DS Final Review: (Fighting!!!!)

Chapter1 Introduction

1. Overview:

- What is a distributed system?

  "A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing message."

  "A collection of independent computers that appears to its users as a single coherent system."

  Key aspect: a number of components/ communication between the components/ synergy: achieve more than the simple sum of individual components.

- Why distributed system?

  Resource Sharing (main motivation): HW Resource/SW Resource/Other(Processing power, BW)

  Benefits of resource sharing: Economy/Reliability/Availability/Scalability

  Detailed: 1) Resource can be add incrementally. 2) They make it easier to integrate different applications running on different computers into a single system. 2) Redundant components reduce the impact of HW and SW failures on users. (Reliability) 4) If designed properly, they scale well with respect to size of the underlying network. (Scalability)

- Example of distributed systems?

  WWW/Web servers and web browsers

- Challenges?

  1) Heterogeneity: means the diversity of the DS in terms of HW/SW/Network/OS/Programming Languages/Implementation by different developers/platform etc.

     Solution:1) Using standard protocols. 2) Using agreed upon message formats and data types. 3) Adhering to an agreed upon APIs. 4) Using Middleware. 5) Portable code

     Remark:

     i) Middleware:is a software layer between distributed application and the operating system that 1) provides a programming abstraction 2) masks the heterogeneity of the underlying network/HW/SW/OS/programming languages. eg. RMI/RPC/DFS. It provides access transparency.

     ii) Mobile Code:is sent from one computer to the other to run at destination (e.g. Java applets):

     Problems: Code that is compiled to run in one OS does not run in another. (Diff HW/OS)

     Solutions: 1) Virtual Machine approach: provides a way of making code executable on any hardware --- compiler produces code that is interpreted by virtual machine. 2) Cross-platform: compilation and code portability is another way that compiles source code to multiple targets.

2) Openness:refers to the ability of extend the system in different way by adding HW/SW resources.
Solution: 1) Publishing key interfaces 2) Allowing a uniform communication over the published interfaces. 3) Ensuring all implementations adhere to the published standards.
Remark: If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future.

3) Security:
3 aspects of security:
i) Confidential: protection from unauthorized individuals.
ii) Integrity: protection from alteration and corruption.
iii) Availability: protection from the means of access.
Solution: Encryption/Authentication/Authorization
Type of security challenges have not yet been resolved completely:
i) Denial of service attacks
ii) Security against mobile code.
Remark: security includes understanding the impact of software bugs.

4) Scalability: A system is considered to be scalable if it can handle the growth of the number of users.
Challenge:
i) Cost of physical resources
ii) Controlling the performance loss (complexity of algorithm)
iii) Resource should not run out (IP address)
iv) Avoiding Performance bottlenecks (design problem→decentralized)

5) Failure Handling:
Solution:
i) Detecting: some types of failures can be detected (checksum)
ii) Masking: some failures that have been detected can be hidden or made less severe (timeout/message retransmission)
iii) Tolerating: impractical to try and handle, it's better to tolerate them (failure is reported back to user)
iv) Recovery: sometimes leads to corrupted data and software can be designed so that it can recover the original state after failure (roll back mechanism/log files)
v) Redundancy: can be made to tolerate failures using redundant components (multiple servers provide same services).

6) Concurrency: Multiple clients can access the same resource at the same time, in some cases for updates
Solution:
i) Make access sequential -→ slow down the system

ii) Semaphores（信号量）supported by OS is a well accepted mechanism.

7) Transparency: hiding the components of a DS from the user and application programmer.

   Types of transparency:

   i) Access: use identical operations to access local and remote resources.
   ii) Location: enables resources to be accessed without knowledge of their physical or network location.
   iii) Concurrency: enables several processes to operate concurrently using shared resources without interference between them.
   iv) Replication: enables multiple instances of resource to be used to increase reliability and performance without knowledge of the replicas by users.
   v) Failure: enables the concealment of faults, allowing users to complete their tasks despite the failure of HW/SW components.
   vi) Mobility: allows the movements of resources and clients within a system without affecting the operation of users.
   vii) Performance: allows system to be reconfigured to improve performance as loads very.
   viii) Scaling: allows the system and applications to expand in scale without change to the system structure or the application algorithm.

