H2020 iP Over IcN– the betTer IP (POINT)

Examples

Deploy Example POINT Topologies and Run ICN, SDN or *-over-ICN Scenarios

List of Authors:

Mays Al-Naday, Sebastian Robitzsch, Mohammed Qasim Al-Khalidi, George Petropoulos

1.Introduction

2. Native ICN Applications Deployment

2.1 Basic ICN Network

2.1.1 Configuration

2.1.1.1 Prerequsitis (credentials and passwordless ssh)

Credentials

Passwordless SSH

2.1.1.2 Management Network

2.1.1.3 ICN Configuration

2.1.2 Network Deployment

2.1.3 Running the Topology Manager

2.2 Example Applications

2.2.1 Video Streaming

2.2.2 ICN Ping

2.2.3 Link-state Monitoring

3. IP-over-ICN Network

3.1 Configurations

3.1.1 IP Endpoint Server

3.1.2 IP Endpoint UE

3.1.3 IP Gateway

3.1.3.1 NAT

3.1.3.2 DHCP Server

3.1.4 Network Attachment Point

3.1.4.1 Prerequisites

3.1.4.2 Host-based Network Attachment Points

3.1.4.3 Routing Prefix-based Network Attachment Points

3.2 Running the Software

3.2.1 Blackadder (Click) and RV

3.2.2 Topology Manager

3.2.3 Network Attachment Points

3.3 Send ICMP Requests from the UE to the Server

4. ICN-over-SDN (Software Defined Network)

4.1 Prepare the Setup

4.1.2 OpenVSwitch

4.1.3 Configuration

4.1.3.1 Prerequsitis (credentials and passwordless ssh)

Credentials

Passwordless SSH

4.1.3.2 SDN Bridge

4.2 Deploy the Network

4.3 Run a Pub/Sub Application

5. Network Simulator 3

6. Dynamic ICN Network

1.Introduction

This document aims to support rapid familiarization with the POINT platform by providing a set of hands-on example network setups that describe how to run the different modules of the POINT platform to construct an ICN or IP-over-ICN network. Each example is self-contained and provides all the necessary instructions to get the network running, without dependency on other examples. Running these examples requires basic knowledge of virtualization softwares, such as VirtualBox, VMware,..etc, as well as familiarity with Linux-based operating systems and interaction with the command line interface. All the examples have been tested using VirtualBox or VMware.

Before diving into these examples, it is recommended to familiarize with modules of the platform by referring to POINT deliverable D3.2 "*Cycle1 Platform Implementation Documentation*" and the README files of each module in their respective directories.

For instructions on how to install the platform and get to know the configuration templates that comes with it, please refer to the HowTo document.

2. Native ICN Applications Deployment

2.1 Basic ICN Network

The objective of this example is to demonstrate how a basic Information-centric network can be setup using the core ICN of our POINT platform (aka Blackadder). For further details about our platform, refer to POINT D3.1.

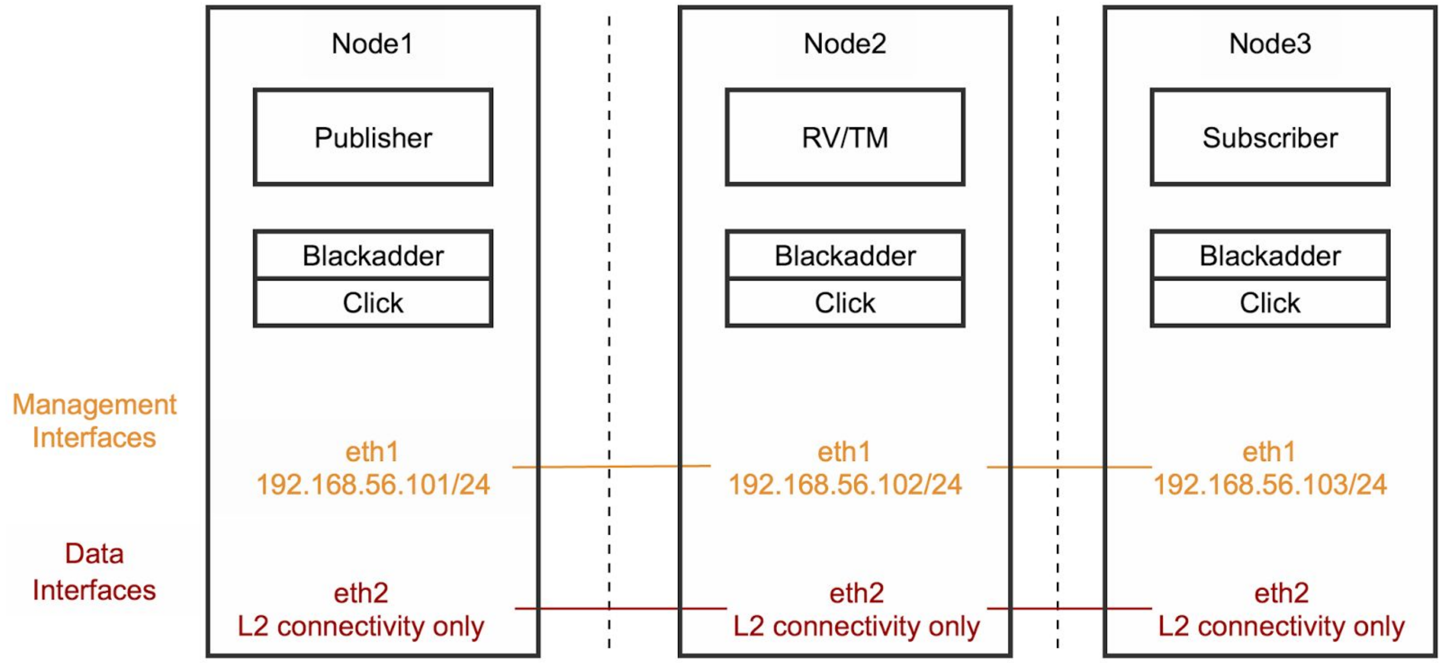

Figure 2.1: Example Basic ICN Setup

The setup consists of three nodes:

- **Publisher**: Provider of Information
- **RV/TM**: Management node operating both the RV/TM core ICN functions. Notice that these function can be operated from two separate nodes; however, for simplicity they have been placed in the same node.
- **Subscriber**: Consumer of Information

Two network interfaces (eth1 & eth2) are required in each node:

- Management interfaces (eth1): these are required for to facilitate the deployment of the ICN topology, including remote login and the transmission of click configuration files to each node, using the deployment tool provided with our platform. For details of how to install the deployment tool, please refer to Section 2.1.3 of the HowTo.
    - Each interface **must be associated with an IP address that provides accessibility to the node**.
    - In this example we associated all interfaces with IPs from the subnet 192.168.56.0/24; however, IP addresses from different subnets can be used as long as they are accessable by the node where you are deploying
- Data Interfaces (eth2): these are required to provide the data connectivity between the ICN nodes. Notice that these interfaces are also used to construct ICN control connectivity, represented by the control path from the RV and the TM to every node in the network. These RV/TM paths are used to communicate control information (e.g. provide a publisher with the FId to transmit information) from the RV/TM to every node.
    - These interfaces don't require IP addresses. If they are assigned IP addresses no error will be produced in this example, but the IP addresses will not be used.
    - In this example, it has been advised to not assign IP addresses to the data interfaces, in order to familiarize the user with this practice as it will be required in other examples.

Notice: It is possible to use a single interface for both management and data connectivity. In this example they have been separated to illustrate a good practice and also to prepare the user for later examples, where this separation is a requirement.

2.1.1 Configuration

2.1.1.1 Prerequsitis (credentials and passwordless ssh)

Credentials

To allow the deployment tool to use the configuration file for constructing the ICN setup and access all three nodes , the latter need to have the same credentials (i.e. username and password) for remote login. To do so, create on each node an account with a fixed username and password (e.g. username = `point`, password = `point`). **This must be the same for all three nodes**.

Passwordless SSH

In order to allow the deployment tool fast access to each node, without having to enter the node password for each remote login process, which can be a very time consuming process, create (if not existent already) a ssh key of the deploying node and copy it to each Nodex in the setup:

```
~$ ssh-keygen -t rsa -b 2048
~$ ssh-copy-id point@192.168.56.x
```

2.1.1.2 Management Network

In each Nodex, configure the management interfaces in your network configuration file (don't forget to replace 'x' in the IP address below with the node number):

```
~$ sudo nano /etc/network/interfaces
```

```
auto eth1
iface eth1 inet static
      address 192.168.56.x
      netmask 255.255.255.0
      broadcast 192.168.56.255
```

2.1.1.3 ICN Configuration

To construct an ICN setup, we have defined a libconfig template that can be used to represent various topologies. for detailed description of this template, refer to Section 3.1 of the HowTo.

For this example setup, the network configuration can be found in `~/blackadder/deployment/examples/basic_3node_icn.cfg`

Notice that, for each node in the configuration file, the IP address of the management interface is used to identify the node for the deployment tool in the field "`tesbed_ip`", while the data interface is used to represent the node connectivity in the "`connections{}`".

2.1.2 Network Deployment

Now that interfaces have been configured, the ICN setup can be deployed from any of the nodes in the setup; or, it can also be deployed from a different, fourth, node as long as it has accessibility to all three nodes. To deploy the network, do (assuming blackadder is in your home directory):

```
~$ cd ~/blackadder/deployment/
~$ ./deploy -c examples/basic_3node_icn.cfg
```

To check if the core ICN platform is running in each node, grap the process ID of click modular router that should be running the ICN configuration file:

```
~$ pgrep click
```

if a process number is returned then the node is running

2.1.3 Running the Topology Manager

After deploying the core ICN nodes, the Topology Manager (TM) application need to run, in order to allow pub/sub relations to be constructed on top of the ICN network. The TM app is the one responsible for constructing delivery paths between publishers and subscribers; without having it running no pub/sub apps can run. The TM uses a topology map of the network, represented by a graphml file named '`topology.graphml`'. The topology map is created by the deployment tool during the network construction phase, for details of `topolgoy.graphml` see Section 3.1.6 of the HowTo.

Now, in this example we have elected Node2 to be the RV/TM of this network and, therefore, Node2 is where you should run the TM app as follows:

```
Node2$ cd ~/blackadder/TopologyManager
Node2$ ./tm /tmp/topology.graphml
```

Now the setup is ready to run publish/subscribe applications as described by some of our examples below.

2.2 Example Applications

2.2.1 Video Streaming

This is a simple, vlc-based, video streaming application, having a video publisher that stores content in a predefined directory being able to publish a video catalogue, which an interested subscriber can subscribed for to view the catalogue and pick a video to be streamed. To run the app, whereby you can stream a video from the publisher of this example to the subscriber, do the following:

- Place a video file, (not too high quality) in the directory (this is where the publisher will look for stored videos) : `/home/point/Videos/`
- In the publisher, cd to `/home/point/blackadder/examples/video_streaming/`
- Allow vlc to run with sudo permissions - many solutions are available on the web for how to enable vlc with sudo (**NOTICE: running vlc with sudo permissions may impose security risks - so if you choose to do this step, you do it at your own risk)**
- In the publisher, run the video publisher: sudo ./video_publisher
- Write the name of the video file (e.g. video file name: `video_sample.mp4`) in the command line window of the publisher
- After that , type "." to indicate the end of the video catalogue
- In the subscriber, run the video subscriber: sudo ./video_subscriber (remember to allow vlc with sudo permissions on the subscriber as well)
- You should receive the video file name on the subscriber command line window, if so then just write the name back in the window and vlc should start now.!

