**CSC 435 Study Log**

Fernando Araujo

Total Word Count: 10929

**Twenty Class Concepts**

**1. Distributed Systems:**

A Distributed System is a group of computers that are all interconnected via a network. All the computers in the system communicate with each other and coordinate what they do in order to achieve a common goal. This communication is mainly based on messages that get sent between the computers in this network. These messages can contain anything from data to just signals to start or interrupt certain processes. This is necessary since there is no global clock and no inter-process communication in such a system. This necessity for coordination can become a main point of failure for the entire system. Due to a lack of test and set instruction, a mutual exclusion issue can sometimes take place.

**2. Mobility**

Mobility can refer to both code and processes that can be transferred across a network for another computer to execute. The main goal of this migration is for a system of computers to be able to share the execution of programs within its members. For example, in the case that one server is very busy, it can share some of its running code with an idle server in order to get more things done at the same time and alleviate its burden. Mobility is split into two different types called Weak Mobility and Strong Mobility.

Using Weak Mobility, a section of code or a process is sent over to another machine. This machine then follows to start running this code from the beginning. While this is the simplest to utilize, since there is no need to track down the current state of the code, it is also less efficient. Since the process needs to be started again, this kind of mobility is not useful for time-sensitive applications. An example of this type of mobility would be JavaScript, since this code is downloaded from the beginning every time you visit a webpage.

Strong Mobility on the other hand, consists of sharing a section of code or process and continue running it from its current state. This benefit also makes it much harder to code, since there is a need to keep track of the current state and to handle all the resources needed to continue the execution on the other machine. There are multiple strategies to accomplish this, such as pushing memory pages, migrating the memory and pulling pages.

**3. UDP vs TCP:**

UDP stands for User Datagram Protocol and it is a very simple and fast connectionless protocol. It extremely efficient but also very unreliable since for every message we send, we do not hear any confirmation back, there is no handshake or acknowledgment and therefore there is no certainty in whether it was received or not. This protocol also does not take into consideration the ordering of the packets and therefore the programmer is the one who has to worry about reordering them on arrival. Even though it has so many disadvantages, its innate speed and flexibility make it a good option when you want to emphasize efficiency. Its downsides can be addressed by implementing specific features on top of this basic protocol

On the other hand the TCP or Transmission Control Protocol, is a protocol that is extremely reliable but much less efficient than UDP. This is because TCP comes with a lot of reliability features that are missing from UDP. One of them, is the fact that it looks for confirmation f arrival for every message sent. If the confirmation is not received, then the message is sent all over again. This addresses any kind of issue that could result from packet losses in the network. It also buffers and reorders all the packages in the case that a package arrived in the wrong order, saving programmer’s time since they do not have to implement this feature themselves.

In the end, the usage of UDP over TCP or the other way around completely depends on the priority of the application at hand. If the overall priority lies on efficiency, then UPD is the right choice for the job. Otherwise, if reliability and ease of use is what you are looking for, then TCP is the right tool. We also must not forget that there is also a middle ground that can be attained by just utilizing UDP as the base and implementing any desired features on top of it.

**4. Thick Clients VS Thin Clients:**

This concept describes the two different ways in which the logic and work can be distributed between the client and the server side of a distributed system. Both of these choices come with their own set of benefits and disadvantages as described in class.

In a Thick Client, as the name suggests, the core of the program’s logic is located in the client side of the system. This is ideal for certain applications, such as for example multiplayer games or other applications that are very processing or graphics cards intensive. This allows us to rely on the user’s computer to do all of those computations for us. Nevertheless, this comes at a large cost in the reals of both security and overall control. Having a thick or fat client makes any updates to the code a very complex process. Also, since the bulk of the code resides at the user’s computer, we run the risk of hackers being able to look at our code and even steal some critical logic of our application.

In a Thin Client, the bulk of the logic is kept in our server and the client therefore mainly serves as a UI to present our responses/calculations to the user. This can be very useful for applications where you want to keep your business logic a secret and don’t want to handle many client updates. The security and the updating of this kind of system is superior. Since the client is light, most updates will just be handled on the server side. This choice is overall simpler to code and maintain than its counterpart. There are still security corners due to its reliance on network traffic, but since the code is securely on the server, it more difficult for the entire system logic to become compromised.

**5. BitTorrent System:**

BitTorrent is a hybrid system with both centralized and de-centralized areas. It is commonly utilized to share files across the internet with many other people. It relies on Nodes and .torrent files or trackers in order to both keep track and distribute sections of a given file that needs to be shared. In order for this system to work appropriately, the good behavior of people contributing to the system needs to be rewarded.

Nodes are essentially the people who are connected to this system and they are organized in groups called swarms which as a whole must contain a full copy of whatever file is being targeted for download. The members of this swarm proceed to share whichever chuck of the file each of them has with another and also simultaneously receive whichever chunk they do not have yet.

The .torrent files are useful because they point to servers called trackers which themselves maintain a list of the different node swarms that contain the file you want to download. This system is very effective at rapidly sharing files amongst a big number of users. While it is often utilized and viewed in the context of piracy, the technology itself is innocent, and is a quite ingenious way of decentralizing a lot of the work related the distribution and sharing of files.

**6. Public Key and Private Key Usage**

In the realm of security, the usage a key pair of Private and Public keys is essential to both secure the document you are sending, and to be able to verify the document’s authenticity. Whenever you create a document and need to be able to share it with the world, but also for your target audience to be able to verify that the document is authentic and comes from you, you must use your Private Key. If you encrypt the document with your Private Key and distribute your Public Key through a reliable hosting website, external users will be able to use it to authenticate the files they download and make sure that they are actually yours.

If instead of making a file completely public, you want to send a secret message or file to a specific person, you must use his Public Key. If you encrypt your message with the Public Key, he can then use his Private Key to decrypt it and you will be sure that only he can read this message.

This entire system relies on two core factors. The first one is that the Private Key always remains secure and is never stolen. The second one is that the hosting website or any other means of the distribution of the Public key must be reliable and will not tamper with it in any ways. If this is not the case, the contents of the files can’t be trusted or accurately authenticated.

