Research Study Log of Distributed System

Rong Zhuang

Total word count: 3454

1. **LAN or Local Area Network**

LAN is a computer network, which locates in a small area. It consists of multiple computers, printers, mobile devices or other facilities. All of them are connected with cables or wifi. Within this network, each device has a unique IP address to identify itself. LAN is a closed group. One device can contact another, send or receive data. Multiple desktop computers can share a printer. The transportation speed within LAN is very quick. The performance is reliable and the communication cost is low.

1. **WAN or Wide Area Network**

Compared with LAN, WAN covers a larger area, cross the cities, countries. WAN is also a computer network, which has the same components with LAN. Most of WAN are connected with switchers. Several LANs can be combined to generate a WAN. Generally, it costs more time to transfer data from one place to another within WAN. There are more delays of data transportation in WAN than in LAN. It is more complicated to manage a WAN.

1. **WLAN or Wireless Local Area Network**

WLAN enables computers or smart devices be connected to network with wireless signals. There is no need to plug cables into devices. There are two popular approaches for connecting, one is mobile signal, and another is wifi. For wifi, there is a router provides the service. Each time a device is trying to connect, it will be assigned a temp IP address. If this device turn off the wifi and it lost connection. Router will recycle the address and assign to a new device. For device, it may be assigned with different IP address for each connection. WLAN reduce the cost to build network since no need to place cables, less limitations.

1. **VLAN or Virtual Local Area Network**

VLAN is like LAN, but it is virtual. It is built from logical view not physical view. There is no limitation of physical location for VLAN. You can build VLAN for several machines which are located in difference LANs. For example, there are two department in a company, Sale Department and Development Department. Each department has its own sub network. Sale Department owns the IP address from 192.168.1.0 to 192.168.1.127. Development department owns the IP address from 192.168.1.128 to 192.168.1.255. With VLAN, these two network segments are combined together.

1. **VPN or Virtual Private Network**

VPN is network which is built upon public network, usually is internet. The communication is encrypted, so it is secure like private network. VPN is widely used in enterprise companies. The VPN gateway implements the remote access by encrypting the data packets and transforming the destination address of packets. One example of the usage for VPN is remote access internal enterprise network. An employee travels to another city or country, he can access company’s private network via VPN, if he can access internet.

1. **MAC Address or Medium/Media Access Control**

MAC Address is a unique identifier of a node in the network. It is an address of a physical device, like network card. It is has 6 bytes(48-bit) address space. They are divided into two parts. The first 24-bits is called Organizationally Unique Identifier(OUI). This part is assigned to the manufactories to identify the device producer. The second 24-bits are determined by the manufacturers themselves. And it is called Extended Unique Identifier(EUI). We have network cards in our desktop, laptop, phone and tablet. Each network card has a global unique MAC address.

1. **DNS or Domain Name System**

DNS is used to convert host name to IP address and vice versa. In the network of internet, each server within it has a unique IP address. We can access any of them if we know its IP address. However, it is hard for people to remember such number address. Instead, it is easy for us to remember meaningful names of these servers. DNS can help us to achieve this purpose because DNS has a table which stores the mapping relationship between IP address and host name. Each server should be registered to DNS, then DNS adds the new IP address and host name to it table. Any client wants to access a server, it will first ask DNS for the IP address. DNS will search the host name in its table and return the IP address to the client. Then client navigates to the destination server with this IP address.

1. **CDN or Content Delivery Network**

CDN is another virtual network, which is built upon Internet. The purpose of CDN is to delivery web pages or data more quickly and more efficiently. The key point for CDN is to avoid any bottleneck or node which may cause latency and instability. One approach to achieve this is to setup more nodes to the places where the network is so busy. The most important technology for CDN is load balance. Whenever a request is received, CDN must find a nearest location of the resource. So the requester can get the resource quickly. Many big websites which requires high volume network throughput put their content to several CDN nodes to accelerate the access speed.

1. **SSL or Secure Sockets Layer**

SSL is a security transportation protocol. It use the encryption technology to guarantee the security of transferring data through internet. There are three main functions for SSL. Firstly, it makes sure that data can be received properly by authorized client and server. Secondly, it makes sure the encrypted data won’t be stolen during transportation. Last, make sure data won’t lost or be changed during transportation.

