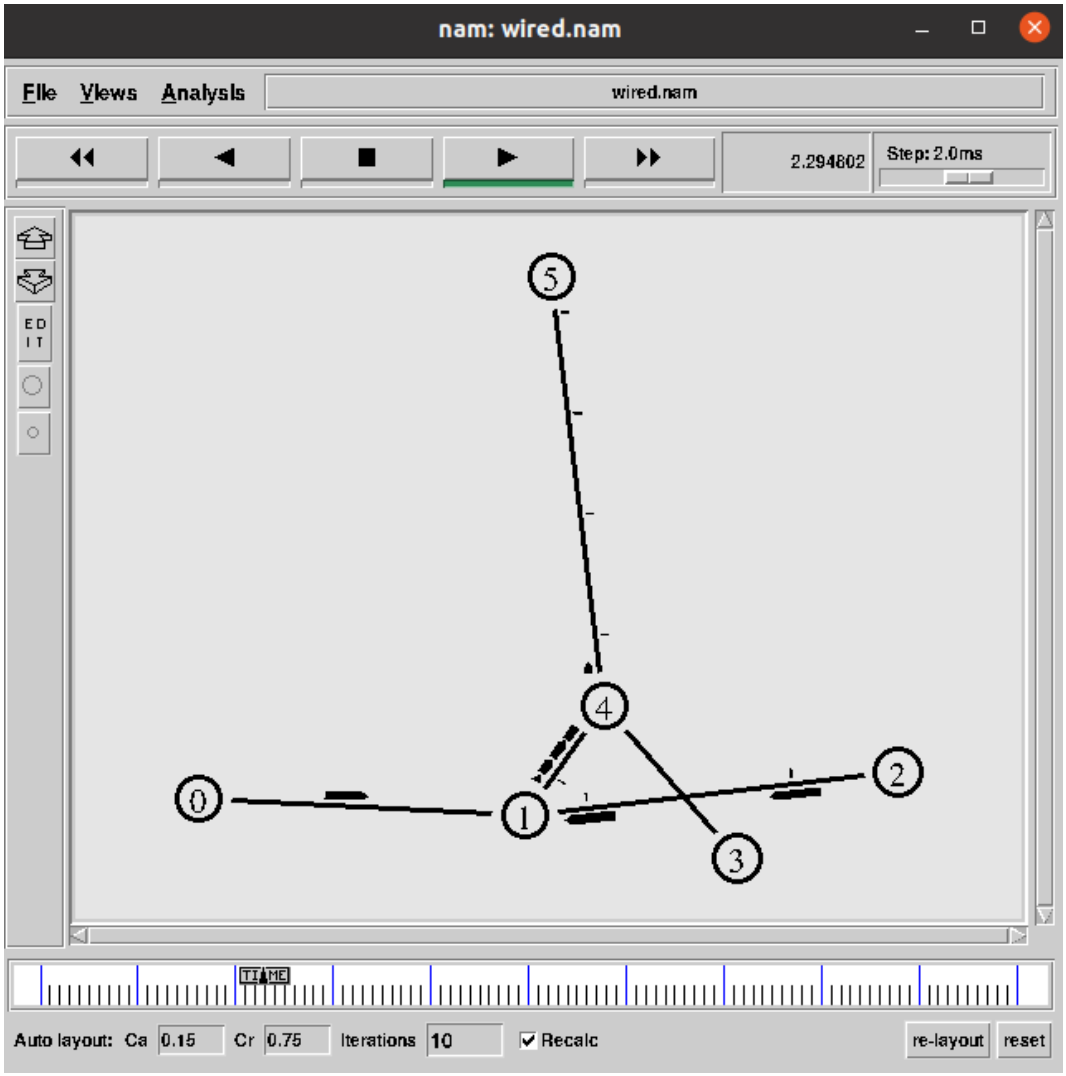
S . NO	EXPERIMENT TOPIC
01	WIRED NETWORK (NS2)
02	WIRELESS NETWORK (NS2)
03	TRAFFIC (SUMO)
04	ROAD NETWORK FROM OPENSTREETMAP (SUMO)
05	WORKSHOP – IOTIFY
06	WORKSHOP – OPPORTUNISTIC NETWORK
07	WORKSHOP – SECURITY ANALYTICS
08	WORKSHOP – MOBILE SECURITY

WIRED NETWORK (NS2)

```
# Create a simulator object
set ns [new Simulator]
# Create a trace file, this file is for logging purpose
set tracefile [open wired.tr w]
$ns trace-all $tracefile
# Create a animation infomration or NAM file creation
set namfile [open wired.nam w]
$ns namtrace-all $namfile
# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
# Creation of link between nodes with DropTail Queue
#Droptail means Dropping the tail.
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n2 $n1 10Mb 5ms DropTail
$ns duplex-link $n1 $n4 3Mb 10ms DropTail
$ns duplex-link $n4 $n3 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 4Mb 10ms DropTail
# Creation of Agents
# Node 0 to Node 3
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
```

```
# Creation of TCP Agent
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
# Creation of Application CBR, FTP
# CBR - Constant Bit Rate (Example nmp3 files that have a CBR or 192kbps, 320kbps, etc.)
# FTP - File Transfer Protocol (Ex: To download a file from a network)
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# Start the traffic
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 10.0 "finish"
# The following procedure will be called at 10.0 seconds
proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
 close $namfile
 exit 0
}
puts "Simulation is starting..."