**7. Synchronous vs Asynchronous calls**

Synchronous calls are exactly just like a regular function call. Whenever this call is performed, the computer has to wait for it to finish executing in its entirety before being able to continue to the next line of code. Programming using this kind of calls is very simple, the calls are easy to follow, and the overall flow of the program is procedural and therefore can be traced down very easily. Nevertheless, having to wait for every call to return could sometimes become exceedingly wasteful.

Asynchronous calls on the other hand, are calls that you make, and you do not have to wait for its execution to complete. In order words, the rest of your code can continue its execution even if this call has not completed its work. Programming in this way is very useful, since it allows your program to keep running in the background and stay responsive even while a given call has not finished executing. However, programming asynchronous calls is very different and therefore may be more difficult to code in general. The programmer may also need to implement listeners and handle interruptions for whenever the asynchronous call actually finishes, and you need to utilize some of its output.

**8. Data Streaming**

Streaming Data is essentially the process of sending a sequence of data chunks in a continuous manner to another computer through a network. This is especially useful for multimedia files such as video and music and it is commonly utilized by streaming services such as Netflix, etc. There are multiple modes in which streaming data can be transferred, such as Asynchronous which does not take time into consideration, Synchronous which respects a maximum delay and Isochronous mode which makes sure the packets do not arrive to early or too late.

Streams can be either simple or complex. In a simple stream, there is only one channel of similar data being sequentially sent. In a complex channel, such as when sending stereo sound or video + sound, the data is being sent via multiple channels that must be synchronized for the media to be played accurately.

While streaming there is always a main compromise between reproduction delays and the quality of the reproduction, since these are both inversely related. The fastest you want to be able to reproduce a streamed media, the lower quality it would need to be, since the main reason for the delay is the buffering of the needed data. The higher the quality, the more you need to wait for the data that comprised that quality image or sound to arrive and buffer.

**9. Blockchain**

The blockchain technology basically consists on creating a decentralized ledger that no one person is able to modify. The entries of this ledger are packed into blocks that are then mined by multiple computers throughout the world. These computers compete with each other in order to solve a puzzle that is called work in order to verify or mine each block. One a block is verified, the computer that solved its puzzle gets credit for doing so. Then this block is added to the chain and the process continues on forever. The work that each block takes to validate is very important, since it makes almost impossible for an evil actor to try to replicate the chain and add fake blocks. The longer the chain is, the more work and time it would take for someone to attempt to do this, and therefore it gets to a point where this is essentially impossible.

This technology is extremely useful since it ensures that the records recorded in the ledger are there forever. For example, companies can use this technology in order to store their financial data. This will ensure integrity of their accounting and literally eliminate many opportunities of internal pilfering. Today, this technology is mainly used for the implementation of cryptocurrencies, such as Bitcoin, Ethereum, etc. They utilize the advantages of this decentralized immutable ledger in order to create a sort of storage of value in the shape of “coins” that can be traded for services across the world. These coins do not rely on any central government and the transactions are both secure and finite.

**10. Encryption and Compression order**

Encrypting data is the process of utilizing a key in order to hide the contents of a file from anyone who does not have access to the corresponding key pair. Without this pair, the file just seems like a string of random data that is impossible to translate or access. Today this is essentially used as one of the main ways to secure communication. Many texting apps even mention that the communication is protected by encryption to assure users that their communication is 100% private.

Compressing is the process in which a compressing program such as 7-Zip looks for trends in the data representation of your file and for empty spaces and trims them down, it tries to represent the same valuable information in a much smaller disk space. Compressing files is one of the main ways to package and store large files. It helps you to waste less disk space and be able to transfer those files much easier through the internet, since they are often much smaller than their uncompressed counterparts.

Both encryption and compression are often combined where there is a need to send a file over a network and you want its size to be smaller and its contents to be secret. Due to the fact that encryption jumbles the file data, it is imperative for the compression to always be done before the encryption. Otherwise, the compression software will be unable to find any patterns in the file and therefore will be unable to save any space.

**11. Data Marshaling**

Data Marshalling is the process of taking data such as objects in a computer’s RAM and reshaping it into a way in which you can send it over the web to a different computer for its consumption. This is often done with the help of external formats that can be easily translated at both ends of the communication, such as XML or CDR. It can also be done with the help of a virtual machine that will serve a middleware communication layer that rests on top of each machine.

There are a series of precautions that need to be taken when trying to marshal data. Sometimes we may try to communicate with computers that have a completely different architecture and therefore it may not be possible to just send our data as-is. In these cases, using XML/CDR or relying on a virtual machine (JVM) may be the best alternatives that we have in order to accurately share our data, without having to explicitly know all the specifics of both of the machines that are trying to communicate at a given time.

**12. Middleware**

Middleware is located in between the operating system, library calls and native instructions of a computer. This layer of software serves as a uniform interface for the programmer. It allows both communication and execution of programs across a vast array of machines with heterogeneous architectures. Java, Sockets, .NET are all examples of middleware.

The utilization of this middleware layer is what has allowed programming languages such as Java to become as popular as they are today. A given program written in Java as an example, can be run in many different devices, ranging from PCs and laptops to mobile phones. They can all communicate with each other without any problems and without having to know anything about each other’s architectural details.

**13. Blocking Queue Usage**

A blocking queue is a data structure that is utilized in order to share data objects safely across different threads. This structure takes the shape of a queue in which objects generated by a given thread can be stored and then accessed by a different thread. This structure was utilized in our class as part of the Blockchain assignment. There, we utilized the Priority Blocking Queue variety of this structure in order to store all the unverified blocks generated by each process. The Priority Blocking Queue sorted these blocks by a specific criterion which was defined as the timestamp of each block.

The utilization of this structure was critical for the Blockchain assignment, since it relied on multiple threads trying to access the same set of data at the same time. Since the Blocking Queue is thread safe, it was able to handle the concurrent generation of new blocks by each process and the consumption of each unverified block by the verification threads. If we did not utilize this structure, the system would run into concurrency issues and there would be a risk of losing access to the data stored in the queue.

