Configuration Manual

MSc Cybersecurity Evening program.

Asad Ali Khan
Student ID: x21168342@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. ……Asad AL Khan ……

Student ID: …… x21168342@student.ncirl.ie ……

Programme: ……MSc Cybersecurity …… **Year:** 2023-2024.

Module: ……Thesis -Research Project ……

Lecturer: ……Michael Pantridge……

Submission Due Date: ……30th April ……

Project Title: …… DDoS Prevention in Home IoT Devices Using Hyperledger Blockchain…….

Word Count: …… **Page Count:** ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	□

Assignments that are submitted to the Programme Coordinator's Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Asad Ali Khan
Student ID: x21168342@student.ncirl.ie

1 Introduction

Software and Tools used.

- Eclipse IDE, sublimetext, NS3 Simulator
- Windows Machine 11 Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz 2.21 GHz with 32GB of RAM and 500 GB SSD
- Kali, ubuntu vms x 2 , windows vm
- Ifogsim simulator
- Blocksim simulator
- Hyper Ledger Framework BC installation on Kali and Ubuntu machines
- HOIC and LOIC tools for educational purposes to simulate DDoS attacks on Ubuntu closed network machines installed with HLF.

Disclaimer: The above tools are detected as malware, I have used them for education use, please don't try it if you don't know what they do. The network must be closed for any testing.
Wireshark for network monitoring
Scappy for packet generation
Nmap for checking for open ports in the system.

2 Installation

Ns3 Network Simulation Installation steps https://www.nsnam.org/
Install NS3 – 3.40 – new version is also out now which is 3.41.

```
Via Bake
$ ./bake.py configure -e ns-allinone-3.41
```

Configure python bindings

```
$ ./ns3 configure --enable-python-bindings
```

Install Cmake tools and C++ plugins to edit the cc files VScode or Eclipse. I am using VScode.
Had to install Python with Sqllite and QT- qmake to run the NetAnim – Which is the animation module in NS3 to show the simulation of Nodes and IOT devices.
https://www.nsnam.org/wiki/NetAnim_3.108
This is to call the actual simulation environment to showcase DDoS attack and traffic behavior.

```
cd netanim
make clean
qmake NetAnim.pro
make
```

```
apt install g++ python3 python3-dev pkg-config sqlite3 cmake
```

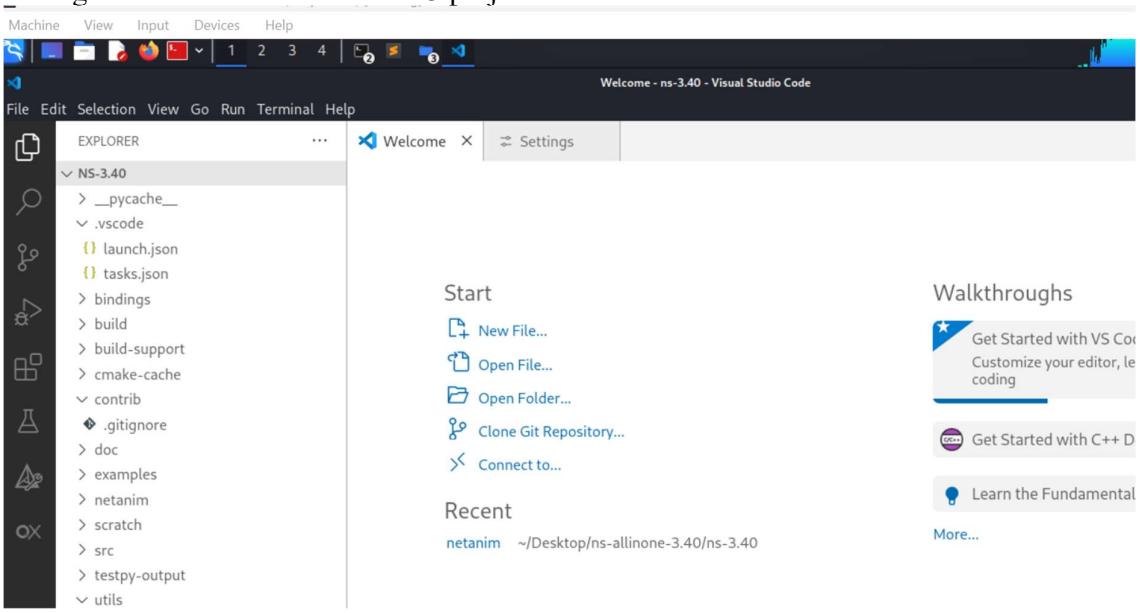
Install other bindings - https://www.nsnam.org/wiki/Installation

```
apt install mercurial unzip
```

Follow the guide to the NSNAM page to get started with the tool for simulation
https://www.nsnam.org/docs/tutorial/html/getting-started.html
Used build command to enable ns3 with examples

```
$ ./ns3 configure --enable-tests --enable-examples -d optimized
```

Configured VS code IDE to edit ns3 project files.