$ns run
```

Result


nam: wired.nam
File
Views
Analysis
wired.nam
2.294802
Step: 2.0ms
EDIT
5
4
0
1
2
3
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset

WIRELESS NETWORK (NS2)

```
#Step 1 initialize variables
set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model

set val(netif)          Phy/WirelessPhy            ;# network interface type WAVELAN DSSS
2.4GHz

set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type

set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq

set val(nn)             6                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)  500   ;# in metres
set val(y)  500   ;# in metres
#Adhoc OnDemand Distance Vector
#Step 2 - Create a Simulator object
set ns [new Simulator]
#step 3 - Create Tracing and animation file
set tracefile [open wireless.tr w]
$ns trace-all $tracefile
set namfile [open wireless.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)
#step 4 - topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

```
#step 5 - GOD - General Operations Director
create-god $val(nn)
#Step 6 - Create Channel (Communication PATH)
set channel1 [new $val(chan)]
#step 7 - Create nodes
$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace ON \
  -macTrace ON \
  -routerTrace ON \
  -movementTrace ON \
  -channel $channel1
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
```

$n5 random-motion 0

#step 8 - Position of the nodes (Wireless nodes needs a location) and any mobility codes (if the nodes are moving)

$ns initial_node_pos $n0 20

$ns initial_node_pos $n1 20

$ns initial_node_pos $n2 20

$ns initial_node_pos $n3 20

$ns initial_node_pos $n4 20

$ns initial_node_pos $n5 50

#initial coordinates of the nodes

$n0 set X_ 10.0

$n0 set Y_ 20.0

$n0 set Z_ 0.0

$n1 set X_ 210.0

$n1 set Y_ 230.0

$n1 set Z_ 0.0

$n2 set X_ 100.0

$n2 set Y_ 200.0

$n2 set Z_ 0.0

$n3 set X_ 150.0

$n3 set Y_ 230.0

$n3 set Z_ 0.0

$n4 set X_ 430.0

$n4 set Y_ 320.0

$n4 set Z_ 0.0

$n5 set X_ 270.0

$n5 set Y_ 120.0

$n5 set Z_ 0.0

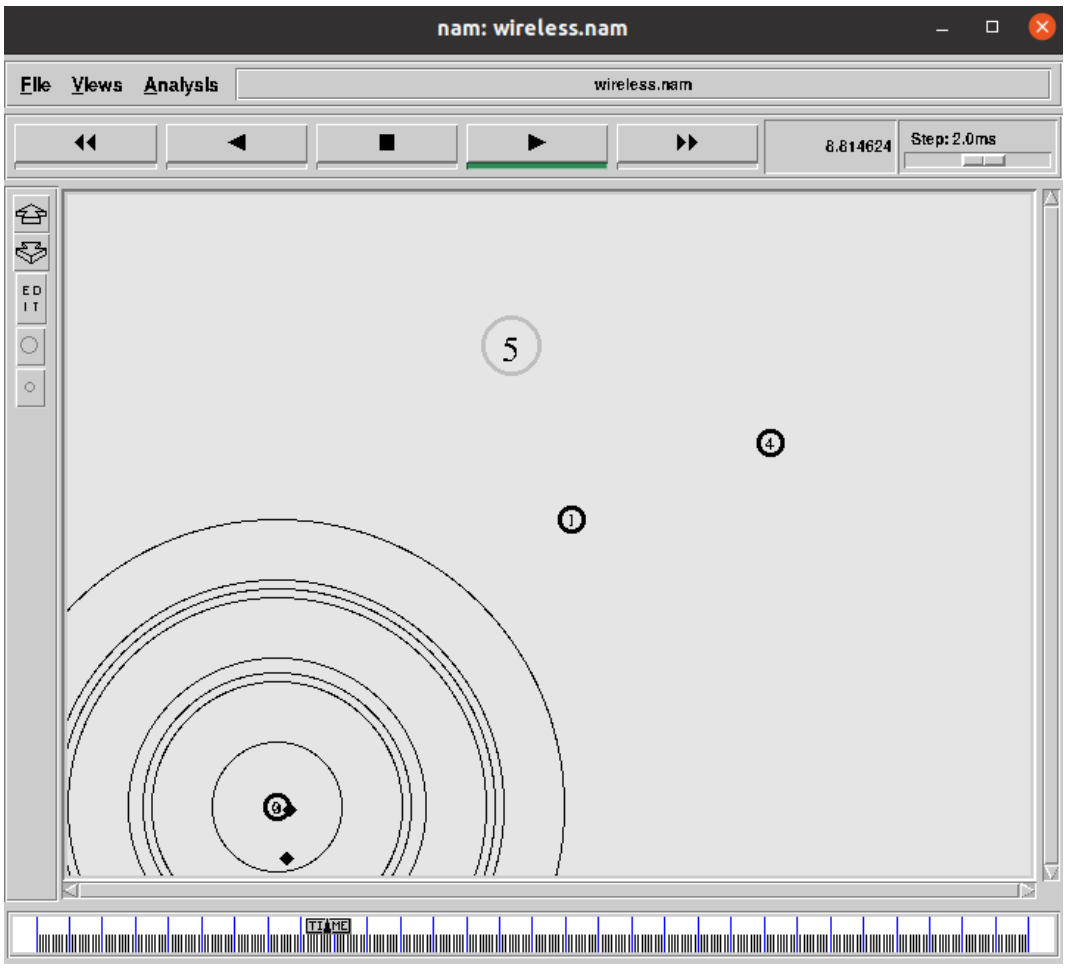
#Dont mention any values above than 500 because in this example, we use X and Y as 500,500

#mobility of the nodes

```
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n1 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n4 setdest 300.0 130.0 5.0"
$ns at 1.0 "$n5 setdest 190.0 440.0 15.0"