**14. Dictionary Attack**

A dictionary attack is a type of brute force attack that is utilized to try to guess someone’s password. It is many times utilized by evil entities in order to obtain access to a person’s account illegally. This is done by utilizing a sorted list of worlds, also called a dictionary, and trying each word one by one as the user’s password. This is type of attack is very efficient since it keeps track of which words have been tried already and can easily guess simple passwords. Nevertheless, more complex passwords that do not utilize complete words as parts of them, or passwords that contain interleaved numbers in them, can be much more challenging to guess with this type of attack. This is one of the main reasons why simple passwords such as “Password” or a specific combination of words are severely discouraged, and many times not even allowed in modern websites.

**15. Password Management**

One dirty secret amongst IT corporations, such as AT&T and eBay that has sparked public outrage in the last couple of years has been the way in which those companies manage the user’s data, specifically their passwords. For a long time, many companies have stored user’s passwords in plain text and sent this unencrypted data across the internet when signing users into their applications or even when sending emails to their users. This poses a huge risk of the user’s passwords being easily stolen by any evil actor that could possibly be sniffing the network packets at a given time. This is especially dangerous, since most people utilize either the same or very similar versions of the same password in multiple accounts. Therefore, if their password for one account is stolen, the evil entity is able to utilize it to log in to other user account’s and steal their identity.

In today’s day and age this kind of password management and transmission is completely unacceptable and often unnecessary. All of the user’s sensitive data, specially his or her password should be transmitted in an encrypted and secure fashion only when completely necessary. As we learned in class, for a user to log in to a website, we do not even need to transmit their inputted password. Instead, we can utilize their password to encrypt a random string and store that result into the database. Whenever a user tries to log in to a website we follow to encrypt the same sting using their input and compare it to the result we have in the database, if they match they are allowed to log in. This way, we never actually have to store their password anywhere, and even if our company’s security is breached, the hacker will never have access to the user’s password.

**16. Code Table**

The utilization of codes in the shape of a code and meaning table is the cheapest way in which we can maintain completely perfect secrecy. Using this strategy, it is impossible to decipher the meaning of each code without actually getting your hands on the table of codes and their corresponding meaning. This strategy consists of having two copies of the table, one at the sender’s side and one at the receiver’s side. When the sender sends a code, the receiver consults his table and looks up the meaning of the code.

There are a couple of very significant downsides to this method. The first downside is that you need to find a safe way to get the table to both the sender and the receiver’s hands. This is essential, since they both need it in order to know the meaning of the code, and if the table is leaked then your communication is no longer secret. The second downside consists on the fact that if the result of the code is observable, then they can only be utilized once. Otherwise, whoever was able to see the result will by default know what each code stands for.

**17. Epidemic Protocols**

Simple and cheap to run algorithms that are initiated by just one node and are spread throughout the network until all nodes are “infected” or have received a given message. An infected node sends a message to its neighbors, trying to infect other nodes. The nodes that have not been infected yet, are “susceptible” which means that they can be infected. Once an infected node is unable to infect any other node, for example when it is surrounded by only other infected nodes, it is removed from the algorithm. Epidemic protocols work in a very similar way to how viruses in real life spread. Comparable to how a person who has the flu, or some other contagious sickness is able to infect people around him, the infected nodes in a network also infect others around them.

This method of spreading messages is very useful for updates, but it is not good for hard removals. Since there is no way to control when each node will get a specific message or update, a removed node may eventually get restored by a pending update. This can be solved by sending death certificates instead of actual removals. This way, the timing of the update will not be able to restore the death (soft removal) of a node.

**18. DHCP**

DHCP or Dynamic Host Configuration Protocol allocates IP addresses to a given machine whenever it is trying to connect to the network. This is very useful because instead of giving each person a new permanent IP address, it allows us to only have a relatively small set number of real IP addresses and allocate them dynamically to machines whenever they are actually needed for connection. This allows a company or school to save money on the number of IP addresses it leases, it also makes it easier to configure connections to the network. Since each machine is provided an IP address dynamically there is no need for each person to manually configure their permanent IP address into their computer whenever they need to connect. The IPs allocated by this protocol are just temporarily loaned to the user, these IP addresses may expire on a variable time basis. One example of this is the IP addresses we are given when connected to DePaul’s network, the IP addresses provided to students on Monday may be reused on Tuesday. Even if the same student connects on a different day, they may get a different IP address allocated to their machine. Unlike NAT, DHCP always provides real IP addresses.

**19. Clock Synchronization**

Since there is no global clock in a distributed system, we need to synchronize the clocks of many machines that are part of the system to accurately keep track of when each event happened in real time. Even when all the internal clocks of the member machines are set to the same origin time, the clocks may eventually vary naturally due to clock skew. Clock skew is a direct result of our inability to accurately measure time in general. Machines generally utilize oscillating quartz crystal to measure time, the differences in the quality and frequency of these internal clocks leads to small differences in time between them that will eventually lead to their times falling out of synch. While the difference in time may seem minute to humans, for a machine that can perform millions of operations per second, this may become critical.

Throughout history, humans have adopted different ways of measuring time in an effort to get closer and closer to an accurate representation of time. This has proven to be challenging because time keeps fluctuating due to the fact that the earth’s spin is continuously slowing down and has led to leap years and more sophisticated time measurement methods being introduced. Similarly, multiple algorithms and tools exist that allow systems of computers to attempt to synchronize their time measurements within a small window of tolerance, such as the Berkeley Algorithm and the Network Time Protocol.

**20. VPNs**

VPNs or Virtual Private Networks are a way for users to connect to a specific intranet remotely through their regular insecure internet service, it can also be utilized to achieve a secure connection between two separate intranets. This is based on the principles of IP security. It consists on encrypting and decrypting data on both sides of the conversation. Both the local computer and the intranet server encrypt the data packets they send and decrypt the data packets they receive with their respective private and public key pairs. This way, even though the communication is happening through an insecure internet service, the data cannot be read by anyone else but these two parties. VPNs are relatively simple to implement and the data protection that they offer is exceptional. This is very useful, since it allows workers to access their company’s or school’s intranet from any part of the world as long as they have an internet connection.

