The intriguing future of blockchain enabled IoT sensor network

Project Workbook

By

Chia-yuan Kuo
Ying Lin
Yunli Wang

July 6th, 2017

Advisor: Hungwen Li

Chapter 1. Literature Search, State of the Art

IoT in agriculture

Internet of things, which offers connectivity between people and objects, is considered to be the new era of network. IoT applications are built in many industrial domains such as healthcare service, food supply chain, safer mining production, transportation and logistics and firefighting.[1] Food supply chain is in the field of Agriculture, which is one of the main fields people seek to improve by IoT.[2] Many farmers have already been benefit from smart farming. [3]Despite of the boost investment in IoT, it is still in its early stage, challenges with regard to security, effective collaboration, cost of implementation are not fully explored in existing researches[4], which would be potential threat to it future process. Some of the challenges are related to the common server-client model. If the server were susceptible, a large number of devices would be influenced.[5]

Blockchain

Since Satoshi Nakamoto proposed Bitcoin in 2008 to the public[6], Bitcoin become more and more popular and be invested by people world widely. As a successful decentralized cyptocurrency, its underlying mechanism which is now called blockchain is explored by researchers. The technology is revolutionary because that it solved two problems: “First, it provided a simple and moderately effective consensus algorithm, allowing nodes in the network to collectively agree on a set of updates to the state of the Bitcoin ledger. Second, it provided a mechanism for allowing free entry into the consensus process, solving the political problem of deciding who gets to influence the consensus, while simultaneously preventing Sybil attacks.”[7]

Based on its concept, blockchain is immutable, transparent, and redefines trust, enabling secure, fast, trustworthy. Therefore, it is potentially perfect solution to overcome problems in the IoT, especially challenges in the IoT privacy and security.[8] Many researchers have been attempted to improve IoT systems by introducing blockchain in. For instance, Feng Tian presented an agri-food supply chain traceability system based on RFlD & blockchain technology. [9] Huh, S proposed a way to manage IoT devices using Ethereum, blockchain computing platform.[6] Kamanashis B and Vallipuram M offered a way to secure data communication in a smart city based on blockchain used security framework.[10] It is no doubt that Blockchain has huge potential to change people’s future life revolutionarily.

State-of-the-Art Summary

Ethereum

Although blockchain attracted people to develop new application of it, it has some limitations: it takes ten minutes to generate a new block and it is not Turing complete that it cannot run loops. [8]Considering the restrictions, Ethereum is now vastly accepted approach to build advanced applications on top of cryptocurrency.

Ethereum is a decentralized platform built upon the concept of blockchain. Ethereum's ledger stores turing-complete programs in the form of Ethereum Virtual machine bytecode. So developers can write Turing-complete programs more flexibly in the way of creating their own arbitrary rules. Once contracts is built in the EVM code , it is independent of owners. Figure 1 shows a overview of the workflow in the Ethereum network.

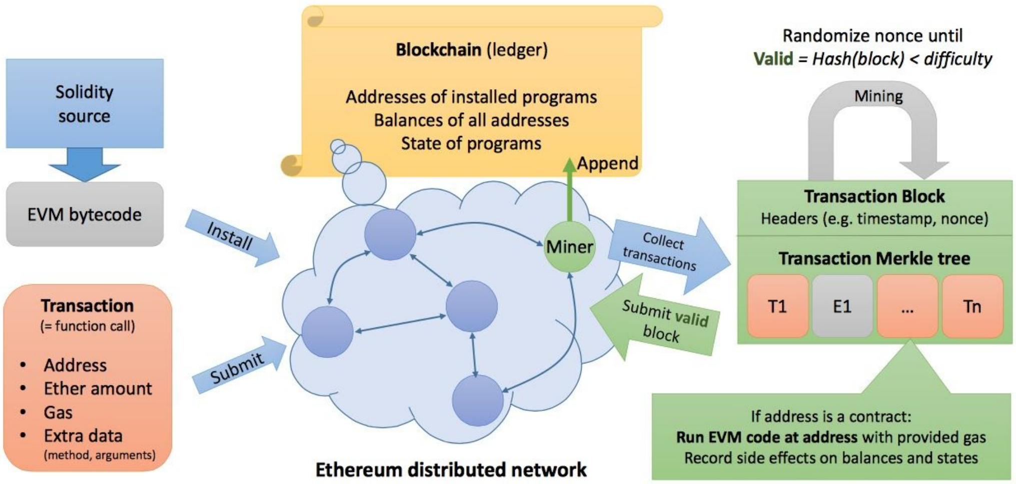


Figure 1: Overview of workflow in the Ethereum network[11]

