H2020 iP Over IcN– the betTer IP (POINT)

HowTo

Installation and Configuration of the POINT Platform

List of Authors:

Sebastian Robitzsch, Mays Al-Naday, George Petropoulos, Mohammed ALKhalidi

1 Introduction

This document describes the steps required to download and install the POINT platform, comprising the core Information Centric Networking (ICN) node (aka. Blackadder), the Topology Manager (TM), the Network Attachment Point (NAP), the Resiliency Manager (RM), various ICN example applications and a collection of complementary components. The document assumes a clean installation of a Linux-based distribution with a number of prerequisites, which will later be outlined. The steps in this document have been tested on Debian 8, Ubuntu 12.04/14.04/15.04, Voyage Linux 0.10 and mac os x. For the purpose of this document, we have used Click version 2.x (latest stable) and POINT cycle-1 release (point-1.0.0). If you need to run NS3 simulations with the POINT platform please refer to the NS3 How To document available in the master directory of your Blackadder installation.

The POINT platform also uses libconfig-based configuration templates to represent a deployable ICN network, in addition to other elements in the network. Each of these templates will be described along their corresponding application. In addition, a description of the node configuration file will also be provided.

2 Installation

2.1 Preparation

Since most of time, the platform installation and running require sudo privileges for users other than root, it is highly recommended that the OS does not prompt for a password when running an application using sudo. To disable sudo password prompts, the user account must be set in the sudoers file to not use a password. To do that, first print out the line that need to be added to the sudoer file by running:

```
~$ echo “$(whoami) ALL=(ALL) NOPASSWD: ALL”
```

copy the output (stdout) then access and modify the sudoers file by running:

```
~$ sudo visudo
```

and paste the copied line at the end of the file.

2.2 Quick Start

This section provides a brief walkthrough of the steps that need to be followed to get the POINT platform installed. For detailed instructions with relevant explanation, please move to the next section.

Install Git:

```
      ~$ sudo apt install build-essential git
```

Compile and install the Click modular router

a. Get Click:
   ```
   ~$ git clone https://github.com/kohler/click
   ```
b. Make Click
   i. move to click directory:
      ```
      ~$ cd ~/click
      ```
   ii. configure, compile and install click:
      ```
      ~$ ./configure --disable-linuxmodule
      ```

```
                    ~$  make && sudo make install
```

Install the Blackadder core (~/blackadder/make-all-clean.sh has all steps in a single script. If the single script is used do NOT run it as sudo!):

c. Get Blackadder (**NOTE**: this command must be called from within a user account other than root, i.e. /home/<USERNAME>):

```
~$ git clone
https://github.com/point-h2020/point-3.0.0.git blackadder
```

b. Install libraries and applications that Blackadder's components require:

```
~$ cd ~/blackadder/
~$ sudo apt install $(cat apt-get.txt)
```

c. compile and install blackadder

```
~$ cd ~/blackadder/src
~$ autoconf
~$ ./configure && make && sudo make install
```

Install the Blackadder API library (C++):

```
~$ cd ~/blackadder/lib
~$ autoreconf -fi
~$ ./configure && make && sudo make install
```

Compile and install MOLY + BAMPERS

```
~$ cd ~/blackadder/lib/moly
~$ make && sudo make install
~$ cd ~/blackadder/lib/bampers
~$ make && sudo make install
```

Compile and install MAPI

```
~$ cd ~/blackadder/lib/mapi
~$ make && sudo make install
```

Compile the Topology Manager (in your TM node):

```
~$ cd ~/blackadder/TopologyManager
~$ make
```

Compile the deployment tool (in your deployment node)

```
~$ cd  ~/blackadder/deployment
```

```
      ~$ make
```

More detailed instructions are given in the following sections.

2.3 Full Instructions

2.3.1 Utilities

2.3.1.1 Prerequisites

Before diving into the compilation of the code, a number of packages need to be installed:

```
$ sudo apt install build-essential git autoconf automake libtool m4
--yes --force-yes
```

There is also another set of packages that need to be installed after obtaining the platform code base. this set is dynamically updated with further packages as the platform evolves and further dependencies are introduced and/or updated.

2.3.1.2 Click Modular Router (CMR)

Click Modular Router is the base platform, over which the POINT platform runs. Therefore, click need to be installed before the POINT platform. Note that although Blackadder should work also with the Click 2.0.1 codebase from September 2011, you might experience package installation problems if you choose to use Click 2.0.1 (e.g., *pkg-Makefile* not found). Therefore it's recommended to install a more recent Click version (e.g., from January 2012 or later) instead.

```
~$ git clone https://github.com/kohler/click.git
~$ cd click
```

If you already have an older version of Click from GitHub, you can get the latest version by running `git pull` in the click directory.