2.2.2 ICN Ping

This app allows ICN nodes to ping each other, similar to conventional IP ping. The app consists of two parts:

- **ping_publisher**: comparable to a 'client' in the IP world and runs on the node that want to issue the ping requests.

- **ping_subscriber**: comparable to a 'server' in the IP world and runs on the node that is to be pinged by other nodes. it basically allows the node to respond to ping requests by issued by different ping_publishers.

The app runs with the following configuration parameters, which can be set through command line arguments:

- `implicit_rv` (`boolean 0|1`): determines whether each ping is a full pub/sub operation involving RV match pub/sub and TM path formation, or only the first ping involves RV/TM while the rest is a communicated directly using the FIDs created and cached from the first ping. The purpose of this variation is to evaluate the RV delay, compared to direct communication. **Notice:** that this is highly dependent on the size of the information data structure maintained by the RV. When only this app is running and nothing else - i.e. the data structure is very small - the difference in delay should be negligible
- `ping_subscriber_NodeID`: The NodeID of the ping_subscriber. This can be the full NodeID (e.g. `00000101`) or only the varying part of the ID (e.g. `101` instead of `00000101`)
- `ping_publisher_NodeID`: The NodeID of the ping_publisher, which should be the same as the NodeID stated in the network configuration file, used with the deployment tool. This can be the full NodeID (e.g. `00000101`) or only the varying part of the ID (e.g. `101` instead of `00000101`). Notice this argument is only set with the `ping_publisher`, not the `ping_subscriber`
- `number_of_ping_requests`: This sets the number of requests to be issued from the `ping_publisher`, default to `1000` if no value is provided for this argument. This argument is only configurable with the `ping_publisher`.

To run the app between two ICN nodes (Node1 and Node3 in the topology of figure 2.1) with RV/TM only involved in the first ping (i.e. `implicit_rv = 0`):

- Consider a scenario where Node1 `(101)` wants to check if Node3 `(103)` is active or not. In this case `Node1` will run the `ping_publisher` and Node3 will run the `ping_subscriber`. Lets also consider to set the number of ping requests to `5`
- In each node, do: `cd ~/blackadder/examples/traffic_engineering/`

- in Node1 do:
    - `sudo ./ping_publisher  0 103 101 5`
- in Node3 do:
    - `sudo ./ping_subscriber 0 103`
- Now, you should start observing ping responses on the command line window, with the first ping response having larger delay than subsequent responses.

2.2.3 Link-state Monitoring

This is a control application that allows the ICN node to periodically announce its presence to its neighbours (i.e. immediately adjacent nodes) and at the same time learn about its neighbours and therefore its connectivity to them, when receiving similar advertisements from each neighbour. The objective of the app is to provide a detection mechanism of link failure scenarios for resilience and path management purposes.

The app operates by sending a 'live' message periodically using the BROADCAST dissemination strategy, which is also link local (i.e. the packet does not traverse more than one link). At the same time, the app maintains a state of all neighbours, from which it received a similar 'live' message, indicating for each neighbor the nodeID and a timestamp of the last received 'live' from this neighbour. Timestamps has a global lifetime setting in the app, which is a configuration parameter that can be set for each node, through a command line argument. **if no setting is provided by the user, the default lifetime is set to 5 seconds**.

When the state of a neighbour exceeds its lifetime without receiving a new message (i.e. the time since the last received 'live' is larger than the set lifetime, which is currently **7 seconds**), the node considers the link between itself and the respective neighbour to have failed and accordingly sends a Link State Notification to the TM, indicating the link that has failed. It is then up to the TM to take respective actions, depending on whether or not it is operating with the extension for resilience or path management.

To run the app, access each node and do:

- `cd ~/blackadder/examples/traffic_engineering`

- `sudo ./broadcast_linkstate_monitor -i NodeID -d lifetime_in_secs`
- When the app runs on all three nodes, Node2 should be receiving 'live' messages from both Node1 (`00000101`) and Node3 (`00000103`)

Notice: for the NodeID, you can give the full nodeID (e.g. `00000101`) or you can provide only the varying part of the NodeID (e.g. `101` instead of `00000101`) and the app will construct the complete ID.

Also notice that this app provides a quick way to check whether your network setup is configured correctly and confirms whether or not each node can see the neighbours it is suppose to connect to, according to the network topology configuration file.

3. IP-over-ICN Network

The main objective of this document is to guide the reader through required steps to set up an IP-to-IP communication an ICN network made up of two nodes. It is considered that the reader has access to POINT's D3.1 deliverable which describes the NAP in further detail.