#the nodes can move any number of times at any location during the simulation (runtime)
$ns at 20.0 "$n5 setdest 100.0 200.0 30.0"
#step 9 - TCP, UDP Traffic
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n2 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns at 1.0 "$cbr start"
$ns at 30.0 "finish"
```

```
proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
 close $namfile
 exit 0
}
#run the simulation
puts "Starting Simulation"
$ns run
```

Result



TRAFFIC (SUMO)

my_nodes.nod.xml

```
<nodes>
<node id="n1" x="-500" y="0" type="priority"/>
<node id="n2" x="-250" y="0" type="traffic_light"/>
<node id="n3" x="-150" y="200" type="traffic_light"/>
<node id="n4" x="0" y="0" />
<node id="n5" x="150" y="250" />
</nodes>
```

my_edges.edg.xml

```
<edges>
<edge from="n1" to="n2" id="1to2" type="3L45"/>
<edge from="n2" to="n3" id="2to3" type="2L15"/>
<edge from="n3" to="n4" id="3to4" type="3L30"/>
<edge from="n4" to="n5" id="out" type="3L30"/>
</edges>
```

my_type.type.xml

```
<edges>
<edge from="n1" to="n2" id="1to2" type="3L45"/>
<edge from="n2" to="n3" id="2to3" type="2L15"/>
<edge from="n3" to="n4" id="3to4" type="3L30"/>
<edge from="n4" to="n5" id="out" type="3L30"/>
</edges>
```

Generate my_net.net.xml from my_nodes.nod.xml, my_edges.edg.xml, and my_types.type.xml

```
netconvert --node-files my_nodes.nod.xml --edge-files my_edges.edg.xml -t
my_types.type.xml -o my_net.net.xml
```

my_route.rou.xml

```
<routes>
<vType accel="1.0" decel="5.0" id="car" length="2.0" maxSpeed="50.0" sigma="0.0" />
<vType accel="1.0" decel="5.0" id="bus" length="12.0" maxSpeed="10.0" sigma="0.0" />
<route id="route0" edges="1to2 2to3" />
<vehicle depart="10" id="veh0" route="route0" type="bus"/>
<route id="route1" edges="2to3 3to4" />
<vehicle depart="30" id="veh1" route="route1" type="car"/>
<route id="route2" edges="3to4 out" />
<vehicle depart="30" id="veh2" route="route2" type="car"/>
</routes>
```

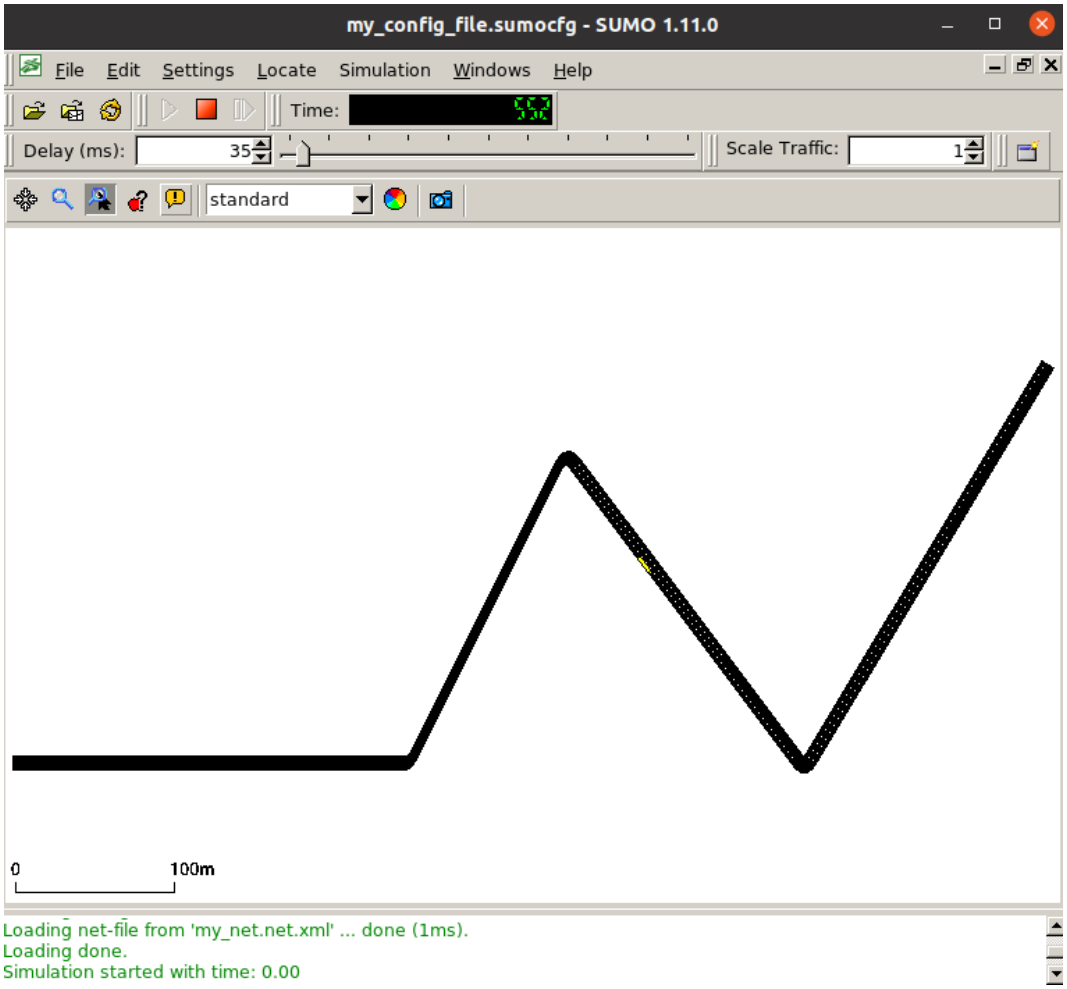
my_config_file.sumocfg

```
<configuration>
<input>
<net_file values=”my_net.net.xml”>
<route_file values=”my_routes.rou.xml”>
<time>
<begin value = “0”/>
<end value= “2000”/>
</time>
</configuaration>
```

display with sumo-gui

```
sumo-gui -c my_config_file.sumocfg
```

Result


my_config_file.sumocfg - SUMO 1.11.0
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms):
35
Scale Traffic:
1
standard
0
100m
Loading net-file from 'my_net.net.xml' ... done (1ms).
Loading done.
Simulation started with time: 0.00

ROAD NETWORK FROM OPENSTREETMAP (SUMO)

Use Openstreetmap.org to download the map and save it to the working directory as map.osm

Use net convert to import the road network stored in map.osm to sumo-network map.net.xml

netconvert –osm-files map.osm -o map.net.xml

Use randomTrips.py to generate a set of random trips for a network

We will get map.rou.xml, map.rou.alt.xml, and trips.trips.xml

python3 $SUMO_HOME/tools/randomTrips.py -n map.net.xml -r map.rou.xml -e 50 -l

map_config.sumocfg

<configuration>

<input>

<net-file value="my_net.net.xml"/>

<route-files value="my_route.rou.xml"/>

</input>

<time>

<begin value="0"/>

<end value="2000"/>

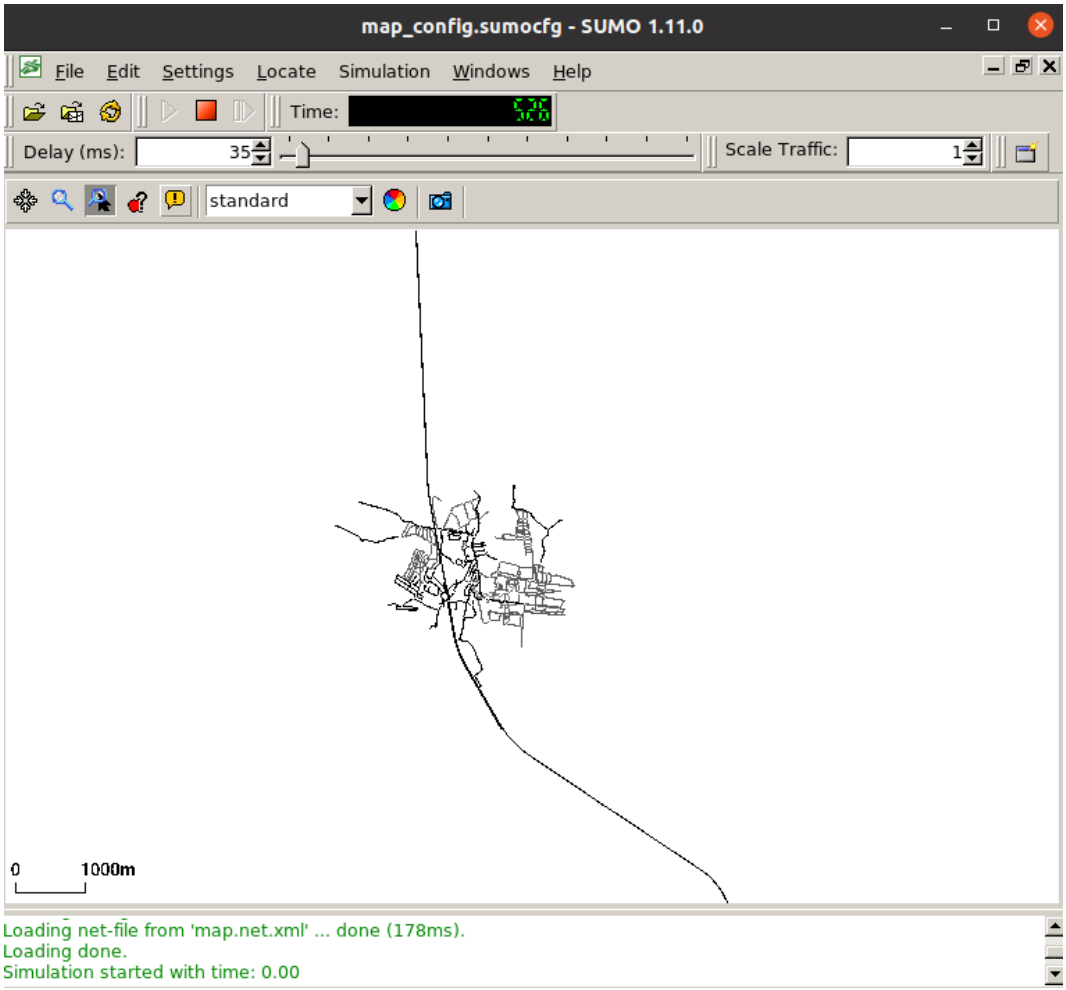
</time>

</configuration>

Display with sumo-gui

sumo-gui -c map_config.sumocfg

Result


map_config.sumocfg - SUMO 1.11.0
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms):
35
Scale Traffic:
1
standard
0
1000m
Loading net-file from 'map.net.xml' ... done (178ms).
Loading done.
Simulation started with time: 0.00

WORKSHOP – IOTIFY

IoTIFY Network Simulator

- A **cloud native** approach, which is horizontally scalable and could be run in any environment. The result is seamless scalability from Cloud, hybrid or event to the local desktops. The IoTIFY network simulator could truly match your IoT cloud platform and scale to test its true performance.
- **Advanced simulation** capabilities such as realistic traffic simulation, weather feeds, location functionality, and custom payload generation, with several helper libraries from a rich Node.js ecosystem.
- An **API driven** interface which can be easily integrated with existing test ecosystem and continuous Integration/ Delivery pipelines. The test results could also be exported to any database of your choice.
- **Scripting** functionality in Javascript ES6, one of the most widely used scripting languages on the planet, we could tap into a rich ecosystem of NPM package ecosystem and could easily integrate Node.js packages into IoTIFY.
- **Multiprotocol support** – The scripting is protocol agnostic, which means testers could easily create several versions of the same test with different protocols. We support COAP, MQTT, HTTP, LWM2M, TCP/UDP out of the box and many other protocols will be supported in the future.