You can choose to install Click with or without kernel support. If you don't intend to run Click or Blackadder in the kernel, you can run:

```
~$ ./configure --disable-linuxmodule
```

Or run (recommended options for ns-3 bindings):

```
~$ ./configure --enable-nsclick --enable-blackadder
~$ make
~$ sudo make install
```

By default many Click packages that Blackadder doesn't need will be compiled and linked with Click, resulting in a large library (and large Click module). To avoid that you can use the mkminidriver tool (*http://read.cs.ucla.edu/click/docs/click-mkmindriver*) or manually delete the elements that are not required before compiling Click.

2.3.2 Core ICN Platform (Blackadder)

To download and compile Blackadder and the user library, first clone the public repository. **NOTE**: this command must be called from within a user account other than root, i.e. /home/<USERNAME>:

```
~$ git clone https://github.com/point-h2020/point-3.0.0.git
blackadder
~$ cd blackadder
```

Install Blackadder further dependencies, listed in `apt-get.txt` file:

```
~$ sudo apt install $(cat apt-get.txt)
```

Compile and install Blackadder:

```
~$ cd ~/blackadder/src/
~$ autoconf
~$ ./configure
~$ make && sudo make install
```

To compile and install Blackadder user library, If needed, you can regenerate the configure file and some other files, such as `Makefile.ins` in each subdirectory, by first running `autoreconf`. Note that this requires that `autoconf, automake` and `m4` have been installed on your system.

```
~$ cd ../lib
~$ autoreconf -fi
~$ ./configure
```

The default installation locations are `/usr/local/include` and `/usr/local/lib`, but the `/usr/local` prefix can be changed by giving a different path with the `--prefix` parameter (e.g.: `--prefix=/path/to`). Also other parameters can be given; run

`./configure --help` for more information.

```
~$ make
~$ sudo make install
```

Both a shared (`libblackadder.la` and `libblackadder.so.*`) and a static library (`libblackadder.a`) are generated. The library is linked with applications by specifying `-lblackadder` as a linker option to (e.g.) `g++`. Also `-lpthread` is normally needed. In case you need to do static linking, add `-static` as an option. The header files that are normally used in C++ programs are `blackadder.hpp` (that implements blocking event handling) and `nb_blackadder.hpp` (for non-blocking event handling).

2.3.2.1 File Structure

`/path/to/binaries/bin/`: all user-space Click-related tools as well as Click executable

`/path/to/binaries/lib/`: all Click-related libraries and all user (.uo) objects for the installed packages, like Blackadder.

2.3.3 Additional Libraries

2.3.3.1 MAPI

```
~$ cd ~/blackadder/lib/mapi
~$ make
~$ sudo make install
```

2.3.3.2 MOLY

```
~$ cd ~/blackadder/lib/moly
~$ make
~$ sudo make install
```

2.3.3.3 BAMPERS

```
~$ cd ~/bladder/lib/bampers
~$ make
~$ sudo make install
```

2.3.4 The Deployment Tool

Compile and install the deployment tool:

```
~$ cd ~/blackadder/deployment
~$ make
```

2.3.5 ICN Applications

The POINT platform comes with a set of ICN Pub/Sub applications. Those are applications that utilizes Blackadder and its API library to communicate with each other over Ethernet or IP networks. The most essential of those applications are the TM as the application that forms the delivery paths, and the NAP as the application that bridges IP networks to ICN. The platform also provides a collection of example Pub/Sub applications that aids in demonstrating the ICN communication, such as the video streaming, the ping and the link state monitoring applications.

2.3.5.1 Topology Manager (TM) / Resiliency Manager (RM)

The Topology Manager is a C++ application that accesses the network using the libraries produced by following the aforementioned steps. The TM provides the basic path formation function according to Shortest Path algorithm, as well as a set of traffic engineering extensions that supports: load balance, resilience and path management.

Compile the Topology Manager

```
~$ cd ~/blackadder/TopologyManager
~$ make
```

The executable module is named `tm`

If at any time of the Topology Manager compilation process, the shared Blackadder library `libblackadder.so.o` cannot be located, please check your permissions to read `/usr/local/bin`. Especially with Debian 8.x, only `sudo` and `root` have access to `/usr` initially.

The steps above will also compile the Resiliency Manager, the executable module is named `rm`.