Configuring to use python bindings
https://www.nsnam.org/docs//manual/html/python.html

DDOS Network Configuration on Virtual box:
Setting up Private Network Configuration: Set up a secure private Network called “Secure” on the virtual box. The devices were assigned internal IP addresses in the same range so they

can communicate with each other without internet access.

```
NetworkName:    Secure
Dhcpd IP:       192.168.2.1
LowerIPAddress: 192.168.2.2
UpperIPAddress: 192.168.2.254
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:               None
Individual Configs:   None

C:\Program Files\Oracle\VirtualBox>
```

Setup 2 x IOT devices connected to a Server Node on kali Linux.

```
sudo apt update && sudo apt upgrade
```

```
sudo apt install wireshark
```

All devices are installed with – Wireshark to check traffic before, during and after DDoS attacks.

Hyperledger Framework Prerequisites:

Install Docker and Docker compose.
Install Golang – the language that HLF uses. It can also use Java.
https://go.dev/dl

```
root@osboxes:/home/osboxes/Downloads# tar -zxf go1.22.2.linux-amd64.tar.gz -C /home/osboxes/
root@osboxes:/home/osboxes/Downloads# curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16555  100 16555    0     0  58292      0 --:--:-- --:--:-- --:--:-- 58498
```

Install node js and nvm

```
=> Cloning into '/root/.nvm'...
remote: Enumerating objects: 365, done.
remote: Counting objects: 100% (365/365), done.
remote: Compressing objects: 100% (313/313), done.
remote: Total 365 (delta 43), reused 165 (delta 26), pack-reused 0
Receiving objects: 100% (365/365), 365.08 KiB | 4.10 MiB/s, done.
Resolving deltas: 100% (43/43), done.
* (HEAD detached at FETCH_HEAD)
  master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /root/.bashrc
=> Appending bash_completion source string to /root/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
root@osboxes:/home/osboxes/Downloads# nvm install 20
bash: nvm: command not found
root@osboxes:/home/osboxes/Downloads# export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
root@osboxes:/home/osboxes/Downloads# [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion
 # This loads nvm bash_completion
root@osboxes:/home/osboxes/Downloads# nvm install 20
Downloading and installing node v20.12.2...
Downloading https://nodejs.org/dist/v20.12.2/node-v20.12.2-linux-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v20.12.2 (npm v10.5.0)
Creating default alias: default -> 20 (-> v20.12.2)
root@osboxes:/home/osboxes/Downloads# node -v
v20.12.2
root@osboxes:/home/osboxes/Downloads# npm -v
10.5.0
root@osboxes:/home/osboxes/Downloads#
```

1. **Install Git**. Git is used for version control and is often required to clone Hyperledger Fabric repositories and related projects.
2. **Install Java JDK** – I installed open JDK https://www.linode.com/docs/guides/how-to-install-openjdk-ubuntu-22-04/

Test Java installation.

https://www.linode.com/docs/guides/how-to-install-openjdk-ubuntu-22-04/

```
root@osboxes:/home/osboxes/Downloads# javac helloworld.java
error: file not found: helloworld.java
Usage: javac <options> <source files>
use --help for a list of possible options
root@osboxes:/home/osboxes/Downloads# javac HelloWorld.java
root@osboxes:/home/osboxes/Downloads# ls -l HelloWorld.class
-rw-r--r-- 1 root root 431 Apr 20 21:42 HelloWorld.class
root@osboxes:/home/osboxes/Downloads# java HelloWorld.
Error: Could not find or load main class HelloWorld.
Caused by: java.lang.ClassNotFoundException: HelloWorld.
root@osboxes:/home/osboxes/Downloads# java HelloWorld.java
error: class found on application class path: HelloWorld
root@osboxes:/home/osboxes/Downloads# java HelloWorld
Hello Java World!
```

Java compiling fine.

3. **Added Sublime Text** for simplicity as an IDE application for working on HLF development in addition to VScode.