- Native Integration with **most popular IoT cloud platform** such as AWS IoT Core and Azure IoT Hub, enabling easy configuration and provision of certificates for millions of devices with ease.
- Unmatched scalability of upto **1 Million device** endpoints or more. Ability to simulate large complex scenarios such as smart city, electric vehicle charging network etc. with customized dashboards.

Virtual Hardware and Sensors

- Emulate the industrial controller, such as Raspberry Pi and Arduino.
- Emulate the most used peripherals in IoT such as GPIO, I2C, LEDs, SPI, USB, Ethernet etc and buses.
- By providing a comprehensive virtual environment for hardware emulation, we allow rapid prototyping of IoT applications at a fraction of the current time and cost. The application developed on virtual hardware could be transferred and run on physical hardware, matching the physical world with the virtual development. Essentially, with hardware simulation, IoTIFY allows you to:
    - Choose a suitable Industrial platform for target applications.

- Choose the appropriate sensors, actuators and gateways for the application.
- Boot desired operating system with web based IDE.
- Interact with virtual hardware environment via UI and command line.
- Validate the control action of IoT value chain by seeing the actual behaviour

Hardware emulation is extremely useful when trying out custom sensors and developing firm- ware for IoT application. With hardware emulation, the time needed to build a full scale proto- type is drastically reduced from months to days. It also acts as a final stage of system validation before making physical prototypes.

WORKSHOP – OPPORTUNISTIC NETWORK

An Opportunistic Network is a network of wirelessly connected nodes. All nodes are mobile and there rarely exists a complete path between source and destination.
Opportunistic Network is one of the most interesting and recent evolutions of mobile Ad-hoc Networks (MANETS)
they are enabled to deliver messages even when there is no connected path between the source and the destination.
If a suitable node is not found, the node simply stores the message and carries it through the network until a better node or destination is found, therefore share similar routing algo's as used in delay tolerant Networks.