2.3.5.2 Network Attachment Point (NAP)

With the underlying ICN platform built, the NAP source code can be compiled now:

```
~$ cd ~/blackadder/apps/nap
~$ make && sudo make install
```

For further information on how to configure the NAP and enable more detailed logging, please run

```
~$ cd ~/blackadder/apps/nap/doc/tex
~$ make
```

and open the generated nap.pdf file.

2.3.5.3 Traffic Engineering Applications

TE example applications include: Broadcast Link State Monitor, ICN Ping Publisher/Subscriber, QoS Publisher/Subscriber, Qos Metadata Provider and a bandwidth test publisher/subscriber.

To compile all examples:

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

~$ make

~$ ./makeBandwidth.sh
```

2.3.5.4 Further Example Applications

2.3.5.4.1 Video Streaming

VLC-based, C++, video streaming Publisher/Subscriber pair of applications. The applications depends on CLI input (i.e. no GUI)

```
~$ cd ~/blackadder/examples/video_streaming

~$ make
```

2.3.5.4.2 Miscellaneous Samples

A collection of simple, C++, applications to test the ICN Pub/Sub communication.

```
~$ cd ~/blackadder/examples/samples

~$ make
```

2.3.5.5 ICN-SDN Application

This application assumes that Blackadder is installed in the machine following the instructions of Section 2.3.2.

Additionally, the google protobuf library needs to be installed:

```
$ sudo apt-get install libprotobuf-dev protobuf-compiler
```

Finally, all the processes require the execution of Click process.

To build the code:

```
$ cd ~/blackadder/apps/icn-sdn
$ make all
```

If the compilation causes errors invoke:

```
$ make clean
$ rm messages/{messages.pb.cc,messages.pb.h}
$ pushd messages
$ protoc -I=. --cpp_out=. messages.proto
$ popd
$ make all
```

To execute the ICN-SDN application server process:

```
$ ./server
```

To execute the ICN-SDN application server with arguments, run:

```
$ ./server node_id host:mac_address openflow:X
```

where `node_id` is the node identifier of the TM, `mac_address` the MAC address of the TM and `openflow:X` the openflow identifier of the attached SDN switch to the TM, e.g. `./server 00000019 host:af:dd:45:44:22:3e openflow:156786896342474`

2.3.5.6 Bootstrapping Application

The bootstrapping application assumes that Blackadder is installed in the machine following the instructions of Section 2.3.2.

To build the code:

```
$ make all
```

To execute the bootstrapping protocol:

```
$ ./bootstrapping
```

The bootstrapping process can also receive additional options, in the form of `-<option(s)>` depending on the environment and execution type. The list of available options is presented below:

- t: Use this option when the node to be bootstrapped is the Topology Manager
- s: Use this option when the node to be bootstrapped has at least one interface attached to an SDN switch
- m: Use this option when the bootstrapping process occurs in a Mininet environment (this option exists to exclude certain operations, e.g. click start and stop, which cannot be handled properly in such emulated environment)

Therefore, if you wish to bootstrap a host attached to an SDN switch, execute the command:

```
$ ./bootstrapping -s
```

If it the node is bootstrapped in a Mininet environment, then execute:

```
$ ./bootstrapping -sm
```

To delete the generated artifacts:

```
$ make clean
```

2.3.5.7 Monitoring Agent

```
$ cd ~/blackadder/apps/monitoring/agent
$ make
```

2.3.5.8 Monitoring Server

```
$ cd ~/blackadder/apps/moose
$ make
$ sudo make install
```

MOOSE also requires MySQL to be installed on either the same machine or another host which is reachable via IP. The installation of a MqSQL server can be achieved by invoking:

```
$ debconf-set-selections <<< 'mysql-server mysql-server/root_password
password point'
$ debconf-set-selections <<< 'mysql-server
mysql-server/root_password_again password point
$ sudo apt install mysql-server -y
```

Note, the password 'point' is given for the installation of the MySQL server user root.

If Debian9 is used MySQL sources must be manually added due to a licensing dispute between Oracle and the Debian community[1].

```
$ wget https://dev.mysql.com/get/mysql-apt-config_0.8.9-1_all.deb
$ dpkg -i mysql-apt-config_0.8.9-1_all.deb
$ apt-get update
```

The database schema is available in the /doc directory of the monitoring server and can be simply imported using the mysql CLI or graphical tools such as PHP MyAdmin. When using the command line execute:

```
shell> cd ~/blackadder/apps/moose/vida/defaults
shell> mysql -h 127.0.0.1 -u point -p
```

[1] https://mysqlrelease.com/2017/06/debian-9-and-mysql-watch-out/