**Forum Posts**

**Hello Thread**

**1)**

[Fernando Araujo](https://d2l.depaul.edu/d2l/le/688219/discussions/threads/2594712/View?searchText=Fernando+Araujo+Chavez)

Fernando Araujo Chavez posted Apr 6, 2019 1:07 PM

Last edited: Saturday, April 6, 2019 2:02 PM CDT

Hello everyone!

My name is Fernando Araujo and I am a full-time MS in Product Innovation and Computing student here at DePaul. I am 26 years old and I was born in Mexico, so I am fluent in both English and Spanish, but I am also working on getting my Japanese up to par.

I am currently taking only online classes, since I am taking Japanese classes here in Tokyo, Japan at the same time as working on my Master's coursework. This means that I will be posting things and watching lectures at very odd hours haha. Before enrolling in DePaul, I worked as a Software Engineer in downtown Chicago for 4 years. I was a full-stack developer mostly focused on Java. I developed and provided support to a very popular higher education effort reporting application.

For my undergraduate studies, I graduated from UIUC with a B.A. in Global Studies which a concentration in Economic Development and a focus on East Asia, I also obtained a minor in Computer Science. Since I studied Japanese during my college years, I decided to come to Tokyo again and spend some time improving my language abilities.

I decided to join DePaul since it seems like a pretty good school. Most of the classes I have taken have been very well taught and the information I learn has been very practical. By the end of my 2 years program, I wish to find a position in Mobile Development, Startup or IT Project Management. My goal is to work for a company where I can not only use my technical abilities but also my language and cultural knowledge. If I could find a good paying job at an international company in Tokyo it would be great!

With regards to my personal hobbies, I love cooking (mostly Mexican and US food). I also like Anime, videogames, etc. My craziest goal is to open an Authentic Mexican restaurant in the middle of Tokyo. I also have a couple of ideas for Mobile Games, if anyone is interested we could talk about it and maybe even work on making something together!

Nice to meet you all!

**2)**

April 9 at 2:37 PM

Hello Janet,

Yeah, it really is. I really like Japan so I have studied here before during my Bachelor's but I really wanted to come back again. Since I joined DePaul and they offer a lot of online classes it was easy for me to do so once more. So, if you are interested, I believe you could do the same in Sweeden too!

How was your time in Sweeden? It sounds like an interesting country to go to. What kind of internship did you do?

**3)**

April 9 at 2:43 PM

Thank you Silviya! I am not a photographer or anything but I do like taking pictures of things/places I find interesting.

Arranging exams abroad is relatively simple. I believe you can find proctoring services in most countries' big cities. So instead of taking them at DePaul for example, you can make an appointment with a proctoring center and then register the appointment date via D2L. You input the proctoring center's information in the D2L exam form and its done. The exam will be sent to the proctor for him to administer it at the appointed time.

If you are unable to find private proctoring services, public libraries or other universities can also help you to proctor those exams. DePaul has made studying abroad quite simple and I would really recommend it!

**4)**

April 9 at 2:57 PM

Hi Raquib,

I am glad you liked my pictures, thank you!

With regards to visiting Japan without any Japanese language ability, it is relatively easy to do so if you plan to visit some of the biggest cities or most popular touristic spots. Tokyo, Osaka, Kyoto, Mt.Fuji are all often visited by foreigners so the local population and businesses have learned enough English to help them around.

People here, in general, are very accommodating to tourists. They try their best to help you out. Nevertheless, a lot of aspects of Japanese culture are quite different so it is best to study it a bit before coming here. This will greatly help to smooth your stay.  Also learning a bit of Japanese, enough to say basic phrases such as "Hello", "Thanks", etc. will go a long way.

If you have any specific questions you would like answered about Japan or you need recommendations on what to do and where to go, just let me know!

**5)**

April 21 at 5:28 AM

Hello John,

It seems like you have a lot of experience in East Asia. It sounds very exciting!

I am currently living in Tokyo while doing my Master's and studying Japanese.

Which part of Japan did you stay for your study abroad? Please let me know if you have any tips or recommendations!

Good luck in class!

**6)**

April 21 at 5:37 AM

**Fernando Araujo**

Hello Patrick!

I am happy to find a fellow student majoring in Product Innovation and Computing!

What do you think of the major so far? How long have you been at it?

Do you think it is risky to have a major that is so new?

I personally have really enjoyed some of the flexibility of this major, but I wish DePaul offered more on-line Management courses. I am currently in Japan, taking online classes, so I have mostly taken the CS/Technical requirements.

Also, I will make sure to try that Ramen place you recommended while I am here!

Good luck in class!

Fernando

**JokeServer and InetServer Thread**

**7)**

April 21 at 5:45 AM

I have so far done almost everything in the server. The only thing I use the client for is to generate a UUID, ask for the user commands and communicate them to the server. Then just wait for the server response in the shape of a joke/proverb.

Keeping it server-side, I believe is ideal for this assignment. The data that you are communicating back to the client is very small, just a string. Also, unlike a game, there is also no heavy workload that needs to be handled on the client side.

Fernando

**8)**

April 21 at 5:58 AM

Yes, I believe that for example:

For userA - you first get Joke A (still have B/C/D)

Then you switch the mode to Proverb mode

For userA- now you will get Proverb B (still have A/C/D)

if you keep switching the mode back and forth, it should give remember which Jokes/Proverbs have not been used per user. Therefore:

If you switch back to Joke mode

For userA - now you will get Joke C (still have B/D left)

I hope I was able to answer your question! Let me know if you have further questions.

Fernando

**9)**

April 21 at 6:02 AM

Last edited: Sunday, April 21, 2019 6:03 AM CDT

Wow, that sounds like a ton of work!

The easiest way that I can find, is to just put your 4 jokes into a collection and literally just call the Collection.shuffle() method. It may not be the most efficient, but it is by far the laziest way that you can go to achieve the task of randomizing the jokes.