Fundamentally different from the traditional network Intermittent connectivity among the nodes lack of end-to-end paths messages are replicated since routing is non-trivial, content dissemination becomes challenging in OppNets.

Application Areas of OppNet's

1) Interconnect mobile and reserve nodes in disaster areas

2) Scientific wildlife monitoring

3) Defence

4) Environment

5) Sensor networks

6) Post-disaster communication.

Interesting projects deploying OppNets

Zebra Net: A princeton university deployed in vast savanna area of central kenya. It tracks wild species to investigate their behaviour and understand interactions. It's a non-intrusive means to monitor large populations roaming in vast areas and is cost-effective.
Dak Net: it is aimed to provide cost effective connectivity to rural villages in India, where deploying standard internet access is costly.kiosks equipped with digital storage and short range wireless communication are built, Mobile Access Points (MAP) mounted on buses, motorcycles, etc exchange data with kiosk wirelessly.

Data Sharing in OppNet

Forwarding a message from its sender to the nodes of interest is big challenge as source and destination nodes are unaware of each other and may never meet.
A trivial solution is flooding entire network with message, but it would clearly saturate network resources and device resources.
A better solution is to replicate to only selected nodes that have more chances to contact and influence others.

Issues in OppNets

1) Medium access layer
2) key management
3) Routing protocol security issue
4) Energy issue
5) Performance issue

Types of protocols for routing OppNets

1) Infrastructure-based protocols
- Infrastructure in form of info stations.
- Infrastructure in forms of message ferries.

2) Infrastructure less protocols
- Direct Transmision
- first contact
- Epidemic Ratting
- spray and wait
- PRoPHET. Probabilistic Routing using History of Encounters and Transitivity.

Modes of Unauthorised access in OppNets

1) Accidental association
2) Malicious association
3) Identity theft (MAC spooping)
4) Denial of service.
5) Man-in-the-middle-attacks
6) Network Injection.

Security Threats

1) Resource deptation attack :- this is because of limited computing, communications, or storage resources in OppNets.
2) Denial of Service (DoS) :- OppNets are subjected to various Dos attacks

from physical layer to application layer
3) confidentiality disclosure : It is because data is transmitted through a number of intermediate nodes.
4) Privacy invasion: As the message needs to be stored in intermediate node, it makes possible to access identity, location, or other sensitive data.

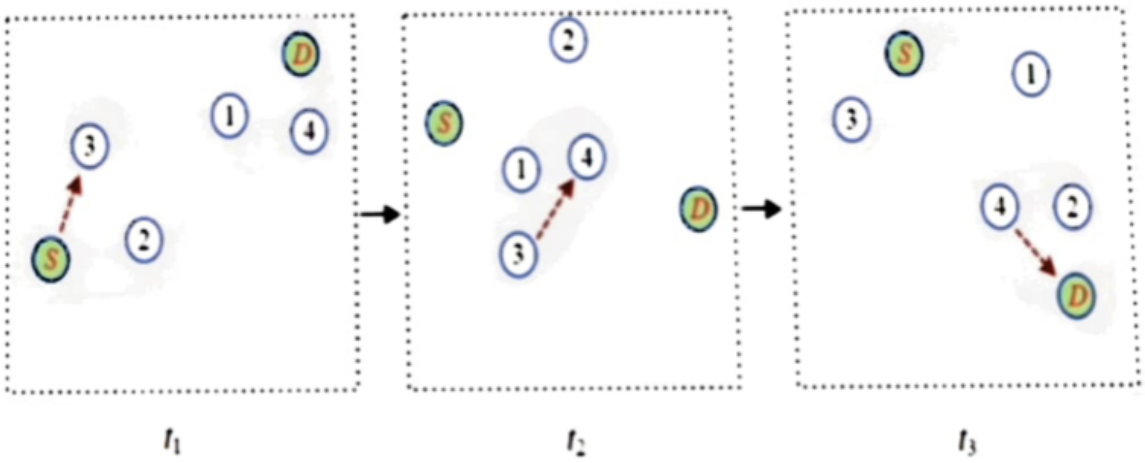


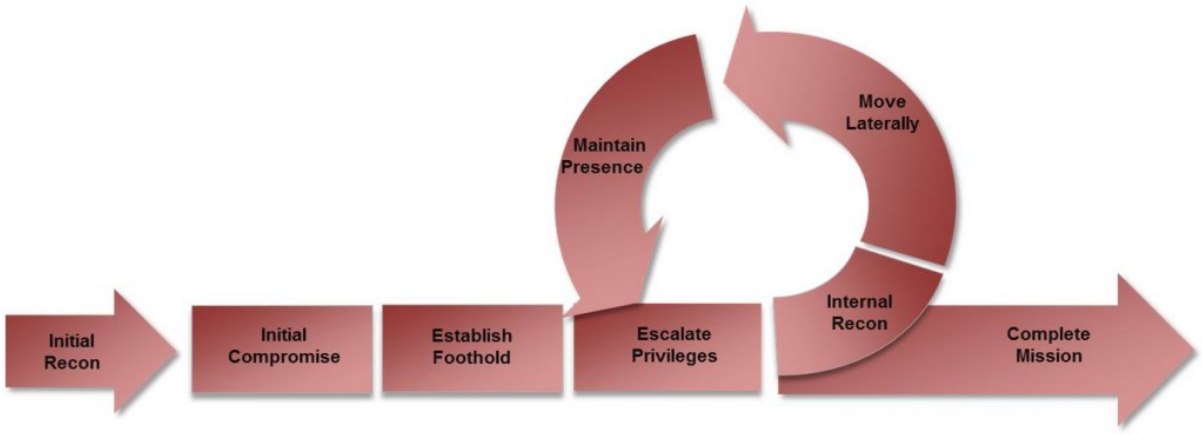
Figure 2: **A message is passed from node S to D via nodes 3 and 4 through the mobility of nodes**

WORKSHOP – SECURITY ANALYTICS

Computer Security

- Processes and mechanisms by which computer based equipment, information, and services are protected from unintended or unauthorized access, change or destruction.