```
mysql> CREATE DATABASE <DB_NAME>;
mysql> USE <DB_NAME>;
mysql> source db_schema.sql;
mysql> source db_static_content.sql;
```

The <DB_NAME> must correspond to the one configured in /etc/moose/*.cfg (by default it is called moose.cfg). It is recommended to use "moose" for <DB_NAME>. More information on MOOSE and its configuration can be found in ~/blackadder/apps/moose/doc/tex (check the MOOSE README.md how to compile the PDF).

2.3.6 Simulation/Emulation Platforms

2.3.6.1 Network Simulator 3

For instruction on installing and running the POINT platform in NS3, please refer to the NS3 HowTo document.

2.3.6.2 Mininet

Mininet is recommended for constructing experimental ICN networks using the POINT platform. Furthermore, installing full mininet also provides OpenVSwitch, which is used in our ICN-SDN forwarding mechanism, described in D3.1. We recommend downloading mininet (version 2.2.1 or higher) from github then installing it, rather than doing apt-get install and obtaining the version supported by package archive. To download and install mininet:

```
~$ git clone git://github.com/mininet/mininet
~$ cd ~/mininet
```

switch to 2.2.1 branch:

```
~$ git checkout -b 2.2.1 2.2.1
```

install mininet with full options (this will also download and install ovs):

```
~$ ~/mininet/util/install.sh -a
```

2.3.7 Doxygen

The codebase is fully documented with doxygen. To compile doxygen:

```
~$ cd ~/blackadder/doc/doxygen
~$ doxygen Doxyfile
~$ cd ~/blackadder/apps/doc/
~$ doxygen doxygen.conf
```

3 Configuration Templates

Configuration files are used in the POINT platform to provide a structural representation of the ICN deployable network, the dynamic attach/detach nodes and the routing prefixes served by the NAP. These templates ought to be customized by the user to suit the setup in consideration. The sections below describe the structure of these templates

3.1 Topology Configuration

3.1.1 Deployable Topology

This template represents the ICN static network to be constructed by the deployment tool. The template consists of global configuration parameters, which are applicable to every node in the network; as well as the network configurations. The latter comprises a list of nodes, with node-specific configurations, each of which has a list of connections that represents the node directional connectivity in the network.

3.1.1.1 Global Configurations

```
BLACKADDER_ID_LENGTH = 8;
LIPSIN_ID_LENGTH = 32;
CLICK_HOME = "/usr/local/";
WRITE_CONF = "/tmp/";
USER = "point";
SUDO = true;
OVERLAY_MODE = "mac";
ODL_ADDRESS = "192.168.132.202";
```

`BLACKADDER_ID_LENGTH:` the length of Scope IDs and Information IDs supported by Blackadder. Currently this parameter has to be configured with the same value as that set for Blackadder at compile time (i.e. the value set for `PURSUIT_ID_LEN` in `src/helper.hh`, `lib/blackadder_defs.h` and, if one use the ns-3 bindings, `ns3/blackadder-model/model/service-model.h`). The current value is set to `8`, indicating `8-byte` length.

`LIPSIN_ID_LENGTH:` the length of the Bloom Filter in bytes. This will be the same for the LinkIDs, internal Link IDs, and Forwarding identifiers. This is currently set to `32,` indicating `32-byte` or `256-bit` length.

`CLICK_HOME:` the absolute path where Click is installed.

`WRITE_CONF:` the absolute path where the deployment utility will remotely copy the Click/Blackadder configurations in each Blackadder node. The same is going to be used to remotely copy the produced topology.graphml file at the network node that will run the Topology Manager.

`USER:`The username of the user that will be used when ssh-ing network nodes (for retrieving mac addresses and running Click) and copying configuration files.

`SUDO:` True if the deployment utility will use sudo when remotely executing commands to network nodes.

`OVERLAY_MODE:` The mode in which Blackadder will run. Currently, Blackadder can run on top of Ethernet (“mac”) or Raw IP Sockets (“ip”).

`ODL_ADDRESS:` The Opendaylight SDN controller IP address.

3.1.1.2 Network Configurations

The network consists of a set of nodes, each of which has a set of connections.

