4+1 Architectural View

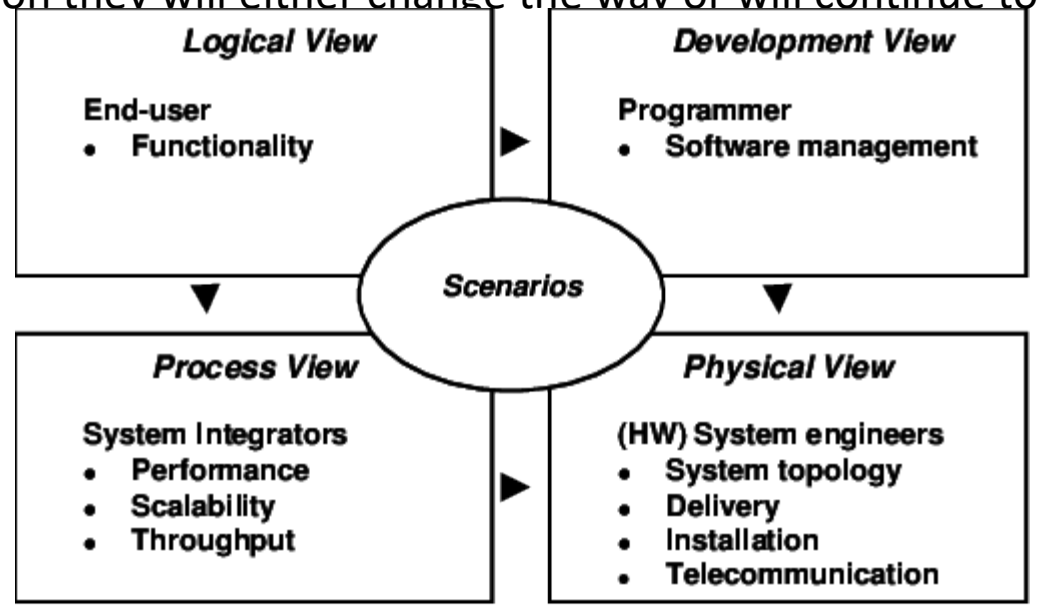
Ajit Kumar(M21CS017)

1.Introduction

In my project, I have implemented block chain on vehicle to vehicle communication. In real life, Whenever accident occur or traffic become high then it that scenario, ahead vehicle will send to message to all the vehicle that is present in communication range. For every vehicle There is fixed communication distance. And once some vehicle receive the message then it also send the message to other vehicle that is present in the range, like that message is transferred to all the vehicle on the road. So we have connected all the vehicle in block chain that is present in the range, to make it secure communication. If it is not secure through block chain then might be in middle can someone change the message.

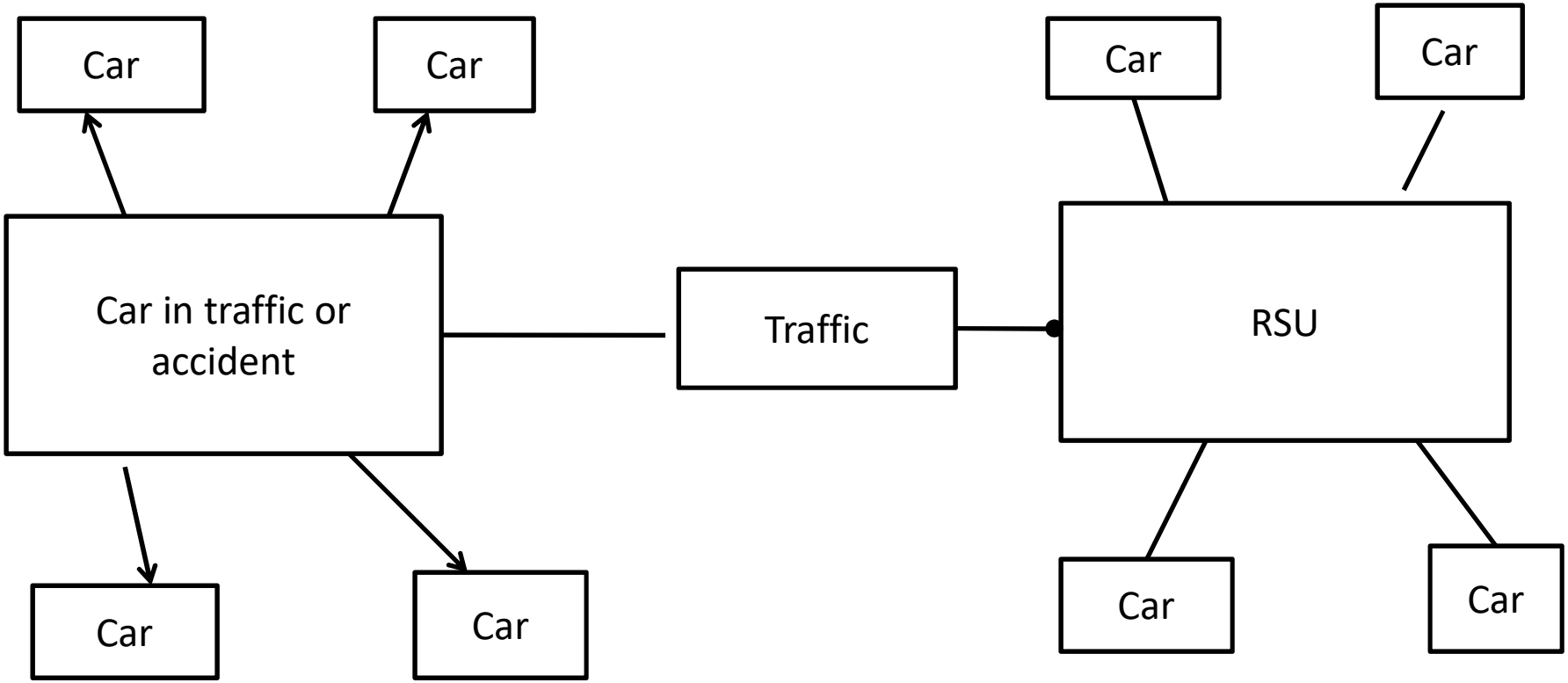
2.Scenario 1(Traffic Jam And Accident):

When traffic is high, vehicle that is present in the range they will receive message by vehicle that is already in traffic. All the vehicle are connected through each other by block chain. According To the situation they will either change the way or will continue to the same way.



2.1. Logical view:

Logical view give the view of user application. In that scenario, whenever traffic is high the vehicle is going to send the message to the other vehicle that is present in range. Message send by the vehicle would be like “Traffic Is High In This Way So Please Change The Way If Possible”, then the vehicle which received the message they will take the decision based on the situation.