As shown in figure 1, Solidity is the most popular JavaScript like language to write smart contracts. Solc is used to compile contracts written in Solidity language. With Solidity, developers can create smart contracts for voting, crowdfunding, blind auctions, multi-signature wallets and more. [12]

As the internal cryptographic token of Ethereum, Ether is designed to pay transaction and computation fees for transactions.

Executing and storing programs on Ethereum network all consume gas, which is converted from Ether. This mechanism is used for guarantee the efficiency of transactions.

In the world of Ethereum, a smart contract based application is called Decentralized App(DApp). DApp aims at making developer's contracts more friendly. DApp can be run on both on a central server and local computer.

References

[1] Xu, LD., He, W., & Li, S. (2014) Internet of Things in Industries: A Survey. IEEE Transactions on Industrial Informatics. (10)4. 2233-2243. doi: 10.1109/TII.2014.2300753

[2] Ramundo, L., Taisch, M.,& Terzi,S.(2016) State of the art of technology in the food sector value chain towards the IoT. Proceeding of 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI).1-6.doi: 10.1109/RTSI.2016.7740612

[3] Perera,C., Liu,C,H., & Jayawardena, S.(2015). The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey.IEEE Transactions on Emerging Topics in Computing.(3)4.585-598.doi:10.1109/TETC.2015.2390034

[4] Nukala, R., Panduru, K., Shields, A., Riordan, D., Doody, P., & Walsh, J. (2016) Internet of Things: A review from 'Farm to Fork'. Proceeding of 27th Irish Signals and Systems Conference (ISSC). 1-6.doi: 10.1109/ISSC.2016.7528456

[5] Huh, S., Cho, S., & Kim S. (2017) Managing IoT Devices using Blockchain Platform. Proceeding of19th International Conference on Advanced Communication Technology. 464 – 467. doi: 10.23919/ICACT.2017.7890132

[6] S. Nakamoto. (2009) Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf

[7] Ethereum White Paper. retrieved July 5, 2017 from https://github.com/ethereum/wiki/wiki/White-Paper

[8] Dorri, A.,Kanhere, S S., & Jurdak, R.(2017) Towards an Optimized BlockChain for IoT. Proceeding of 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI). 173-178. PA, USA. available in IEEE Xplore Digital Library.

[9] Feng,T. (2016) An agri-food supply chain traceability system for China based on RFID & blockchain technology. Proceeding of 13th International Conference on Service Systems and Service Management (ICSSSM). 1-6.doi: 10.1109/ICSSSM.2016.7538424

[10] Biswas, K., & Muthukkumarasamy, V. (2016). Securing Smart Cities Using Blockchain Technology. Proceeding of 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference

on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS).1392 – 1393. doi: 10.1109/HPCC-SmartCity-DSS.2016.0198

[11] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., …, Zanella-Beguelin, S. (2017). Formal Verification of Smart Contracts: Short Paper. Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security. Vienna, Austria. 91-96. doi:10.1145/2993600.2993611

[12] Solidity develop. (n.d). Solidity. Retrieved July 5, 2017, from https://solidity.readthedocs.io/en/develop/

Chapter 2. Project Justification

As discussed in the previous chapter, researchers have been focusing on IoT and blockchain separately for a while but are only starting to combine the benefits of these two technologies together. Therefore, the information we could find in this topic is somewhat limited. Inspired by the huge potential of these two technologies, we think it would be a really good opportunity for us to explore the possibilities and move the frontier.

