**CSC 435 Study Log – Xilong Yu**

Professor Elliott—Winter 2020

**20 concepts**

1. Middleware

Middleware is the hidden layer sitting between the operating system and the user applications or devices. It is often called “software glue”. It enables the user applications or device to communicate the operating system. For exampe, a keyboard driver is a middleware. Without a keyboard driver, you can’t use your keyboard. Middleware reorganized the resources and services provided by the underlying systems and provide the resources and services for the upper layer applications or processes to consume.

The distributed system can be orgnized in a middleware layer. It reorganizes the resources and services offered by multiple single computers and provides the a common interface for the top-layer application and processes.

1. CAP theorem

CAP theorem is also called Brewer’s theorem. It is proposed by computer scientist Eric Brewer.

C stands for Consistency, which means that every read gets the most recent write or an error. A stands for Availability, which means that Every request gets a non-error response, but it doesn’t guarantee that it is the most recent write. P stands for Partition tolerance, meaning that the system continues to operate despite a network partition failure(a node is cut off from the network). The theorem states that trade-off has to be made between consistency and availability when a network partition failure happens.

1. Byzantine failure

It’s also called byzantine fault or interactive consistency, byzantine agreement problem, etc. This term takes its name from the “Byzantine General Problem” which describes a situation where all the members(troops, components, teams) of the whole army must agree on the same stragegy so as to avoid catastrophic failure. For example, they must agree on a strategy deciding whether or not to attack or retreat. Catastphones happen if they fail to agree upon the same strategy. The tricky part is that even though they have a concerted strategy, the implementation of that stategy is still difficult because some of the members of the army might not be reliable.

When it comes to the distributed system, some components of the system may fail. However, different nodes/components may make different judgement about whether or not a component has failed.

1. Distributed Hash Table

A distributed Hashtable is similar to a regular hashtable except that it is implemented in a distributed fashion. It is particularly useful in peer-to-peer content-sharing system such as BitTorrent, because it enables the resources to be distributed across the internet.

1. Overlay network

An overlay network is a network structure in which each node is linked to a set of other nodes. The set of nodes that a node is linked to are called the neighbors of the node.

1. LWP(lightweighted processes)

This is a combination of the user-level threads model and kernel-level thread level. A light weight process runs in a regular process, and the regular process(heavyweight process) can contain more than one lightweight process. Besides that, the system can offer a user-level thread package. The user-level thread package is just like a regular thread package implemented on regular user-level threads, meaning that all operations on threads are operated without involving the kernel. The thread package is shared by seveval lightweight processes, and each lightweight process can run its own thread. The advantage of a LWP is that the context switch is completely in user space, resulting in high performance.

1. Concurrency transparency

A distributed system can have many shared resources. The user often make asychronous request to access the resources. It’s OK if users just read without updating the shared resources. When two or more users try to access the sources at the same time, we need to make sure that they don’t interfere with each other. We need to create the illusion to the users accessing the shared resources that they have exclusive access to the shared resources.

1. Remote Procedure Call(RPC)

Remote Procedure Call means that there’s a program/procedure sitting on the remote machine and waitting to be called and executed by another machine(the client machine) through the network, which can be local or global. When we call the procedure on the client machine, it’s just like calling a local method. The method acutually takes the data to the remote machine for it to run. The communication basically goes through the client program/procedure, client stub, client operating system, network/(global or local), server system, server stub, processing on the server procedure, remote server operating system, network transmission, client operating system, client stub, client procedure/program. An example of a RPC I can think of is calling a stored procedure on a remote database with a program.

1. Multicast communication

Multicast communication in distributed system is an very important topic. It simply means sending data to multiple receivers. The implementation of multicast communication can be diffult and can cost huge management effort before the coming of peer to peer technology and the development in structured overlay management. There are different ways of setting multicast communication including application level tree-based multicasting, flooding-based multicasting, gossip-based data dissemination and so forth.

1. Round-Robin scheduling

It’s one of the pre-emptive algorithms used by process and network schedulers in computing. In a round-robin algorithm, equal portions of time slices are assigned to each processes. It is organized in a cicular order and handles all processes without priority. Coming requests are placed in a request queue and executed by the processor one after another. If the task is finished, the task will be removed from the request queue, if the request is not finished, the request will be placed in the end of the request queue and wait to be executed by the processor. Round-robin scheduling is a very basic and easy algorithms used by processes and network design.