2.2Development View:

For implementation of the above scenario we have taken three following:

a. Block class
b. Car
c. RSU

a. Block:

Each class contains their function to implement the required functionality. Block class conation five parameter data, previousHash, hash, timestamp and trueness. It contains five following function:

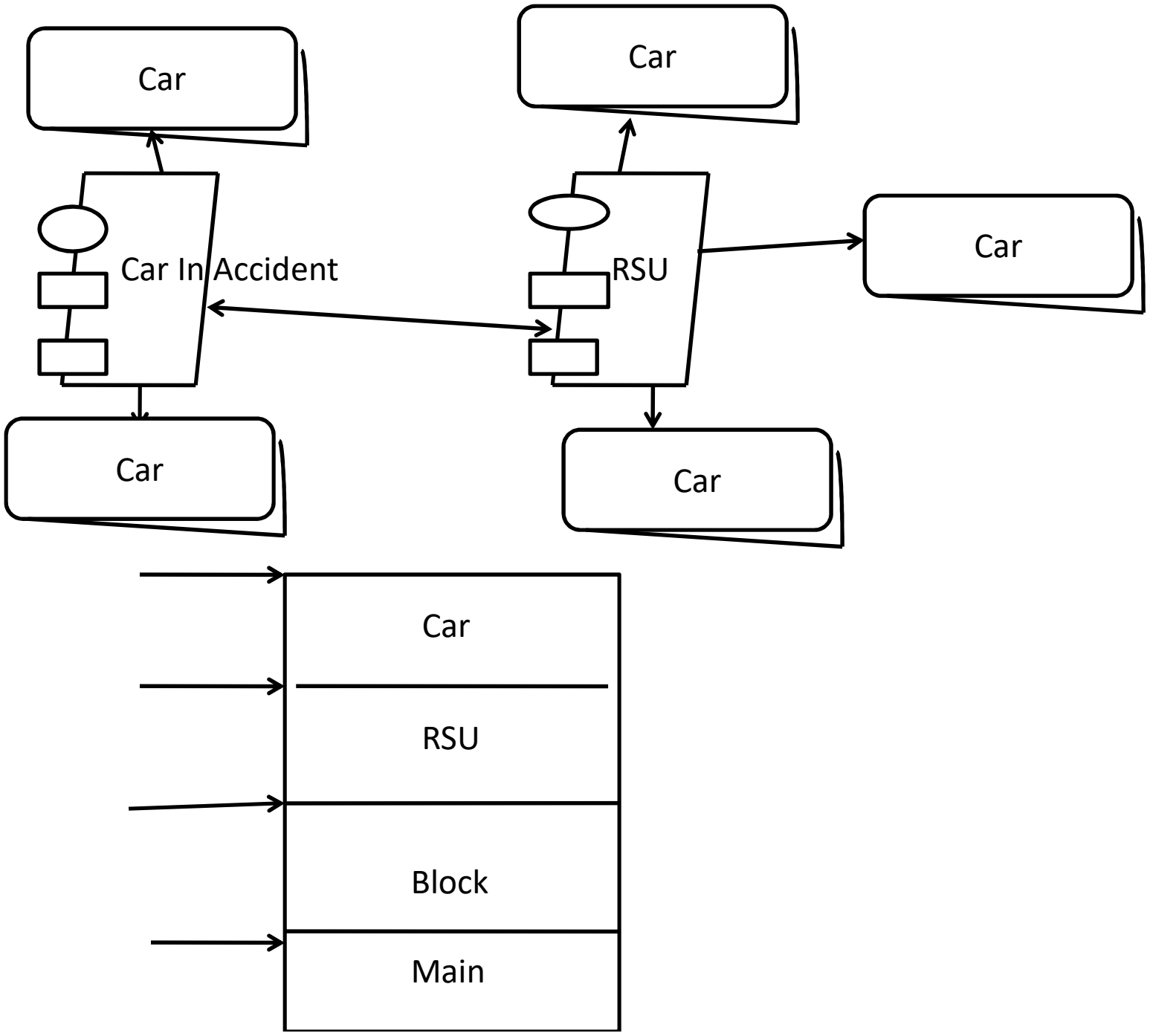
(i)Constructor: this constructor function is used to initialize the block that take three input as argument. This is data, previousHash, trueness.

(ii)getTruthValue: This function is used to get the trueness value.

(iii)setTruthValue: This function is used set trueness value.

(iv)current_block_hash_calculate: This function return the hash value for current block.

(v)Sha_256: This function is used to calculate the current hash value for block.


Car
Car
Car In Accident
RSU
Car
Car
Car
Car
RSU
Block
Main

b. Car:

This class is used to represent all the car. It contains two parameter:

(i) x_cordinate: It represent x_coordinate of the car.

(ii) y_cordinate: It represent y_coordinate of the car.

It has following five function:

(i)Constructor: It is used to initialize the parameter of car class.

(ii) get _x_coordinate: It is used to get the x_cordinate of the car.

(iii) get_y_coordinate: It is used to get the y_coordiante of the car.

(iv) set_x_coordinate: It is used to set the x_coordinate.

(v) set_y_coordinate: It is used to set the y_coordinate.

C.RSU

This class is represent the road side unit side unit. It also receive the message and transmit the message to the vehicle. It contains three parameter :

(i) x_coordinate: It represent the x_coordinate of RSU.

(ii) y_coordinate: It represent the y_coordinate of RSU.

(iii) message: It represent the message that is received from the vehicle.

It has following three function:

(i) Constructor: It is used to initialized the parameter for class RSU.

(ii) getXcoordinate: It is used to get the x_cordinate of RSU.

(iii) getYcoordinate: It is used to get the y_cordiante of RSU.

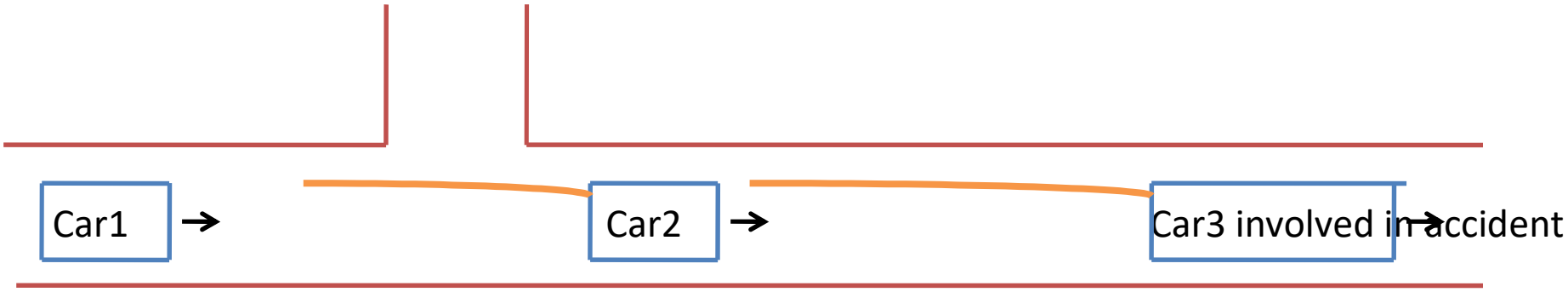
2.3.Process View:

Inside the main function we have taken two variable :

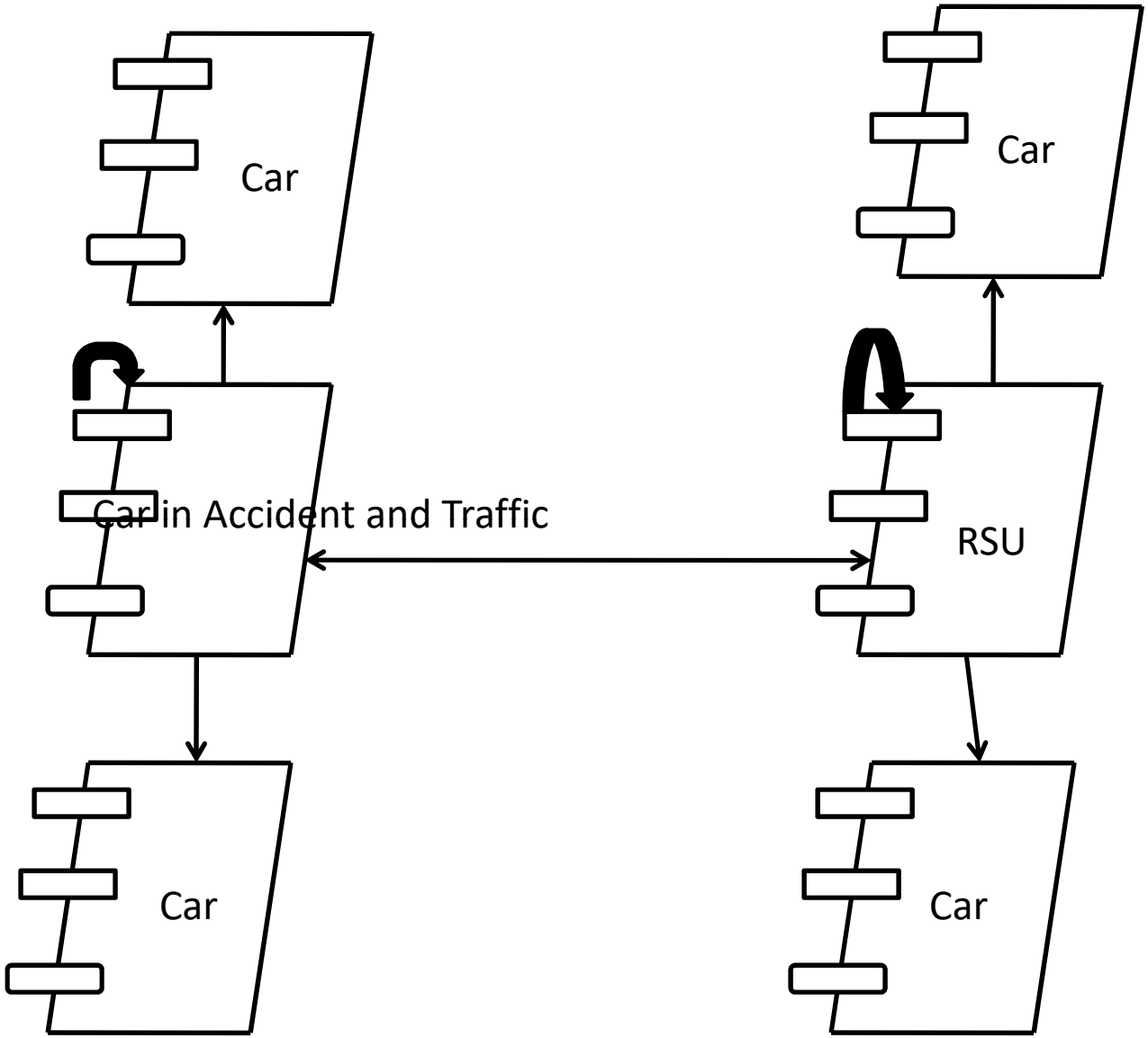
(i)Val: It's value represent the scenario. If it is less than or equal to 0.5 then traffic is high or else accident occur.

(ii) Trueness: It represent the truth value of information. It is used to provide the correct information and ignore the incorrect message.

So once a vehicle receive Val is less than or equal to five and trueness value is greater than 0.5 then it take the decision according to the message, and if trueness value is less than 0.5 then it ignore the massage and it is ignored as it contains wrong information. When trueness value is greater than 0.5 then ether it will change the way or else continue going on the same way it depends upon the scenario and also transmit the message forward to another vehicle than is in range of that vehicle .