We think that by combining IoT and blockchain together, we may be able to take advantage of both of them. The major benefit of IoT is the flexibility and diversity of devices available for our disposal. Devices range from computing devices to small sensors. However, managing these devices can be challenging. Existing architecture relies heavily on centralized or cloud-based server to manage devices and facilitate their communications. This creates a problem that when the server becomes unavailable, the devices may lose functionality. And we may not be able to add, remove or change devices during server downtime. Blockchain technology might be a viable solution as it does not require a server to function. All the information about every single member is stored locally on every participating devices. As long as there are enough devices connected to each other, the network remains fully functional. Even in the case that some devices are down due to hardware failures or maintenance, other devices can still handle any change to the network. We can add, remove and change devices normally. Once the nodes that was previously offline come back online, they will be able to synchronize the information very quickly and communicate with the latest network.

On the other hand, the benefit of blockchain is its ability to run without any centralized server. All the information regarding all the member in the network is self-contained and replicated across the entire network. One major downside is the computation resources required to compute the hash value whenever a new block is added. With IoT, some devices are more capable are calculating hash values so we may be able to offload computationally intensive tasks to the more capable nodes so that the network can function smoothly.

Chapter 3. Project Requirements

Requirements

The application will have the following functionalities:

1. Adding a device

When a device is added to the network via another device that is already in the network, it will be validated and receive all information about the network.

2. Maintain current network information when device is disconnected

If a device is disconnected from the network, it will not be able to communicate with other part of the network. The network should be able to keep the device information. Once the network connectivity restores, the said device would automatically be incorporated into the network without validating again.

3. Adding device into a partially disconnected network

When some part of the network is disconnected from the other devices, they should still form a partially connected network that functions normally. Also, it should still be able to receive new device request. Once the network connectivity is restored, both the current device and the new device will be added back to the network and the network will become one integrated network as before.

Chapter 4. Dependencies and Deliverables

Dependencies

Integrating our project design with Ethereum framework

Because our scenario is slightly different from typical Ethereum use cases, we need to understand the framework better before we can adapt it to suit our needs.

Satisfy the computational needs required by the framework

The simulated application may be able to run on a modern desktop/laptop computer, we still need to determine whether it is feasible for it to be run on a low-end device by looking at its CPU and memory consumption.

Designing and implementing a user interface that is compatible with the framework

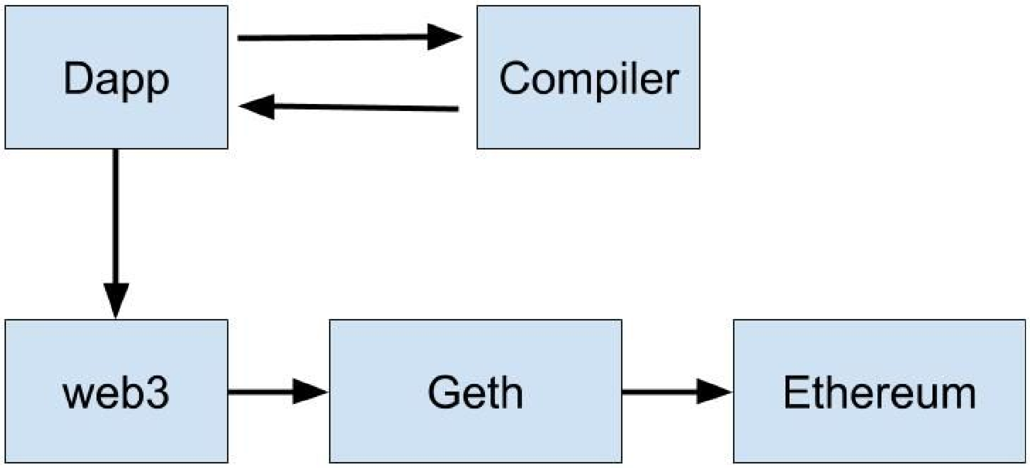
Even though Ethereum framework provides a user interface for developing Dapps, it may not be very straightforward to implement our user interface with it. We need to design our user interface to be compatible with the existing framework while maintaining functionality.