Before transportation occurs, both server and client will be authorized first. Client sends a ‘hello’ message to server to setup a new conversation. Server send back a message which contains the information to create a new main key. Client gets the information and generate a main key and encrypt it with public key provide by server. Server receives the main key and acknowledge the key. After above steps, client will send data encrypted by the public key to server.

1. **HTTPS or Hypertext Transfer Protocol Secure**

HTTPS is the secure version of HTTP. The basic of HTTPS is SSL. It is widely used in internet, especially for banking and online shopping stores.

1. **Distributed Computing**

Distributed Computing use multiple independent machines to do calculation. In the reality, there are some calculation cases, like in math, biology, which need enormous time to finish the calculation. It is nearly impossible to do this if using a central machine, even it is powerful. In Distributed Computing, a master node will divide the calculation task to small pieces and assign each of them to a worker node. Each worker node only works on a piece of task, which is small enough for them to afford. After finishing the calculation, the result will be send to another separate node, which will merge the results all together and output it.

1. **Cloud Computing**

Compared with Distributed Computing, Cloud Computing is a more practical, it is an implementation. Cloud Computing is a service which charges the cost by the volume of usage, eg. CPU numbers, memory size or time. The most difference or advantage of Cloud Computing is, it is configurable. All of the resources like server, storage, software application, network bandwidth and services can be changed according to the actual requirement. It is convenient for customers to use, since they don’t need to care about the hardwares, they just focus on their business calculation. They spend a little effort on maintaining the system and can buy the service as need and return back if the calculation is finished. If they need more resources, they can easily extend them.

1. **Cloud Storage**

Cloud Storage is an extension of Cloud Computing. Since after the calculation output is created by the cloud computing, there must be some locations to store these results. From technical point of view, Cloud Storage required network technologies and distributed file systems. It manages plenty of different servers, provide a uniform interface for the end user. The data may be replicated to different locations globally to provide user a stable and friendly access experience. Users can easily access their files through multiple kinds of devices at any time, at any location with internet access.

1. **SaaS or Software-as-a-Service**

Saas provides on-demand software through internet technology. The SaaS service provider install softwares in their server. Customers can buy the service according to their business needs. They will be charged based on the what service are subscribed and how long the service is used. This business mode is very popular for small and medium companies, since they don’t need to invest a lot on the hardwares and software licenses.

1. **REST or Representational State Transfer**

REST is a not real software or industry standard, but a design style for web application architecture. This kind of style is called RESTful sytle. REST reduce the complexity of development, and improve the extendibility of system. It is much simpler than traditional web services which are built upon SOAP and WSDL.   
REST defines the interface how resources are accessed. In each interface it shall contains following methods: post, put, get, delete. They are the implementation for HTTP operations.

1. **JSON or Javascript Object Notation**

JSON is a data format used for exchanging data between web applications. It is lightweight and easy for human to understand. It is independent to server side programming languages. It has becomes an actual industry standard in today’s network world. JSON defeats XML format because it is cleaner and smaller than XML. There are too many redundant tags in XML and takes more time to generate and parse. JSON overcomes these both disadvantage of XML format.

JSON has two types: JSON Object and JSON Array. The former one is used to represent a single object which may contains several attributes. For example:

{"firstName":"Rong","lastName":"Zhuang","email":"jojozhuang@gmail.com"},

The latter one represents a collection of objects. Each object is a JSON Object.

{

"students":[

{"firstName":"Rong","lastName":" Zhuang ","email":" jojozhuang@gmail.com "},

{"firstName":"Bill","lastName":"Gates","email":"gates@microsoft.com"}

]

}

1. **GFS or Google File System**

GFS is an extendable distributed file system. It is invented by Google to store huge amount data generated by search engine. It consists of millions of cheap and ordinary machines. All of these single machines are connected together and are coordinated by GFS. GFS can provide high performance calculation on extremely large original data.

GFS has a master node and several worker node. The master node is the brain of this file system. It stores the metadatas, chunk indexes and location of worker nodes. Master node dispatch task to worker nodes and associate them work properly. Worker node is responsible for storing and retrieving data. Files are actually separated into chunks, one chunk may have several small chunks. The whole GFS system is built upon these chunks, they have several layers.