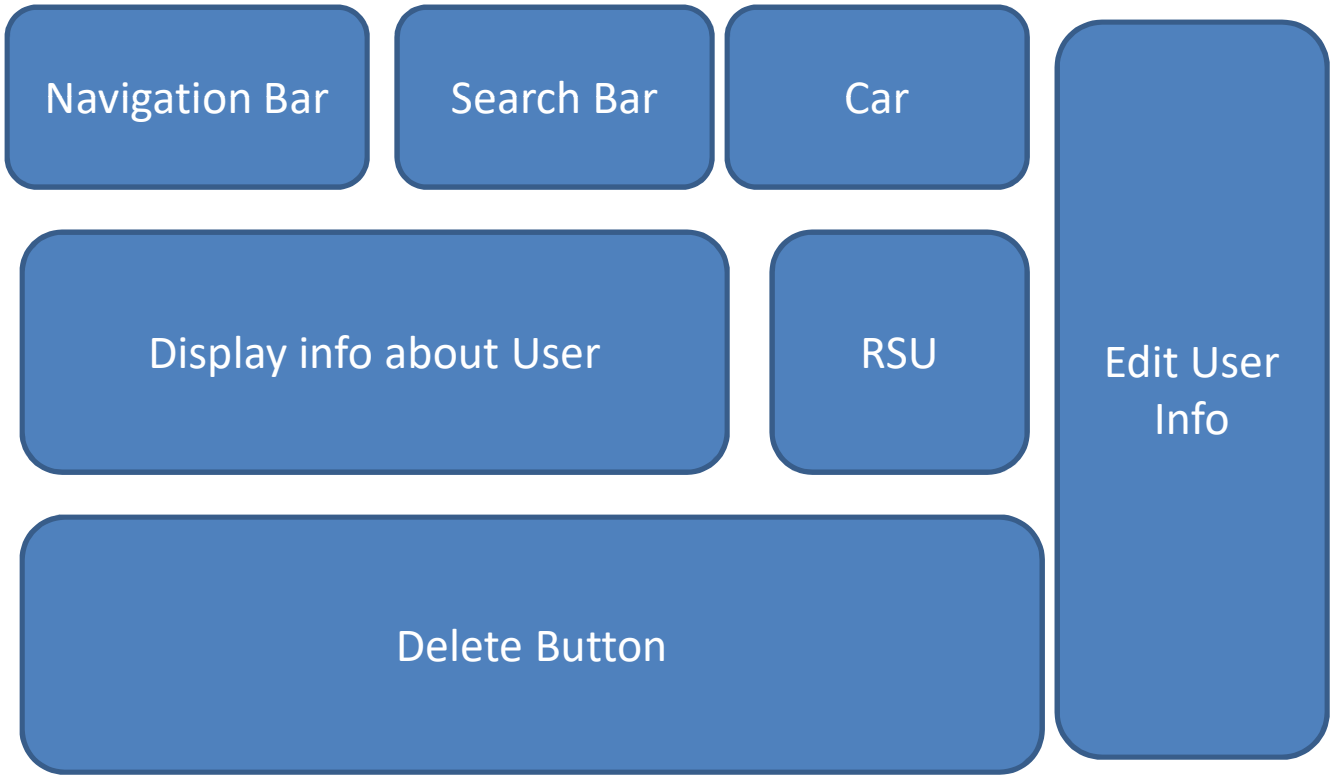


In the above scenrio Car3 is involved in the accident and car2 is n range of car3 So it able to Transmit the message. But car2 is not in range of car1 so message is not reachable to car1.


Car
Car
Car in Accident and Traffic
RSU
Car
Car

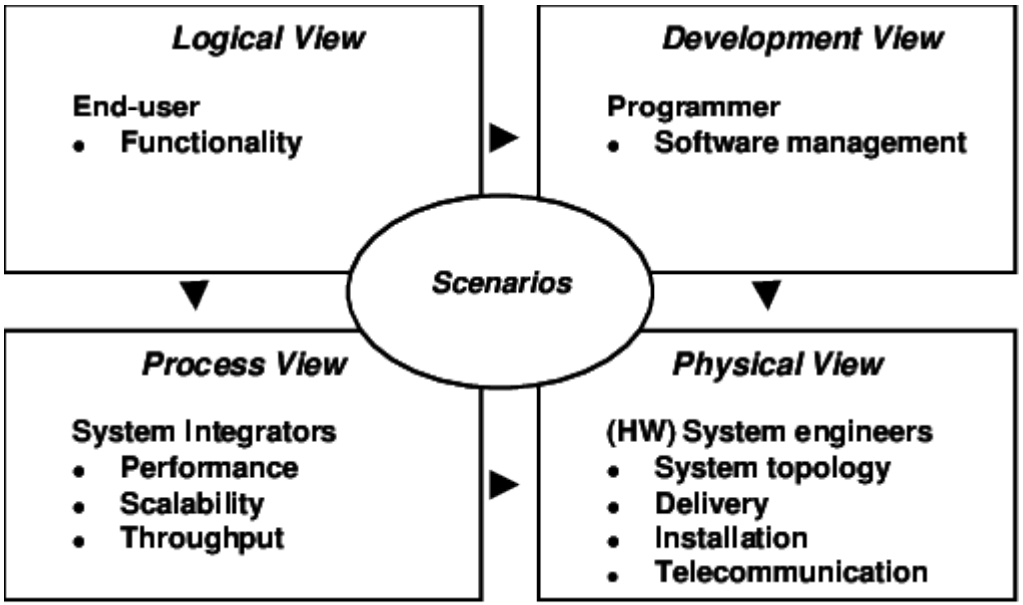
2.4.Physical View:

The physical architecture takes into account primarily the non-functional requirements of the system such as availability, reliability , performance , and scalability. The software executes on a network of computers, or processing nodes . The various elements identified— networks, processes, tasks, and objects—need to be mapped onto the various nodes. We expect that several different physical configurations will be used: some for development and testing, others for the deployment of the system for various sites or for different customers. The mapping of the software to the nodes therefore needs to be highly flexible and have a minimal impact on the source code itself.



3.A Block chain-Based Application System for Product Anti-Counterfeiting

The trade in counterfeit goods is growing and is affecting the sales and profits of companies affected by this phenomenon. To ensure the identification and traceability of real products throughout the supply chain, this paper is the first to propose a fully functional blockchain system to prevent product counterfeiting. Enterprises only need to pay very low transaction fees, and they no longer need to worry about the possibility of obtaining counterfeit products.



3.1.Logical View

It's gives the view of user application. In that scenario ,It contains manufacturer , customer and seller.

1) **Manufacturer Role:** For the seller's part, the provided functions include adding new seller's address on contracts, adding the number of products that the seller can sell, and retrieving information on sellers so that the latest sales status can be retrieved.