Deliverables

Other than the standard deliverables, we plan to deliver a “things to not try” document that records all the failed attempts made during the project so that we can learn how to spot patterns when developing new application with new framework. We hope it will help others build intuition about common pitfalls in blockchain and IoT development.

Chapter 5. Project Architecture

Project Architecture Diagram



In our project, we have the following blocks:

Dapp: The Dapp is a short of “decentralized application” which is a combination of front-end and smart contracts, where the smart contracts are written in Solidity. The Dapp receives compiled byte codes from the Compiler, deploys the contracts from the byte codes, and further execute the contract functions.

Compiler: The compiler receives and compiles smart contracts in Solidity from Dapp into byte codes, then sends them back to the Dapp.

web3: The web3 makes RPC requests based on the smart contracts from the Dapp.

Geth: Geth is a local Ethereum node that communicates with Ethereum network to deploy and interact with the contracts in the network.

Ethereum: Ethereum contracts are code running on the global Ethereum decentralized P2P network.

Chapter 6. Project Design

[

Describe your project design. Types of design artifacts that you provide might include:

- UML diagrams (class diagrams, sequence diagrams, etc.)
- UI Mockups
- Other artifacts you deem necessary to describe your project in detail.

CMPE Students may want to consider the following alternatives:

- System Block Diagram (Detailed block diagram for each block)

Chapter 7. QA, Performance, Deployment Plan

[

Describe the testing, performance evaluation, and deployment plan for your project

CMPE students may want to consider the following alternatives

- Testing: Different tests done or to be done on the project and their outcomes
- Algorithms
- Bill of materials

Evaluated: Workbook Assignment 2

]

Chapter 8. Implementation Plan and Progress

- Setting up a programming and execution environment

  We setup our programming environment based on the following Github links:

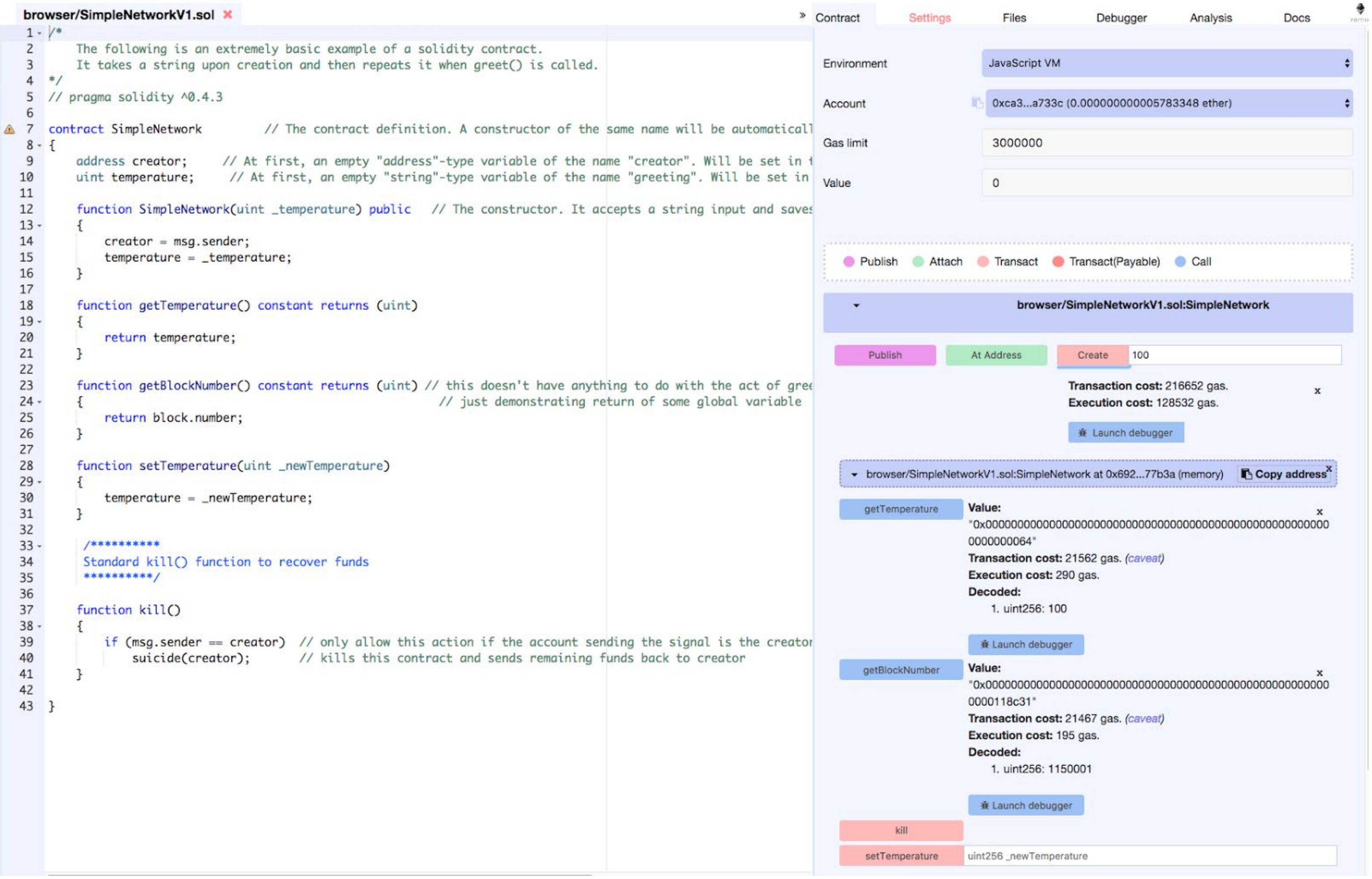
  - Dapp: https://ethereum.github.io/browser-solidity
  - Compiler: https://ethereum.github.io/browser-solidity
  - web3: https://github.com/ethereum/web3.js
  - Geth: https://github.com/ethereum/go-ethereum
  - Ethereum: https://github.com/ethereum/remix