1. **Big Table**

Big Table is also invented by Google. It is a distributed storage system, which is used to handle huge amount of data. Compared with traditional relational database, Big Table is non-relational database. It is actually a multi-dimension sorted map. It can be deployed to thousands of machines and can handle PT level data volume.

1. **MapReduce**

MapReduce is a programming model. It is a design style not implementation. MapReduce is used to analyze big data. It has two steps: Map and Reduce.

In the map step, all of the data will be extracted and converted to key-value pair.

In the reduce step, the key-value pairs will be merged together and generated to new key-value pair or single result directly.

In the reality application, these two steps may be composited and more steps will be created for a complex analysis.

1. **Hadoop**

Hadoop is popular distributed computing system. It is written in java and it has two core components, HDFS(Hadoop Distributed File System) and MapReduce. The responsibility for HDFS is to store huge amount files or data. And MapRedeuce provides the calculation ability on these data. Generally, Hadoop has a name node and several data node.

1. Definition
   1. What is distributed system?

A distributed system contains multiple independent computers/machines/servers. When these servers are running, they may access same resources, communicate with each other, or work on a time consuming task together. No matter how complex the system is and no matter what kind of internal components are, to the outside world, they are treated as a single system. The client even doesn’t know there may be hundreds and thousands of machines behind the user interface.

* 1. Features for distributed system
     1. Complexity

Compared with single node system, a distributed system is much more complicated. For example, data is not stored in a single machine, but in many locations. One node has to communicate with others to get all of the data it requires to finish the job. And other challenges like how to choose strategy of fault tolerance, how to deal with access conflicts on same resource or how to cooperate hundred and thousands of servers. It requires more efforts to keep the system being correct, stable and robust.

* + 1. Multi-task support

A distributed system shall have the ability to work on multiple tasks simultaneously. This is the main advantage of distributed system, compared with a standalone system. Obviously, multi-thread programming is required for this support.

* + 1. Robustness

Another advantage of distributed system is robustness, compared with the single node system. In distributed system, if a machine is dead or shutdown, it will be replaced by another machine immediately. System will find a substitute machine to continue handle the task. The end-user may not realize this happens.

* + 1. Uniform interface

In most cases, the internal structure of a distributed system is very complicated. However, there is no need to expose it to the outside world. Instead, a uniform interface will help user to get hands on quickly and efficiently. To end-users, they just don’t need to know how the distributed system works internally. A friendly UI with consistent inputs and outputs is enough.

* + 1. Transparency

Whatever the architecture of the distributed system is and however the complexity of the system, it is always transparent to the end user. Transparency includes following aspects:

* Access – How a resource is accessed is unknown to end user. The user just cares about what is returned but not how it is returned.
* Location – Where the resource is located is unknown to end user. There may be lots of duplicated resources in several machines or locations. To the end user, he/she just wants to know whether the resource is obtained but not where it comes from.
* Migration – The resources may be moved to other places, but the end user doesn’t realize this situation and he/she actually doesn’t need to know that, since the system still works as nothing has been changed.
* Relocation – Sometimes, the resource the user is asking for may not be available from the original location. So the system may relocate the user to another machine or sub system to get it. The end user has no idea about the details of this procedure.
* Replication – Generally, a distributed system creates duplicated data copies in different locations. This is so-called data replication, which is an approach to make the system more robust. End user doesn’t know whether he/she is accessing the original data or a duplicated one.
* Concurrency – Since a distributed system can handle multiple tasks simultaneously, it often happens that one resource may be accessed by several users at the same time. The distributed system must solve this conflict problem, keep atomic transaction for each user and prevent from creating dead lock. For each end user, he/she doesn’t realize the existence of other rivals.
* Failure – Error/exceptions are hard to completely avoid. In distributed system, when a failure occurs, the task would be executed again on another machine or at another time. End user totally has no idea what has happened, since the result is still returned correctly.
  1. Interface Definition Language

IDL is used to define uniform interfaces for the distributed system to communicate with other systems. Generally, it has at least two arguments: one is the length of the data; another is actual data contains a string with that length.