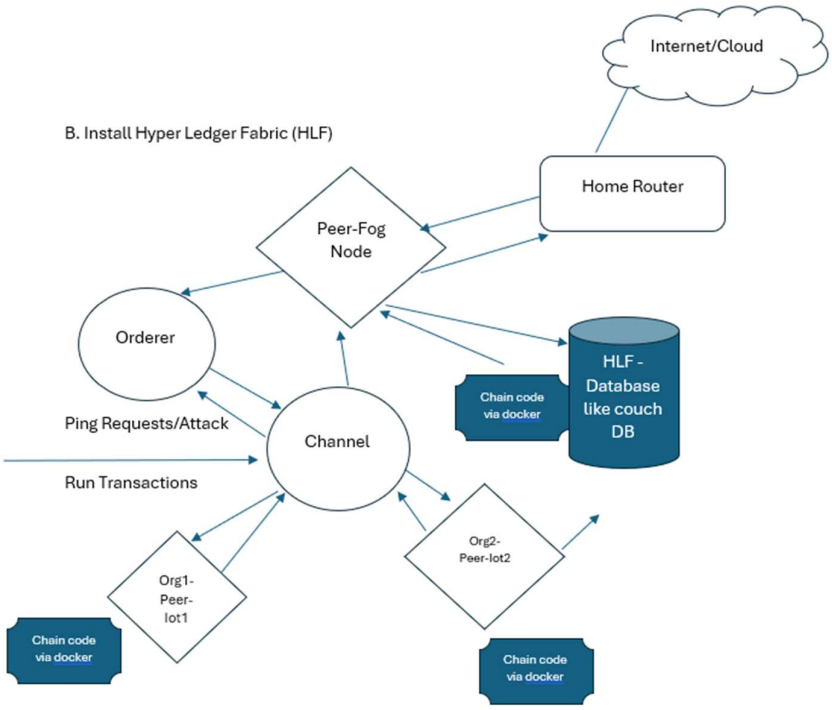


Fig2. Attack concept, network setup – HLF setup on 2 IoT devices connected to a Node device.

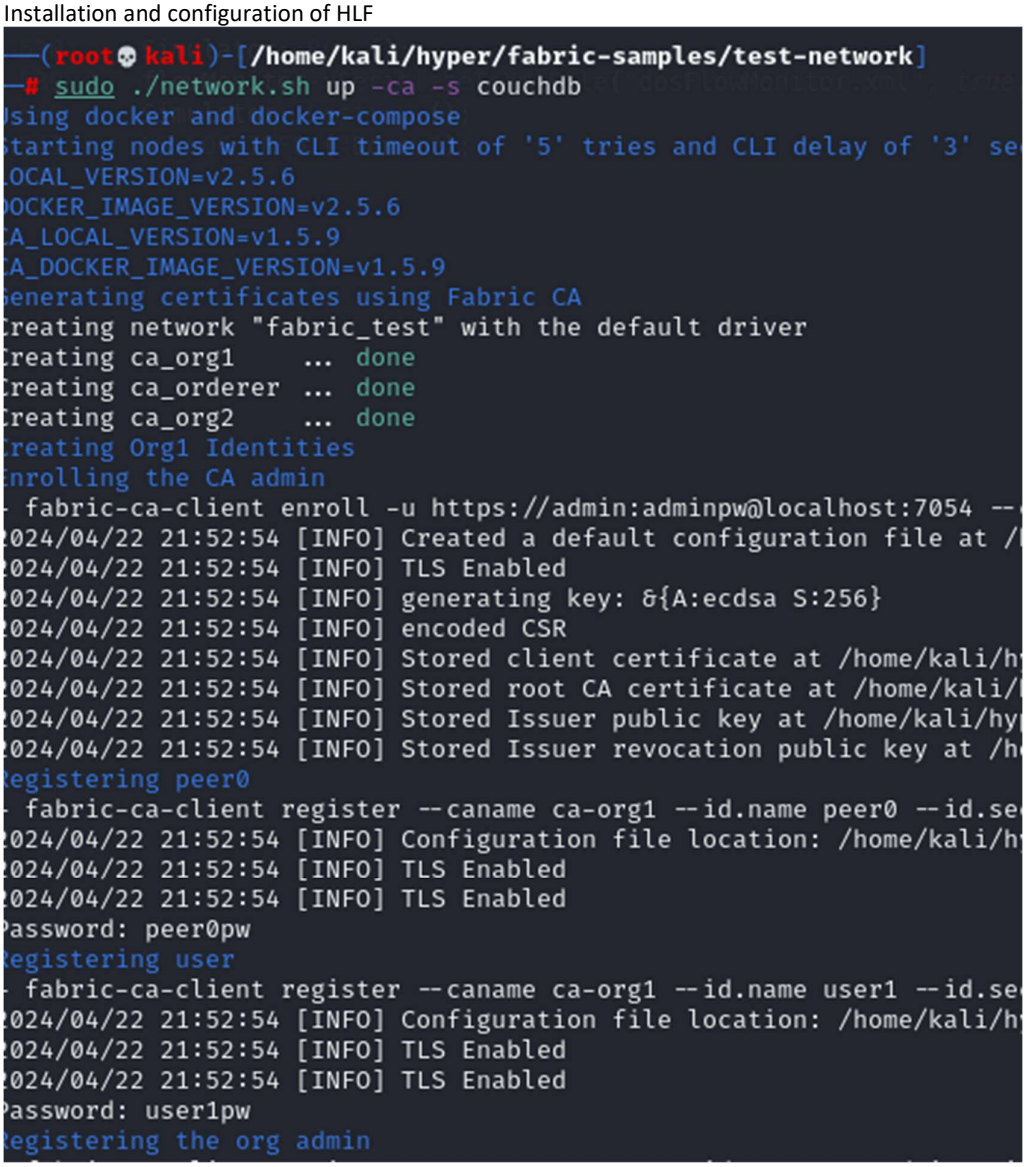
HLF Installation:

```
curl -sSL https://bit.ly/2ysbOFE | bash -s
```