- Acquiring development tools.

  - Dapp: The Dapp is a combination of front-end and smart contracts.
  - Compiler: The compiler compiles smart contracts in Solidity
  - web3: The web3 makes RPC requests based on the smart contracts from the Dapp.
  - Geth: Geth is a local Ethereum node that communicates with Ethereum network.
  - Ethereum: Ethereum is a platform to run Ethereum contracts.

- Implementing a prototype of our project

  - We use browser version compiler to compile our prototype written in Solidity.
  - In the following diagram, we set sensor temperature as 100 degree.
  - If we run the function getTemperature(), we get 100 in uint256.
  - If we run the function getBlockNumber(), we get 1150001 in uint256.



browser/SimpleNetworkV1.sol

```
/*
    The following is an extremely basic example of a solidity contract.
    It takes a string upon creation and then repeats it when greet() is called.
*/
// pragma solidity ^0.4.3

contract SimpleNetwork        // The contract definition. A constructor of the same name will be automaticall
{
    address creator;     // At first, an empty "address"-type variable of the name "creator". Will be set in t
    uint temperature;     // At first, an empty "string"-type variable of the name "greeting". Will be set in

    function SimpleNetwork(uint _temperature) public   // The constructor. It accepts a string input and saves
    {
        creator = msg.sender;
        temperature = _temperature;
    }

    function getTemperature() constant returns (uint)
    {
        return temperature;
    }

    function getBlockNumber() constant returns (uint) // this doesn't have anything to do with the act of gree
    {                                                  // just demonstrating return of some global variable
        return block.number;
    }

    function setTemperature(uint _newTemperature)
    {
        temperature = _newTemperature;
    }

     /*********
     Standard kill() function to recover funds
     *********/

    function kill()
    {
        if (msg.sender == creator)  // only allow this action if the account sending the signal is the creator
            suicide(creator);       // kills this contract and sends remaining funds back to creator
    }

}
```

Environment: JavaScript VM

Account: 0xca3...a733c (0.000000000005783348 ether)