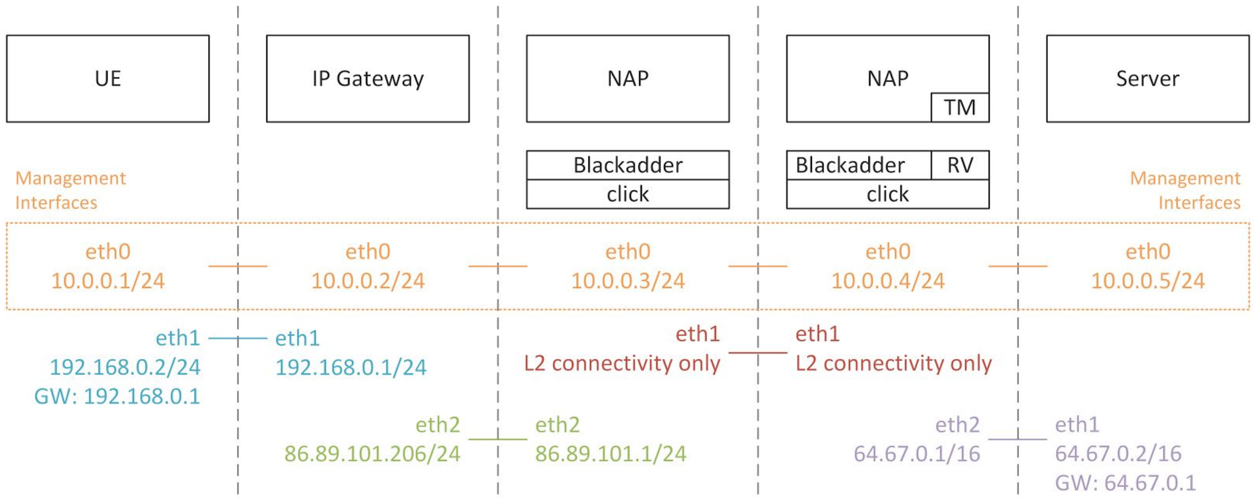

Figure 3.1: IP-over-ICN set-up described in this section

If not mentioned otherwise, it is expected that the user configures the interfaces in all nodes (UE, IP gateway, NAPs and Server) according to Figure 1. All eth0 interfaces coloured in orange are managed interfaces which allows to communicate among all five nodes using standard IP tools, e.g., ssh, scp, telnet or ping.

IMPORTANT: here we describe a NAP (host-based) to NAP (routing prefix-based) scenario to guide the reader through a simple example, where both NAP deployment scenarios are showcased. The deployment mode (host-based / routing prefix-based) of the NAP does not inflict any change in the network, apart from the number of hosts connected to the NAP. Therefore, selecting the deployment mode of the NAP should solely be to suit the scenario of the IP network connected to it.

3.1 Configurations

3.1.1 IP Endpoint Server

The server can be configured either way (static/automatic address assignment). Just ensure that the default GW is 64.67.0.1.

To make this setting permanent add the following line to you interface configuration file (either /etc/network/interfaces or /etc/network/interfaces.d/eth1.conf):

```
iface eth1 inet static
      address 64.67.0.2
      netmask 255.255.0.0
      gateway 64.67.0.1
```

3.1.2 IP Endpoint UE

This IP endpoint has a single interface which must be physically connected to the gateway's eth1 interface. As this how-to guides the user through the set-up of a DHCP server on the IP gateway, the UE's interface should be configured to automatically obtain its IP configuration. Simply verify that /etc/network/interfaces (or in case a dedicated configuration file per NIC is created in /etc/network/interfaces.d/*) has the following entry:

```
iface eth1 inet dhcp
```

and for eth0:

```
iface eth0 inet static
   address 10.0.0.1
   netmask 255.255.255.0
```

Furthermore, ensure that UE's kernel has the appropriate routing table entry to reach the server on 64.67.0.2 via the IP gateway instead of the management interface eth0. To do so,

```
~$ route -n
$ Kernel IP routing table
Destination Gateway     Genmask Flags Metric Ref      Use Iface
0.0.0.0      192.168.0.1 0.0.0.0  UG   1024   0        0 eth1
10.0.0.0     0.0.0.0    255.255.255.0    U    1000    0    0 eth0
192.168.0.0 0.0.0.0    255.255.255.0    U    0     0     0 eth1
```

3.1.3 IP Gateway

The IP gateway must provide NAT with eth2 as the outgoing (public) interface and eth1 as the internal one facing the local IP endpoints.

3.1.3.1 NAT

To do set up the NAT between eth0 and eth1 invoke the following commands:

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
$ iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
$ iptables -A FORWARD -i eth2 -o eth1 -m state --state
RELATED,ESTABLISHED -j ACCEPT
$ iptables -A FORWARD -i eth1 -o eth2 -j ACCEPT
```

To make the NATting a permanent setting which applies after reboot simply add the lines above to /etc/rc.local before the exit 0 statement.

3.1.3.2 DHCP Server

It is recommended that the IP assignment of IP endpoints connected to the IP gateway should be conducted automatically via DHCP. The default DHCP server in Debian-based repositories can be installed via:

```
~$ apt install isc-dhcp-server
```

Now tell the software to bind on eth1 for DHCP requests by changing the value of INTERFACES in /etc/default/isc-dhcp-server to:

```
INTERFACES=”eth1”
```

Now the subnet range has to be configured used by the DHCP server to respond to DHCP request from IP endpoints seeking for an IP. To do so add the following lines to /etc/dhcp/dhcpd.confg:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.2 192.168.0.254;
   option routers 192.168.0.1;
}
```

Before restarting the DHCP server to apply the changes above, ensure that eth1 has been set up according to Figure 1 (192.168.0.1/24). Now restart the DHCP server:

```
$ service isc-dhcp-server restart
```

Sometimes the DHCP server cannot be started automatically when rebooting the system due to some delay in configuring the network interfaces. To overcome this issue add the line above to /etc/rc.local.

3.1.4 Network Attachment Point

3.1.4.1 Prerequisites

As the Kernel is not aware of the NAP, ICMP destination unreachable responses must be switched off:

```
$ sudo iptables -I OUTPUT -p icmp --icmp-type destination-unreachable
-j DROP
```

In order to make this a permanent setting simply add the line to /etc/rc.local, before the exit 0 command.

3.1.4.2 Host-based Network Attachment Points

For setting up the host-based scenario first copy the corresponding NAP example configuration file the directory where the NAP binary resides:

```
$ cd point/apps/nap
$ cp doc/nap-host.cfg nap.cfg
```

The configuration file reflects the scenario illustrated in Figure 3.1. Only change the values if you have different interface names or IP address allocations.