Greetings

Fernando

**BlockChain Thread**

**10)**

June 2 at 6:17 AM

Yes, I did something similar.

Created the compareTo method and a Comparator class for my blocks in order for the PriorityQueue to work properly. I ended up comparing the timestamps in the blocks only when both blocks being compared were not null.

**11)**

June 2 at 6:15 AM

My response may be a little late, but maybe it helps to compare different implementations for the future.

I personally just created a very simple object that contained the public key and the process number. I stored those objects in an array. So it was very simple and quick.

**12)**

June 2 at 6:13 AM

Yes, I completely agree with you Tylor and many other people in this thread.

This assignment was actually quite challenging and took me a lot longer to complete than any of the previous ones. There are parts that I am still not very sure if I did correctly.

At the same time, I did really enjoy exploring a bit about the implementation of block chains. I only wish that we ventured a bit more into the actual usage of this technology front the user side. I personally think that this is one of the main reasons why bitcoin and other applications of this technology have not become mainstream. There is no user-friendly application for this yet out there.

**General Postings Thread**

**13)**

# **Blockchain Projects**

Fernando Araujo Chavez posted Jun 2, 2019 6:25 AM

Hello everyone!

So now that we are all done with the BlockChain assignment I wanted to ask for everyone's opinion about the BlockChain technology as a whole.

I think that we all know about cryptocurrencies and their boom in the past few years, but other than this one specific usage, I have not heard of many useful projects actually being developed that utilize this technology. I have heard of a few game/collection gimmicks such as Cryptokitties (which is not really a game), and similar projects but I have not heard of anyone actually fleshing out their idea in a useful and user-friendly way. Does anyone know of any examples of useful and practical projects that people are being developed? I for myself believe that there are a ton of uses for this technology that have not been exploited so far and would really like to see what people think.

**14)**

June 6 at 1:33 PM

Talking about utilizing the blockchain to keep politicians honest, I actually have been thinking about a very practical use for this technology for over a year now that not many companies are targeting. I believe voting, especially for important government elections could be made much more transparent with the use of blockchain.  
  
While I have found some companies that are trying to do this in the US, such as the ones mentioned in the following article, I believe that the main target should be more unstable governments where fraud is a more prevalent phenomenon.

<https://medium.com/coinbundle/using-blockchain-for-voting-3287817291dc>

For example, I believe that utilizing blockchain technology in countries like Mexico, where the theft and burning of paper ballots are prevalent in literally every single election could do wonders for both its democratic process. Being able to cast a vote digitally into a decentralized and immutable ledger could radically increase the confidence that the general citizen will have on the nation's institutions. Getting an actual receipt or code that you could then utilize to look up your actual vote could not only ensure citizens that their vote was counter but it will also discourage the purchasing of votes by the drug cartels and unscrupulous politicians. I honestly would love for such a system to be adopted in places that really need it like Latin America, Africa, and other developing nations.

**15)**

June 2 at 6:46 AM

I personally believe that this is not a practice unique to big companies at all. From my experience, a lot of companies of all sizes do this still today. I know this from experience and from friend's anecdotes.

The problem is that most times this is done out of "convenience". It is easy for technical support to be able to look up someone's password and replicate whatever bug is happening to a specific user. There isn't often a will to change the status quo or to implement ways to emulate users without logging in as one since this takes both developer's time and money.

This is also often the case with passwords in general. Database passwords, server passwords, you name it. I have seen them stored in a word document at a shared folder or the company's intranet. While this is a horrible practice, it does not surprise me in the least and until a big hacking incident happens to them I don't believe that a lot of companies will change their ways.

**16)**

June 2 at 6:55 AM

I believe that similar to many other creative industries, game development is a very underpaid and overworked career. I think this is related to the whole mentality of "Well as long as I am doing what I love, I don't mind earning less". Which, while I understand it, I believe it is a major reason why game developers are underpaid, to begin with.

There needs to be a push for game developers to be recognized as what they are, highly skilled developers. Yes, while they may be working in a very popular industry, it is not like they are not bringing in millions in profits to their companies. Companies may think that they are easy to replace and that there are many other "geeky gamers" who will easily replace them if they ever complained, but I honestly believe that if there was a well-organized protest or strike during the production of a major game, things may begin to change.

**17)**

June 2 at 6:35 AM

Thanks for posting this Arynn.

I believe that the strategies detailed in your linked articles are very useful for any programmer to take into consideration and try to adapt to his or her programming routine.

I personally feel quite frustrated whenever I try to look for a bug for hours upon hours only to find out that it was caused by a small typo or some other minor mistake. This actually happened to me during the last BlockChain assignment.

I thought I had everything 80% done but once I proceeded to test it my signatures and hashes did not match up. I spent more hours than I can admit staring at the code and adding logging in order to find out that I was not verifying my signatures correctly.

I always tell myself to take a break and go out for a walk whenever I get frustrated with coding. But the reality of the matter is that many times when confronted with deadlines it does not feel very productive to go for a walk or rest when in order to try and solve an urgent issue.

**18)**

# **University IRB Training**

Fernando Araujo Chavez posted May 15, 2019 9:29 AM

I am currently doing the IRB Training as part of one of the extra credit projects. Other than it being quite interesting I really wanted to share with you an article that highlights the importance of both this training and all the regulations related to human research that are in place today.

The article is about the Stanford Prison Experiment that took place in 1973. It is briefly mentioned in the training module 4. It is a very interesting case of what happens when a human research experiment goes wrong and wanted to share it with you all.

<https://www.simplypsychology.org/zimbardo.html>

**19)**

April 21 at 6:08 AM

Pedro,

I 100% support your post. GitHub is an amazing asset for any developer out there. It is not only a professional tool that you will most likely utilize at some point during your career, but it also encourages good coding habits and helps you build a nice portfolio of projects to talk about during your next interview.

I honestly wish I had known more about GitHub and its importance earlier in my career. Even today, I still struggle to keep all my projects up to date and committed to my personal GitHub account.

Great post!