Gas limit: 3000000

Value: 0

browser/SimpleNetworkV1.sol:SimpleNetwork

100

Transaction cost: 216652 gas.
Execution cost: 128532 gas.

browser/SimpleNetworkV1.sol:SimpleNetwork at 0x692...77b3a (memory)

Value:
"0x0000000000000000000000000000000000000000000000000000000000000064"
Transaction cost: 21562 gas. *(caveat)*
Execution cost: 290 gas.
Decoded:
1. uint256: 100

Value:
"0x0000000000000000000000000000000000000000000000000000000000118c31"
Transaction cost: 21467 gas. *(caveat)*
Execution cost: 195 gas.
Decoded:
1. uint256: 1150001

uint256 _newTemperature

Chapter 9. Project Schedule

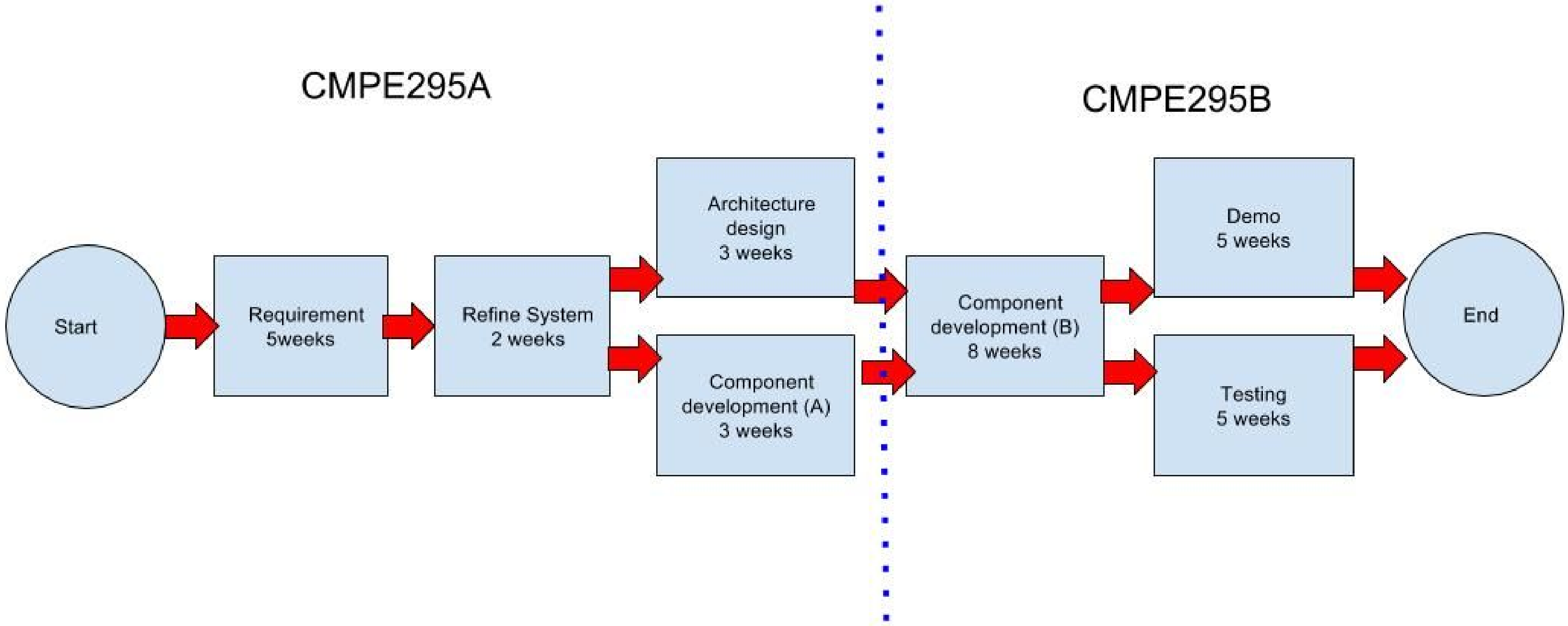
TASK ASSIGNMENTS

In this project, every member will participate almost all of the tasks, including defining and refining system, designing architecture, testing, and demo. But, due to complexity of component development, we consider to separate the component development into three parts, and each of us works on one of them.

