School of Computing & Information Sciences

MC^2: P2P Based Network Masquerading for Mission Critical Clouds

Francois D'Ugard, Florida International University
Ming Zhao, PhD. Florida International University

IPOP

Problem

Virtualization is an increasingly popular approach to address rising information technology costs and complexity. Cloud service providers pool hardware resources and allocate them based on the requests of their users. In order to efficiently allocate these resources, providers must aggregate users of different requirements and workloads onto the same physical infrastructure.

This approach increases the likelihood that a malicious user can collocate a virtual machine alongside a target virtual machine in order to extract information or disrupt the cloud's functioning in some way. By observing the flow of network traffic among a group of cooperating virtual machines a malicious user is easily able to identify the critical virtual machine within the group.

My role in the project was the design and implementation of a new VPN controller to connect virtual machines using IPV6 addresses. The controller routes packets among the virtual machines randomly in order to masquerade the communication patterns. Making it difficult to identify the mission critical virtual machine.

System Design

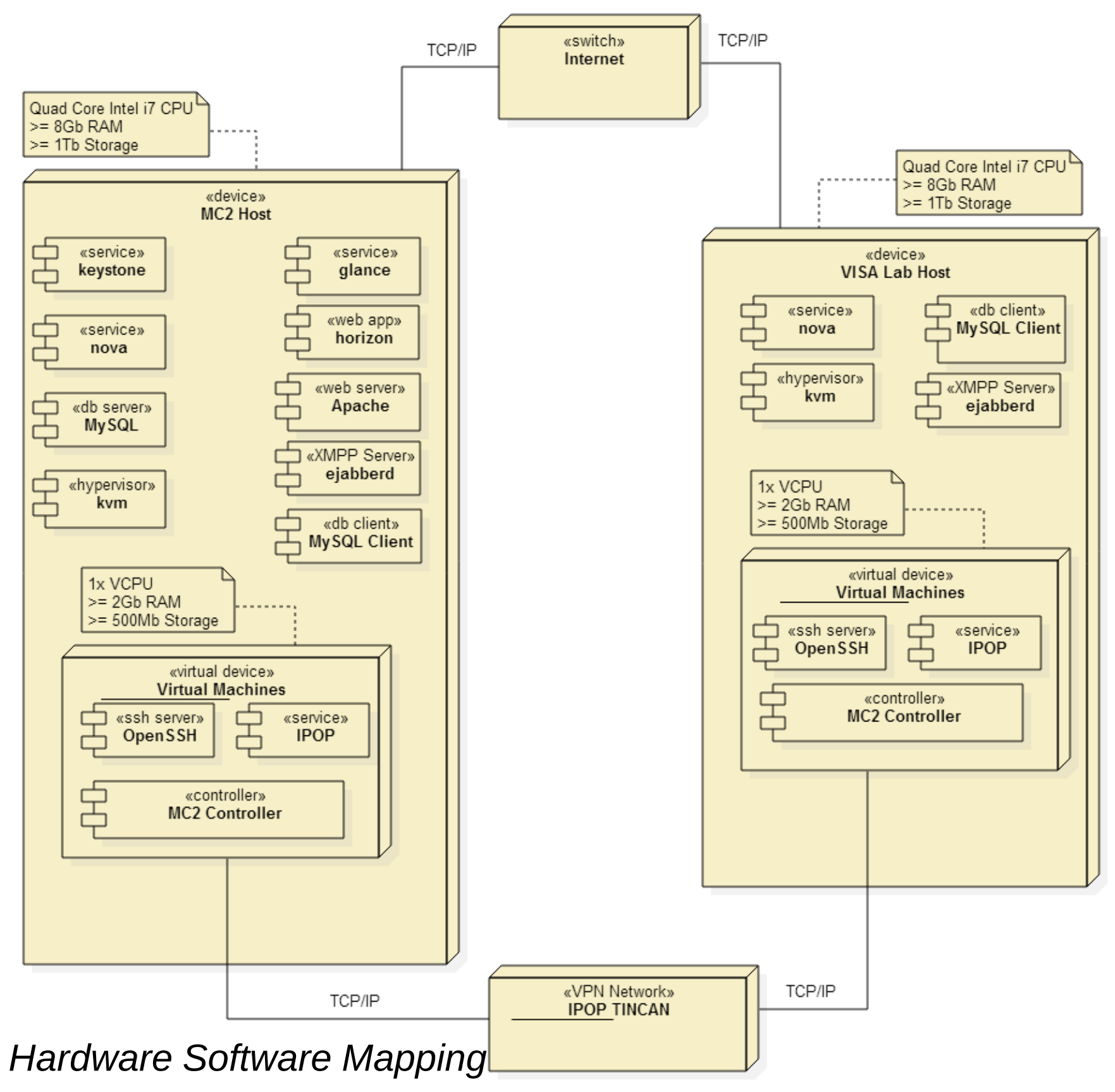


Figure 1. Hardware Software Mapping

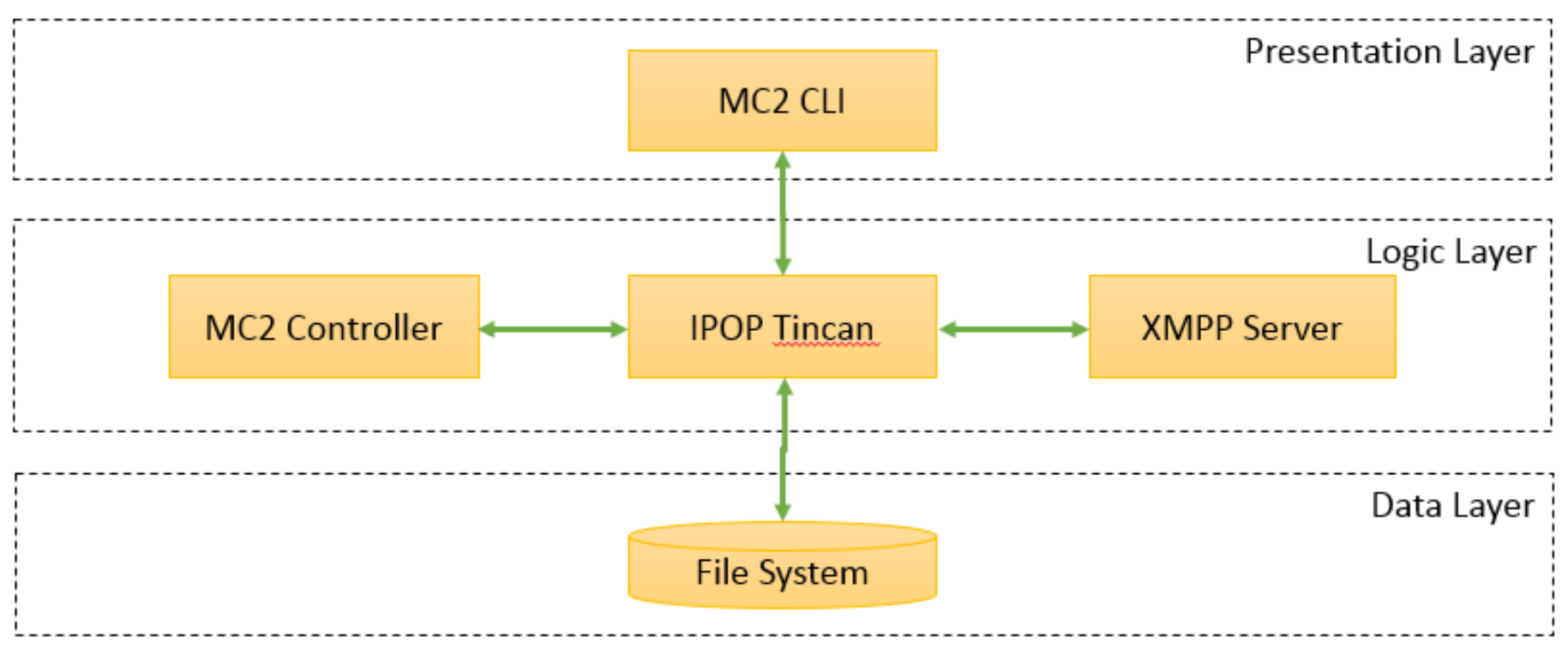


Figure 2. 3 – Tier Software Architecture

