support for heterogeneous DBs.
- Security risk because the authentication is performed at the DB server level only.

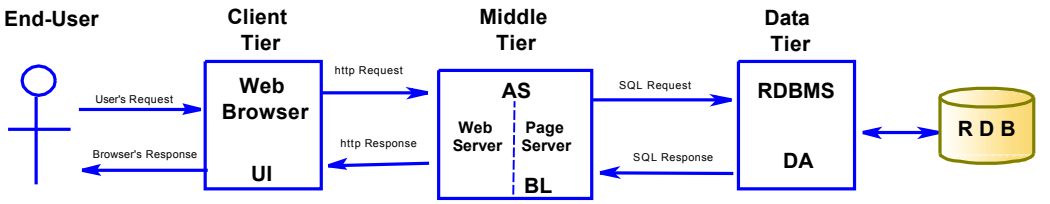
II.3 3-Tier Architecture

II.3.1 Diagram



In 3-tier client/server system architecture, the business logic is centralized in a specific middleware called application server.

II.3.2 3-Tier Web Architecture



The 3-tier Web architecture is a special case of 3-tier architecture such that the client is always a Web browser, and the application server is composed of a Web server and a page server.

II.4 2-Tier vs. 3-Tier Architectures / Models / Paradigms

2-Tier	*3-Tier*
Great number of concurrent connections	Multiplexing – less number of connections to DB

2-Tier	*3-Tier*
to the DB servers	servers
No access to mainframes	Access to mainframes using gateways and open OLTP
Non-transactional (conversational instead)	Transaction-oriented system. High throughput
Support for synchronous communication only	Support for both synchronous and asynchronous communications
Homogeneous data management	Heterogeneous data management
Suitable for small and medium applications	Suitable for large, mission-critical enterprise systems
One programming language	Multiple programming languages
Low cost	High cost
Easy to manage	Complexity in management
Good performance	Low performance
Less secure	More secure
No support for Internet & Web	Support for Internet & Web
Less technical skills	More technical skills
Client-Server infrastructure	Server-Server infrastructure
No support for distributed C/S applications	Support for distributed C/S applications
Peer-to-Peer communication style	Peer-to-Peer, MOM, and RPC communication styles

II.5 n-Tier / Multi-Tier / Components-Based Architecture

- $n \geq 3$.
- The business logic is centralized on the middle tier ($n = 3$), or distributed on the internal tiers ($n > 3$) between all the component-based servers.
- Flexibility on scalability.
- Modularity on maintaining components.
- Great use of ORB services (CORBA).
- Reusability.
- Communication between components (in addition to the communicating processes).
- Support of heterogeneous and multi-platforms environment.
- Internet & Web services may be provided.
- Suitable for large, mission-critical systems.
- Support for synchronous and asynchronous communications.
- Reliability.
- High performance.
- Improved security.

II.6 Case Studies

II.6.1 2-Tier Architecture

II.6.2 3-Tier Architecture

II.6.3 Mixed Architectures

II.7 Official Exams

2012 – S2 – Q2

2011 – S2 – Q6

2012 – S1 – Q2, Q3

2011 – S1 – Q2

2010 – S1 – Q1, Q4

2009 – S2 – Q1

2009 – S1 – Q1

2009 edu – S2 – Q3, [Q4]

2008 – S1 – Q1, Q4

2008 edu – S1 – Q4

2007 – S2 – Q1, Q4

2007 – S1 – Q1, Q3, Q4

2006 – S2 – Q4

2006 – S1 – Q4

2005 – S2 – Q1, Q4

2005 – S1 – Q4, [Q3]

Chapter III: NOS Middleware

III.1 Definitions

1. NOS middleware is a software that assures the interface between client software and server software. In a typical client/server architecture, NOS middleware is located on the middle tier while the client tier is dedicated to the client (also called front-end), and the data/information tier is dedicated to the server (also called back-end).

2. NOS middleware is a set of software services constructed on the top of a transport protocol in order to allow the exchange of clients' requests and servers' responses transparently.

III.2 Functions

- Procedures of establishment and closing of connections.
- Execution of requests and reception of results.
- Process initialization at different sites.
- Global directory of services.
- Remote data access.
- Security & integrity (authentication, encryption, etc.)
- Monitoring (computers, printers, etc.)
- Process termination.
- Requests & results caching.
- Provide a high level API for applications.
- Mask the heterogeneity of hardware & software systems.
- Make the distribution of applications transparent (portability of applications).
- Interoperability of heterogeneous applications.
- Provide distributed services (distributed time services, distributed security services, etc.).

III.3 Transparency Meaning

1. Transparency means to let everyone thinking that there is no C/S system.
2. Transparency means hiding the network and its servers from the users, and even from the application programmers.