2. Computer Networks vs. Distributed Systems:

CN: is a collection of spatially separated, interconnected computers that exchange messages based on specific protocols. Computers are addressed by IP addresses.

DS: Multiple computers on the network working together as a system. The spatial separation of computers and communication aspects are hidden from users.

DS is based on CN.

3. Consequences/Characteristic of DS

1) Programs can execute concurrently. 2) There is no single global notion of time. 3) System components can fail independently. 4) The capacity of the system to handle shared resource can be increased by adding more resource to the network.

4. Computer Networks:

1) Internet: consists of a large number of interconnected collection of computer networks of different types

   Feature:

   1) Computers interacting by message passing using a common means of communication (Internet protocol)
   2) Many different services (applications) → WWW, email, file transfer

3) A number of intranets linked by backbones.
4) Internet Service Providers (ISP), that provides access to the services on the services on the Internet while providing local service such as email and web hosting.
5) A backbone network link with high transmission capacity.
6) Communication via Satellite, fiber optic cables and other high-BW circuits.

2) Intranets: is a portion of the Internet that is separated by organization.
Feature:
1) A boundary that can be configured to enforce local security policies.
2) Several local area connection (LANs) linked by backbone connections.
3) A connection to the Internet via a router allowing users within the Intranet to access services on the Internet.
4) Firewalls to protect an intranet by preventing unauthorized messages leaving or entering by filtering incoming and outgoing messages.

3) Wireless networks: allows the integration of small and portable computing devices (laptop), hand-held devices (mobile phones), wearable devices, devices embedded applications (washing machines) into ds.

Three popular paradigms: Mobile Computing/ Ubiquitous Computing/ Internet of Things

5. Web Pages:
Static web pages: allow data to be made available for retrieval.
Dynamic web pages: allow users to interact with resources by taking user input, executing programs and returning results.

Chapter2 Models:

1. Concept of Physical model, Architectural model and Fundamental model:
   - Physical model: underlying HW elements
   - Architectural model consider:
     i) Architectural elements – components of the system that interact with one another.
     ii) Architectural patterns – the way components are mapped to underlying system.
     iii) Associated middleware solutions – existing solutions to common problems

     E.g. client-server, peer-to-peer
   - Fundamental model: the non-function aspects of the DS. (interaction model/failure model/security model)
     reliability/security/performance
   - Difference between A & F model:
     1) Fundamental model is more about property.
     2) 2) Architectural model you can “point” it out, e.g. “this is process”, but fundamental model can’t, you can’t say, “this is reliability.”

2. Communicating Entities:
   1) From a system perspective:processes/threads/nodes(sensors)
   2) From a programming perspective:Object (a class)/Components/Objects and components are usually used within an organization, while web services are usually seen as providing public interfaces.

3. Interfaces (use to communicate): is an important part of designing objects, components and web services.
   - Programmers are only concerned with abstraction offered by the interface, they are not aware of the implementation details.
   - Programmers also need not know the programming language or underlying platform used to implement the service.
   - So long as the interface does not change (or that changes are backwards compatible), the service implementation can change transparently.

4. Communication paradigms
   From low-level to high-level:
   - Interprocess communication: relatively low-level of support for communication between processes in a distributed system. (e.g. shared memory, multicast communication, sockets)
   - Remote invocation: based on a two-way exchange between communicating entities in a distributed system and resulting in the calling of a remote `operation, procedure or method. (e.g. request-reply protocols, RMI, RPC)
   - Indirect communication:
     space uncoupling – sender do not need to know who they are sending to
     time uncoupling – senders and receivers do not need to exist at the same time.(e.g. group communication/publish-subscribe system/message queues/ tuple spaces/distributed shared memory)

5. Roles and responsibilities
   - Client: a process that initiates connections to some other process
   - Server: a process that can receive connections from other process
   - Peer, can be seen as taking both the role of client and server.