1. Clock synchronization

In a single machine, time is not ambiguous. In a distributed system, however, achieving agreement on time is not easy. This is because different local machines may have different clocks. Therefore, synchronizing all clocks in a distributed system is very complicated. If a machine has a UTC receiver, we usually use UTC(Universally Coordinated Time) as a worldwide standard for time synchronization. However, UTC is not precise. There can be an error of -1or +1 due to random atmospheric fluctuation.

1. Interprocess communication(IPC)

Sometimes different programs/processes need to cooperate to complete a common goal, for example, a multithreading application is often composed of multiple programming doing different works. IPC needs kernel intervention. Basically when process A context is switched to process B, what happens is the switch from user space of process A to a kernal space, the kernal will do a series of operations including chaning the memory map and flusing the translation lookup buffer, which is to maintain the state of process A. Then the context is switched from kerel space to user space of process B. When the context is switch back, the same process happens. That’s why a context switching from process to process is usually costly.

1. Multiplexing

It is equivalent to many to one, which means we put many conversations onto one transmission facility. We do so because it cost less for one shared facility to carry multiple transmission signals than tranmitting it with multiple channels. A reverse process of multiplexing is called demultiplexing, which extracts the original channels on the receiver end. A device that performs multiplexing is called a multiplexer, short name as MUX, and a device that performs the opposite process is called a demultiplexer.

1. AMQP (advanced message queuing protocol)

The Advanced Message Queuing Protocol is an open standard application layer protocol for message-oriented middleware. The features of AMQP includes message orientation, queuing, routing, reliability and security. Just like protocols like SMTP, HTTP, FTP, which have created interoperable systems, AMQP defines the rules for the message producer and consumer to follow in order to send or receive messages. AMQP is different from java messaging service (JMS) in that java messaging service defines that rules in the writing of the API interface, AMQP, however, is a wire-level protocol. A wire-level protocol is an application layer protocol and it describes how data are exchanged from one end to another. It’s a lower-level protocol than JMS. In AMQP, the data is tranmitted as a stream of bytes. Therefore, any tool that can create and translate data that conforms to this data format can interoperate with each other, no matter what progamming language it is using.

1. Jitter

Jitter is referred to as packet delay variation, which means the variance in time delay between data packets over a network. The time it takes for packets to be sent is not the same as the time in which he will arrive at the end point. And if one packets encounters some delays in the intermediate node, the packet will arrive later than expected. This is especially true in global internet environment, compared to a local network. If you are sending a lot of data to the end point. Some data packet will arrive earlier than others. But sometimes, we would want to data packet to arrive in chronological order so that we can process them in the end point in order, for example, a video streaming system. To reduce the bad effects of jitters, we can use jitter buffers. Normally speaking, the larger the buffer is, the better performance we achieve. But buffers are not endless sometimes, so we often need to make compromises between increasing buffer size and dropping some data packets.

1. Symmetric cryptography

It means a cryptographic system where the sender and recipient of the messages use the same key. Those systems include DES, AES and so forth. This is in contrast to public key methods such as RSA, in which the sender and the receiver use different keys. Compared to the public key methods, the Symmetric key methods are computationally faster. Therefore, RSA is sometimes used to transmit a DES key that will later be used for the descryption of the ciphers.

1. DHCP

Dynamic Host Configuration Protocol, which is a network protocol used with local network. It belongs to the application level of the OSI model. It mainly has two functionalities: first, automatically assign IP addresses to users so that the users don’t need to manually configure IP addresses on the computers. Second, used by internal network administrator to manage all the local computers.

1. Mobile IP

The fullname is Mobile Internet Protocol. It is a communication protocol that allows the users’ devices to have the same IP addresses when moving from one network to another. Thus, the communication and session or connection will not be dropped because of the changing of the IP addresses.

1. Kerberos

Kerberos is a network authentication protocol. It can be seen as a security system that help clients set up a secure channel with any server in a distributed system. The security is based on shared secret keys. There are AS and TGS, which respectively represents Authentication Server and Ticket Granting Service. The AS handles a login request from the user. If the user is authenticated, it will get a key that can be used to set up secure channels with servers. TGS is there to handle the setting up of secure channels. TGS hands out tickets, which is special messages used to tell the servers that the client is anthenticated.

1. One-time Pad

One-time pad is a “perfect encryption” because it cannot be cracked. It requires the use of a one-time preshared key which has the same size as or longer than the message to be sent. Basically, the plaintext is paired with a random secret key and the cipher text is generated by doing bit operation between the plaintext and the secret key. The ciphertext is impossible to decryt if the key is truly random, never fall into a repetive pattern, kept completely secret and the encrypted message have a uniform frequency distribution.