**20)**

April 21 at 6:24 AM

James,

While I agree that the blockchain and its main feature of decentralization are extremely powerful tools, I believe that we have yet to learn how to harness its power effectively.

With the rise of Bitcoin, Litecoin, Etherium, and many many other coin offerings, the dramatic rise and fall of their prices and the overall craze that this has caused, the image of the blockchain has been somewhat tainted.

While I certainly believe in the power of this technology and even have some ideas that could benefit from its unique features, I have not seen many successful projects come out of it (other than just speculative assets/new coins). Today many of the projects, are either meaningless or overhyped. Just look at <https://www.cryptokitties.co/>

I also think that due to the recent craze/crash of the overall crypto scene, the popular image of the Blockchain has gone from "The cool new thing" to more of a "Get rich quick pyramid scheme". While looking at blockchain in this way is overly simplistic, popular opinion does impact the willingness of companies to invest in this technology.

What do you think?

Fernando

**21)**

[Study Log - Forum Posts Format](https://d2l.depaul.edu/d2l/le/688219/discussions/threads/2674384/View)

Fernando Araujo Chavez posted Jun 6, 2019 12:45 PM

Hello,

I have a question related to the Study Log.

I know that as part of the second section of the log, we are supposed to add all of our contributions to the Forum posts.

Is there a specific format that is expected for this? We are instructed to just add our original content, but should we also include the date of the post and the thread/post title? Or is this just irrelevant information?

Hopefully, someone has some input on this before the deadline.

22)

I turned in the IRB Training, but I still do not see the grade added.

Hopefully, it isn't just me.

-Fernando

**Free-form Study Log**

**Distributed Systems:** A group of multiple computers or other machines that interact with each other and work together for a common goal. When the users interact with this, they interact with the system as a whole.

Desirable characteristics - shares resources, flexible, scalable, robust and secure

**Transparency:** How much communication or functionality of the system is visible to the client

* **Access –** whether the user knows which specific part of the system they are accessing
* **Location –** whether the users knows the physical location of the system or its data
* **Concurrency –** whether the user is aware of multiple users sharing the same data at the same time
* **Replication –** whether the user data is replicated somewhere else, before it settles
* **Failure –** whether the user knows when there was a failure within the system
* **Mobility –** does the user know where exactly the system is located and if it has moved to a new place in the past
* **Performance –** whether the physical configuration of the system has changed, power, etc.
* **Scale up –** whether the size of the system has changed, it has grown.

**Open Systems:** Publicly available system with its own interfaces (ways of communicating with the system). People can contribute to it freely with defined interfaces but open and flexible implementation. A big example of this would be Linux

**Interface Definition Language (IDL) –** Language used to specifically define how an interface should be structured. It tells us what functions, parameters, values, etc. should be utilized in the interface. IDLs should be fully defined, stable, public and not under centralized control. They encourage development collaboration and mobility.

Disadvantages – Many times very expensive to start, hard to encourage companies to invest. Often very complex with multiple policies.

**Fully Distributed Algorithms –** Algorithms that are decentralized, spread around multiple machines without any global clock. The member machines only have access to local information and even if one machine dies, the algorithm can continue.

Disadvantages – Most time very complex, often less efficient and more prone to failures than non-distributed algorithms.

**Inet Assignment lessons** – Learned how to utilize sockets to both communicate (wait for connection and listen) with other processes using worker/listener threads. Basic foundation of the Server/client implementation for JokeServer.

**Synchronous Calls**

* Waits for response – blocks the process until it finishes
* Good for individual machine – if procedural programming
* Very inefficient – specially when in a distributed system, since waiting for responses from other members could be a huge waste of processing time.
* Network speed dependent – even if other machines in the network respond quickly, the speed of the network could be very costly
* Blocking failures – if a message is lost or the machine that you are waiting for goes down, you can block any other calls from happening.
* Very simple to program

**Asynchronous Calls**

* Much harder to program
* Does not wait for a response
* Great for networks – the network speed does not matter as much, since the main process is not locked waiting for a response
* More efficient – since it does not have to wait for a response, it can keep working.
* No Blocking – running independent of actual response

**When Should we use Synchronous or Asynchronous?** Depends if we want to make a tradeoff. Do we want to introduce more complexity to the system in order to be able to do more work, and don’t idly wait for a response? (**Toolbox**)

**False Assumptions of Distributed Systems**

* Reliable Network – Not compared to a local machine
* Secure Network
  + No, they are not secure, anyone could be sniffing the network at any given time
  + Never send sensitive data in plain text form
* Homogenous Network – not everyone is running the same hardware across the network
* Unchangeable Topology
* Latency is zero – No it is not, a packet may take longer depending on the location and routing that it takes to get there. Can create a lot of synchronization issues
* Infinite Bandwidth – No, with a lot of people using the same network, the bandwidth is quite variable.
* Zero Transport Cost – No, now a days with the repeal of Net Neutrality, ISPs may start charging different fees based on the kind of data you send, how much, how fast.
* One Administrator – No, there could be a multitude of administrators

**Transactions**

* There is always a before and after state, never anything in between for the user. (For example: Withdrawing money)
* Critical Section: Is the area of code that needs to be locked for data to not be overwritten or modified before settling a transaction.
  + Critical-Section problem – making sure that multiple threads do not access the same data simultaneously, otherwise the transaction results may be compromised.
  + Semaphores – used to regulate thread access to the critical section. One at a time once each transaction is settled.
* Roll Back – ability to go back to the previous state, in case there was an error in the transaction
* Commit – When a transaction is verified to be correct, committing finalizes it and gets rid of the option to roll back.
* Checkpoint – A point in time that can be saved, the system as a whole can be rolled back to this time. This will undo all transactions done after the checkpoint.

**Transaction Characteristics**

* Atomic: A transaction does not have any steps in between. There is just a before and an after
* Consistent: Transactions follow the system rules or invariants. They can’t settle on an invalid value.
  + Example of an invariant: final result can’t be a negative number
* Isolated: Two transactions running at the same time do not interfere with each other. This is why semaphores and locks are used in the critical section.
* Durable: After a transaction is committed, it cannot be rolled back. All of the changes made by this transaction are there to stay.