3.1.4.3 Routing Prefix-based Network Attachment Points

For setting up the routing prefix-based scenario copy the corresponding NAP example configuration file to the directory where the NAP binary resides:

```
$ cd point/apps/nap
$ cp doc/nap-prefix.cfg nap.cfg
```

The configuration file reflects the scenario illustrated in Figure 3.1. Only change the values if you have different interface names or IP address allocations.

3.2 Running the Software

3.2.1 Blackadder (Click) and RV

The doc directory of the nap folder comes with an example Blackadder configuration file which reflects the scenario illustrated in Figure 3.1. Only change this file if interface names or IP addresses have been changed.

First, deploy the network of two ICN nodes where the TM and the RVZ are running on the same machine as the NAP. To do so Blackadder comes with a deployment tool which requires sudo privileges to run the underlying click platform. Please ensure that the sudo command does not require a password input on the machines where the NAPs are running (see Section 2.1.1 in the HowTo document) and that SSH keys were distributed:

```
NAP1~$ ssh-copy-id -i .ssh/id_rsa.pub point@10.0.0.3
NAP1~$ ssh-copy-id -i .ssh/id_rsa.pub point@10.0.0.4

NAP2~$ ssh-copy-id -i .ssh/id_rsa.pub point@10.0.0.4
NAP2~$ ssh-copy-id -i .ssh/id_rsa.pub point@10.0.0.3
```

Now the topology can be deployed by running the deployment tool:

```
$ cd point/deployment
$ ./deploy -c examples/ip-over-icn.cfg
```

Note, this starts the RV too.

3.2.2 Topology Manager

Next, start the topology manager:

```
$ cd point/TopologyManager
$ sudo ./tm /tmp/topology.graphml
```

3.2.3 Network Attachment Points

As both ICN nodes have Blackadder running and RV and TM are up, the NAPs can be started to server their IP endpoints. Note, it does not matter whether the NAP facing the UE or the one facing the server is started first.

```
$ cd point/apps/nap
$ sudo ./nap -c nap.cfg
```

3.3 Send ICMP Requests from the UE to the Server

This completes the setup and IP traffic can be sent from the UE to the server and vice versa. Test this by issuing a ping from the UE towards the server over the deployed ICN network:

```
$ ping 64.67.0.2 -c 3
```

4. ICN-over-SDN (Software Defined Network)

This example illustrates the steps required to construct an ICN-over-SDN network setup, using the core ICN part of the POINT platform (aka Blackadder), over which any ICN application can run, including the NAP app to construct an IP-over-ICN-over-SDN setup. Notice that our ICN-SDN implementation comes with two types of SDN connectivity setups:

1. pipeline (`tables`), which allows for pipelined flow tables and requires one bridge per multiple interfaces, and
2. bridges, which also provides pipelined tables but creates a single bridge per interface.

This example shows how to construct the ICN-over-SDN network following the pipeline implementation. The topology considered in this example is shown below in Figure 4.1:

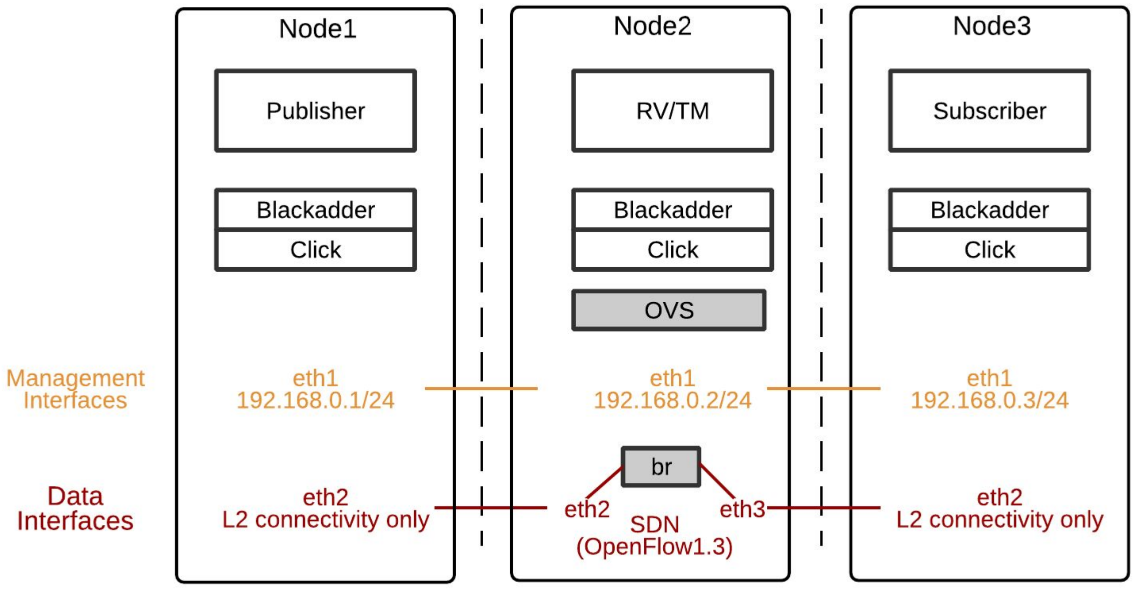

Figure 4.1: ICN-over-SDN Topology

In this topology, management and data connectivity are separated and provided through different interfaces. However, it is possible to provide both networks jointly through the same interface.