* 1. Distributed Algorithm

When designing a distributed system, we are actually designing a distributed algorithm. And we must understand that there are some limitations for a distributed system.

* Each node/component has only limited information locally, but no whole picture of the state.
* Each node makes decision based on its limited partial knowledge.
* One machine’s failure doesn’t affect to others or ruin the whole system.
* No global clock exists.
  1. False Assumption
* Network is reliable, always can get entire data , no data lost
* Network is secure, no spy, won’t be stolen
* Network is homogeneous, same interfaces, same structures
* Topology doesn’t change, resource location is fixed
* Latency is zero, no latency, network speed is quick
* Bandwidth is infinite, can handle tremendous throughput.
* Transport cost is zero, no charge when sending or receiving data.
* There is an administrator, a big brother monitors everything.
  1. Transaction
     1. Characteristics
* Atomic: The transaction’s inner is black box, no matter how it is complicated.
* Consistent: After execution, the balance of constants is not broken. Resources/values are just moved from only place to another, not disappear, no new objects from nothing.
* Isolated: No disturbance between concurrent transactions.
* Durable: Changes becomes immutable once the transaction is confirmed.
  + 1. Test and Set
* Use ‘Test and set’ to preserve atomic operation in local machine.
* Hard to use ‘Test and set’ for remote machine, which has different memory spaces.
* Hard to implement shared memory to make distributed system coordinate well.

1. Architecture
   1. Architecture Styles

* Layered: One layer can only communicate with the nearest layer. Suppose a new request comes, it is handled by the first layer. Then, the first layer calls the second layer. Layers are invoked one by one from top to down or vise versa. Later, the latest layer finishes the work, and it returns the result to the previous layer. The result is transferred back at the opposite direction.
* Object-based: Each component in the distributed system is abstracted to one object. Each object has its own properties and functions. One object may have no, one or multiple relationships with others. These relations are created depends on the business model and system design. With these relationships, one object can call others to get a specific result, invoke an event, or just send out notifications.
* Data-centered: In this style, all of the data are stored in a centralized server. Each component needs to contact the data center to get the required information. The benefits of this style is, no synchronization is required, since there is only one copy of the data. If we are building a heavy server, we can consider to use this style. The disadvantage is obvious, the system will collapse if data center goes down. So it is recommended to have another shadow data center which can take the responsibility immediately if the data server shuts down.
* Event-based: This style is based on Publish-Subscribe design pattern. The components are not tied together. Each component can be a publisher, meanwhile, it can be a subscriber as well. The publisher invokes the event and notifies other components which have subscribed the event. The subscriber receives the notification and takes corresponding actions. The benefits of this style is, components are decoupled with each other. It is easy to create, update or remove component without any impact to the whole system.
  1. Application Layer
* User-interface Level: The responsibility of this level is to accept inputs, transfer them to processing level and display outputs. There is no much business logic in this level. It’s a bridge between end-user and distributed system. The real implementation of this level can be a webpage, a console or a touchable screen.
* Processing Level: This level is the kernel of distributed system. All of the business logics and functionalities are located in this level. It receives requests from the user-interface level, processes it and sends the result back. It may contains several components, each of them has a specific responsibility. In the reality, examples for this level can be a web server or a cloud service.
* Data Level: No much business logics in this level, either. It just stores data and provides the data retrieving service for the processing level. Though itself can be complicated, however, to the process level, it likes black box, just returns the data that the processing is asking for. It can be a separated and independent system, eg., a database.
  1. Alternative Client/Server Organization
     1. Heavy Client

Client contains all of the user-interface and application levels, even partial data level. This kind of client can provide users with great experience of quick response, rich UI functions and smooth interactions. End user feels no latency when operating the system. Compared with heavy server organization, there is less network throughout between client and server. However, it has some disadvantages. More work is required if a new version is released. All of the clients need to be upgraded. Some tricky problems may occurs when different versions are existing in the same system. And the system administrator has to pay more attention to maintain the system and keep it running properly. If any error/exception occurs, client has to handle it locally. And it is also hard for programmers to diagnose issues. Because each client is running in a special environment. This makes it hard to reproduce the issue and it becomes impossible to debug at client side.