- Computer security also includes protection from unplanned events and natural disasters.
- Cyber Security refers to the technologies, processes, and practices designed to protect networks, devices, applications, and data from any cyber-attacks. Cyber security may also be known as information technology (IT) security.

Attack Life Cycle



Cybersecurity Framework

A baseline of network operations and expected data flows for users and systerns is established and managed Detected events are analyzed to understand attack targets and

Methods. Event data are aggregated and correlated from multiple sources and sensors. Impact of events is determined. Incident alert thresholds are established

Causes of Security Related Issue

- Protocol Error: No one gets it right the first time.

- Software Bugs: Is it a bug or feature?
- Active Attack: Targeting specific devices, BotNets, DDoS (amplification attacks).
- Configuration Mistakes: Very common form of problem.

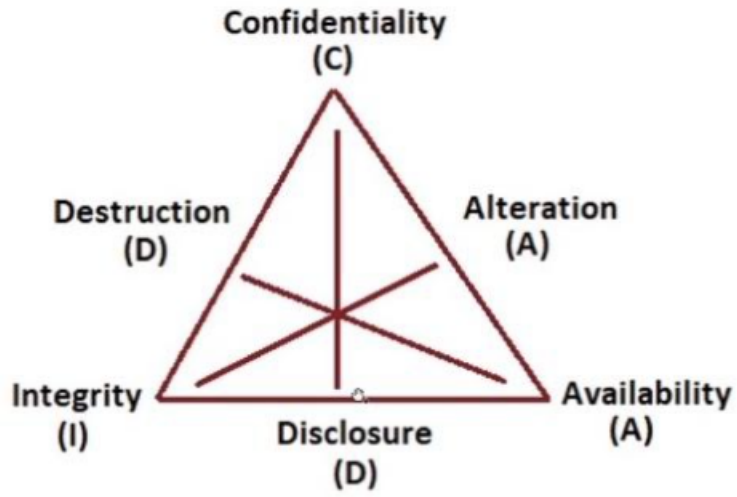
Passive VS Active Attacks

- Passive Attacks
  - Eavesdropping.
  - Offline cryptographic attacks.
- Active Attacks
  - Replay
  - Man-In-The-Middle.
  - Message Insertion.
  - Spoofing (device or user).
  - Denial of Service.
  - Protocol specific attacks.

What are we trying to protect?

- Infrastructure: Routers, switches, and associated data
- Hosts, services, Mail, DNS, ...
- Data: Files, databases, .. .
- Users: Passwords, privileged accesses

The CIA Triad



- Confidentiality: Access to information is restricted to those who are privileged to see it.
- Integrity: Having trust that information has not been altered during its transit from sender to intended recipient.
- Availability: Information or resources are accessible when required.

Security Services

- Authentication: Process of verifying the claimed identity of a device, user/application.
- Authorization: Rights and permissions granted to a user, device or application that enables access to resources.
- Access Control: Means by which authorized user has access to resources.
- Encryption: Mechanism by which information is kept confidential.
- Auditing: Process that keeps track of networked activity.

Policy Framework

- Typical policy framework for a University is an "Acceptable Use Policy"
- AUP: Allowed Activity, Disallowed Activity, Monitoring, Consequence, Process.
- Typical: Computing and network for University-related use only.
- Typical: Shall not interfere with use of computing, network of others. Copyright must be respected, Violators be denied access. Use of computing and network is not private and can be monitored by IT Staff.
- Policy is important as it is a question of behaviour and discipline, not technology.

The Age of Artificial Intelligence and Machine Learning

- The goal of ML is to identify and exploit hidden patterns in "training" data. The patterns learnt are used to analyse unknown data, such that it can be grouped together or mapped to the known groups.

WORKSHOP – MOBILE SECURITY

Mobile Security Threats

Web-Based Mobile Security Threats. Web-based threats are subtle and tend to go unnoticed. They happen when people visit affected sites that seem fine on the front-end but, in reality, automatically download malicious content onto devices.
Mobile Network Security Threats. Network-based threats are especially common and risky because cybercriminals can steal unencrypted data while people use public WiFi networks.