**Forum Postings:**

Mitchell Klich

Nice to meet you here, Mitchell. I used to have a dog too. But I don't have enough time to take care of it. So I had to give it to someone else. Glad to know that you don't have the same problem.

Goran Somic

Hi, Goran. I like documentaries too. Two of the most recent documentaries I really like are American Factory and One Child Nation. If you haven't seen them, I highly recommend.

Matt Jacobs

Hi, Matt. Can you recommend three of your favorite movies?

reply to George Katsigiannis

I don't think it's a good idea to use username to identify a user, unless we have some mechanism to force the user to use different username. That means we need to store all the taken/registered username on the server.

reply to Clark Elliott

UUID seems to be a good choice. It seems like a new uuid will be generated each time the client app starts. So do we assume the conversation ends when a user quit or the client app closes?

reply to Nardos Tessema

I have that concern too when I first use UUID. Then I was told the UUID is meant to be generated uniquely. UUID collision is vanishingly small. We usually don't need to worry about that.

reply to George Katsigiannis

Looks like you are storing the user-server conversation state on the server using UUID as an identifier? A concern I'm having is: if you store the user conversation state on the server, how do you remove it when the conversation is no longer active?

reply to Nardos Tessema

Yes. On a web server, which is connected with a browser, we can do that with a session. I'm not very good with a socket. Can we do something similar on a socket?

Blockchain:

Xilong Yu

yesterday at 9:18 AM

Thanks Simon, this is useful.

thread safety

Xilong Yu posted Feb 25, 2020 10:00 AM Last edited: Tuesday, February 25, 2020 10:09 AM CST

This page automatically marks posts as read as you scroll.Adjust automatic marking as read setting

Here is my design: I'm broadcasting the new blockrecord each time and I defined the global/canonical blockchain as a list. My logic goes like this, pop a puzzle from the priority queue, check if it's solved(first check), abandon it(if solved) or solve the puzzle, check again(second check), abandon it or broadcast it, when the server receives the broadcast block, check again(third check), abandon it or add it to the block chain. The last check and add is done in a locked block to make sure it's thread safe.

Feel free to tell me what do you think of this design.

yesterday at 10:26 AMLast edited: Tuesday, February 25, 2020 10:27 AM CST

Sorry, I just realized that this is not thread safe since the canonical blockchain is not a shared object. Each server/process maintains its own blockchain...

yesterday at 3:19 PMLast edited: Tuesday, February 25, 2020 3:20 PM CST

Using a PriorityBlockingQueue will make sure it's thread safe. So before the queue is being popped, the three servers will have the same queue with the same order. The problem I'm worrying about is that the three servers may work out the puzzle at the same time and broadcast the new blockrecord at the same time, but each blockrecord may arrive at different servers in any order.

yesterday at 5:44 PM

Yes, Thank you Nardos. VerificationTimestamp will solve cases where there are no ties. As for the tie, do you mean we select the block with smaller process number?

Network lab

material sharing (http authentication, session, cookie, browser)

This page automatically marks posts as read as you scroll.Adjust automatic marking as read setting

Hi class, I found a useful video while doing the http lab:

<https://youtube.com/watch?v=GxiFXUFKo1M>

Xilong Yu

February 14 at 9:24 PM

material sharing (event-based architecture)

<https://www.youtube.com/watch?v=rJHTK2TfZ1I>

Xilong Yu

February 20 at 11:56 AM

material sharing (hash encryption api in java)

<https://www.youtube.com/watch?v=dh8ura4rVUM>

**thoughts and notes from the textbook and other resources:**

**Chapter 1.**

* The distributed system mainly have characteristics:

1. Collection of automonous computing elements, which means that the distributed system is composed of multiple nodes and process. We need to decide how those multiple nodes communicate and cooperate with each other.

2. Single coherent system, which means that the whole distributed system works as one single entity. The end users would not be able to be aware of the fact that the system is composed of multiple nodes. It appears to the users as one single node that is providing all the services and information.

**Chapter 2.**

* What’s special about a distributed architechture?

The major difference between a distributed system and a centralized system is the way components of the system is organized and how they interact with each other. In a centralized system, most of the software components are implemented on a centralized server. In a distributed system, however, software components are dispersed across many different machines. Some of the most typical distributed architectural styles include: layered architechture, object-based architectures, resource-centereds architectures and event-based architectures.