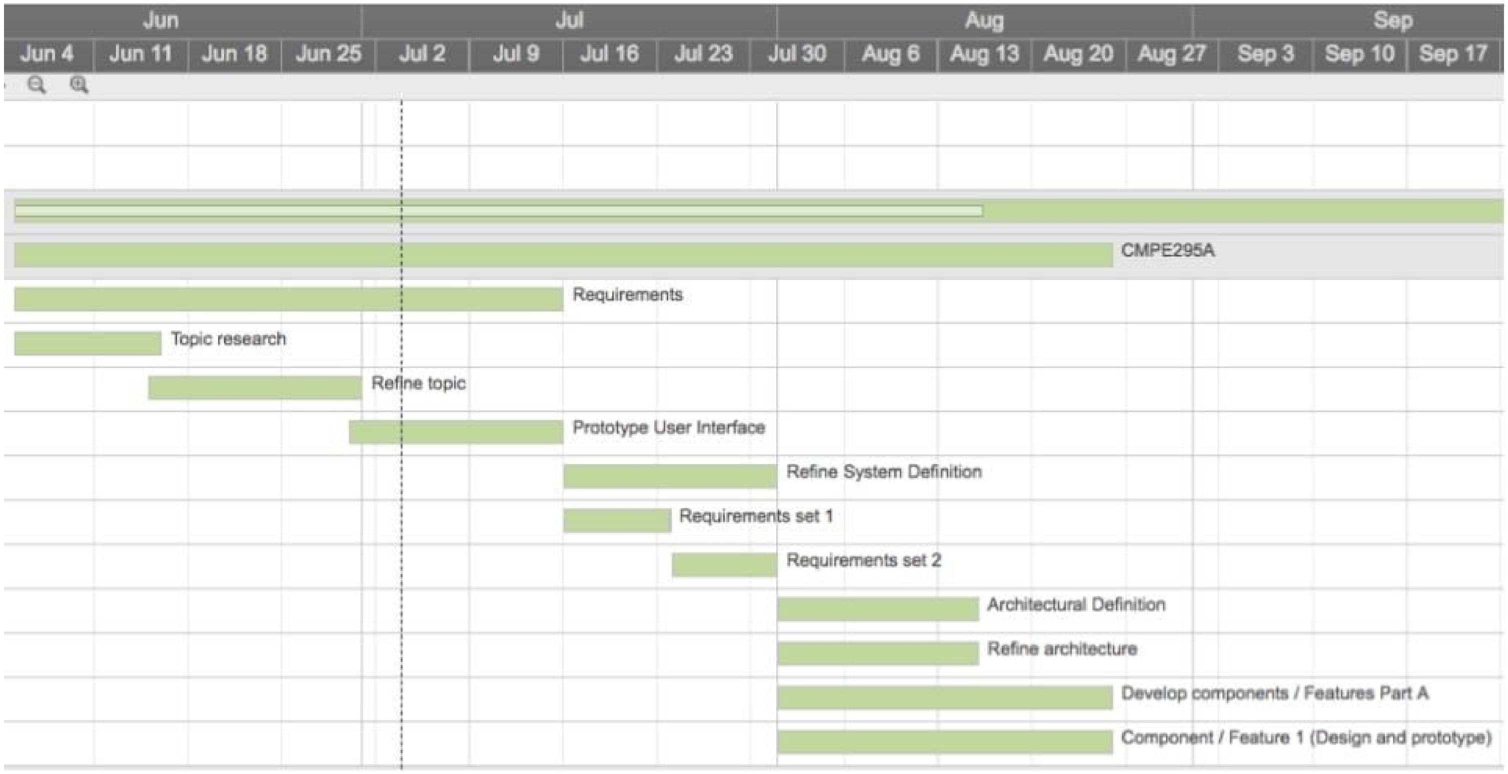
TASK SCHEDULE

We submit two types of schedule charts for review, including Gantt chart and Pert chart which both cover our task to be done in the two semesters CMPE295A and CMPE295B. Please see the following diagrams.