6. Placement
   - mapping services to multiple servers: a single service may not make use of a single processes and multiple processes may be distributed across multiple machines.
   - caching:storing data at places that are typically closer to client or whereby subsequent accesses to the same data will take less time.
   - mobile code: transferring the code to the location that is most efficient, (e.g. running a complex query on the same machine that stores the data, rather than pulling all data to the machine that initiated the query)
   - mobile agents: code and data together (used to install and maintain software on a users computer, the agent continues to check for updates in the background.)

7. Architectural Patterns
- Client-Server: Clients invoke services in servers and results are returned. Servers in turn can become clients to other services.
- Peer-to-Peer: Each process in the systems plays a similar role interacting cooperatively as peers (playing the roles of client and server simultaneously)

8. Compare C-S with P2P model:
- If all parts are P2P, it will have security problem. Because in C-S, servers is a computer you can trust, like Google, Amazon… (Some company you can trust), but peers are all independent, you need to trust people.
- In CS, you can attack Server and system will be crashed. But in P2P, you must attack them all.
- Big file sharing ---- P2P is better. If use C-S pattern, all the file need to transmit to server, then to the client. Use too much BW.

9. DS Architecture Variation:
   1) A service provided by multiple servers:
      - Objects maybe partitioned or replicated across servers.
      - The communication between S2S could be P2P. (Scalability)

   2) Proxy server and caches
      - Cache is a store of recently used objects that is closer to client.
      - New objects are added to the cache replacing existing objects.
      - When an object is requested, the caching service is checked to see if an up-to-date copy is available (fetched in not available).
      - Always works when web server and client are not in the same country.
      - Not have processing unit in proxy server, just a look-up table, not a real server.

   3) Mobile Code and Agents
      - Mobile Code is down loaded to the client and is executed on the client (e.g. applet) (只能从服务器刀客户端进行单方向的移动)

        Why need Applet Code?
        1) Otherwise will use a lot of server CPU.
        2) Servers don't need to know the execution. (E.g. Bank need calculate something on his own, because it uses some information the bank didn't want to use calculator on the server side)

      - Mobile agents are running programs that includes both code and data that travels from one computer.

   4) Network Computers and Thin clients
      - Network Computers: download their operating system and application software from a remote file system. Applications are run locally.
      - Thin Clients: application software is not download but runs on the computer server – e.g. UNIX. (Not suitable for highly interactive

graphical activities, like video, a lot of overhead). Running no process at all, just display the result.

10. Layering: Middleware/Platform.

11. Fundamental Models: allows distributed system to be analyzed in terms of fundamental properties regardless of the architecture. These models help understand how the non-functional requirements are supported.
    Interaction/Failure/Security (openness of DS make them prone to various type of attacks)

12. Interaction Model: model the interaction between processes of a ds.
    - 2 important aspects of interaction modeling (影响进程交互的两个重要方面):
      1) Performance of communication channel(信道性能):
         3 important performance characteristics of communication channels: Latency/Bandwidth/Jitter
      2) Computer clocks and timing events (计算机时钟和时序事件):
         - ✓ Each computer in a ds has its own internal clock.
         - ✓ The timestamps between two processes can vary : Initial time setting being different/difference in clock drift rates
    - Variations of Interaction Models:
      1) Synchronous system model (assumes known bounds on)
         - ✓ time to execute each step of a process
         - ✓ message transmission delay
         - ✓ local clock drift rate
      2) Asynchronous system model (assumes no bound on)
         - ✓ process execution speed
         - ✓ message transmission delays
         - ✓ clock drift rates