Mobile Device Security Threats. Physical threats to mobile devices most commonly refer to the loss or theft of a device. Because hackers have direct access to the hardware where private data is stored, this threat is especially dangerous to enterprises.

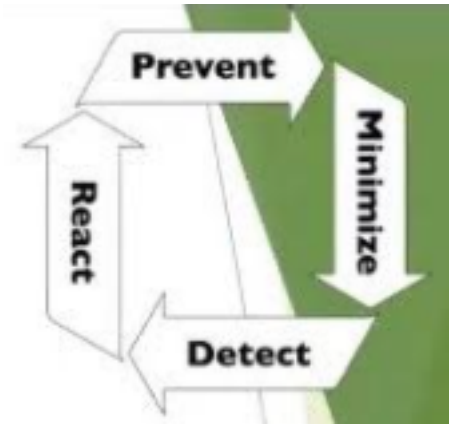
Why is mobile security important?

Nowadays nearly all the tasks that you could only perform on a computer are achievable on mobile devices as well.
more sensitive information will be stored on peoples' mobile devices than before. Employees are even able to do work on their mobile devices.
more risks for proprietary information leaks as well.
Additionally, the number of attempts of cybercrime has been increasing steadily in the recent years.
This is even more important for Android because it is the most targeted platform due to its widespread usage and open source properties.
The need for security is greater than ever for not only consumers, but large enterprises as well.

Security Philosophy

Finite time and resources: Humans have difficulty understanding risk.

Safer to assume that most developers do not understand security and most users do not understand security.



Security philosophy cornerstones

- Need to prevent security breaches from occurring
- Need to minimize the impact of a security breach
- Need to detect vulnerabilities and security breaches
- Need to react to vulnerabilities and security breaches swiftly

Security flaws in Android

The security flaws allow hackers to exfiltrate media files such as photos, videos, and call-recordings. Additionally, criminals could access real-time data, such as microphone data, GPS, and location data. The attackers could also freeze the affected Android phones, rendering them unresponsive.

Security Model in Android

Android is fundamentally based on a multi-party consent-model: An action should only happen if all involved parties consent to it.
If any party does not consent, then the safe-by-default choice is for that action to be blocked.
This is different to the security models that more traditional operating systems implement, which are focused on user access control and do not explicitly consider other stakeholders.

Devices built on top of AOSP (Android Open-Source Project)

Android Google Mobile Services-certified devices
Ex. Pixel, Samsung, devices that use the "Android" name Includes Google apps (GMail, Maps, GMSCore, etc.) Build images must be approved by Google (series of test suites) Devices built on AOSP Ex. Amazon Fire tablets Doesn't include Google apps. "Android-compatible" means the device complies with the Android Compatibility Definition Document (CDD)

Hardware Abstraction Layer Attacks

In Broadcom Company's Wi-Fi chipset, which is used in most of the Android and iOS
devices, a highly dangerous vulnerability was exposed which could allow the attacker to get the control of the complete system by using Booby-Trapped Signals.
Attackers can also use applications to spy on the user's data using Audio Channels,
GPS, Camera and other components as Google Play Services that offers some unsafe features to the applications which can be exploited by the intruders.
If user disables these features, it affects the normal functioning of the application.
Malicious applications installed in to the system can also use the Audio channels to steal the password. Such applications send deceitful commands through speaker of the victim's system to its microphone which behaves as confused deputy.
TalkBack Accessibility Service of Android is used to accomplish the purpose of this attack, as it reads out the password typed by the victim to the attacker.

Application Based Attacks

Applications request for the privileges from the system at the time of installation for smoothly running later on.
It has been found that the Applications which are installed from websites request for unnecessary permissions which are not required for performing their job. On Android, there are no restrictions on how an application
can be written that are required to enforce security; in this respect, native code is sandboxed as interpreted code.
The attackers are misusing the flaws in native libraries and the third party libraries found in the Applications; they use these libraries to request for combined permissions.

Motivation for Malware Developers

Dark Web: A marketplace for criminals to buy and sell mobile malware, which are often sold as component of software packages.

Permission System Exploitation: Permission System forms the important component of Android security mechanism. it restricts the applications from accessing users personal information.

Malware developers are taking advantage of weaknesses of not only Permission System but also exploiting Android Infrastructure susceptibility to attacks. Users simply accept permissions to be able to use the application; giving knowingly or unknowingly consent to the Attackers to get into the system. After crossing this checkpoint malicious application can gain root access to the system by exploiting kernel level vulnerabilities and can initiate Privilege escalation attacks

Lack of Awareness Regarding Security Protocols and Poor Developing Practices: Android developers while developing the application do not give much emphasis on secure communication of data. Using SSL certificates helps in securing the data from intruders. Because of the high cost involved in debugging android applications using SSL certificates, developers prefer using development servers consisting of unauthorized certificates for debugging. If SSL/TLS code is written incorrectly or steps for establishing the secured connection are not followed carefully it can lead to cryptographic attacks.

Lack of Documentation: Lack of documentation by developers can be misleading for the users and for other developers who are not directly involved in coding but with different modules of application development.