Verification

The functions of the system were unit tested using unittest, the Python unit testing framework. Integration testing within the major subsystems was performed using the bottom-up testing strategy.

During system testing, we verified the functionality of our systems components including the random path generator.

In particular, our system testing included the following:

- Usability testing of the command line tools that provide Compatibility testing of IPOP Tincan and the MC2 Controller.
- Functionality testing of the random path generating algorithms.

Current System

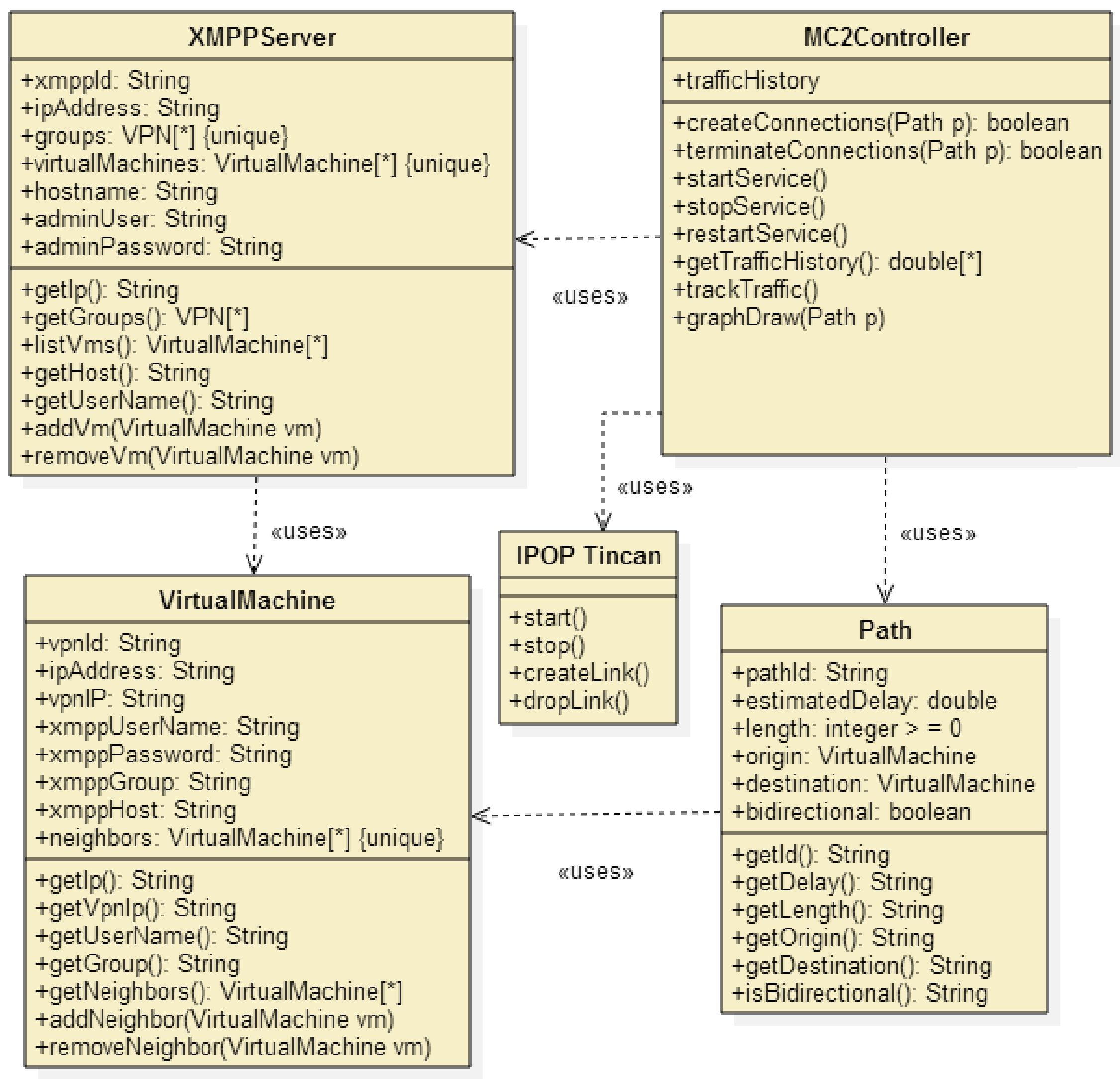
Our project leveraged OpenStack, an open-source Cloud Framework solution used by many organizations.