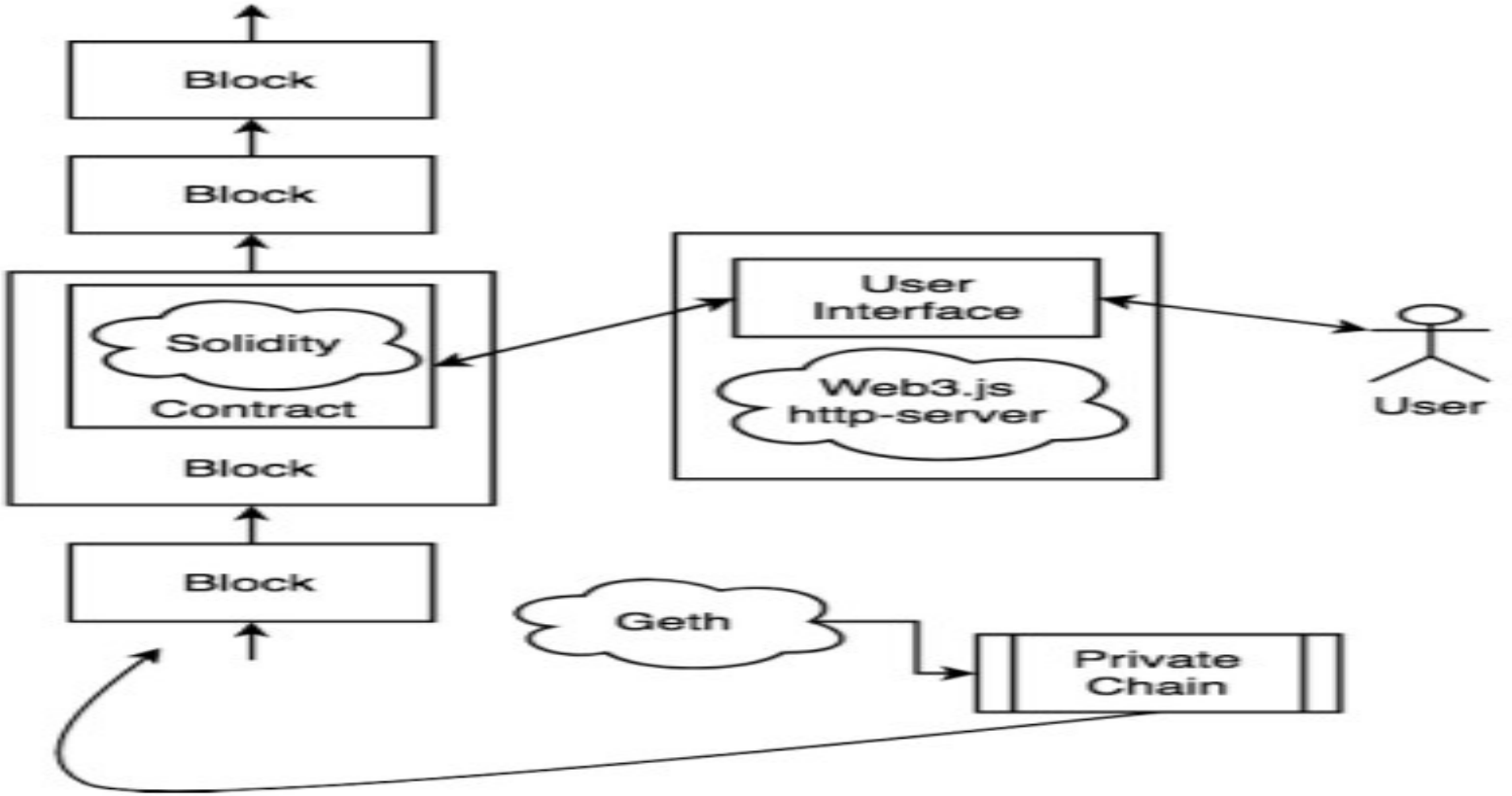
2) **Seller Role:** For the consumer's part, the seller can use the system's functions to encrypt the verification information with a private key, and the consumer can use the seller's public key to verify if the seller is what he claims to be. After buying and selling, the seller specifies the purchaser's address in the contract for the manufacturer to obtain the information.

3) Consumer Role: In the seller's part, the consumer can verify whether the seller has a sales relationship with the manufacturer and also verify whether the seller's stock hasn't been yet sold out. In the manufacturer's part, the consumers can prove that their identity is consistent with their address and in the case of a well-preserved contract address, the consumers can obtain individual purchase records and product status in their product.

2.2Development View:

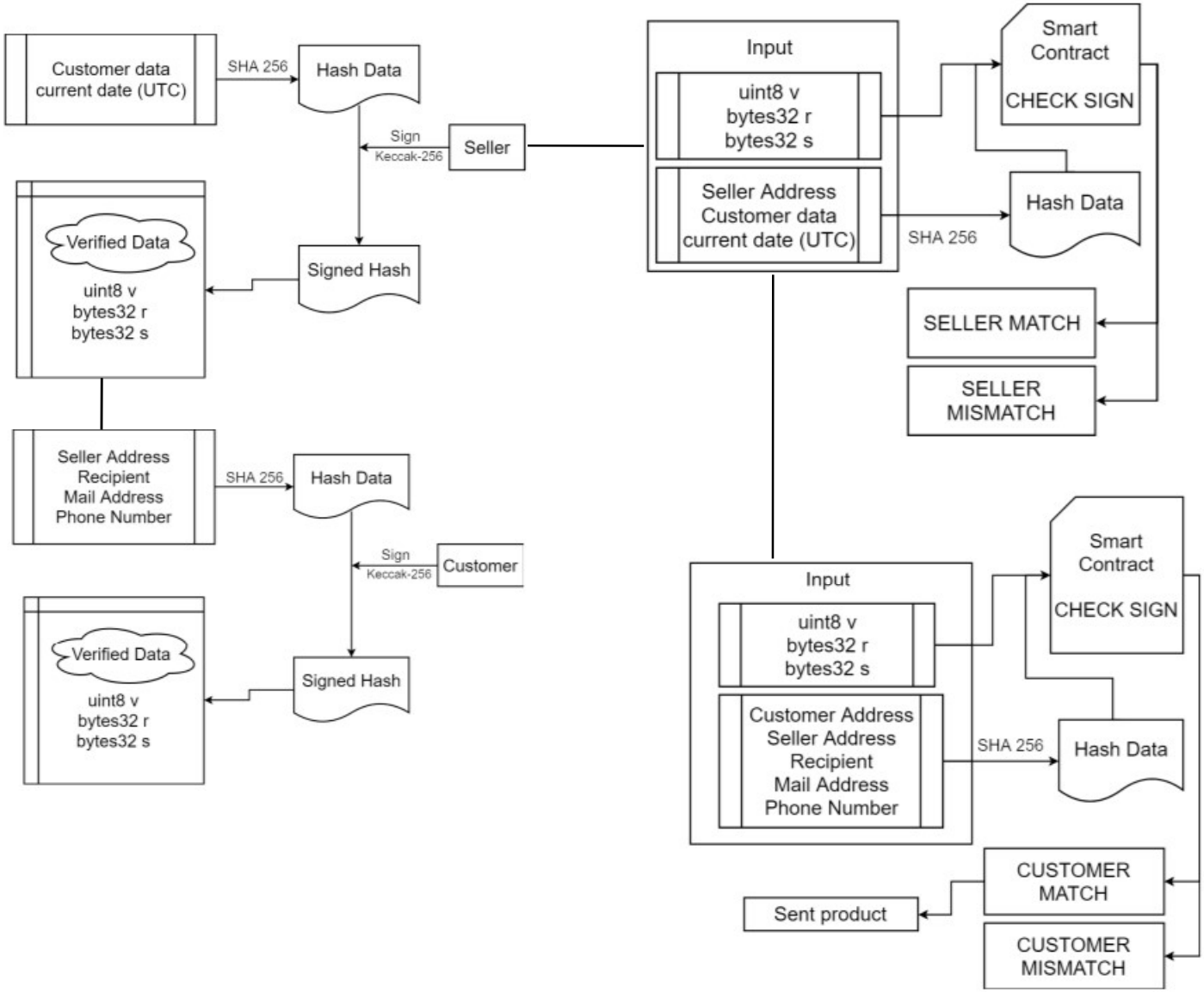
The proposed system uses Ethereum as the back end Block chain operating system and uses Ethereum's proprietary programming language Solidity as the high-level programming language for writing smart contracts. Solidity supports inheritance, libraries importing, etc. Solidity is designed for Ethereum Virtual Machine(EVM). Unlike Bitcoin's scripts, Solidity provides loops and it is Turing complete.