* + 1. Heavy Server

All of the database and application levels are in the server side. Partial of the user-interface level can also be put into this server. The biggest benefits is that no need to distribute new releases to all of the clients, just need to upgrade the server with latest version. The source codes are easy to maintain. The maintenance fees are also lower. Support packages for the new features and bug fixing can be delivered quickly and smoothly. Only the server need to be upgraded. There are also some disadvantages. Firstly, there are much more communications between clients and server. Latency may occurs, and the user experience is not so good. Secondly, processing level becomes a performance bottleneck since most of the communications concentrate in this level. When more business logics added to the system, the server becomes hard to be stable, robust and efficient. Third, the server may store some sensitive data for the client. So, security is another issue.

* + 1. Choice

Each approach discussed above has its own advantage and disadvantage. It’s hard to say which one is better. And there is no uniform standard to choose heavy client or heavy server. The choice depends on requirements and business scenarios. Most commonly, these two approaches are combined in a large scale distributed system. For example, if we are building a cloud service, for example, a service which provides weather forecast, we may choose heavy server. The data and processing level are both in the server side. Client has no business logics, it just displays the weather forecast by calling the public APIs published by the heavy server. Another example, if we are building an online game, we may choose heavy client. For a game, it requires high FPS animation and high quality of sounds. With heavy client, end-user can enjoy the fantastic animations and sound. Clients connect to the server only when necessary.

1. Processes
   1. Process and Thread

These two concepts come from operating system. Process is a smallest unit, which can allocate resources(cpu processing time, memory) and handle requests. For any application, there must be at least one process. Threads are embedded into process. They are workers for process. Threads are not able to own resources by themselves, instead, they share the resources with other threads, which are all belong to the same process. Processes can use IPC(InterProcess Communication) to communicate with each other.

* 1. Virtual machine

Virtual machine is an interface above some layer or system. It contains specific machine instructions, invokes them and simulates as another system. It can locate above the operating system or under it.

1. Communication
   1. Protocol Level
      1. Protocol Layer

* Application: High level, focus on functionality.
* Presentation: Different data form, texts, pictures, videos.
* Session: Setup conversations between application, maintain their states.
* Transport: Wrap the transportation process, make sure data is transferred successfully.
* Network: It’s about routing. The job for it is to determine how to find an efficient path to transfer packets from the start point to the end point.
* Data Link: Basic data format, this layer considers how to encode or decode packet.
* Physical: Facilities or hardware used to transfer data, eg, cable, light, signal and etc.
  + 1. UDP or User Datagram Protocol
* One way, from sender to receiver data
* Low cost, efficient, without setup
* But no guarantee of the transportation, the sequence of packets may be different from sending sequence, data may be lost
  + 1. TCP or Transmission Control Protocol
* Two way, each node can send or receive data
* Reliable, packets in order
* Costly, setup is required
* Buffer is required, maybe delay
  1. Sync/Async Calls
     1. Synchronous Call
* Executed in sequence, easy to implement
* Call will wait for the response, block may happen
* Inefficient because of wait
  + 1. Asynchronous Call
* Executed not sequentially, much more complex
* Implemented with multi-thread
* Main process won’t hang, no wait
* More efficient
* Hard to debug
  1. Socket/Port
     1. Socket

Socket is a low layer implementation of server or client. The concept of server or client is relative. A server can be a client to other servers. A server socket is the one which provides services via receiving requests and sending packets back. A client socket is the one which asks for services via sending requests and receiving feedbacks. To socket itself, it doesn’t care. The job for it is just to send or receive data packets. Each socket has an IP address and a port.

* + 1. Port

Port is a channel for transferring packets. Each socket has a specific port number. There can be multiple sockets in one server. They may have different ports, so they can work on different channels. We can define one port for TCP communication, another port for UDP communication. Both of them won’t disturb each other. So this makes it possible that one server can be setup with the capability to handle multiple communication services.

* 1. Data Transfer
     1. Data Format

Different systems may have different data formats. When they try to communicate with each other, a unified data format is necessary for transportation. So the receiver can understand and consume the data received from sender.

* + 1. Marshalling/Unmarshalling

Marshaling is the process that the sender converts objects to unified data format, like xml. XML is a commonly used data format. Unmarshaling is the opposite procedure that the receiver converts XML to objects or other formats which the receiver can consume.