- Despite robust networking options within OpenStack's networking subsystem, it does not provide a mechanism to mask the communication patterns among virtual machines.
- If a malicious user were to observe the communication patterns among virtual machines finding the mission critical virtual machine is as easy as identifying the machine that receives the most inbound communication.

Proposed System

A P2P Based Network Masquerading system that chooses a random path to route packets within specified latency boundaries at specified intervals. Our system depends our cloud network being a complete graph. This allows us to generate a random path of any length dynamically.
Website: http://mcc-dev.cis.fiu.edu

Object Design



XMPPServer
+xmppId: String
+ipAddress: String
+groups: VPN[*] {unique}
+virtualMachines: VirtualMachine[*] {unique}
+hostname: String
+adminUser: String
+adminPassword: String
+getIp(): String
+getGroups(): VPN[*]
+listVms(): VirtualMachine[*]
+getHost(): String
+getUserName(): String
+addVm(VirtualMachine vm)
+removeVm(VirtualMachine vm)

MC2Controller
+trafficHistory
+createConnections(Path p): boolean
+terminateConnections(Path p): boolean
+startService()
+stopService()
+restartService()
+getTrafficHistory(): double[*]
+trackTraffic()
+graphDraw(Path p)

«uses»
«uses»
«uses»
«uses»
«uses»