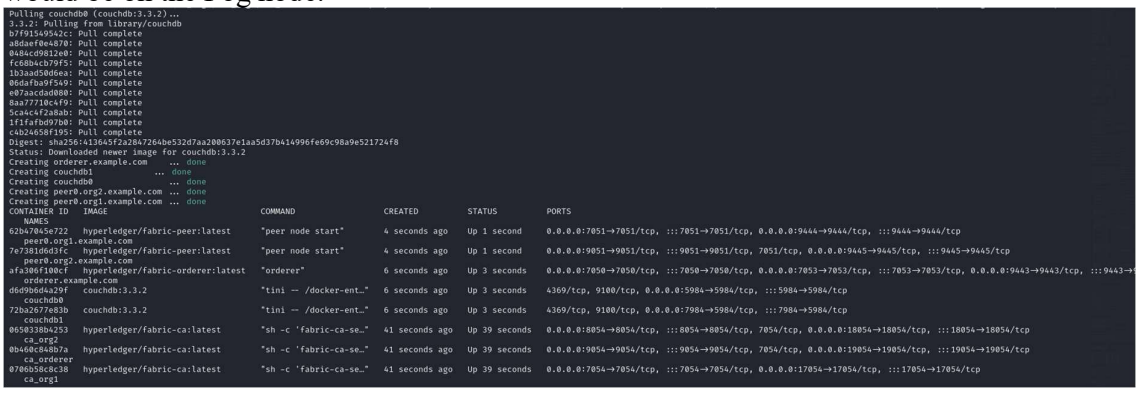
change path

```
export PATH=<caminho para o local de download>/bin:$PATH

Deploy the test network
cd fabric-samples/test-network with flags to create channel for communication and also
chaincode for consensus mechanism.
    - 'up' - bring up fabric orderer and peer nodes. No channel is created
       - 'up createChannel' - bring up fabric network with one channel
       - 'createChannel' - create and join a channel after the network is created
       - 'deployCC' - deploy the fabcar chaincode on the channel
       - 'down' - clear the network with docker-compose down
       - 'restart' - restart the network

 Creating a channel for communication of nodes
./network.sh createChannel

Query the Chaincode using default channel
peer chaincode query -C mychannel -n fabcar -c '{"Args":["queryAllCars"]}'
```

Installation and configuration of HLF



Setting Up Couch DB and network setup completion with 2 nodes and one orderer which would be on the Fog node.



Command to check for docker containers running on the network.

```
docker ps -a
```

```
┌──(root㉿kali)-[/home/kali/hyper/fabric-samples/test-network]
└─# docker ps -a

CONTAINER ID   IMAGE                               COMMAND                  CREATED         STATUS         PORTS
 NAMES
62b47045e722   hyperledger/fabric-peer:latest      "peer node start"        5 minutes ago   Up 5 minutes   0.0.0.0:7051→7051/tcp, ::
 peer0.org1.example.com
7e7381d6d3fc   hyperledger/fabric-peer:latest      "peer node start"        5 minutes ago   Up 5 minutes   0.0.0.0:9051→9051/tcp, ::
 peer0.org2.example.com
afa306f100cf   hyperledger/fabric-orderer:latest   "orderer"                5 minutes ago   Up 5 minutes   0.0.0.0:7050→7050/tcp, ::
 orderer.example.com
d6d9b6d4a29f   couchdb:3.3.2                       "tini -- /docker-ent…"   5 minutes ago   Up 5 minutes   4369/tcp, 9100/tcp, 0.0.0.
 couchdb0
72ba2677e83b   couchdb:3.3.2                       "tini -- /docker-ent…"   5 minutes ago   Up 5 minutes   4369/tcp, 9100/tcp, 0.0.0.
 couchdb1
0650338b4253   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   6 minutes ago   Up 6 minutes   0.0.0.0:8054→8054/tcp, ::
 ca_org2
0b460c848b7a   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   6 minutes ago   Up 6 minutes   0.0.0.0:9054→9054/tcp, ::
 ca_orderer
0706b58c8c38   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   6 minutes ago   Up 6 minutes   0.0.0.0:7054→7054/tcp, ::
 ca_org1
```