III.4 Transparency Types

- **Physical Localization Transparency**
- **Namespace Transparency**
- **Logon Transparency**
- **Distance Access Transparency**

- **Distributed Time Transparency**
- **Failure Transparency**
- **Administration Transparency**

III.5 NOS Communication Styles

III.5.1 Peer-to-Peer (P2P)

- All NOSs offer peer-to-peer interfaces that let applications communicate using close-to-the-wire send/receive semantics.
- The term "peer-to-peer" indicates that the two sides of a communication link use the same protocol interface to perform a network conversation.

III.5.2 Remote Procedure Call (RPC)

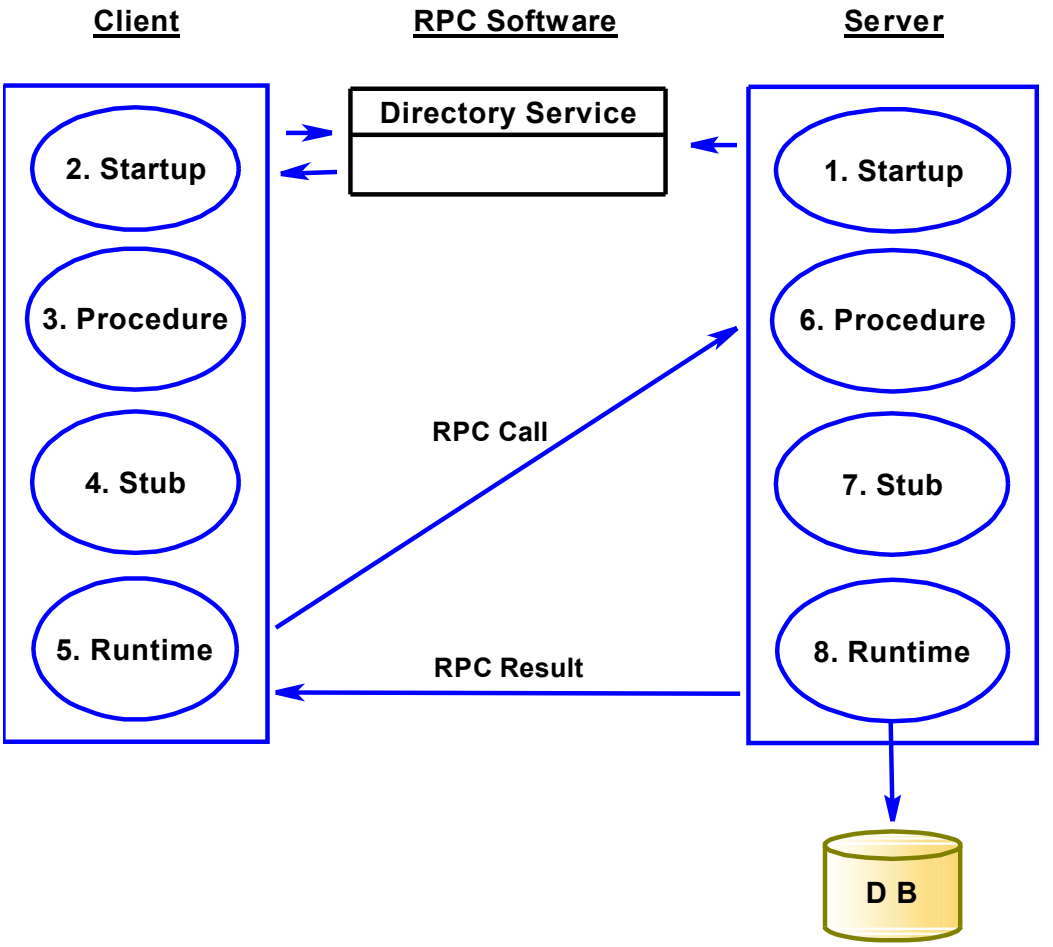
- The RPC hides the complexity of the network using the classical mechanism of calling procedures.
- A client process (or thread) calls a procedure that is located on a remote server and hangs its proper functioning until the results from the remote procedure are received.
- The procedure's arguments (also called parameters) are passed / transmitted as we do in an ordinary procedure.
- The RPC software gathers the parameters' values at execution time, constitutes a message, and sends it to a distance server.
- The distant server receives the request, disassembles the arguments, calls the appropriate procedure, and finally returns the response to the client process (or thread) through the RPC software (telephone-like metaphor).

III.5.2.1 Processing a Typical RPC

A typical RPC is processed as follows:

1. The client creates a message (RPC call).
2. The client encodes the procedure's arguments.
3. The client sends the RPC message to the remote server.
4. The server accepts a demand of RPC execution.
5. The server decodes the procedure's arguments according to the representation of the local machine.
6. The server dispatches the message to the adequate procedure.
7. The server builds the response returned by the procedure and then encodes it.
8. The server sends the corresponding message/result back to the client.
9. The client waits for the result from the remote server and then decodes this result according to the internal representation of the local machine.