* 1. Remote Procedure Call
     1. Steps of RPC

Client->Local OS->Remote OS->Server->Local OS->Client OS->Client

* + 1. RPC Exchange Protocols
* R
* RR
* RRA

1. Security
   1. Encryption
      1. Symmetric key encryption

* Only one key. It is used to encrypt and also used to decrypt.
* It is efficient and quick.
* It is simple but the problem is how to distribute the key to both sides.
  + 1. Non-symmetric key encryption
* Two keys used. One key for encryption, another for decryption.
* Not efficient than symmetric key encryption.
* When the two keys are generated, one key can be selected as private key, another used as public key.
* Public key can be sent to the user and used to encrypt.
* Private Key can be kept in server and used to decrypt.
  1. Certificate
     1. Prerequisites
* The public key is authorized and real.
* The private key is very safe and unknown to others.

1. Assignment
   1. JokeServer
      1. JokeServer

* JokeServer holds the data of joke/proverb list, and also holds the identities and their states of the clients.
* Joke Server provide two services. One is to handle joke/proverb request, another is for admin purpose. These two services are implemented by sockets with different ports.
* When a joke/proverb request comes, JokeServer dispatches the task to a Worker in a separate thread.
* When JokeServer is starting up, an AdminListener service is started up as well.
  + 1. Worker
* A worker is one-time working unit. It is created by JokeServer when a new joke/proverb request comes. After finishing the task, it will be destroyed.
* There may be multiple instances of Worker running at the same time, they work simultaneously without interfere with others.
* Worker gets request and other relevant information from JokeServer, generates a joke or proverb(depends on the current server mode), sends back to client.
  + 1. AdminListener
* The instance of AdminListener works on a separate thread and only monitors at the specific admin channel(port).
* Similar with JokeServer, when a new admin request comes, AdminListener creates a new instance of AdminWorker and dispatches the request to it.
  + 1. AdminWorker
* Like Worker, the AdminWorker is also a one-time working unit. After finish handling the admin request, it will be destroyed.
  + 1. JokeClient
* JokeClient runs as console, waiting user’s input.
* It monitors at a specific port.
* It gets the name from first input, receive the second input as request and sends to JokeServer to get joke or proverb.
* After receiving the joke or proverb from Joke Server, displays it to end user in console.
* There can be multiple instance of JokeClient, one client for one user.
  + 1. JokeClientAdmin
* JokeClientAdmin also runs as console, but it sends Admin commands. The user of the client is an administrator.
* It is setup to monitor a different port with JokeClient.
* It gets command from administrator, then sends it to Joke Server. The AdminListener receives the request and switch server mode accordingly.
  1. MyWebServer
     1. MyWebServer
* MyWebServer does nothing but monitors specific port to wait the http request from client browsers.
* When a new http request comes, it creates an instance of HttpWorker and dispatches the request to it.
  + 1. HttpWorker
* HttpWorker is a one-time working unit. It will be destroyed once its job is finished. There can be multiple instances of HttpWorker running at the same time.
* The request processing logic is more complicated than Joke Server request. HttpWorker validates the http request first. If it is valid, then parses the parameters from the request to identify what exact request it is.
* There are three types of requests: a) looking for the structure of a folder; b) looking for the content of a file; c) calling a fake CGI command to get dynamical calculation result.
* After processing is finished, HttpWorker sends back responses with different http response code, MIME types(text, html, image) and different content accordingly.
* The execution logs are stored to local files.
  1. Mimer
     1. MyWebServer
* Same class as the one in above MyWebServer. The difference is a new instance of BCListener will be created when MyWebServer is starting up and it monitor at a specific port.
  + 1. HttpWorker
* Same class as the one in above MyWebServer, the only difference is that it can handle customized file type, eg. xyz.
  + 1. BCListener
* The instance of BCListener is created by MyWebServer along with the system’s setup.
* It monitors at a specific port, waiting the xml stream sent by BCHandler.
* When new request comes, it creates instance of BCWorker and assigns request to it.
  + 1. BCWorker
* The instance of BCWorker receives data which is in XML format, unmarshales it and converts to object by using XStream. Finally, displayes the content to end user in console.
  + 1. BCHandler
* BCHandler is located in client’s machine. It is used to support client browser to handle customized file, eg. xyz. It is a tool which enables web browser deal with this kind of file properly. Some configurations have to been set before running.
* BCHandler is invoked when a file with extension xyz is opened from web browser. File path is provided by the web browser.
* It reads the content of the file and converts the content to XML format. Then sends the converted xml data to BCListener.
  1. HostServer
     1. HostServer
* HostServer monitors the main port, waits for new request, and creates AgentListener to handle it.
* Each time it finds a new port and assigns it to the newly created instance of AgentListener.
  + 1. AgentListener
* There can be multiple instances of AgentListener. They works independently.
* Each instance of AgentListener monitors at a specific port.
* When new request comes, it creates instance of AgentWorker and assign request to it.
  + 1. AgentWorker
* AgentWorker is a one-time working unit. There can be multiple instances of AgentWorker, which are working for the same AgentListener.
* Two main functions provided by AgentWorker: State Update and Migration.
* When the ‘Update’ request comes, AgentWorker first finds the current state from AgentHolder, updates it(increment by 1), and sends back the state to client.
* If a ‘Migration’ request comes, AgentWorker first contacts HostSever to get a new valid port number. Then a new instance of AgentListener will be setup at this port. AgentWorker returns back the result to client and shuts down the current AgentListener.
  + 1. AgentHolder