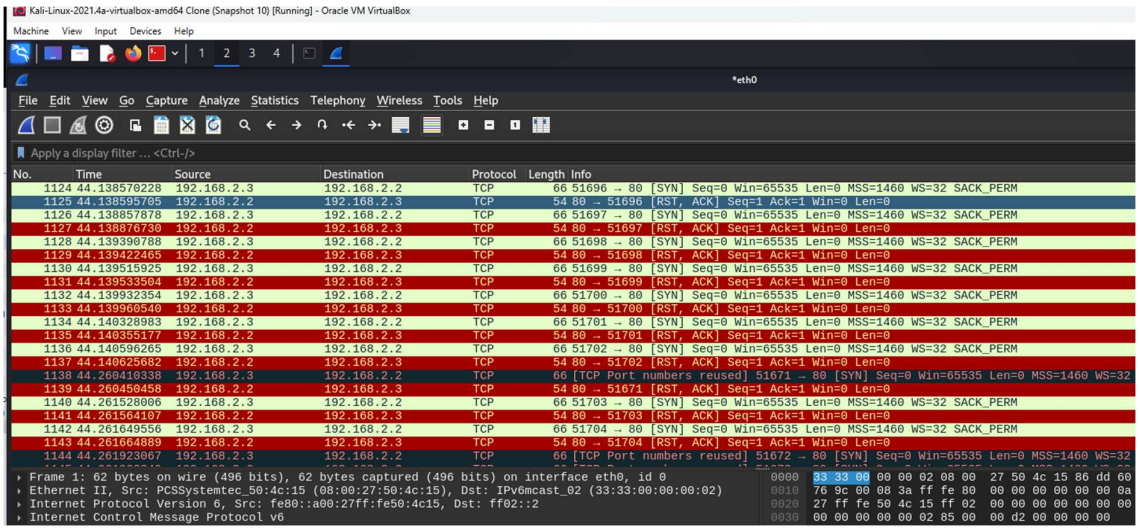
Channel activation on HLF to communicate with the intended IoT nodes only. Nodes/devices not authorized cannot communicate inside the channel.

```
ndorsement":' '{' '"mod_policy":' '"','  '"policy":' null, '"
"' '}' '},' '"values":' '{' '"AnchorPeers":' '{' '"mod_polic
 '"',' '"value":' null, '"version":' '"0"' '}' '},' '"versio
n":' '"0"' '}' '}}}}'
++ configtxlator proto_encode --input config_update_in_envel
2024-04-22 21:54:22.431 EDT 0001 INFO [channelCmd] InitCmdFa
2024-04-22 21:54:22.442 EDT 0002 INFO [channelCmd] update →
Anchor peer set for org 'Org2MSP' on channel 'iotchannel'
Channel 'iotchannel' joined
```

DDoS Attack performed from Windows 10 devices on IOT3 using HOIC DDoS too in an internal network setting, isolating the attack internal to the network:

3. Experiments Results

DDOS attack using HOIC program through Kali,



Prerequisites to check npm, node, and python

root@osboxes:/home/osboxes/Downloads/fabric-samples/test-network# npm -v
10.5.0
root@osboxes:/home/osboxes/Downloads/fabric-samples/test-network# node -v
v20.12.2
root@osboxes:/home/osboxes/Downloads/fabric-samples/test-network# python --version
Python 2.7.18
root@osboxes:/home/osboxes/Downloads/fabric-samples/test-network#

./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go

./network.sh deployCC -ccn iotchannel -ccp ../asset-transfer-basic/chaincode-go -c iotchannel

Created a channel for IOT devices to communicate.
./network.sh createChannel -c iotchannel

Setting up couch db database for transactions
sudo ./network.sh up -ca -s couchdb