III.5.2.2 RPC Protocol Model



1. [Server] Services advertisement and publishing in the directory service.
2. [Client] Finding an appropriate server that provides the requested service (as RPC).
3. [Client] Service selection from the directory service.
4. [Client] Packing procedure's arguments.
5. [Client] Making the RPC call.
6. [Server] Receiving and accepting the RPC call.
7. [Server] Unpacking procedure's arguments.
8. [Server] Executing the procedure that may access a DB and finally sending the result back to the client.

III.5.2.3 RPC Message Format

Any message sent by the RPC protocol has the following format:

- Message ID
- Message type (CALL or REPLY)
- RPC version number (RPC library version)
- Remote program
- Remote program version
- Remote procedure
- Authentication fields
- Procedure's arguments (CALL) or results (REPLY)

The format has a variable length since the number of the arguments of the called procedure is variable.

III.5.3 Message-Oriented Middleware (MOM)

- Every DAD (Distributed Application Development) needs a MOM.
- MOM provides the easiest way to create inter-enterprise C/S systems.
- MOM allows the exchange of messages in a C/S system by means of a queuing system.
- The applications communicate with each other over the network by sending and receiving messages in queue systems.
- MOM's messaging and queuing allows clients and servers to communicate across a network without being linked by a private, dedicated, and logical connection.
- The clients and servers can run at different times (mail post-like metaphor).
- A message-queue based system can be used to develop applications that should support asynchronous processing. In this environment, the messages are sent from the client to the system, the system will put the received messages in a queue, and later, the system may filter the messages and process a part of them or the overall messages.
- The queue may be implemented as FIFO or based on a priority policy.
- MOM greatly uses both types of queues (FIFO or priority-based).
- MOM can also use specific triggers to process certain types of messages that need particular care.
- MOM can be a coordinator of transactions in an Internet online environment.

III.5.4 MOM vs. RPC

Feature	***MOM***	***RPC***
Metaphor	Post office-like	Telephone-like
Style	Queued	Call-Return
Message filtering	Yes	No
Asynchronous Processing	Yes. Queues and triggers required	Limited. May be implemented using threads
Partner need to be available	No	Yes
Sequencing order	No preferred order	The server must be up first
Transaction / transactional support	Yes for some products using queues	No. Requires transactional RPC
Communication	Post a message then wait	Listen-Connect-Accept Request-Response
Performance	Low	Fast
Scalability	Hard	Easy
Administration & monitoring		Commercial products are available
Load balancing / equilibrium load	Requires a monitor to manage more than one single queue. Queues may be implemented as FIFO or priority-based policy	Can be accomplished with performance monitors and application management systems

Feature	*MOM*	*RPC*
Temporal relationship between the client and the server	Asynchronous. The client and server can be up or down independently of each other	Synchronous. The server must be running/up before the client can send a request

III.6 Client & Server Expectations from an OS

III.6.1 Client Expectations

A client may access from an OS resources that are local to this client such as file access, directory access and move, program execution, settings change, etc.

III.6.2 Server Expectations

A server may ask from the OS resources such as memory access, file access, program execution, going to a folder, etc. The expected services can be classified into basic services and extended services.

Basic Services

Basic services are part of standard OS such as:

- Efficient memory management
- Virtual memory management
- Process management
- Inter-process communication
- Thread management
- Priority of tasks
- Input/Output management
- Multi-user file system with high performance
- Security Management
- Pre-emption
- Semaphores
- Inter-task protection
- Extensions to DLLs

Extended Services

Extended services are supplementary software programs and components added to the OS such as:

- System administration
- System optimization & high performance / fine tuning
- DB and transaction services
- Internet services
- Object-oriented services
- Network clock
- Network extensions of the OS
- Binary large objects support
- Total directories and yellow pages of the network
- Ubiquity (existing at the same time) of the communication
- Etc.

III.6.3 Executing Clients & Servers on Different C/S Systems

There are many reasons such that clients and servers may execute on different C/S systems:

- Existence of many technologies like Java, JavaBeans, Enterprise JavaBeans.
- Existence of OSs that manage objects (like ActiveX, CORBA objects) on the network.
- Existence of different network protocols on the Internet. These protocols manage objects using, for example, remote method invocation or IIOP.
- Existence of technologies that encapsulate data.
- Existence of many middleware software products that facilitate the communication between C/S components.
- Existence of a variety of communication styles like MOM, RPC, etc.

III.7 Official Exams

2012 – S2 – Q4, Q5, Q6

2012 – S1 – Q4, Q5

2011 – S2 – Q2, Q5

2011 – S1 – [Q3, Q4]

2010 – S2 – Q4

2009 – S2 – Q3, [Q4]

2009 edu – S2 – [Q1]

2009 – S1 – [Q2, Q3]

2008 edu – S1 – Q2

2007 – S2 – Q2, [Q3]

2006 – S2 – Q1

2006 – S1 – Q2, Q3

2005 – S2 – Q3

2005 – S1 – Q2

Chapter IV: DB Servers

IV.1 DB Server Architecture Types

In client/server computing, there are two types of DB server architectures: a centralized architecture and a distributed architecture.