13. Failure Model:
    1) Omission failure:refers to cases where a process or a communication channel fails to perform what is expected to do. (该做的没做)
       i) Process omission failure:
          - Normally caused by a process crash.(may not detect)
          - Repeated failures during invocation are an indication.
          - Timeouts can be used to detect this type of crash
          - A crash is referred to as a fail-stop if other processes can detect certainly that the process crashed.

       ii) Communication channel failure:
          - Process p inserting the message m to its outgoing buffer (send)
          - The communication channel transporting m from p's outgoing buffer to q's incoming buffer.
          - Process q taking the message from its incoming buffer (receive)

Send omission failure/Receive omission failure/Channel omission failure

2) Arbitrary failure (Byzantine failure):refers to any type of failure that can occur in a system. (A process begins to execute arbitrary steps)
    - Intendedsteps omitted in processing
    - Message contents corrupted
    - Non-existent messages delivered
    - Real messages delivered more than once.

3) Timing failure: these failures occur when time limits set on process execution time, message delivery time and clock rate drift. (relevant to synchronous system) (Sometime expect something occur in certain time) ---- Clock/Performace
   (指定事件间隔内对客户没有响应)

14. Reliability of one-to-one communication
    2 properties of Reliable communication: validity: any message in the outgoing buffer is eventually delivered to the incoming message buffer/integrity: the message is identical to the one sent, and no messages are delivered twice.

15. Security Model: Security of a distributed system is achieved by securing processes, communication channels and protecting objects they encapsulate against unauthorized access.
    Possible threats:
    - Threats to processes: Server and clients cannot be sure about the source of message. Source address can be spoofed.
    - Threats to communication channels: Enemy can copy, alter or inject messages.
    - Denial of service attack: overloading the server or otherwise trigger excessive delay to the service. (用大量无意义的请求攻击服务，使得重要的用户不能使用它，这称为拒绝服务攻击)
    - Mobile code: performs operations that corrupt the server or service in an arbitrary way.

    Addressing security threats:
    - Cryptography and shared secrets: encryption
    - Authentication: provide identities of users
    - Secure channel: A secure channel is a communication channel connecting a pair of processes on the behalf of its principles.

Chapter3 Interprocess Communication:

1. Introduction
   API for UDP:
- Provides a message passing abstraction
- Is the simplest form of Interprocess Communication (IPC)
- Transmits a single message (called a datagram) to the receiving process

API for TCP:

- Provides an abstraction for a two-way stream
- Streams do not have message boundaries
- Stream provide the basis for producer/consumer communication
- Data sent by the producer are queued until the consumer is ready to receive them
- The consumer must wait when no data is available

Data Representation:

Deals with how objects and data used in application programs are translated into a form suitable for sending as messages over the network

2. The API for Internet protocol
   - Processes use two message communication function: send & receive
   - Communication may be synchronous or asynchronous:
     Synchronous: Both send and receive operations are blocking operations.
     Asynchronous:send operation is non-blocking.receive operation can be blocking or non-blocking(not exist)

3. Message Destinations
   - A port usually has exactly one receiver (except multicast protocols) but can have multiple senders
   - Location transparency is provided by a name server, binder or OS

4. Socket: provides an end point for communication between processes
   Socket properties:
   - Same socket can be used both for sending and receiving messages
   - Processes can use multiple ports to receive messages
   - Ports cannot usually be shared between processes for receiving messages, but recent changes allow multiple processes to listen on the same port.
   - Any number of processes can send messages to the same port.
   - Each socket is associated with a single protocol (UDP or TCP)

5. UDP Datagram Communication (e.g. Domain Name Service-- DNS)
   - Both the sender and the receiver bind to sockets
   - Message Size:
     1) Receiving process defines an array of bytes to receive message
     2) If the message is too big, it gets truncated
     3) Protocol allows packet lengths of 2^16 bytes but the practical limit is 8 kilo bytes.
   - Blocking:
     1) Non-blocking sends and blocking receives are used for datagram communication
     2) Operation returns when the message is copied to the buffer
     3) Outstanding or future invocations of the receive on the socket can collect the messages