**Processes –** Running program

* Can be run locally
* Can be run through the network
* One program can start multiple processes (like browsers)
* Process instructions are run by the CPU sequentially one at a time
* They can use the available resources like memory, ram, etc.

**Shared Memory** – This is the way in which locally processes can share information with each other. They just share the same memory space and this way they can read and write data in that shared memory and use it together (Inter-Process Communication).

For distributed systems, the concept of shared memory has not been reliably implemented. Therefore, processes are unable to access memory that is being used by other processes through the network. Instead, we utilize messages in order to synchronize and communicate between processes.

**Atomic Action –** Actions that can happen simultaneously without leaving the CPU.

* Test and Set instructions for local systems.
* Distributed systems do not have this, instead they need to use monitor processes to handle shared resources.

**Distributed Shared Memory Systems –** The ideal, the holy grail for distributed systems. If we could map the memory of all the machines in the system together via a middleware, then we would be able to use IPC on distributed systems. Nevertheless, so far this is just a dream and trying to implement this has been unsuccessful. Too slow, buggy and inefficient.

**UUIDs (Universally Unique Identifiers)**

* Completely unique string or number that no one else will use
* They are often used as addresses to a specific file, location, computer,etc. (Like URLs)
* Using very large numbers that are randomly generated could often suffice depending on how much data we are handling.
* Always keep a one to one relationship (no two people have the same UUID)

**JokeServer Lessons –** For the JokeServer assignment I was able to implement both a client and a server by utilizing the foundational code from the InetServer. This assignment centered itself on teaching us the tradeoffs between a fat client and a fat server and how this plays a role in preserving user data/state. We were given the freedom to choose whichever strategy we wanted. I personally chose to utilize a fat server. I believed that keeping most of the data and logic centralized would be much easier to code. I really enjoyed this assignment and will look back to it whenever I need a refresher on client/server design and communication.

**Messages –** the way in which we can communicate between processes in a distributed system

**Problems with Messages**

* Buffers are full – receiver can’t get the message because its too busy
* Lost message – packet that contained the message doesn’t arrive
* Acknowledgment is lost – packet sent back in order to acknowledge the receipt of a message never arrives, leaving the sender waiting or having to resend it.
* Delayed message – may take too long to arrive
* Wrong order – may need to be buffered and reorganized before its consumption
* New Process joins - it may need to be updated with the current system state via messages

**Send and Receive**

* Both sender and receiver need to be up and running at the time of communication
* Both sender and receiver need to know each other’s endpoint or address + port
* Need to make sure the buffers are the right size to handle the message volume
* Message Oriented Middleware – may be useful to help with redirecting messages or temporary storing them when the receiver is done.

**Electronic Health Care Systems –** Special system that handles very sensitive personal data. This comes with unique challenges in the shape of confidentiality, data storage, privacy and design. The main design question consists on whether we should store the data logically or remotely. This then will dictate the kind of measures we need to take in order to protect this sensitive date. How secure it should be, how robust the system and the alerts it communicates should be, etc.

Keep it in the person vs centrally stored?

* In person is good for hospital, since they can deny liability
* Centrally stored would be a lot more useful, and hospital would take care of the data better

Even though a lot people think that Fax is safe, this is completely false. Any encrypted email would be much safer than fax.

**Sensor Networks –** A network of sensors that communicate with each other and try to compile data. These can be organized in various ways. We need to consider the battery life available, the possibility of data aggregation and dependence on few bigger sensors that can be in charge of transmitting the data to our central storage system.

**Distributed Hash Table –** Distributed System Structure that allows for members to join or leave the group at will. The data is spread amongst the nodes based on their provided node id. It is very useful for organizing machines in an fashion independent of their location. Extremely general purpose, such as hosting a lot of data, this helps spread the burden amongst all the members of the DHT. (**Toolbox**)

**Bit Torrent (Toolbox)**

* Sharing data amongst peers
* Nodes organized in swarms exchange chunks of the desired file
* The complete file should be found within a swarm
* .torrent /Trackers – lists of active nodes with a given file
* Often used for piracy but the technology is great
* Leechers: Members who don’t upload just download
* Seeders: Members who keep contributing to the swarm even when not downloading
* KEY POINT: Reward good behavior for the system to work properly

**Middleware** – layer that is between the programs and the operating system. They offer a uniform interface for the programmer to utilize. Examples include Java, Sockets, .Net, etc.

**Message Protocols** – compromise between reliability or efficiency

Send and Forget – most efficient but least reliable (UDP Universal Datagram Protocol), we send and won’t hear back if it was received or not. (horrible for sensitive communication if a packet is lost)

Send and Confirm - TCP – there needs to be confirmation of arrival and if didn’t we will resend it. We will also buffer and reorder packages that got there in wrong order. But it is less efficient.

If we don’t care if we lose a few packets but we need them to arrive in a specific order – maybe a bidding protocol, very reliable in one aspect but we don’t need guaranteed delivery. We don’t always need ALL the features of TCP.

Don’t care about execution efficiency – TCP/IP most reliable and emphasize programmer efficiency (XML) (easier to debug)

Care about execution efficiency – UDP and implement whatever you need.

Marshall all your data as XML, but then in production you can translate it into a single binary string and send it over. This would be a nice compromise since you are sending more data in smaller space.

**HTTP** – connectionless protocol, just connect get response and disconnect

**TCP/IP** – connection holding protocol, keeping the connection going for months

**Why take a connectionless protocol (HTTP) and run it on top of a connection protocol (TCP/IP) ?**

Its stupid, but it is also brilliant – Setting up a TCP/IP connection is very easy, so setting that connection up and then sending HTTP over it became the go to thing.

**Security -** The longer the key the better the encryption

**Symmetric key encryption**

* Most common type of encryption, used often for passwords
* It is less costly than Public/Private keys, faster, and still very good
* The workhorse of cryptography – used whenever you don’t need a public key
* There is only one key, so this is difficult to handle. May need to be sent encrypted or in person.
* (You would have to send the key encrypted with a public key, decrypt it and then use the symmetric key to decrypt the data)

**Signing** – using a private key to encrypt something, sharing the public key, that key works to open the file it means it is signed by you.