IV.1.1 Centralized DB Server

A centralized DB server (also called a DBMS) is a software system that is responsible for managing a centralized DB. A centralized DB is a physical DB that was mapped from one logical DB.

IV.1.2 Distributed DB Server

A distributed DB server (also called a DDBMS) is a software system that is responsible for managing a distributed DB. A distributed DB is a collection of two or more physical DBs that were mapped from one logical DB. Each physical DB is managed by a DBMS and allocated to a specific site, and all the DBMSs are linked to each other via a communication network.

IV.2 Transaction Management

- The idea of distributed systems without transaction management is like a society without contract law.
- Under the control of a TP monitor, a transaction can be managed from its point of origin – typically on the client – across one or more servers, and then back to the originating client.
- The transaction becomes the contract that binds the client to one or more servers.
- In an ideal world, all C/S programs are written as transactions.

IV.2.1 Transaction Definition

A DB transaction is a logical unit of works with ACID properties.

IV.2.2 ACID Properties

Atomicity (A)

The transaction is an indivisible unit of work with "all or nothing" contract. That means all the SQL statements in the transaction are either executed successfully (i.e., committed) or aborted (i.e., rolled back).

Consistency (C)

The transaction will transform the DB from a consistent state to another consistent state. In other terms, the DB will remain consistent (i.e., correct, valid, integrated) after the execution of the transaction.

Isolation (I)

If two or more transactions are running concurrently (i.e., in parallel, at the same time), each transaction has the "feeling" that is running alone, and, therefore, it's isolated from the other transactions.

Durability (D)

Durability is also called persistence or permanency. Once a transaction has committed its work, all the changes made by this transaction in the DB are durable, and, thus, they cannot be undone.

IV.2.3 Transaction Structure

A DB transaction starts with a begin_transaction and ends with either a commit_transaction or a rollback_transaction (also called abort_transaction). The commit has as effect that all the changes made by this transaction in the DB are durable, while the effect of the rollback is that all the changes made by this transaction in the DB are undone (i.e., cancelled, aborted), and, consequently, the DB will return to its initial state before the execution of the DB transaction.

IV.3 Transaction Models

IV.3.1 Flat Transactions

A transaction is called flat because all the work done within a transaction's boundaries is at the same level.

IV.3.1.1 Centralized Flat Transactions

- The whole centralized flat transaction is managed by one and only one DBMS.
- Suitable for centralized DBMS.

IV.3.1.2 Distributed Flat Transactions

- A distributed flat transaction is a global transaction that is decomposed into multiple sub-transactions.
- Suitable for distributed DBMSs
- The global transaction is controlled by a transaction coordinator, and each sub-transaction is managed by the local DBMS responsible for managing data located at this site.
- The synchronization of updates at different sites is done using the 2PC (two-Phase Commit) recovery protocol.

2PC Recovery Protocol

The 2PC recovery protocol is a special protocol that can be used to manage distributed DB transactions. It's formed of two phases:

- Phase 1: Voting phase done by all the transaction participants.
- Phase 2: Decision phase made by the transaction coordinator.

IV.3.1.3 Limitations of Flat Transactions

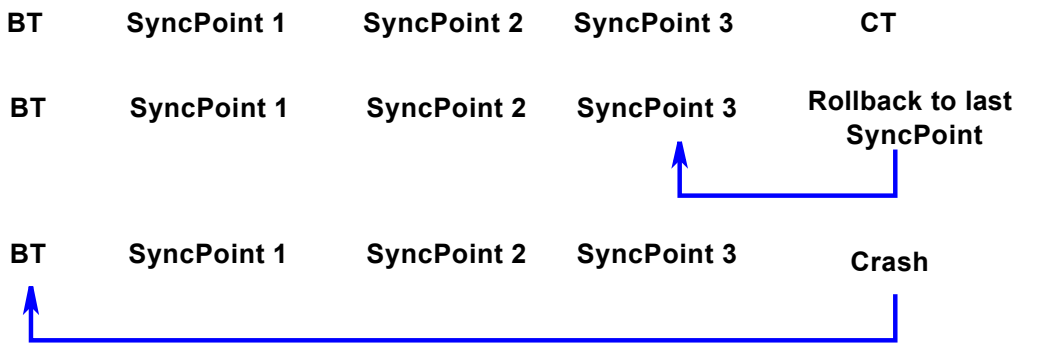
- We cannot rollback/abort a part of the transaction.
- Flat transactions are not suitable for business transactions with humans in the loop (classical GUI C/S transactions).
- Flat transactions are not suitable for business transactions that span long periods of time, typically computer-aided design (CAD) transactions.
- Flat transactions are not suitable for business transactions with a lot of bulk to deal with huge amount of data.

- Flat transactions are not suitable for business transactions that span across companies or the Internet.