Useful channel commands

Enabling Peers – First we need to provide path to Peer

export PATH=${PWD}/../bin:$PATH

export FABRIC_CFG_PATH=$PWD/../config/
export PATH="/usr/bin:$PATH"
go path
`export PATH=$PATH:/usr/local/go/bin`

check paths after running test chaincode
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/pe
er0.org1.example.com/tls/ca.crt
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin
@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051

For docker to run you need to have the ban to the bin
export PATH="/usr/bin:$PATH"

Create a channel.
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl

User full network restart commands in Linux
sudo service network-manager restart

sudo service networking restart

export GOPATH=$HOME/go
export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
export PATH=$HOME/fabric-samples/bin:$PATH
export PATH=$PATH:$GOPATH/bin
export FABRIC_CFG_PATH=$HOME/fabric-samples/config/

Invoking chaincode to test a query to check communication between devices.

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C iotchannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C iotchannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'

Query result below from iot2 device on the default chain code supplied.

Install VScode- download the deb file from the VScode site into the downloads.
sudo dpkg -i /path/.deb
Writing Chain code for IOT channels and devices in VS-code
Add it to the Gochain folder with admin access
chown -R osboxes/home/osboxes

---

3 Code for BC execution on IoT devices and prevention of DDoS.

Smart contract for IoT – was not able to be implemented due to errors in configurations.

Chain code for IoT devices to exchange parameters for trading information.
2 x IoT devices named Alice and Bob which are smart temperature sensors that will be exposed to DDoS attack.

Parameters to exchange information based on BC contract in the chain code. These include device id , device Ip address, Owner, and Reading of the sensor.

Function to check if the device already exists in the HLF ledger, if not then add the trusted device based on the parameter to the ledger and update the BC block.

```
package chaincode

import (
    "encoding/json"
    "fmt"

    "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

// SmartContract provides functions for managing IoT devices
type SmartContract struct {
    contractapi.Contract
}

// Device represents an IoT device
type Device struct {
    ID            string `json:"ID"`
    Name          string `json:"Name"`
    IPAddress     string `json:"IPAddress"`
    Owner         string `json:"Owner"`
    SensorReading int    `json:"SensorReading"`
}

// InitLedger adds a base set of IoT devices to the ledger
func (s *SmartContract) InitLedger(ctx
contractapi.TransactionContextInterface) error {
    devices := []Device{
        {ID: "device1", Name: "Temperature Sensor", IPAddress:
"192.168.0.1", Owner: "Alice", SensorReading: 25},
        {ID: "device2", Name: "Humidity Sensor", IPAddress:
"192.168.0.2", Owner: "Bob", SensorReading: 60},
        // Add more initial devices as needed
    }

    for _, device := range devices {
        deviceJSON, err := json.Marshal(device)
        if err != nil {
```

```
            return err
        }

        err = ctx.GetStub().PutState(device.ID, deviceJSON)
        if err != nil {
            return fmt.Errorf("failed to put device to world state:
%v", err)

        }
    }

    return nil
}

// CreateDevice registers a new IoT device in the ledger
func (s *SmartContract) CreateDevice(ctx
contractapi.TransactionContextInterface, id, name, ipAddress, owner
string, sensorReading int) error {
    exists, err := s.DeviceExists(ctx, id)
    if err != nil {
        return err
    }
    if exists {
        return fmt.Errorf("the device with ID %s already exists", id)
    }

    device := Device{
        ID:            id,
        Name:          name,
        IPAddress:     ipAddress,
        Owner:         owner,
        SensorReading: sensorReading,
    }
    deviceJSON, err := json.Marshal(device)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(id, deviceJSON)
}

// ReadDevice retrieves details of an IoT device from the ledger
func (s *SmartContract) ReadDevice(ctx
contractapi.TransactionContextInterface, id string) (*Device, error) {
    deviceJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
        return nil, fmt.Errorf("failed to read device from world state:
%v", err)
    }
```

```
    if deviceJSON == nil {
        return nil, fmt.Errorf("the device with ID %s does not exist",
id)
    }

    var device Device
    err = json.Unmarshal(deviceJSON, &device)

    if err != nil {
        return nil, err
    }

    return &device, nil
}

// UpdateDevice updates details of an existing IoT device in the ledger
func (s *SmartContract) UpdateDevice(ctx
contractapi.TransactionContextInterface, id, name, ipAddress, owner
string, sensorReading int) error {
    exists, err := s.DeviceExists(ctx, id)
    if err != nil {
        return err
    }
    if !exists {
        return fmt.Errorf("the device with ID %s does not exist", id)
    }

    device := Device{
        ID:            id,
        Name:          name,
        IPAddress:     ipAddress,
        Owner:         owner,
        SensorReading: sensorReading,
    }
    deviceJSON, err := json.Marshal(device)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(id, deviceJSON)
}

// DeleteDevice removes an IoT device from the ledger
func (s *SmartContract) DeleteDevice(ctx
contractapi.TransactionContextInterface, id string) error {
    exists, err := s.DeviceExists(ctx, id)
    if err != nil {
        return err
    }
```

```
    if !exists {
        return fmt.Errorf("the device with ID %s does not exist", id)
    }

    return ctx.GetStub().DelState(id)
}

// DeviceExists checks if an IoT device with a given ID exists in the
ledger
func (s *SmartContract) DeviceExists(ctx
contractapi.TransactionContextInterface, id string) (bool, error) {
    deviceJSON, err := ctx.GetStub().GetState(id)
    if err != nil {
        return false, fmt.Errorf("failed to read device from world
state: %v", err)
    }

    return deviceJSON != nil, nil
}

// TransferDevice transfers ownership of an IoT device to a new owner
func (s *SmartContract) TransferDevice(ctx
contractapi.TransactionContextInterface, id, newOwner string) (string,
error) {
    device, err := s.ReadDevice(ctx, id)
    if err != nil {
        return "", err
    }

    oldOwner := device.Owner
    device.Owner = newOwner

    deviceJSON, err := json.Marshal(device)
    if err != nil {
        return "", err
    }

    err = ctx.GetStub().PutState(id, deviceJSON)
    if err != nil {
        return "", err
    }

    return oldOwner, nil
}

// GetAllDevices returns all registered IoT devices from the ledger
func (s *SmartContract) GetAllDevices(ctx
contractapi.TransactionContextInterface) ([]*Device, error) {
    resultsIterator, err := ctx.GetStub().GetStateByRange("", "")
```

```
    if err != nil {
        return nil, err
    }
    defer resultsIterator.Close()

    var devices []*Device
    for resultsIterator.HasNext() {
        queryResponse, err := resultsIterator.Next()
        if err != nil {
            return nil, err
        }

        var device Device
        err = json.Unmarshal(queryResponse.Value, &device)
        if err != nil {
            return nil, err
        }/
        devices = append(devices, &device)
    }

    return devices, nil
}
```