On the system, the public smart contract is based on Ethereum's Blockchain. In this paper, for ease of testing, we use Geth to build a Private chain and push the smart contract on this Private chain, so that the Private chain simulates the situation of the Public chain. Plus use Mist for account balance and contract information management. The user interface seen by the user is a web page. The server side of the web page is made using the http-server suite, which was provided by node.js and web3.js is used as the link between the smart contract and the user interface. The Private Chain and Address information can be connected after setting the server. The overall system relationship is shown in the following diagram.



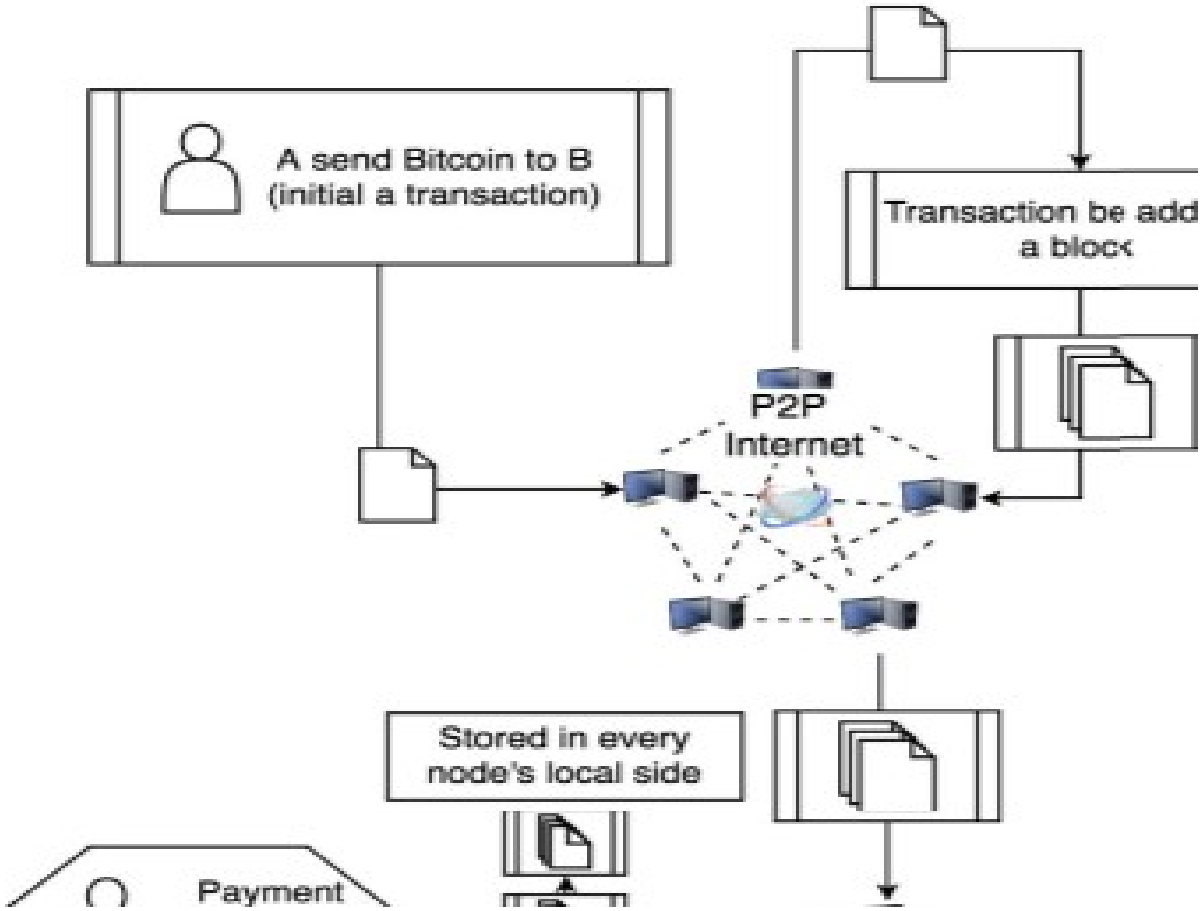
2.3.Process View:

Before conducting a transaction with the seller, the consumer needs to ask the seller for a proof of identity. The consumer will provide the seller with a message that is to be encrypted. Then, the seller will call a function to encrypt the message, the function will concatenate the message and the current time, and will proceed to encrypt them. The system will then return the v, r, s, and the encryption time back. The seller will then send v, r, s, the encryption time, and the seller address to the customer. After the consumer acquires the essential information to verify the identity of the seller, the consumer will then call our system verification function to verify whether the seller identity is correct. The customer will have to input the v, r, s, the encryption time, the seller address, and the message that the customer asked the seller to encrypt. The function will then return True if the seller is who he claimed to be. In contrast, return False if the seller is not legit. After being added into contract product owner field by seller, the consumer can send the shipment address information which was encrypted by the consumer's private key and the hash value of the encrypted information to be verified by the manufacturer. The customer has to provide the seller address, the recipient name of the product, the mail address of the recipient, and the phone number of the recipient. The system will sign these data with the consumer's private key, and provide these data to the manufacturer. The manufacturer will use our system verification function to verify whether the encrypted information is from the customer. The manufacturer will have to input the consumer address, the seller who sells a product to the seller, the recipient of the product, the mail address of the recipient, phone number of the recipient, and the encrypted verification


Customer data
current date (UTC)
SHA 256
Hash Data
Sign
Keccak-256
Seller
Verified Data
uint8 v
bytes32 r
bytes32 s
Signed Hash
Seller Address
Recipient
Mail Address
Phone Number
SHA 256
Hash Data
Sign
Keccak-256
Customer
Verified Data
uint8 v
bytes32 r
bytes32 s
Signed Hash
Input
uint8 v
bytes32 r
bytes32 s
Seller Address
Customer data
current date (UTC)
SHA 256
Smart
Contract
CHECK SIGN
Hash Data
SELLER MATCH
SELLER
MISMATCH
Input
uint8 v
bytes32 r
bytes32 s
Customer Address
Seller Address
Recipient
Mail Address
Phone Number
SHA 256
Smart
Contract
CHECK SIGN
Hash Data
CUSTOMER
MATCH
Sent product
CUSTOMER
MISMATCH

2.4.Physical view:

Physical view deal with system topology, delivery, installation, telecommunication. It contains a sender bit coin to B which is a customer to seller transfer . All the transaction are connected peer to peer and whenever a new customer make a new transaction then transaction is added into block after verification of the transaction. Once system is verified then transaction become successfully to the seller.