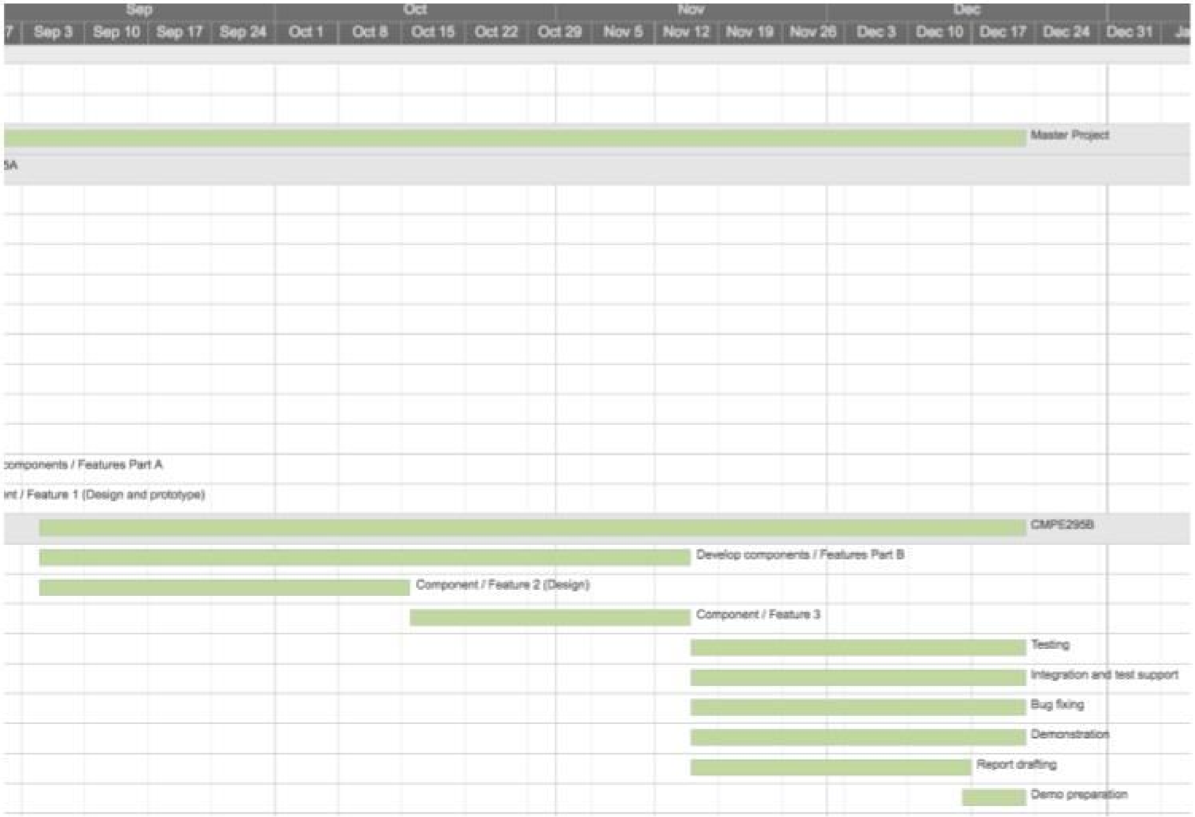
PERT DIAGRAM for CMPE 295A and 295B



GANTT DIAGRAM for CMPE295A



GANTT DIAGRAM for CMPE295B


Sep
Oct
Nov
Dec
Sep 3
Sep 10
Sep 17
Sep 24
Oct 1
Oct 8
Oct 15
Oct 22
Oct 29
Nov 5
Nov 12
Nov 19
Nov 26
Dec 3
Dec 10
Dec 17
Dec 24
Dec 31
Ja
Master Project
5A
components / Features Part A
nt / Feature 1 (Design and prototype)
CMPE295B
Develop components / Features Part B
Component / Feature 2 (Design)
Component / Feature 3
Testing
Integration and test support
Bug fixing
Demonstration
Report drafting
Demo preparation