IoT device Health Check and update Ledger:

Invoke RecordDeviceHealthMetric function
peer chaincode invoke -o orderer.example.com:7050 -C mychannel -n mychaincode --peerAddresses peer0.org1.example.com:7051 --peerAddresses peer0.org2.example.com:7051 --tls --cafile $ORDERER_CA --waitForEvent -c '{"Args":["RecordDeviceHealthMetric", "device1", "0.75", "0.50", "1000"]}'
// RecordDeviceHealthMetric function with arguments "device1", "0.75" (CPU utilization), "0.50" (memory usage), and "1000" (network bandwidth)

IoT device health check
Query GetDeviceHealthMetrics function
peer chaincode query -C mychannel -n mychaincode -c '{"Args":["GetDeviceHealthMetrics", "device1"]}'
//This command queries device health metrics for "device1" from the ledger.

This will work provided all identity checks are completed via HLF using X.509 certificates.

Sample device enrollment code.
const enrollment = await caClient.enroll({ enrollmentID: 'device1', enrollmentSecret: 'password' });
    const certificate = enrollment.certificate;
    const privateKey = enrollment.key.toBytes();

If unauthorized access is done for example in case of DDoS attack, return error , block transaction , don't update ledger and drop packets.

```
func (t *SensorChaincode) submitSensorData(stub shim.ChaincodeStubInterface, args []string) pb.Response {
  // Check if the correct number of arguments are provided
  if len(args) != 2 {
    return shim.Error("Incorrect number of arguments. Expecting 3: SensorID, SensorData, ipaddress")
  }
```

```
    sensorID := args[0] // arguments on what the id is
    sensorData := args[1] //arguments on what data needs to be sent
sensorIpAddress := args[1] //arguments on checking the white list ip addresses.

    // Validate sensor data (e.g., format, range, etc.)
    // Implement your validation logic here

    // If unauthorized, return an error
    if !isAuthorized(sender) {
        return shim.Error("Unauthorized access. Transaction rejected.")
    }

    // If authorized, update the blockchain with sensor data
    err := stub.PutState(sensorID, []byte(sensorData))
    if err != nil {
        return shim.Error(fmt.Sprintf("Failed to submit sensor data: %s", err))
    }

    return shim.Success(nil)
}
```