Q2. Identify two design patterns used in the app and explain each of the design patterns with the following subsections:

a) Problem Statement,

b) Context,

c) Forces/Constraints and

d) Solutions - Static & Dynamic schematics,

e) Consequences.

Ans: Two design pattern used in this application:

(i). Pipe-and-filter

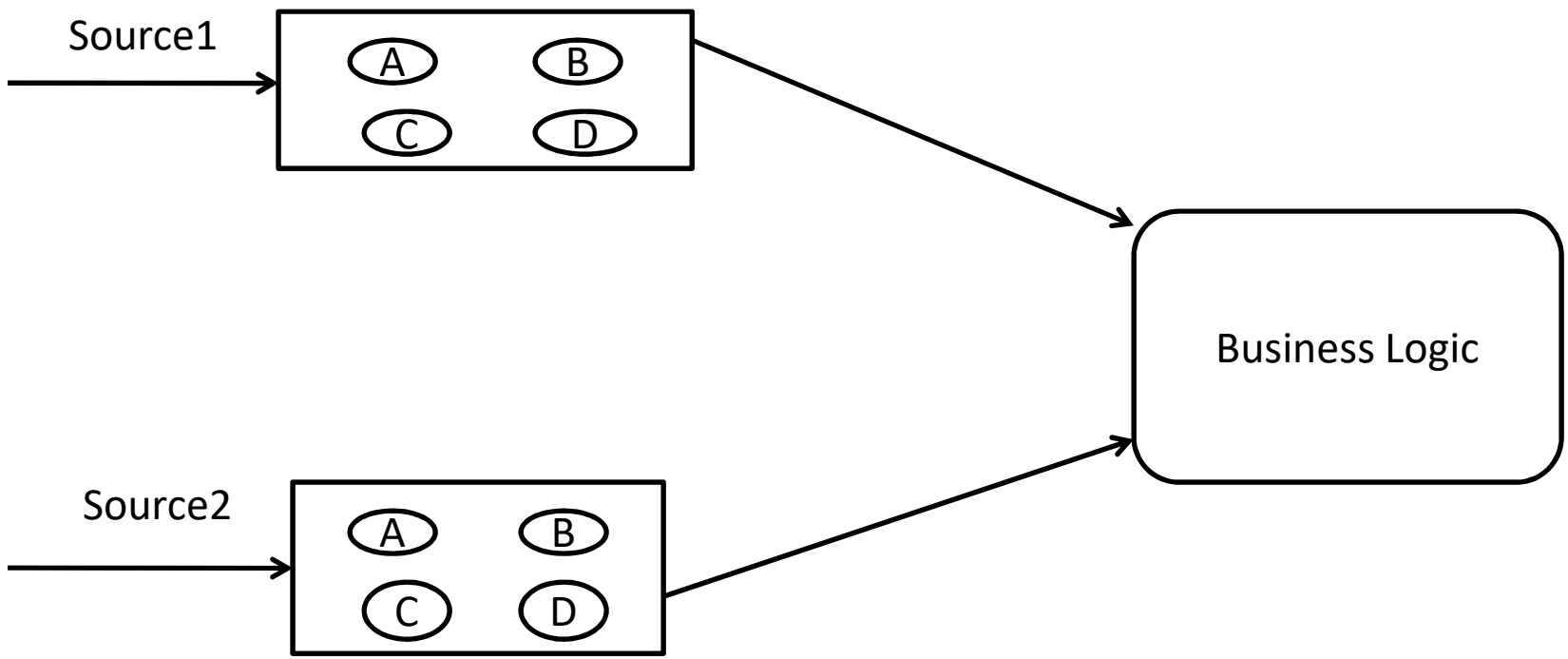
(ii). Peer-to-peer

(i).Pipe and Filter:

Problem Statement:

A very simple, yet powerful architecture, that is also very robust. It consists of any number of components (filters) that transform or filter data, before passing it on via connectors (pipes) to other components. The filters are all working at the same time . The architecture is often used as a simple sequence, but it may also be used for very complex structures.

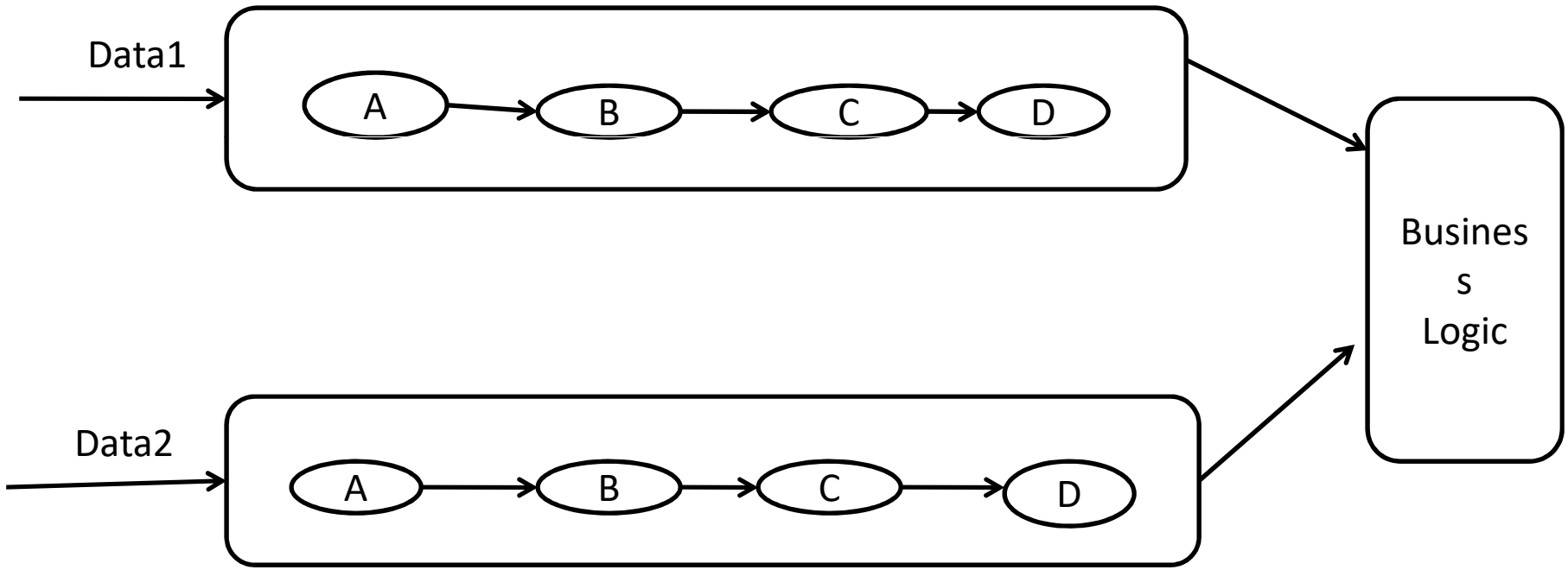
Let us consider a monolithic approach . In figure we can identify the problem with processing the data in monolithic approach. A application receive the data from two resource. Separate module receive the data and processing it. After processing it , would pass the data to business logic phase.