Used as model class, which stores data for requests.

* 1. AsyncJoke
     1. AsyncJokeServer
* Same with the one in JokeServer assignment.
  + 1. Worker
* Same with the one in JokeServer assignment. Only one difference, it doesn’t send joke/proverb back immediately. Instead, it waits for 40 seconds and then sends it to client via UDP.
  + 1. AdminListener
* Same with the one in JokeServer assignment.
* When a new admin request comes, AdminListener creates a new instance of AdminWorker and dispatches the request to it.
  + 1. AdminWorker
* Same with the one in JokeServer assignment.
  + 1. AsyncJokeClient
* AsyncJokeClient runs as console, waiting end user’s input.
* It is setup to monitor specific port.
* It sends a request to AsyncJokeServer to get joke or proverb.
* After sending the request, create an instance of UdpWorker.
  + 1. UdpWorker
* UdpWorker instance is created by AsynJokeClient.
* It only receives udp messages, no sending. It monitors at separate port.
* After receiving the joke or proverb from Async Joke Server, displays it to end user in console.
  + 1. AsyncJokeClientAdmin
* Same with the one in JokeServer assignment.
  1. Distributed Intelligent Agent
     1. NameServer
* NameServer is the administrator of the DIA system. It holds the list of servers, agents and groups.
* NameServer provides two services. The first service is to provide the current states of the DIA system in html format. The second service is to response to the requests from servers or agents in UDP channel.
* When NameServer is setup, an instance of CommunicationListener is created and monitors at a specific port.
  + 1. CommunicationListener
* This listener setups a UDP socket. Its responsibility is to handle the requests from servers and agents, see below.
* It accepts the registration request for server and agent. And updates these states to name server.
* It also accepts the request for client to find a new valid host server for migration.
* It gets notification when an agent is killed. This agent will be removed from the list.
  + 1. HostServer
* HostServer is working for the client. There can be multiple instances in the DIA system.
* It can be setup with default port or with specified port.
* Each hostserver monitors at a specific port, dispatches request to AgentListener.
* AgentListener contacts with NameServer to register new server via UDP.
  + 1. AgentListener
* The instance of AgentListener is called Agent. It is created by HostServer. And it monitors a new port which is determined by itself.
* When it is created, it sends response to the client and begins to monitor at its port.
* When a new request comes, it dispatches the request to AgentWorker.
* AgentListener contact with NameServer to register new agent via UDP.
  + 1. AgentWorker
* The instance of AgentWorker is created by Agent(AgentListener). There may be multiple instances of AgentWorker running at the same time.
* Each Agent Worker handles the request of ’State update’, migration and killing itself.
* AgentWorker contacts with NameServer to notify that the agent is killed.
  + 1. UdpHelper

It is a helper class used to send and receive UDP messages with specified IP and port.

* + 1. LogHelper

It is a helper class used to store logs to local files.