4.1 Prepare the Setup

1. Get three nodes - possibly Virtual Machines (VMs) - prepared with Linux based operating system. We have tested this example over Ubuntu 14.04 server edition; however, if losing the desktop is too much for you at this stage, Ubuntu Desktop will do fine as well.
2. Make sure that each machine has at least three interfaces, with the exception of Node2 having 4 interfaces:
   a. Internet interface (`eth0`): this should connect the node to the Internet. if your nodes are VMs, this interface can be configured as NAT interface in VBox or a bridged interface to the NIC of your host machine, which connects to the Internet. **Notice:** this interface is not shown in the diagram as it does not contribute to the network in the examples
   b. Management Interface (`eth1`): this interface is used for managing the network through standard IP tools such as, ssh, scp , ping,..etc.
   c. ICN (data) Interface (`eth2 and eth3 for Node2`): the interface(s) used to establish ICN connections between the nodes. These interface do not require IP addresses, and if one is provided no error will occur, but it will just be ignored; except for Node2, as the interfaces **must not** be given IP addresses
3. Install click and the POINT platform on all three nodes. For instructions on download, installations and configurations, please refer to the HowTo document

4.1.2 OpenVSwitch

OpenVSwitch is required in Node2 to provide SDN connectivity between Node1 and Node3 based on our ICN-SDN forwarding mechanism, described in D3.1. You can either install OVS (version 2.3.0) by itself, or install mininet (2.2.1) with '`–nfv`' options (if the node is intended to host a mininet cluster in the future) and it will come as part of it. For instructions on installing mininet, please refer to the HowTo.

4.1.3 Configuration

4.1.3.1 Prerequsitis (credentials and passwordless ssh)

Credentials

To allow the deployment tool to use the configuration file for constructing the ICN setup and access all three nodes , they need to have the same credentials (i.e. username and password) for remote login. To do so, create on each node an account with a fixed username and password (e.g. username = `point`, password = `point`). **This must be the same for all three nodes**.

Passwordless SSH

In order to allow the deployment tool fast access to each node, without having to enter the node password for each remote login process, which can be a very time consuming process, create (if not existent already) a ssh key of the deploying node and copy it to each Nodex in the setup:

```
~$ ssh-keygen -t rsa -b 2048
~$ ssh-copy-id point@192.168.0.x
```

4.2.1.2 Interface Configuration

configure the management interface on each node by editing the `interfaces` file as as follows (use any editing tool of your preference):

```
~$sudo nano /etc/network/interfaces
```

add the following lines (replace X with the host address of the corresponding node):

```
auto eth2
interface eth2 inet static
     address 192.168.0.X
     netmask 255.255.255.0
     broadcast 192.168.0.255
```

save and exit the file

4.1.3.2 SDN Bridge

In Node2, the SDN bridge (`br`) need to be created and configured to support OpenFlow1.3. Then both eth2 and eth3 should be added to the bridge and configured with openflow port numbers. To create the bridge, do (in Node2):

```
~$ sudo ovs-vsctl add-br br
```

Configure the bridge to support OpenFlow1.3:

```
~$ sudo ovs-vsctl set Bridge br protocols=OpenFlow13
```

Add interfaces eth2 and eth3 to the bridge and configure them with port numbers 2 and 3 respectively:

```
~$ sudo ovs-vsctl add-port br eth2 -- set interface eth2 options:key=flow ofport_request=2
```

```
~$ sudo ovs-vsctl add-port br eth3 -- set interface eth3 options:key=flow ofport_request=3
```

Now, Node2 is ready to operate as an SDN forwarder in the ICN network and you can move to deploying the ICN-over-SDN network.

4.2 Deploy the Network

The ICN topology of Figure 4.1 can be found in `blackadder/deployment/examples`. The configuration file is named `icn_over_sdn.cfg`. To familiarize with the configuration file, refer to Section 3 of the HowTo.

Notice: Blackadder does not run on the SDN forwarder, as the latter merely performs Flow-based forwarding. Therefore, the domain RV/TM MUST not be the SDN node.

To deploy the ICN-over-SDN network:

```
~$ cd ~/blackadder/deployment/
~$ ./deploy -c ./examples/icn_over_sdn.cfg -l
```

Now you can verify that the network is operational by checking click is working in Nodes 1 and 3:

```
~$ pgrep click
```

Check the TM is working on Node1 (the last process number):

```
~$ pgrep tm
```

Check that your ICN forwarding flows are placed correctly in Node2:

```
~$ sudo ovs-ofctl -O OpenFlow13 dump-flows br
```

This should look like the following:

```
 cookie=0x0, duration=190.369s, table=0, n_packets=0, n_bytes=0,
priority=200,ipv6,dl_dst=00:00:00:00:00:00,ipv6_src=::128.0.0.0/::128
.0.0.0 actions=output:2,resubmit(,1)

 cookie=0x0, duration=190.37s, table=0, n_packets=0, n_bytes=0,
priority=100,ipv6,dl_dst=00:00:00:00:00:00 actions=resubmit(,1)

 cookie=0x0, duration=1764.563s, table=0, n_packets=8, n_bytes=648,
priority=0 actions=NORMAL

 cookie=0x0, duration=190.368s, table=1, n_packets=0, n_bytes=0,
priority=200,ipv6,dl_dst=00:00:00:00:00:00,ipv6_dst=::4000:0:0:0/::40
00:0:0:0 actions=output:3
```

```
 cookie=0x0, duration=190.367s, table=1, n_packets=0, n_bytes=0,
priority=100,ipv6,dl_dst=00:00:00:00:00:00 actions=drop
```

4.3 Run a Pub/Sub Application

You can test the operation of the network by running any of the provided Pub/Sub applications (including the NAP). Here, we will use the Pub/Sub ping app for simplicity.

- In Node1:

  ```
  ~$ cd blackadder/examples/traffic_engineering
  ```

  - Run the ping publisher (request 5 pings):

    ```
    ~$ sudo ./ping_publisher 0 3 1 5
    ```

- In Node3:

  ```
  ~$ cd blackadder/examples/traffic_engineering
  ```

  - Run the ping subscriber:

    ```
    ~$ sudo ./ping_subscriber 0 3
    ```

If you see the pings flow between the two nodes then your test is complete and the network is fully operational. Well Done..!