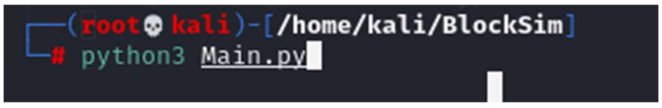
Docker Images – to see all the images downloaded command -- docker images.
Docker inspect peerid for example id 32715a1fc99e
4 peers in 2 x iot devices have been created

dev-peer0.org2.example.com-basic_1.0.1-650b7b4f5a8545d710651dc01edee8cf83518ef4b36a67a08be061ba14da653a-c0a713c435d5298d7e406d76938453c6295ec064a3b830682335e59970b8be78 latest 32715a1fc99e 3 hours ago 146MB
dev-peer0.org1.example.com-basic_1.0.1-650b7b4f5a8545d710651dc01edee8cf83518ef4b36a67a08be061ba14da653a-20086c6927e952735a27a90ceb7546bf364568aa2d1f2d3b0a8fde53dfe8f69d latest f945e70ceb23 3 hours ago 146MB
dev-peer0.org2.example.com-iotchannel_1.0.1-c26839f4b62a92b2d4f61f32c752f584785b1319b9899c0206cf9b6406aec653-d3b2f284144e06b7a5c02faf4a436294beac8107a1b170d9568c85c311085957 latest 929eafd081b5 5 hours ago 146MB
dev-peer0.org1.example.com-iotchannel_1.0.1-c26839f4b62a92b2d4f61f32c752f584785b1319b9899c0206cf9b6406aec653-fb78d9a364cbc6e9dfac1f7a3cab9f186ef6ba4c04007e93ebe79acf4faee222 latest 68641a63b007 5 hours ago 146MB

Error response from daemon: could not choose an IP address to advertise since this system has multiple addresses on interface enp0s3 (2a02:8085:a13f:d400:30f5:b382:f70e:5a01 and 2a02:8085:a13f:d400:f2b2:e6a6:b87b:77f8) - specify one with --advertise-addr

Appendix:

Blocksim command to run python script based on input configuration file to produce metrics related to blockchain transactions. This would generate results based on input parameters that can be tweaked in the python files that comes with the GitHub package.



Blocksim tests

Sim bloc *tools for simulation*
https://github.com/dsg-titech/simblock?tab=readme-ov-file

Blocksim – Very Basic – base model, which displays results using a python script – Display results in Excel based on defined parameters. You can change parameters like how many nodes, time interval, delays, transaction sizes etc.

No. of Gateways	Total No. of Devices	Total No. of Blocks	Blocks per Chain	Max TX List Size	Total Transcations	Average Transaction Latency	Transaction Throughput	Simulation Duration (secs)
2	20	44	22	2	200	0.011113424	20.16857827	9.916415395

	Gateway Node ID	Tx ID	Sender Node ID	Receiver Node ID	Tx Creation Time	Tx Reception Time	Tx Insertion Time
0	a	27440685686	1	a	0.730312513	0.730351295	0.730586785
1	a	23124667614	1	a	1.037450632	1.039979267	1.040017038
2	a	58025658298	1	a	2.550591725	2.550596747	2.551229676
3	a	60018075161	1	a	3.465271415	3.465584985	3.466068182
4	a	63543793607	1	a	4.565362441	4.565403302	4.586081151
5	a	69178111606	1	a	5.113384424	5.11394751	5.114829445
6	a	53052698521	1	a	6.711109043	6.711449953	6.718420705
7	a	31827579992	1	a	7.173056257	7.173183076	7.173549451
8	a	7582290679	1	a	8.259498639	8.259863271	8.260111099
9	a	10127613168	1	a	9.803035035	9.803609065	9.803929395
10	a	58556546935	2	a	0.756028451	0.75741081	0.757742441
11	a	95828212620	2	a	1.333090824	1.333443723	1.333858823
12	a	15648256448	2	a	2.84858609	2.849944665	2.850187733
13	a	3343080944	2	a	3.989483794	3.990571687	3.991436389
14	a	63611354039	2	a	4.214614243	4.214673964	4.214778563
15	a	74125809628	2	a	5.141426675	5.143941021	5.155730828
16	a	12104822736	2	a	6.915304673	6.916090544	6.916635084
17	a	87699028305	2	a	7.916848482	7.917040383	7.925296135
18	a	85664643852	2	a	8.520957433	8.521062494	8.52129558

Results | InputConfig | GatewayBlockchains | GatewayTransactions | transaction_lat