Layered Architecures: components at a higher lay can make a call to components in a lower layer. The call usually happens in a from top to down fashion, rarely in the opposite direction.

Resource-based architectures:

Three layered DNS Server:

**Chapter 3.**

* Thread implementation

A thread package is often used to provide operatons to create and destory threads. There basically two ways to implement a thread package. First way: create a thread library which is completely in the user space. Second way: let the kernel know the threads and schedule them.

Advantages of the user-level thread library: the creation and destruction of a thread is low-cost, since all thread management is in the user’s address space. The cost of a thread creation is determined by the memory allocation of setting up a thread stack(each thread will has its own stack space). The cost of thread destruction is the cost of freeing the stack memory. Both operations are very cheap. Another advantage of a user-level threads is thread context switching is easy, which takes only a few instructions. Only the values of CPU registers need to be stored and later reloaded. No need to perturbate the memory caches. Disadvantage: The disadvantage is due to the deployment of the many-to-one threading model. If one thread makes a blocking system call, the whole process which the thread belongs to will be blocked.

Advantages of the kernel-level thread library(one-to-one threading model): every thread operation is carried out by the kernel, avoiding the one thread blocking the whole process problem. But each thread operation in this model requires a system call, which means that switching thread context is equal to switching a process context.

* A multi-threaded server

A multi-threaded server has more than one threads, in comparison to single-thread server. A multi-threaded server does a better job dealing with and dispatching concurrent client requests. It can take full advantage of the server resources, rather than keep client requests waiting in line for the resources that is locked due to the previous user request. A multi-threaded server can make full use of the server rather than keep it idle.

* Concurrent server vs iterative servers

A concurrent server doesn’t handle the request by itself. More precisely, it doesn’t handle the requests on one thread. It passes or dispatches the coming requests to other servers or threads so that the main thread is responsible for the dispatching not the processing. This makes a concurrent server much more responsive and can immediate concurrent requests, which hugely enchances performance, efficiency and user experience.

For an iterative server, the server deals with the request, return a response to the client and wait for the next request. No other request is being responded while the server is handling the current request. An interative server has its value too. Basicly, it’s lightweight and easy to implement and the cost of adopting a iterative server is low. So if we don’t require high cocurrency and want to cut budget. An iterative server can do the job.

* How to interrupt a server

Solution 1: Disconnect the client and restart the client. When the connection between the client and the server is cut off. There will be an exception or some other mechanisms implemented on the server side, which will stop the client-server conversation.

Solution 2: Implement the server that can receive out-of-band data from the clients. The out-of-band is data that can be processed by the server with priority. To do this, we can let the server listen to a different end point to which the out-of-band data is sent. This end point has a higher priority than end points/ports where normal data is sent.

Solution 3: Send out-of-band data or the original request through the same connection.The current data is viewed as urgent data. When urgent data is detected at the server, the server is interrupted.

Stateless vs stateful servers

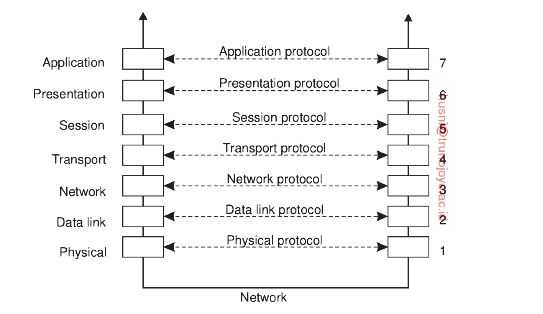
**Chapter 4**

* Two types of communication Service

Connection-oriented service, for example, telephone.

Connectionless service: mailbox, UDP

* OSI (Open System Interconnection Model) seven layer protocol



Application: The layer is closest to the end user. It provides interfaces for software applications, and is responsible for communicating with other softwares. HTTP, HTTPS, FTP, HTML, POP are in this layer.

Presentation: This layer is responsible for converting data received from the receiver and transforming it to formats easier to be transmitted.

Session: Responsible for maintaining the conversation/dialogue between the client and the server.

Transport: Encaptulate the data and header into a data packet and transmit the data packet from the source to the destination. TCP, UDP protocol are on this layer.

Network: Responsible for the delivery of message and determines the data transmission paths/routings.

Data link: Provides node to node data transfer. It has two sub layers: logical link control(LLC) and Media access control(MAC).

Physical: It is responsible for managing the communication between and raw data transmassion between physical devices. It converts the digital bits into electrical signals.