5. Network Simulator 3

5.1 Introduction:

The main objective of this document is to guide the reader through the required steps to set up an ICN network of various sizes and topologies simulated in an NS3 environment and how to run Pub/Sub applications on different nodes setting various traffic loads on the network. It is not intended for NS3, Click or Blackadder installation and compilation process, for such objectives please see the “How To.pdf” and “NS3 How To.pdf” documents available in the doc directory of your Blackadder installation “`<BLACKADDER PREFIX>/doc/`”. In this setup we assume that the mentioned components are already installed and compiled to support NS3 – Blackadder simulations. Figure 1 shows a simple NS3 – Blackadder simulation example consisting of 5 nodes. In this setup node 3 has the TM/RV role in the network while nodes 1 and 5 run publish, subscribe applications and nodes 2 and 4 are forwarders in the network.

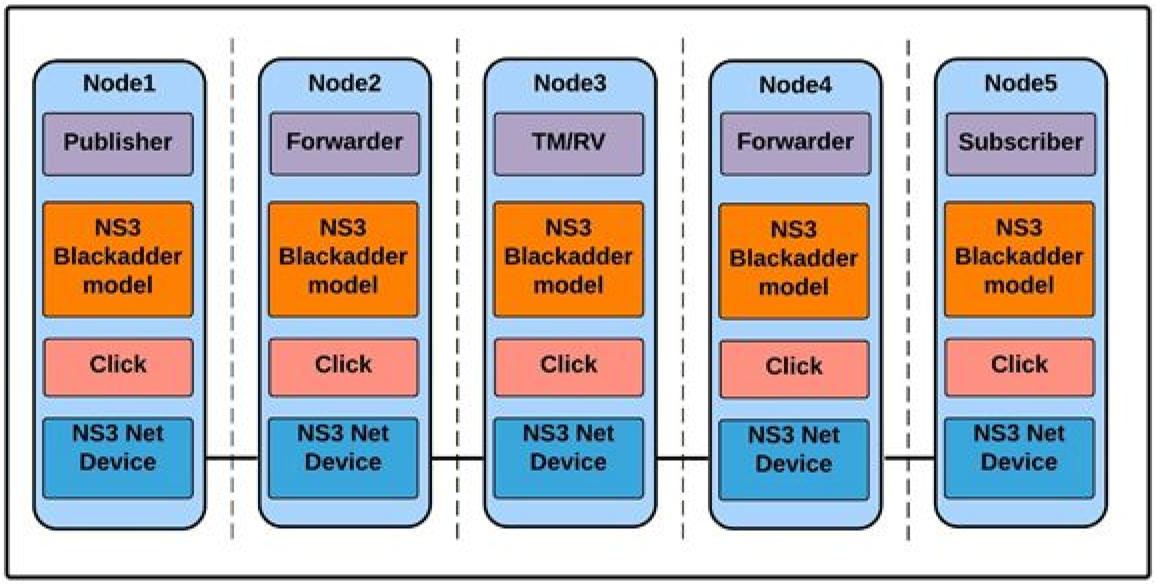

Figure (1) Basic NS3 – Blackadder Simulation Example

5.2 Configuring an NS3 Network:

Creating the Blackadder configuration file can be done manually by defining all the network nodes, their connections and all Pub/Sub applications in the network. This process is described in the “NS3 How To.pdf” document available in the doc directory of Blackadder “`<BLACKADDER PREFIX>/doc/`”. This approach can be suitable for small networks but is found to be a time-consuming and error-prone process for large scale networks with different topologies and node connectivity degrees since multiple parameters must be manually added with a specific format. Therefore a script has been provided that allows for defining different random network topologies with any number of nodes into a Blackadder configuration file. This script is available in:

```
<BLACKADDER PREFIX>/ns3/config-generator/
```

To use this script you will need R programming language installed on your machine (preferably R Studio). Open the script file and edit the number of nodes you would like to be simulated (5 in this example) at line 11 of the code as follows:

```
g <- barabasi.game(5, directed = FALSE)
```

Make sure that the node links always stay undirected (directed=FALSE) in order to allow the auto generator code to create bidirectional links for the blackadder topology. Worthy to note that the random network graph function above is just an example and you are free to use any other available functions you may find suitable for your network topology. Also the node connectivity links are generated randomly and do not necessarily reflect the links of this example.

Now source and run the script in R and the blackadder configuration file (ns3_topology.cfg) will be automatically generated in your R workspace. After generating the Blackadder configuration file, you will need to define the TM/RV node and Pub/Sub applications to run on the nodes. In this example node 3 will be the TM/RV of the network, node 1 will run a publisher application and node 5 will run a subscriber application, Thus the configuration file should be edited as follows (needed modifications are highlighted in red):

1) Add TM/RV to the role definition of node 3 as follows:

```
 {
label = "00000003";
role = ["RV", "TM"];
connections = (
{
to = "00000002";
Mtu = 1500;
DataRate = "100Mbps";
Delay = "10ms";
},
{
to = "00000004";
Mtu = 1500;
DataRate = "100Mbps";
Delay = "10ms";
}
);
},
```

2) Add the publisher application to node 1 as follows:

```
{
label = "00000001";
role = [];
connections = (
{
to = "00000002";
Mtu = 1500;
DataRate = "100Mbps";
Delay = "10ms";
},
);
applications = (
{
name = "Publisher";
start = "2.34";
stop = "14.87";
}
);
},
```

3) Add the subscriber application to node 5 as follows:

```
{
label = "00000005";
role = [];
connections = (
{
to = "00000004";
Mtu = 1500;
DataRate = "100Mbps";
Delay = "10ms";
},
);
applications = (
{
name = "Subscriber";
start = "2.34";
stop = "14.87";
}
);
},
```

5.3 Deploying the NS3 Network:

After defining the pub/sub applications of the example, the configuration file is ready for deployment. To deploy the NS3 example use the configuration file created earlier and deploy by running:

```
./deploy -c /<PATH TO CONFIG FILE>/ns3_topology.cfg -s
```

By using -s (--simulate) flag, the deployment tool will create all necessary click configuration files, the topology file and NS3 simulation (C++) code. The generated file "topology.cpp" will contain the NS3 code necessary to run the network configuration.