4) Messages are discarded if no socket is bound to the port

- Timeout
  1) Receive will wait indefinitely till messages are received
  2) Timeouts can be set on sockets to exit from infinite waits and check the condition of the sender
- Possible failures:
  1) Data Corruption: checksum can be used
  2) Omission failure: buffers full, corruption, dropping
  3) Order: messages might be delivered out of order

6. TCP Stream Communication (e.g. HTTP/FTP/SMTP)

- Features of stream abstraction
  1) Message sizes: no limit
  2) Lost messages: acknowledgment and retransmission
  3) Flow control: attend to match the speed of process
  4) Message duplication or ordering: message identifiers are associated with IP packets to enable recipient to detect and reject duplicates and reorder messages in case messages arrive out of order.
  5) Message destinations: establish a connection

- Steps involved in established a TCP stream socket

  Client:
  1) Create a socket specifying the server address and port
  2) Read and write data using the stream associated with socket

  Server:
  1) Create a listening socket bound to a server port.
  2) Wait for clients to request a connection (Listening socket maintains a queue of incoming connection requests)
  3) Server accepts a connection and creates a new stream socket for the server to communicate with client, retaining the original listening socket at the server port for listening incoming connections. A pair of sockets in client and server is connected by a pair of streams, one in each direction. A socket has an input and output stream.

  When an application closes a socket, the data in the output buffer is sent to the other end with an indication that the stream is broken.
- TCP connection issues:

  A pre-agreed format for the data sent over the socket/blocking is possible at both end/if the process support thread, it's better that a thread is created to each connection so that other clients will not be blocked.

- Failure Model:
  1) Use checksum to detect and reject corrupt packets
  2) Use sequence numbers to detect and reject duplicates
  3) Timeouts and retransmission is used to deal with lost packet
  4) Under severe congestion, TCP streams declare the connections to be broken
  5) When communication is broken, the processes cannot distinguish

between network failure or process crash
6) Communicating process can't definitely say whether the messages sent recently were received

7. External data representation and marshalling(Binary format & Text format)
   - Data structure in programs are flattened to a sequence of bytes before transmission
   - Diff computer has diff data representations (ASCII vs. Unicode)
   - 2 ways interpret data in different formats:
     1) Data is converted to an agreed external format before transmission and converted to the local form on receipt.
     2) Values transmitted in the sender format, with an indication of the format used.
   - Marshalling: Process of converting data to the form suitable for transmission
   - Unmarshalling: Process of disassembling the data at the receiver

8. CORBA's common data representation (Binary format)
9. Java object serialization(Binary format)
   - Serialization refers to the activity of flattening an object to be suitable for storage or transmission.
   - Deserialization refers to the activity of restoring the state of the object
   - During remote method invocation, the arguments and results are serialized and deserialized by middleware
10. Extensible markup language (XML) (Text format)
   - A markup language is a textual encoding representing data and the details of structure.
   - XML:
     1) defined by W3C
     2) tags describe the logical structure of the data
     3) extensible
     4) tags are generic
     5) tags describe the data
     6) tags together with namespaces allow the tags to be meaningful
     7) since data s textual, the messages are large causing longer processing and transmission times and more space to store.

     takes more space/easy to read

   - Compare text format and binary format:
     Advantages: readable/easy to debug
     Disadvantages: takes a lot of space for transferring, binary format are more competitive, saving more BW/security issue is more serious

11. JSON (text format)
   - less text/use a lot in web system, because javascript is kind of hot language/very convenient

12. Group communication

- A multicast operation allows group communication – sending a single message to number of processes identified as a group.
- Uses of multicast: fault tolerance based on replicated services/finding discovery servers/better performance through replicated data

13. IP multicast:

- Allows a sender to transmit a single packet to a set of computers that form the group.
- class D(1110-→224)
- The sender is not aware of the individual recipients.
- IP multicast API:
  1) available only for UDP
  2) an application can join a multicast group by making its socket join the group
- failure model: omission failures are possible. Messages may not get to one or more members due to a single omission.