* Processes involved in a remote procedure call operation

There are basically ten steps involved in a RPC. The message goes through the client program/procedure, client stub, client operating system, network/(global or local), server system, server stub, processing on the server procedure, remote server operating system, network transmission, client operating system, client stub, client procedure/program.

(original notes from the textbook)  
1. The client program calls the client stub.

2. The client stub builds a message and calls the local operation system.

3. The client operation system sends the message to the remote operation system.

4. The remote OS gives the message to the server sub.

5. The server stub unpacks the parameters and calls the server.

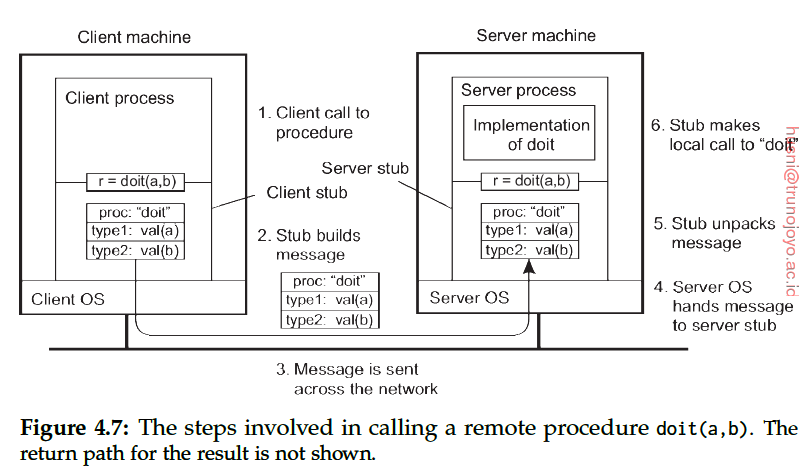
6. The server does the work and returns the result to the stub.

7. The server stub packs the result in a message and calls its local operating system.

8. The server’s operating system sends the message to the clients operating system.

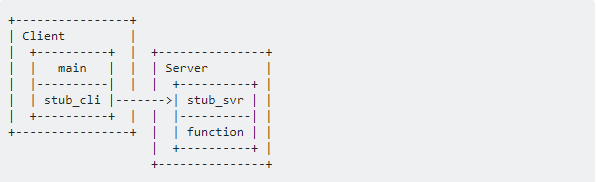
9. The client gives the message to the client stub.

10. The stub unpacks the result and returns it to the client.



* IDL(interface definition language)

An IDL is used to set up Remote Procedure Call (RPC) communication between clients and servers. Essentially, you're using an IDL to define the client-server interface so that the RPC mechanism can generate the code stubs needed to call functions across the network.



In the above example, instead of calling function in the same program, main calls a client stub function (with the same prototype as function) that is responsible for packaging the information and moving it to another process through the wire.This may be the same computer or another machine, it doesn't really matter-one of RPC's benefits is being able to move servers around at will.(referred to stackover flow)

* Atomic multicast

When we multicast message to multiple receivers, we want to guarantee that either all group members get the message or none of them get the message. Four properties are required from an atomic multicast protocol. Validity: if a correct participant multicasts a message, then all correct receivers will receive it. Uniform: Agreement: if one correct participants receives a message, then all correct participants will receive the message. Uniform Integrity: a message is received by each participants at most once and only if it was previously multicast. Uniform total order: if any correct receivers receives message 1 before message 2, then every other receiver must receive the messages in the same order.

**Chapter 6**

* Atomic clock

Atomic clock is invented in 1948, which makes possible to measure time more precisely than before and free from the effect of the earth wiggling and wobbling. The definition of a second is the time it takes for the cesium 133 to make exactly 9192631770 transitions.

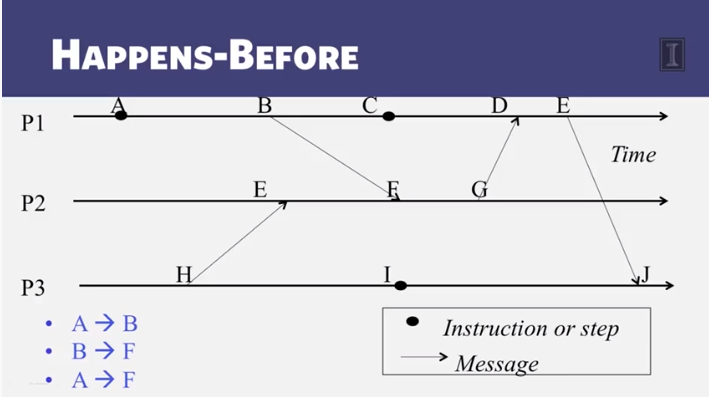
* Network Time Protocol

NTP is a network clock synchronization protocol. Basically, if we know when we send the message, when we receive the message and how much time the request stays on the server, we can estimate the time from a server. NTP is one of the oldest network protocols in clock synchronization. It is intented to synchronized all computers in a group to be within a few milliseconds of UTC.