5.4 Running the NS3 Example:

First of all, you will need to copy topology.cpp that was created during the deployment process into the appropriate location in the NS3 directory `<NS3 PREFIX>/ns-3.X/examples/blackadder/`.

It will be called examples/blackadder/example3 in NS3.

```
cp /tmp/topology.cpp <NS3
PREFIX>/ns-3.X/examples/blackadder/example3.cc
```

Now edit `<NS3 PREFIX>/ns-3.X/examples/blackadder/wscript` to add the example by appending the following lines:

```
obj = bld.create_ns3_program('example3', ['core', 'point-to-point',
'blackadder', 'applications'])

obj.source = ['example3.cc', 'publisher.cc', 'subscriber.cc',]
```

Now build NS3:

```
cd <NS3_PREFIX>
```

```
./waf build
```

And finally you can run the example simulation with

```
./waf --run examples/blackadder/example3
```

6. Dynamic ICN Network

To bootstrap a simple 3-node ICN topology, the standard Blackadder deployment tool used large configuration files, similar to the `~/blackadder/deployment/examples/sample_topology.cfg`, which was introduced above. However, this might be a time-consuming and error-prone process, since multiple parameters must be manually added with a specific format. In addition to this, if there is a requirement to update the topology, this single configuration file needs to be updated and ICN processes will restart in every topology node, which does not allow the seamless operation of the non-affected ones.

The updated deployment tool suggests a more dynamic approach for the ICN topology formation and configuration, which enables the addition and deletion of nodes without disrupting the operation of non-attached ones. ICN topologies can now be dynamically deployed with small configuration files each time, instead of a complex and error-prone one.

The following examples will show how to extend an existing topology configured in deployment file `~/blackadder/deployment/examples/basic_3node_icn.cfg`, adding a new node, and also removing it. The topology to be created is presented in Figure 6.1. DS abbreviation means that node will act as the Deployment Server, while TM and RV indicate that node will act as the Topology Manager and RendezVous node.

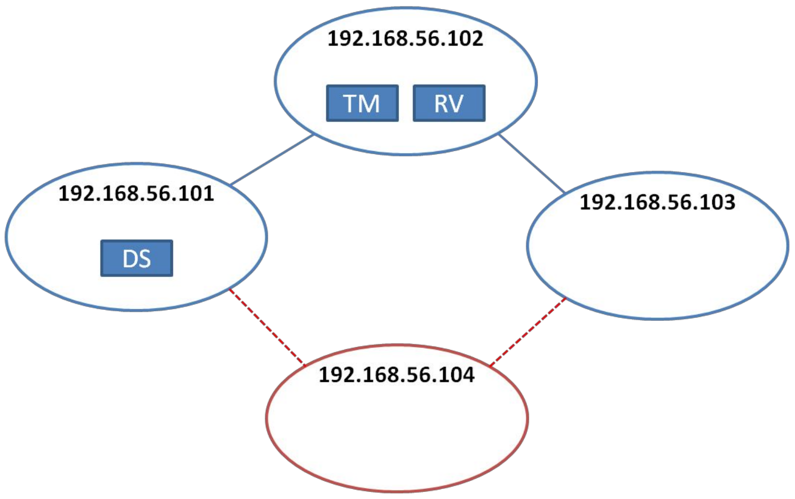

Figure 6.1: Sample topology for attaching and detaching a node

To deploy the basic 3-node example and also start the deployment server (DS) at node 192.168.56.101, login to node 192.168.56.101 and execute:

```
$ ./deploy -c examples/basic_3node_icn.cfg --dynamic
```

This commands calculates and prepares all the necessary files, and copies them to each node, starts the Click process in all 3 nodes, as well as TM process in node 192.168.56.102, and finally starts the deployment server in node 192.168.56.101 at port 9999, and waits for incoming deployment requests. Hence, the blue-colored nodes (192.168.56.101-103) of Figure 6.1 are properly deployed.

6.1 Attach a new node:

To add node 192.168.56.104, the user can now login to node 192.168.56.104 and use the `~/blackadder/deployment/examples/new_node.cfg` file, aiming to add the node to the existing topology, and connect it with nodes 192.168.56.101 and 192.168.56.103.

To send the deployment request to the deployment server at 192.168.56.101, execute:

```
$ ./client 192.168.56.101 examples/new_node.cfg
```

The deployment server will process the incoming request, generate the new topology graphml file, generate and send the new Click files, as well restart the Click and TM processes in nodes 192.168.56.101 and 192.168.56.103. Note that node 192.168.56.102

will not be affected. Hence, now the complete topology including 4 nodes (192.168.56.101-104) is fully functional and the user can run any of the existing Blackadder samples.

6.2 Detach an existing node:

If for some reason, node 192.168.56.104 has to removed from the topology, the user must login to 192.168.56.104 and use the configuration file `~/blackadder/deployment/examples/delete_node.cfg`.

To send it to the deployment server, execute:

```
$ ./client 192.168.56.101 examples/delete_node.cfg
```

Upon reception of such request, the deployment server will delete the node `192.168.56.104`, update all required Click and TM files in all topology nodes and also restart all respective processes. Hence the topology will return to the original 3-nodes state.