Chapter4 Remote Invocation

1. Introduction:
   - Remote Procedure Call model –an extension of the conventional procedure call model
   - Remote Method Invocation model – an extension of the object-oriented programming model

2. The Request-Reply protocol: the most common exchange protocol for implementation of remote invocation in a distributed system.
   - 3 abstract operations: doOperation/getRequest/sendReply

3. Design Issues:
   - Failure model can consider: Handling timeouts/Discarding duplicate messages/Handling lost reply messages-strategy depends on whether server operations are idempotent/History
   - 3 main design decision: (RRP 协议通常采用一些措施来保证所要求得消息传递保障)
     1) retry request message (重发请求消息)
     2) filter duplicates (过滤重复请求)
     3) results retransmission (重传结果)

4. Exchange protocols:
   3 protocols that produce differing behaviors in the presence of communication failures are used for implementing various types of request behavior
   - the request (R) protocol
   - the request-reply (RR) protocol
   - the request-reply-acknowledge reply (RRA) protocol

5. Invocation Semantics (使用 RRP 协议)

Middleware provides a level of semantics:

1) Maybe invocation semantics
   The remote procedure call may be executed once or not at all.
   Problem:客户不能判断服务器端是否执行程序

2) At-least-once invocation semantics(客户端收到异常后会重复请求--RPC)
   Either the remote procedure was executed at least once, and the caller received a response, or the caller received an exception to indicate the remote procedure was not executed at all.
   Problem: arbitrary failure (如果远程操作不是 idempotent,会很麻烦), 远程对象的服务器 crash, 增加信息量(利用 timeout 解决)

3) At-most-once:(RMI, not idempotent method need this operation)
   The remote procedure call was either executed exactly once, in which case the caller received a response, or it was not executed at all and the caller receives an exception.
   Solution:在服务器 filter duplicates 或者 record results 可以保证方法不被重复执行。
   Problem: 加重服务器处理的负担

6. Transparency
   - Location and access transparency are goals for remote invocation
   - But in some case complete transparency is not desirable, due to
     1) Remote invocations being more prone to failure due to network and remote machines.
     2) Latency of remote invocations is significantly higher than that of local invocations.

   So, many implementations provide access transparency but not complete location transparency.

7. Client-server communication
   - Client-server communication normally uses the synchronous RR communication paradigm
   - TCP or UDP can be used, -- TCP involves additional overheads:
     1) redundant acknowledgements
     2) needs two additional messages for establishing connection
     3) flow control is not needed since the number of arguments and results are limited

8. HTTP: an example of a RR protocol

9. Remote Procedure Call: enable clients to execute procedures in server processes based on a defined service interface. (不同进程之间的调用，但是相同语言，环境)

   Important Concepts:
   - Communication Module: implements in terms of retransmission of

requests, dealing with duplicates and retransmission of results.

- Client Stub Procedure: behave like a local procedure to the client. Marshals the procedure identifier and arguments, which is handed to the communication module. Unmarshalls the results in the reply.

- Dispatcher:Select the server stub based on the procedure identifier and forwards the request to the server stub.

- Server stub procedure: Unmarshalls the arguments in the request message and forwards the request to the server stub.

Remark: stub 的主要功能是对要发送的参数进行 marshal (可理解成一种打包操作) 和对接收到的参数(或返回值)进行 unmarshal 的操作(解包过程)

10. Object-Oriented Concepts
    - Objects: consists of attributes and methods. Objects communicate with other objects by invoking methods, passing arguments and receiving results.
    - Object References: can be used to access objects, Object references can be assigned to variables, passed as arguments and returned as results.
    - Interfaces: define the methods that are available to external objects to invoke.
    - Actions:objects performing a particular task on a target object.
    - Exceptions: are thrown when an error occurs. The calling program catches the exception.
    - Garbage collection (有必要提供一种手段在不再需要对象时释放其占用的空间): is the process of releasing memory used by objects that are no longer in use. Can be automatic or explicitly done by program.