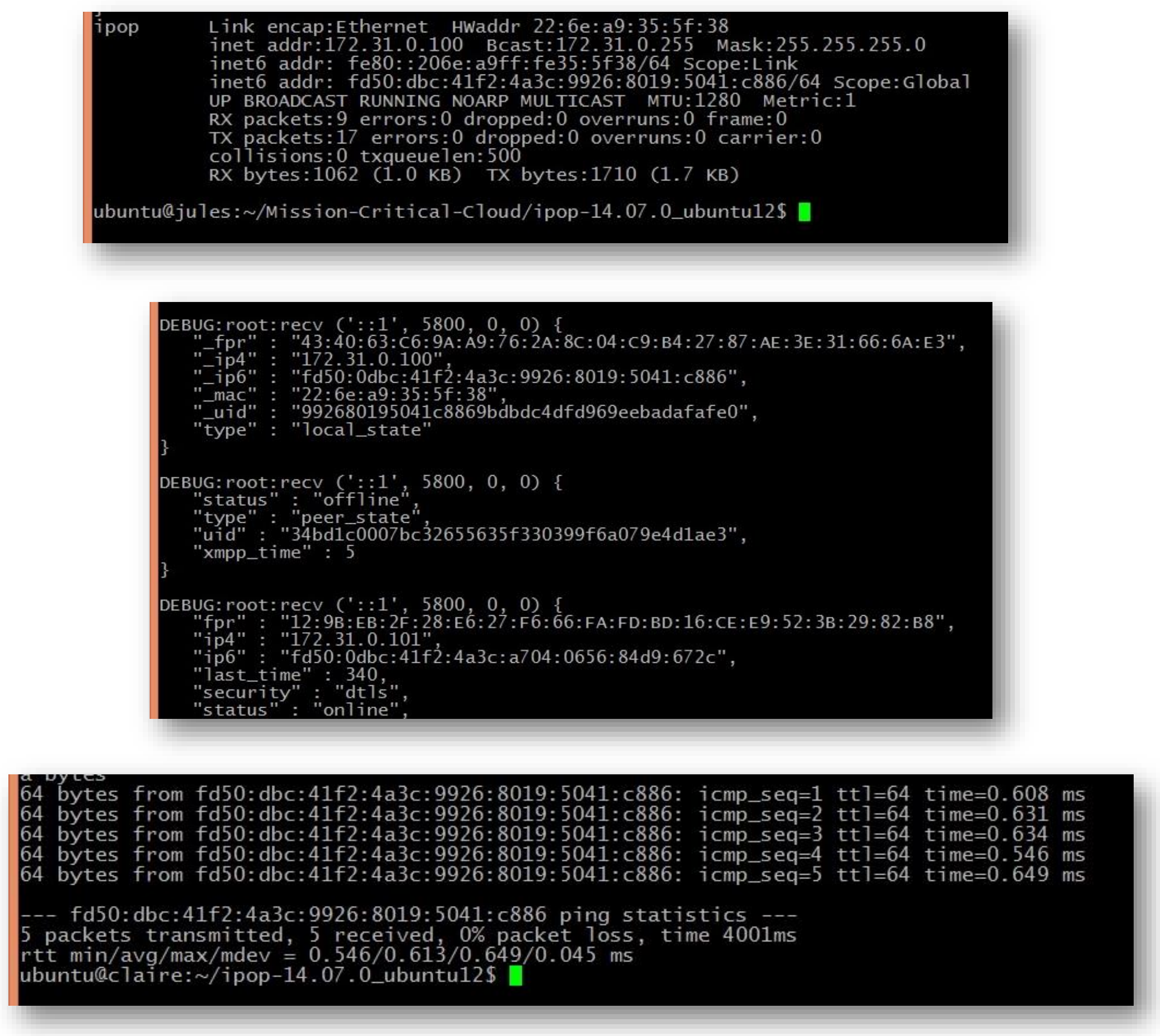
IPOP Tincan
+start()
+stop()
+createLink()
+dropLink()

VirtualMachine
+vpnId: String
+ipAddress: String
+vpnIP: String
+xmppUserName: String
+xmppPassword: String
+xmppGroup: String
+xmppHost: String
+neighbors: VirtualMachine[*] {unique}
+getIp(): String
+getVpnIp(): String
+getUserName(): String
+getGroup(): String
+getNeighbors(): VirtualMachine[*]
+addNeighbor(VirtualMachine vm)
+removeNeighbor(VirtualMachine vm)

Path
+pathId: String
+estimatedDelay: double
+length: integer > = 0
+origin: VirtualMachine
+destination: VirtualMachine
+bidirectional: boolean
+getId(): String
+getDelay(): String
+getLength(): String
+getOrigin(): String
+getDestination(): String
+isBidirectional(): String