The source of the public key has to be reliable for the system to be trusted. Also your private key need to be kept safe. (This can be solved in a business setting, if you lose the key you would have to pay an agreed upon money)

To verify that a document has not been changed, you can hash the original document. Compare the hash between both of them and if they are the same then it has not changed.

You can both sign and verify no changes by signing the hash value of the file with your private key. (this doesn’t encrypt the document though, but it is faster)

**Public vs Private Key (Toolbox)**

Public key: key you give to others to decrypt

Private key: key you keep yourself to encrypt

If you can use the public key and decrypt the file into a readable version, then it is the real file. This means that the only person on the world who could have encrypted it was the creator with the private key. This means the file has been signed.

The only problem is that you have to trust that you are using the “real” public key from the original creator.

**Threads** – they share the same process, so they share memory and if one goes crazy it can kill all the rest.

**Process Control Block** - is much more expensive than a Thread but it also provides a lot of protection. If one goes crazy, it doesn’t affect the rest.

**Lightweight processes** = Kernel level threads

Lightweight processes are cheaper than full processes and can be assigned to a user level thread to offer more protection.

**Dispatcher/worked model** - Jokeserver is an example of this. You get a request at the dispatcher and create a new worker to handle the work of the request for every connection.

**Networked User Intefaces –** Apps in two machines talk to their own middleware, local OS and send things through the network.

**XWindow System** – supported full screen interface back in the day, have one application that manages all the windows UI, multiple servers and a display server.

The application client has to become the windowing server and the application server becomes the windowing client.

**Daemons** – process that runs on its own (handles communication and then sends the client request to the server). The advantage is that the Endpoint of the server does not need to be hard coded and the request can be therefore forwarded to different servers or load balancing.

**Superserver** – Program that checks if the server is running, if it is then it send the info back to client for it to connect to the server. If it is NOT running, it starts the server first. (Advantageous for the cases when running a server is expensive and it is better to turn it off when not in use)

**Quality of Service –** Manage the quality of streaming

Packets take some time to arrive. For the user to not realize this delay we use the buffer to wait a bit before reproducing the media. The experienced delay seems much smaller. (Compromise wait to buffer to prevent interruptions once starting to play)

The bigger the buffer the least interruptions. To make sure there are no delays, you would need to download the ENTIRE file, that’s the only way.

(Netflix recalibrates sometimes and degrades the signal to prevent service stoppage)

**Interleaved transmission** – Helps avoid big degradation of quality even if you lose a package. Sending data out of order, this way you don’t lose a big chunk of continuous data and it is less noticeable during playback.

Compromise: you need to download and store more packages before start playing

**Blockchain**

* Public ledger that can’t be changed by one person
* You need work like hashing by guessing the seed so it meets a criteria
* Race against time, to solve work before other nodes
* These computers are not doing anything useful other than generate heat. Uses same energy as the country of Ireland.
* If you use the previous answer as part of the next hash, it is extremely difficult for anyone else to try to recreate or fake your chain.
* Hash together the previous block hash + the current block’s data + the seed to make the chain very hard to fake.

**One time pad** – byte padding that serves as a way to send perfectly secret messages. But each pad can only be used once on a given message otherwise you run the risk of creating a pattern that can be deciphered. The more bits you pad, the better the encryption. Pad it with a random bit string that is the same size as the message itself.

**Epidemic**

**Calculating average of network** - In order to calculate the average value of the network, each node exchanges their numbers with a neighbor. They take both of those numbers and divide them by two, after this both have the same averaged value. This keeps going throughout the entire network, until all of them get the same averaged value. This final value ends up being the exact value of the entire network. The more time you allow the network to calculate this, the closer and closer the networks values will get to the average value.

**Calculating the size of the network** – In order to calculate the full size of the network, every node gets a 0 except for the first node which gets a 1. Then each node keeps taking the average of their value and a neighbor’s value and divides them by two. This keeps going throughout the entire network until all nodes come up with the same value. By taking the inverse of this value, you get the exact size of the entire network.

**Dynamic Host Configuration Protocol (DHCP)** – It serves to allocate dynamically on a need basis REAL IP addresses to any new members of the network. Allows you to recycle IP addresses and connect a lot of users with a limited number of real IP addresses available.

**Network Address Translation** – provides fake NAT IP addresses, but these addresses can’t be used for the internet, just for local networks. Mostly used at homes or offices to connect multiple electronics to the web via a router.

**How Network Address Translation Works** – Routers use this to translate the fake NAT IP addresses into the real IP address. Puts it into a table and waits for responses from the internet before transmitting it back to the user’s computer. This allows multiple machines at the user’s home to connect to the internet, all having different NAT IP addresses, but all getting internet access through the one IP address the router utilizes to access the internet.

**Virtual Private Networks (VPN) –** utilize IP Security to establish secure communication

* Based on end to end encryption
* Allows connections between multiple intranets, or remote connection to a intranet
* Basically grabs data packages and stores them within encrypted data packages before sending it.

**Clock Synchronization**

* Big issue for distributed systems synchronization and timing
* Clocks across the world are too inaccurate
* New ways of measuring time are being considered (current Cesium 133 standard is not enough)
* Berkeley time algorithm
* Lamport logical clocks
* Vector clocks

**Bluetooth –** Used to connect low power electronics

* Uses 2.4ghz frequency, which is full of interference
* Uses Adaptive Frequency Hopping – Switching 1600 times per second between sub-band to handle this interference
* Easy to lose connection, depending on current traffic
* Focus on saving energy, this makes it take a while to connect sometimes.

**Distributed Systems and Humans**

* Most powerful processor known – Human brain
* Bringing the human brain online would overload our current networks / processing power
* There is evidence that humans can see some seconds into the future – this is a hard thing to code for (Daryl Bern’s studies)
* Graphene based devises – being used to make contact points and take readings or stimulate primate brains