11. Distributed Object Concepts(进程中可以包含多个对象，本地 or 远程。调用其他进程的对象叫远程调用,即使同一台机器上，不同地址空间的调用也叫远程调用)
    - Remote Objects: An object that can receive remote invocations is called a remote object. A remote object can receive remote invocations and local invocations. Remote objects can invoke methods in local objects as well as other remote objects. (提供可被远程调用方法的对象叫远程对象)
    - Remote Object Reference: is a unique identifier that can be used throughout the distributed system for identifying an object. (远程对象引用可以作为远程方法调用的结果返回，例如对象 A 可能会从对象 B 得到一个对象 F 的远程引用)
    - Remote Interface: defines the methods that can be invoked by external processes, Remote object implement the remote interface. (remote object 实现接口后可被远程调用,一个远程对象可以实现对个远程接口)
    - Actions in ds: An action could be executing a remote method defined in the remote interface or creating a new object in the target process. Actions are invoked using Remote Method Invocation (RMI).
    - Exceptions (e.g. 因分布引起的超时异常)

- Garbage collection in ds

12. Implementation of RMI
- Communication Module: responsible for communicating messages between server and client.
  - ✓ Use 3 fields from the message: message type/request ID/ remote object reference
  - ✓ responsible for implementing invocation semantics (服务器端和客户端两个通信模块共同负责)
  - ✓ query the remote reference module to obtain the local reference of the object and passes the local reference to the dispatcher for the class
    (服务器端的通信模块先调用 remote reference module 获取 local object reference 替换收到消息里面的 remote object reference，然后为其调用 dispatcher)
- Remote reference module:responsible for creating remote object references and maintaining the remote object table, which is used for translating between local and remote object reference.
  - ✓ remote object table contains: remote object reference held by the process/Local proxy
  - ✓ Entries are added to the remote object table when: A remote object reference is passed for the first time/when a remote object reference is received and an entry is not present in the table
  - ✓ Servants are the objects in the process that receive the remote invocation.
- The RMI software：this is the software layer that lies between the application and the communication and object reference modules. (由应用层对象和通信模块、远程引用模块之间的软件层组成)
  - ✓ Proxy: Plays the role of a local object to the invoking object. There is a proxy for each remote object. (每个远程对象都有一个本地代理，让客户端能够明确调用远程对象的方法)
    Remark:在客户端里,每个远程对象都在代理完成 1)传递请求 2) 将要传输的参数或者对象进行编码解码
    1) Marshalling the reference of the target object, its own method id and the arguments and forwarding them to the communication module. (编码+传送给 communication module)
    2) Unmarshalling the results and forward them to the invoking object (从 communication module 那里读回结果并把结果返还给调用对象)
  - ✓ Dispatcher: there is one dispatcher for each remote object class. Responsible for mapping to an appropriate method in the skeleton based on the method id.
  - ✓ Skeleton: responsible for 1) Unmarshalling the arguments in the request and forwarding them to the servant. 2) Marshalling the results from the servant to be returned to the client.

13. Distributed garbage collection: A distributed garbage collector ensures that a remote object continues to exist as long as there are local or remote object references to the object. If no references exist then object will be removed and the memory will be released.

Chapter5 Indirect Communication

1. Introduction
   - Indirect communication is defined as communication between entities in a ds through an intermediary with no direct coupling between the sender and receiver
     - ✓ Space uncoupling: sender does not know or need to know the identity of the receiver(s)
     - ✓ Time uncoupling: sender and receiver can have independent lifetimes, they do not need to exist at the same time. (与 synchronous &asynchronous 无关)

2. Group Communication: offers a sp

3. Publish/Subscribe System
4. Message Queues
5. Shared memory approaches
6. Tuple Spaces