IV.3.2 Chained Transactions

IV.3.2.1 SyncPoints / SavePoints

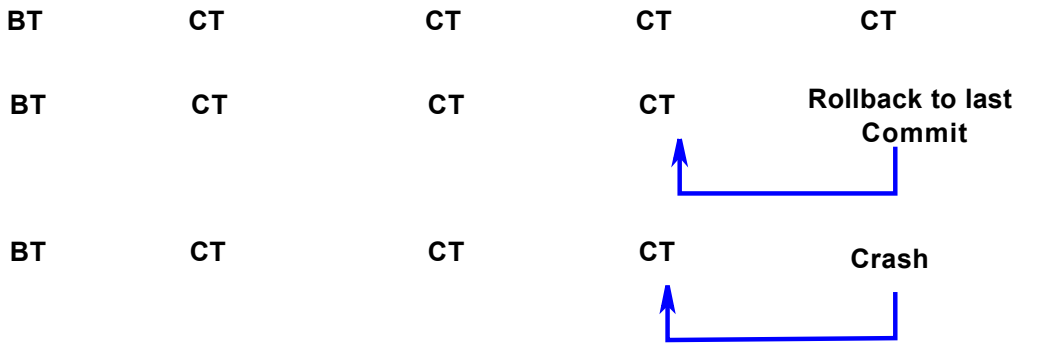
- The simplest form of chaining is to use SyncPoints – also known as SavePoints – within a flat transaction that allows periodic saves of accumulated work.
- The SyncPoint lets us rollback work and still maintain a live transaction. In contrast, a commit ends a transaction.
- A transaction can be divided into a series of activities that can be rolled back individually.
- SyncPoints are not durable.
- If the system crashes during a transaction, all the data accumulated in SyncPoints are lost.



SyncPoints are not durable

IV.3.2.2 Chained Transactions

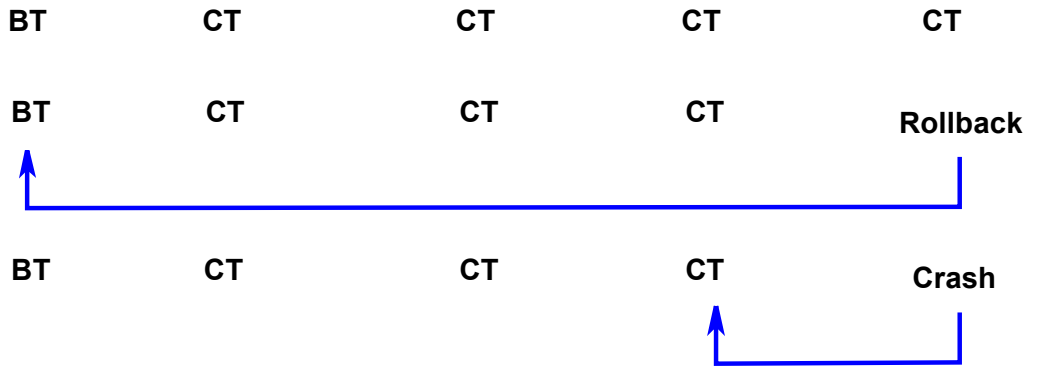
- Chained transactions are a variation of SyncPoints that make the accumulated work durable.
- They allow us to commit the work while staying in the transaction.
- But what we lose is the ability to roll back an entire chain of work.
- In chained transactions, commits are durable.



Chained Transactions: Commits are durable

IV.3.2.3 Sagas

- Sagas extend the chained transactions to let us roll back the entire chain.
- In sagas, commits are durable but can be rolled back.

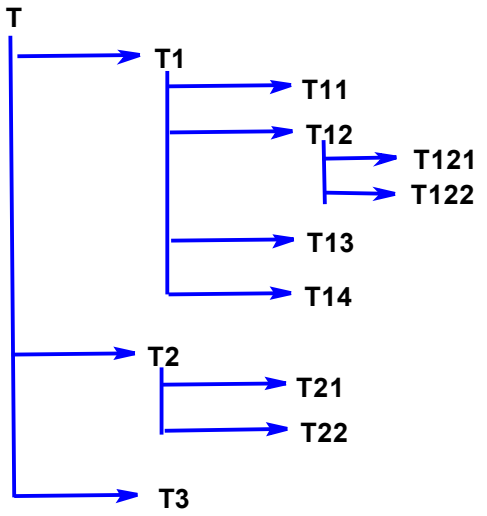


Sagas: Commits are durable but can be rolled back

IV.3.3 Nested Transactions

- One main/parent transaction and many dependent/child sub-transactions.
- Nested transactions provide the ability to define transactions within other transactions. They do that by breaking a transaction into hierarchies of sub-transactions. The main transaction starts the sub-transactions that behave as dependent transactions. A sub-transaction can also start its own sub-transactions, thus making the entire structure very recursive.
- Each sub-transaction can issue a commit or a rollback for its designated pieces of work. When a sub-transaction commits, its result is accessible only to the parent that spawned it. A sub-transaction's commit becomes permanent after it issues a local commit and all its ancestors commit. If a parent transaction makes a rollback, all its descendent transactions are rolled back, regardless of whether they issued a local commit.