Figure 3. Class Diagram

Screenshots



```
ipop      Link encap:Ethernet  HWaddr 22:6e:a9:35:5f:38
          inet addr:172.31.0.100  Bcast:172.31.0.255  Mask:255.255.255.0
          inet6 addr: fe80::206e:a9ff:fe35:5f38/64 Scope:Link
          inet6 addr: fd50:dbc:41f2:4a3c:9926:8019:5041:c886/64 Scope:Global
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1280  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:1062 (1.0 KB)  TX bytes:1710 (1.7 KB)

ubuntu@jules:~/Mission-Critical-Cloud/ipop-14.07.0_ubuntu12$
```

```
DEBUG:root:recv ('::1', 5800, 0, 0) {
   "_fpr" : "43:40:63:C6:9A:A9:76:2A:8C:04:C9:B4:27:87:AE:3E:31:66:6A:E3",
   "_ip4" : "172.31.0.100",
   "_ip6" : "fd50:0dbc:41f2:4a3c:9926:8019:5041:c886",
   "_mac" : "22:6e:a9:35:5f:38",
   "_uid" : "992680195041c8869bdbdc4dfd969eebadafafe0",
   "type" : "local_state"
}

DEBUG:root:recv ('::1', 5800, 0, 0) {
   "status" : "offline",
   "type" : "peer_state",
   "uid" : "34bd1c0007bc32655635f330399f6a079e4d1ae3",
   "xmpp_time" : 5
}

DEBUG:root:recv ('::1', 5800, 0, 0) {
   "fpr" : "12:9B:EB:2F:28:E6:27:F6:66:FA:FD:BD:16:CE:E9:52:3B:29:82:B8",
   "ip4" : "172.31.0.101",
   "ip6" : "fd50:0dbc:41f2:4a3c:a704:0656:84d9:672c",
   "last_time" : 340,
   "security" : "dtls",
   "status" : "online",
```

```
64 bytes from fd50:dbc:41f2:4a3c:9926:8019:5041:c886: icmp_seq=1 ttl=64 time=0.608 ms
64 bytes from fd50:dbc:41f2:4a3c:9926:8019:5041:c886: icmp_seq=2 ttl=64 time=0.631 ms
64 bytes from fd50:dbc:41f2:4a3c:9926:8019:5041:c886: icmp_seq=3 ttl=64 time=0.634 ms
64 bytes from fd50:dbc:41f2:4a3c:9926:8019:5041:c886: icmp_seq=4 ttl=64 time=0.546 ms
64 bytes from fd50:dbc:41f2:4a3c:9926:8019:5041:c886: icmp_seq=5 ttl=64 time=0.649 ms

--- fd50:dbc:41f2:4a3c:9926:8019:5041:c886 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.546/0.613/0.649/0.045 ms
ubuntu@claire:~/ipop-14.07.0_ubuntu12$
```

Figure 6. MC2 Controller Screen Shots

Requirements

FUNCTIONAL:

- Join a VM to the network.
- Disconnect a VM from the network at any time.
- Set up an open-source XMPP server for the collaborating VMs to discover each other and join the network.
- Randomly choose a communication path for packets.
- View the system help.

NON-FUNCTIONAL:

- Support the execution of unmodified applications that use the standard TCP and UDP protocols.
- Not impose constraints in the network infrastructure.
- Make all features available through command-line user interface suitable for a computer savvy user.

Implementation

```
• G = ( V, E )
Multi-hop Path( source s, destination d, hops h )
        if s == d || h <= 1
                return  send_to ( d, h -1 )
        Return send_to ( random.choice ( G.V ), h – 1 )

•  The function MST() returns a minimum spanning tree.
•  The function RDFW() finds a random depth first walk.
Random Source Routing ( Graph G, Max Latency ml )
        if ml < MST ( G )
                Return None
        Return RDFW ( G )
```