* Lamport Timestamps

It is very difficult to synchronize real time across processes. So the Lamport timestamps algorithm provides a simple way to determine the order of events in a distributed system. The timestamps acts as virtual time, not real time. The algorithm follows some simple rules:

1. A process increases its counter before each event in that process;
2. When a process sends a message, the message also contains counter value;
3. When receiving a message, the counter of the receiver is updated to max(local clock, message timestamp)+1



Comparison between Lamport timestamps and Vector timestamps

Lamport timestamps assigns integer clocks to events; it obeys causality; it can’t distinguish concurrent events;

Vector timestamps also obeys causality; by using more space, it can also identify concurrent events.

* Berkeley’s Algorithm

Berkeley’s Algorithm is a clock synchronization technique used in distributed systems, assuming that each node in the network either doesn’t have an accurate time source or doesn’t have an UTC server. The algorithm goes like this:1) An individual node is chosen as the master node from a pool nodes in the network. It acts as a master and rest of the nodes act as slaves. Master node is chosen using a election process/leader election algorithm. 2) The master node periodically reach out to slaves nodes and gets clock their time using Cristian’s algorithm.

**Chapter 7**

* Coordination VS Synchronization

Coordination and Synchronization are similar but different. Coordination is more general than synchronization. We can say that coordination includes synchronization. There are different types of synchonizaiton such as process synchronization and data synchronization. Coordination in distributed system is much more difficult to implement than in a single machine.

**Chapter 9**

* Four types of sercurity threats:

Interception: Unauthorized access to the service or data. For example, The data transmission between two parties is illegalled obtained by a third person.

Interruption: A service or resouce cosumption is interfered by some other process. For example, a ddos attack can be classified as interruption security threat.

Modification: Authorized tampering of the original data or service. For example, a sql injection attack which changed or removed some of the data in a database.

Fabrication: An additional data which doesn’t exist before is generated. For example, a databased is breached and new previledged user accounts are added to it.

Four security messures:

Encryption: Encrypt the data before tranmission so that an attacker can not understand it even though he can obtain the data. Encryption can also support integrity checks, since modified data will be encrypted differently from original data.

Authentication: To identify a user before allowing him to access the resource. A most common example is user login. A user need to type in the correct username and password in order to access corresponding resources provided by the sites/server.

Authorization: This is a procedure after the authentication. Before accessing a certain resource, the user needs to be further analyzed if he has the authority to do that action. Its like different users may have different previleges of that system. For example, a VIP account can enjoy many more services from a websites than a regular account.

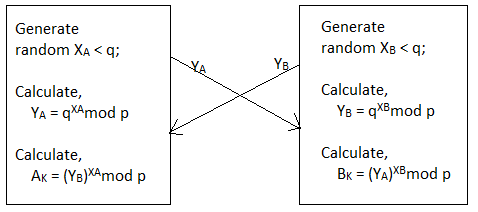
Auditing: It’s used to record user actions on the server. What operations certain users did, when and how, are all recorded for later analysis.

* Secret key distribution

An important problem in cryptography is how to establish keys for use in cryptographic protocols such as DES and AES, especially when the two parties are widely separated. Public key methods such as RSA provides one solution. One important key exchange method is Diffie-hellman Key exchange. Basically, A and B can create a private key K in the following way:

1. A or B chooses a big and secure prime number p and a primitive root α (mod p). p and x can be public.
2. A chooses a secret random x (1<=x<=p-2), and B chooses a secret random y in the same range.
3. A sends αx to B and B sends αy to A.
4. With the messages they have received, they can calculate the key K, since K = (αx)y=( αy)x. (mod p)

A and B can then go on and encrypt their messages with K or get some other key based on the common K they have.



* Certificate Authority

A certificate Authority is a trusted third party that issues digital certificate, which certifies that a public key is owned by a certain entity. This enables others to rely on the signatures to test the validity of the private key corresponding to the certified public key. An important use of cerficate authority is to sign certificate used in https, which is a very secure protocol for the network communication.