Hierachy of Parent/Child Recursive Relationship between Transactions



IV.4 Official Exams

2011 – S2 – Q3

2011 – S1 – [Q2]

2010 – S2 – Q2

2009 – S2 – Q5

2009 edu – S2 – [Q5]

2007 – S1 – [Q2]

2006 – S2 – [Q2]

2005 – S1 – Q1

Chapter V: TP / Transactional Monitors

V.1 Definition

A TP (also called transactional) monitor is a specific kind of operating systems for transaction management and processing.

V.2 TP Monitor Functions / Tasks

- **Process Management**
- **Transaction Management**
- **C/S Communication Management**

A TP monitor can be used to design and develop transactional systems with high performance and huge amount of data.

The transactions managed by a TP monitor can be very complex and may take an important processing time to finish.

A TP monitor also provides a framework for running middle-tier server applications and components.

V.3 Advantages / Benefits

- **C/S Application Development Framework**
- **Firewalls of Protection**
- **High Availability**
- **High Performance**
- **Data Integrity**
- **Data Security / Auditing**
- **Load Balancing**
- **MOM Integration**
- **Scalability of Function**
- **Reduced System Cost**

V.4 TP Monitor Types

V.4.1 TP-Lite

TP-Lite is simply the integration of TP monitor functions in the DB engines.

V.4.2 TP-Heavy

TP-Heavy is a TP monitor as defined in the beginning of this chapter. TP-Heavy includes process management, load balancing, global transaction synchronization, interfaces to multiple resource managers, and error recovery.

V.4.3 TP-Lite Vs. TP-Heavy

Feature	*TP-Lite*	*TP-Heavy*
Problem nature	Can solve simple problems only	Can solve both simple and complex problems
DB transaction	Local transaction	Global transaction
Resource management	Can manage homogeneous resources	Can manage both homogeneous and heterogeneous resources
Process management	Processes allocated to stored procedures	Processes are pre-started and managed as server classes
RPC	Proprietary RPC	Non-proprietary RPC
C/S invocation	No support for alternative communications like conversations, queues, or publish-and-subscribe.	Very open to different communication styles like conversations, queues, and publish-and-subscribe

V.5 TP Monitor Editors & Products

- **BEA Technologies's Tuxedo**
- **IBM's TX Series**
- **WEB's Digital TP**

V.6 Official Exams

2012 – S1 – Q6

2010 – S2 – Q5

2009 edu – S2 – [Q5]

2007 – S1 – [Q2]

2006 – S2 – [Q2] **2005 – S1 – Q1**

Chapter VI: C/S Groupware

VI.1 Definitions

1. Groupware is a software package that supports the creation, flow, monitoring, and tracking of non-structuring information in direct support of collaborative group activity.

2. Groupware is a collection of technologies that allows us to represent complex processes that center on collaborative human activities.

VI.2 Synonyms for Groupware

- **Collaborative Computing**
- **Workgroup Computing**
- **Knowledge Management**
- **Computer-Supported Cooperative Working**

VI.3 Groupware Components / Technologies

VI.3.1 Multimedia Document Management

The groupware document management paradigm is a generalization of the electronic imaging file cabinet. The basic unit of storage in groupware is the document. Multimedia documents can handle multiple data types including text, images, graphics, voice clips, and video.

VI.3.2 Workflow

Workflow is the "up and coming" client/server technology that can be used to automatically route events and work from one program to the next in structured or unstructured client/server environments (Workflow River Model).

VI.3.3 E-Mail

E-Mail component is a groupware technology that allows the partners to send and receive messages and documents of different types asynchronously. That means you can use e-mail to send something to others without making a real-time connection.

VI.3.4 Group Scheduling & Calendaring

There's extensive innovation behind the electronic scheduling of meetings, sharing calendars and "to do" lists, and all that good stuff.

VI.3.5 Group Conferencing / E-Meetings

Conferencing or "electronic meetings" is another native groupware technology. Millions of PC users are now discovering the wonders of conferencing through electronic bulletin boards. We can divide client/server conferencing technology into two types: real-time and anytime. Real-time conferences allow groups to interactively

collaborate together (like video-conferencing technology). Anytime conferences allow people to participate in group discussions when and where they want.

VI.4 Groupware Editors & Products

- **IBM's Lotus Notes / Domino**
- **Novell's GroupWise**
- **Netscape's Suitespot**
- **Microsoft's Exchange**
- **Microsoft's Sharepoint**

VI.5 Advantages / Benefits

- Groupware can transform a company by challenging the way people communicate with each other.
- Groupware can positively change the business process of the organizations.
- Groupware allows direct contributors to collaborate on a job using Client/Server networks.
- Groupware is useful and helpful to manage and track the product through its various phases.
- Groupware allows departments to develop and deploy their own groupware applications.
- Most groupware products support open APIs that allow third parties to add new functions on the top of the foundation.