Many task in monolithic module are very similar but for each task module has been designed separately. The processing task perform by each module has changed as business requirement. However, the processing task performed by each module, or the deployment required for each job, could change as a business requirement is updated. Some tasks might be compute-intensive and could benefit from running on powerful hardware, while others might not require such expensive resources.

Context:

Break down the processing required for each stream into a set of separate components (or filters), each performing a single task. By standardizing the format of the data that each component receives and sends, these filters can be combined together into a pipeline. This helps to avoid duplicating code, and makes it easy to remove, replace, or integrate additional components if the processing requirements change.



Forces and Constraints:

(i) Easier to scale
(ii) Allow code reusability
(iii) More loose coupling

Even above points are there but following statement will be also affect:
(i) Complexity of code is more in distributed system.
(ii) Incase of unreliable data flow, the design is vulnerable.
(iii) One failure in pipeline can affect all pipeline.

Static and Dynamic solution:

Static pipeline has fixed operation. Dynamic pipeline has various operation and it is called when It is required or requested.

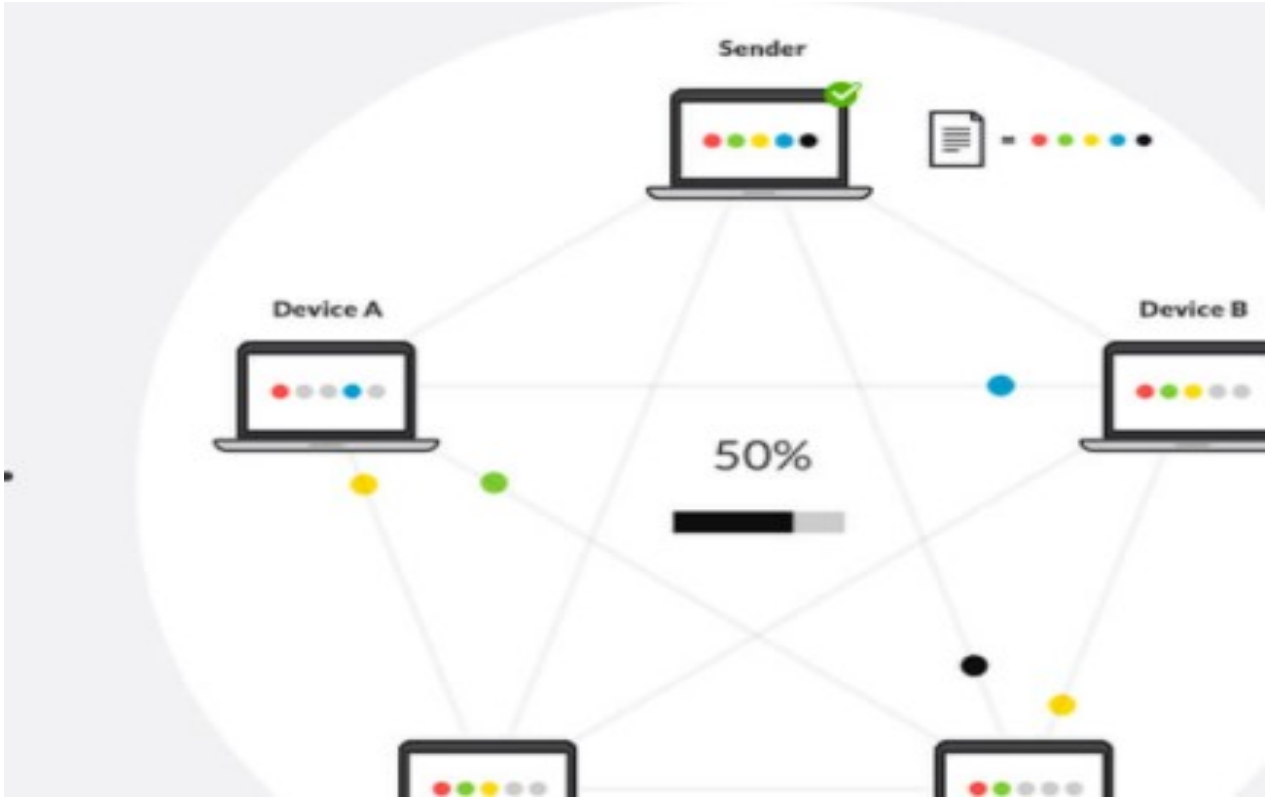
Consequences:

By using pipeline-and-filter, the processing steps performed by an application aren't independent , or they have to be performed together as part of the same transaction. The amount of context or state information required by a step makes this approach inefficient. It might be possible to persist state information to a database instead, but don't use this strategy if the additional load on the database causes excessive contention.

Peer-to-Peer:

Problem Statement:

The peer-to-peer model differs in that all hosts are equally privileged and act as both suppliers and consumers of resources, such as network bandwidth and computer processing. Each computer is considered a node in the system and together these nodes form the P2P network. The early Internet was designed as a peer to peer network where all computer systems were equally privileged and most interactions were bi-directional. When the Internet became a content network with the advent of the web browser, the shift towards client-server was immediate as the primary use case on the internet became content consumption



Context:

(i) Every node must update the block-chain.
(ii) If there are multiple query hits, the client selects from one of these peers.
(iii) Now when one peer requests for some file, this request is sent to all its neighboring nodes i.e. to all nodes which are connected to this node. If those nodes don't have the required file, they pass on the query to their neighbors and so on.

Forces and Constraints:

(i) No access request emergency because system is decentralized.
(ii) Its implementation is some more simple as compare to other design.

However, Following constraint are present:
(i) Security maintaining is big challenge.
(ii) Performance is fixed in case of peer-to-peer.

Static and Dynamic Solution:

In case case of static peer-to-peer ,the address of the node within the network is not changed. But in case of dynamic , Ip address of every node might not be same . Once the connection will terminate or close for some times then node will get other address.

Consequence:

(i) Performance depend upon complexity of transaction. If number of transaction is more then It performance is decreased.

(ii) Every transaction perform on any node , it's history is available to every node.