Figure 4. Routing Algorithms

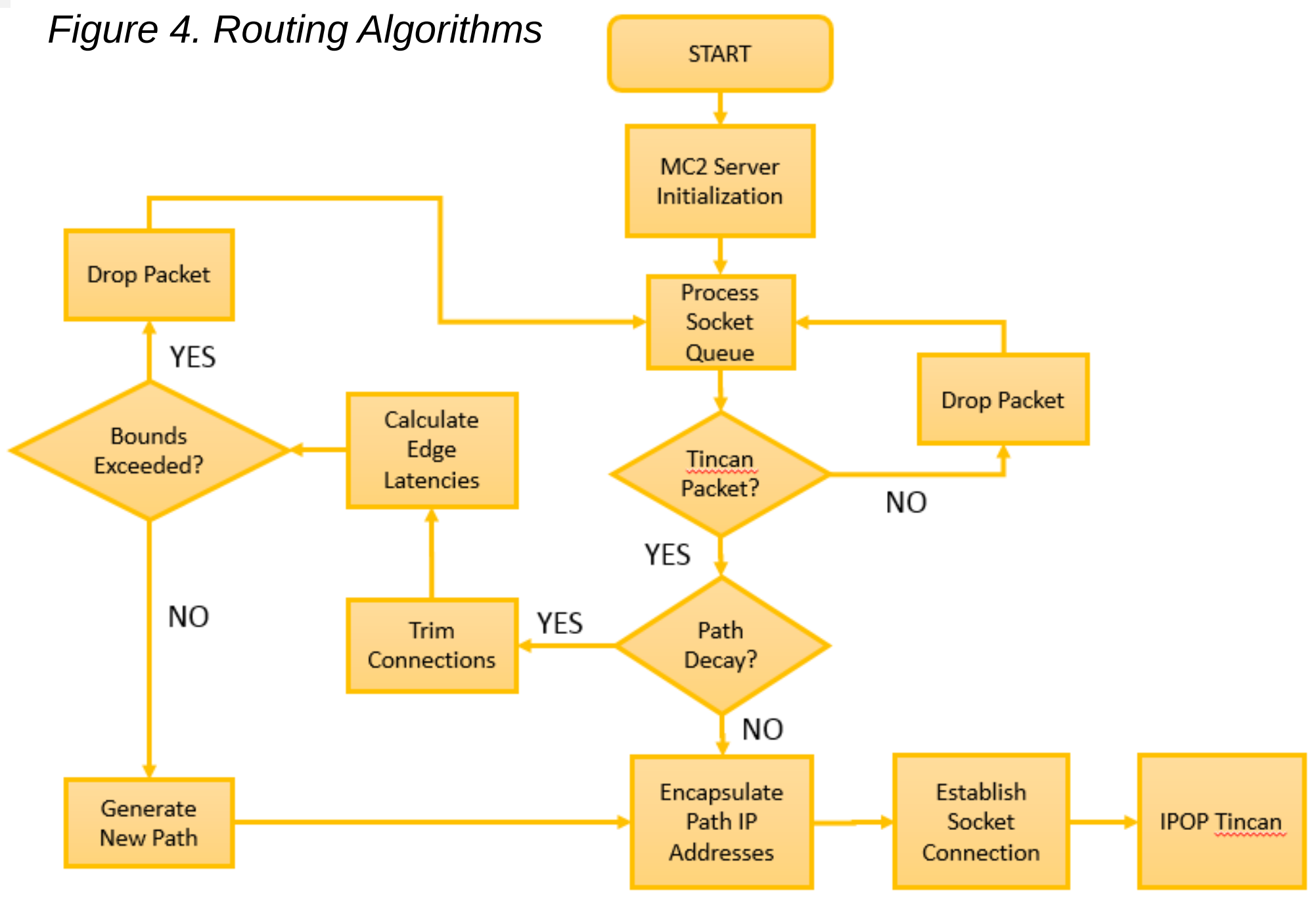


Figure 5. MC2 Controller Event Diagram

Summary

Mission-critical VMs running in clouds may be at risk of attack. These risks are only expected to increase as businesses and governments embrace the Cloud business model as a way of creating cost-effective and easily scalable IT solutions. We propose a solution that transparently enhances VM security using existing building blocks in an innovative way, with the potential of incurring little performance overhead as the state of the art of those basic functionalities advances.

My main contribution to the project was:

- The design and implementation of a new IPOP controller.
- This controller interacts with IPOP Tincan to randomly generate a communication path to route packages within the VPN.

Our proposed system can be extended and improved by:

- Exposing its features through the OpenStack Dashboard.
- Better error handling when no random path is found with acceptable latency boundary.

Acknowledgement

The material presented in this poster is based upon the work supported by the VISA. Lab at FIU I am thankful to the help that I received from my mentor Dr. Ming Zhao and IPOP's author Dr. Renato Figueiredo.