VI.6 Official Exams

2012 – S2 – Q1

2011 – S2 – Q4

2010 – S1 – Q2

2008 – S1 – [Q2]

2008 edu – S1 – [Q1]

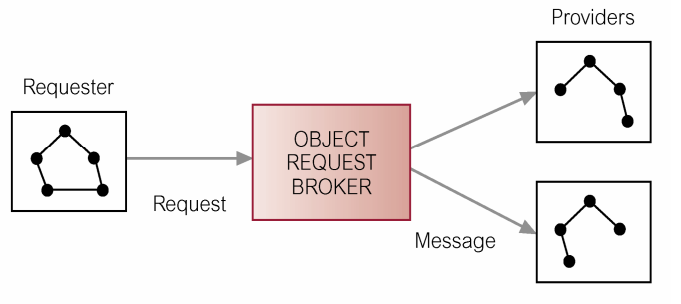
2006 – S2 – Q3

Chapter VII: CORBA

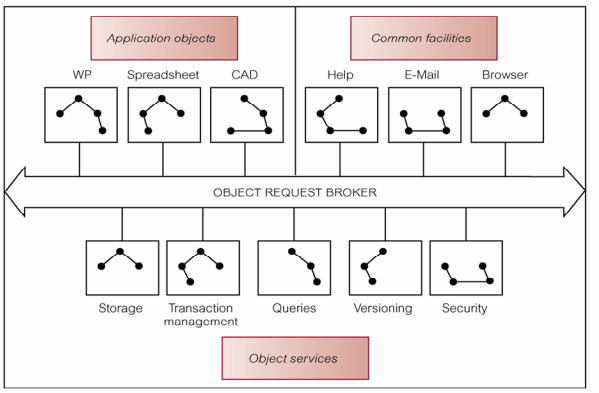
VII.1 Definition

CORBA is a platform- and language-independent framework for building distributed systems.

VII.2 Object Model



VII.3 Object Reference Model



VII.4 Fundamental Concepts of CORBA

- CORBA is an industry standard developed by the OMG to aid in distributed object programming.
- CORBA allows programs written in various languages with varying implementations running in separate locations to communicate with each other as easily as if they were in the same process address space.
- CORBA is simply a specification.
- CORBA describes the architecture made of cooperative services.
- CORBA objects can run on any platform, located anywhere on the network, and can be written in any language that has IDL mappings.
- CORBA defines the architecture of ORB-based environments.
- CORBA provides static and dynamic mechanism for clients to issue requests to objects.
- CORBA is an example of the DOM.
- Transparency is a crucial goal of CORBA.

VII.5 CORBA Main Elements

VII.5.1 Client

Client is the requestor and consumer of services from the server.

VII.5.2 Stub

- Stub is a proxy. A proxy is an object that controls the access to other objects.
- In a distributed computing environment, a stub stands for a client side object participating in distributed object communication.
- The stub acts as a gateway for client side objects and all outgoing requests to server side objects that are routed through it.

VII.5.3 Object Request Broker (ORB)

ORB is the central mechanism of CORBA. It's a communication bus of distributed systems.

VII.5.4 Skeleton

- Skeleton is also a proxy.
- In a distributed computing environment a skeleton stands for a server side object participating in distributed object communication.
- A skeleton acts as gateway for server side objects and all incoming clients requests are routed through it.

VII.5.5 Server

Server is the service provider that responds to clients' requests.

VII.5.6 Interface Definition Language (IDL)

IDL is the language that can be used to define CORBA interfaces.

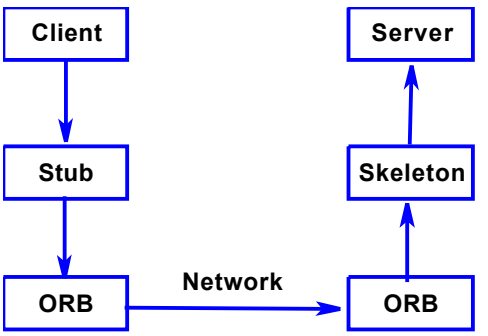
VII.5.7 Internet Inter-ORB Protocol (IIOP)

IIOP is a protocol that makes it possible for distributed programs written in different programming languages to communicate over the Internet.

VII.5.8 Interoperable Object Reference (IOR)

An IOR is the "contact details" that a client application uses to communicate with a CORBA object.

VII.6 Call Path from a Client to a Distributed Object



VII.7 CORBA Repositories

VII.7.1 Interfaces Repository

- This repository contains all information about CORBA interfaces.
- It provides type information about CORBA interfaces and other entities defined in IDL.
- Interface repository is a runtime component in the ORB architecture that can be used to obtain information on IDL types, e.g., object interfaces.
- The purpose of interface repository is to maintain information about IDL files.

VII.7.2 Implementations Repository

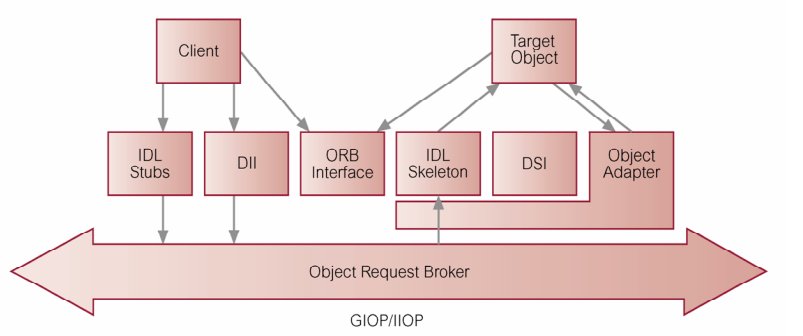
- This repository contains all information about CORBA implementations.
- This repository contains information that allows the ORB to locate and activate implementation of objects.
- Implementation repository is a database / file that stores information about CORBA server applications
- Functions of the implementation repository:
  - Registering the server with the implementation repository.
  - Manually running a server.
  - Maintaining a registry of known servers.
  - Recording which server is currently running and which port and host its uses.

VII.8 CORBA Services

- **Naming Service**
- **Event Management Service**
- **Life Cycle Service**
- **Persistent State Service**
- **Object Transaction Service**
- **Concurrency Service**
- **Relationship Service**
- **Externalization Service**
- **Query Service**
- **Licensing Service**
- **Property Service**

- **Time Service**
- **Security Service**
- **Notification Service**
- **Trader Service**
- **Collections Service**

VII.9 CORBA ORB Architecture



VII.10 Official Exams

2011 – S1 – Q1

2009 – S1 – Q4

Chapter VIII: Overview of Networking

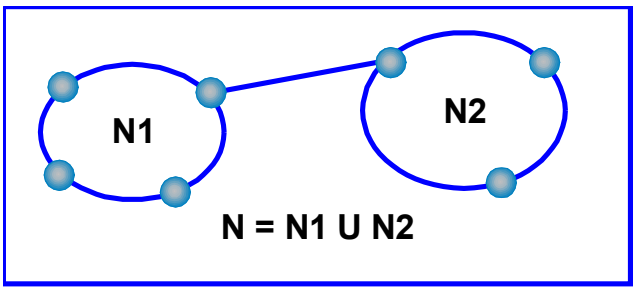
VIII.1 Network Definition

An interconnected collection of autonomous electronic devices (such as computers, mobile phones, printers, routers, switches, hubs, iPads, iPods, etc.) that are capable of exchanging information.

VIII.2 Network Types

VIII.2.1 internet

internet = inter | net = interconnected networks = network of networks



VIII.2.2 Internet

The Internet is the biggest public WAN network worldwide.

VIII.2.3 Intranet

An intranet is a private network within an organization.

VIII.2.4 Extranet

An extranet is a private network that relates two or more organizations together.

VIII.3 Network Categories

VIII.3.1 LAN

VIII.3.1.1 SAN

VIII.3.2 WAN

VIII.3.2.1 MAN

VIII.3.3 WAN vs. LAN

Feature	*WAN*	*LAN*
Distance	Distances up to thousands of kms	Distances up to a few kms
Link	Link autonomous computers	Link computers that cooperate in distributed applications
Network	Network managed by independent	Network managed by users (using

Feature	*WAN*	*LAN*
	organization (using telephone or satellite links)	privately owned cables)
Data Rate	Data rate up to 33.6 kbps (dial-up via modem), 45 Mbps (T3 circuit)	Data rate up to 2,500 Mbps (ATM), 10 Gbps is under development
Protocol	Complex	Simpler
Routing	Use point-to-point routing	Use broadcast routing
Topology	Use irregular topology	Use bus or ring topology
Error Rate	About $1 : 10^5$	About $1 : 10^9$

VIII.4 Network Topologies

Bus

Ring

Star

VIII.5 Network Protocols

A network protocol is a set of rules that determines how messages between electronic devices (mainly computers) are sent, interpreted, and processed.

TCP/IP

TCP/IP stands for Transmission Control Protocol / Internet Protocol.

SPX/IPX

SPX/IPX stands for Sequenced Packet eXchange / Internetwork Package eXchange.

NetBIOS

NetBIOS stands for Network Basic Input/Output System.

APPC

APPC stands for Advanced Program-to-Program Communications.

DECnet

DECnet stands for Digital Equipments Corporation network.

AppleTalk

WAP

WAP stands for Wireless Application Protocol.

VIII.6 Communication Time (CT)

CT = C0 + nb_of_bits_in_message / transmission_rate
C0: fixed cost of initiating a message, known as the access delay.

Example: Access Delay C0 = 1 s
Transmission Rate = 10,000 bps

- To send/transfer 100,000 records at the same time, 100 bits per record, we need: CT = 1 + (100000 * 100) / 10000 = 1,001 s

- To send/transfer 100,000 records, 100 bits per record, one record at a time, we need: CT = 100000 * [1 + (100 / 10000)] = 101,000 s
  Reason: Access delay!