```
network = {
    nodes = (
    {
        testbed_ip = "A.B.C.D";
        running_mode = "user";
        label = "00000001";
        role = ["RV","TM"];
        operating_system = "Darwin";
        connections = (
            {
                to = "00000002";
                src_if = "eth0";
                dst_if = "eth1";
            }
        );
    });
};
```

3.1.1.2.1 Node Configurations

`testbed_ip:`_The management IP address (in dotted decimal format) to which the deployment utility will copy the configuration files and remotely execute commands. Notice, this is only used for deployment purposes, once the network is deployed the IP address will not be used anymore.

`running_mode`: The mode in which Blackadder will run in this node. Use "user" for user-space (as user-space process)

`label:` The NodeID (i.e. node identifier) in the ICN network. The label is used when sending requests to the Rendezvous Node. The Topology Manager also keeps track of the nodes in the network using their labels. The size of the label must be `BLACKADDER_ID_LENGTH` bytes.

`role:` if omitted or role[] then the network node has no special functionality. Use role["RV","TM"] if the node is the Rendezvous Node and the Topology Manager or use the above keywords separately to place the (extra) functionalities to different nodes.

`operating_system:` defines the operating system of the node. By default, this is set to linux; for mac os x, this should be set to `Darwin`. for OVS, it should be set to `ovs`

`connections:` defines the list of outgoing connections from the node. The definition of a node connection will be described below.

3.1.1.2.2 SDN Node Configurations

These only need to be specified for an SDN node, should one or more exists in the deployable topology.

`sdn_implementation ("tables"` or `"bridges"):` defines which implementation to be considered for the SDN nodes. For details about tables vs. bridges, please refer to POINT D3.1https://www.point-h2020.eu/wp-content/uploads/2015/09/POINT_D3.1-First_Platform_Design.pdf .

`bridge`: The bridge name of the SDN bridge.

`openflow_id:` The Openflow identifier of the SDN switch, as provided by the SDN controller. It must be used when the switch has to be configured by the Opendaylight controller instead of ovs commands. Such identifier has the following structure, "openflow:311143139923".

3.1.1.2.3 Node Connections

`to:` the `NodeID` of the destination node

If the `OVERLAY_MODE = "IP":`

`src_ip:`the source IP address that will be used when sending to the Raw IP Socket.

`dst_ip:`the destination IP address that will be used when sending to the Raw IP Socket.

If the `OVERLAY_MODE = "mac":`

`src_if`: the network interface of the source MAC address. The deployment will use this and remotely acquire the respective MAC address. E.g. use "eth0" .

`dst_if:` the network interface of the destination MAC address. The deployment will use this and remotely acquire the respective MAC address. E.g. use "eth0" .

For an SDN node, instead of `src_if/dst_if` configure:

`src_pt`: The port on the SDN bridge, from which the connection originates

`dst_pt`: The SDN port where the connection terminates.

In the case that the administrator wants to configure the fast failover groups of this port, then an additional parameter should be included:

`ff_pt`: The port on the SDN bridge, which will be the alternative in case of failure of the src_pt.

Those parameters (i.e. `src_if/_pt`, `dst_if/_pt`) can be used in conjunction with each other when the connection is between a blackadder node and a SDN node.

3.1.2 Dynamic Topology

The tool aims to build dynamic topologies on the fly by extending the standard Blackadder deployment tool to process new node addition or deletion requests. For that purpose, an initial topology file has to be prepared. This topology file can be as minimum as possible (containing only one node), or fully described (as happened in all Blackadder installations). The only requirement is that the file must contain node(s) with TM and RV roles, otherwise the software execution will exit.

A minimum configuration file, e.g. topology.cfg, is:

```
BLACKADDER_ID_LENGTH = 8;
LIPSIN_ID_LENGTH = 32;
CLICK_HOME = "/usr/local/";
WRITE_CONF = "/tmp/";
USER = "point";
SUDO = true;
OVERLAY_MODE = "mac";
network = {
    nodes = (
    {
        testbed_ip = "10.1.1.1";
        running_mode = "user";
        label = "00000001";
        role = ["RV","TM"];
    });
};
```

This first instance of the topology can be deployed from one machine, in this case 10.1.1.1, which from now on will also act as the deployment server listening for incoming node addition or deletion requests.

3.1.2.1 Attach a New Node

To add a new node, an additional configuration file has to be prepared, e.g. new_node.cfg, which looks like the one below:

```
BLACKADDER_ID_LENGTH = 8;
LIPSIN_ID_LENGTH = 32;
CLICK_HOME = "/usr/local/";
WRITE_CONF = "/tmp/";
USER = "point";
SUDO = true;
OVERLAY_MODE = "mac";

network = {
    nodes = (
    {
        testbed_ip = "10.1.1.2";
        running_mode = "user";
        label = "0000000x"; //node to be added
        role = [];
        connections = (
            {
              to = "10.1.1.1"; //use IP address, instead of label
             src_if = "eth0";
             dst_if = "eth0";
            }
        );
    });
};
```

This file indicates that a new node with IP address 10.1.1.2 is going to be added (label 0000000x indicates node addition), which will connect to existing node 10.1.1.1 on the source and destination interfaces eth0 and eth0. The file semantics are almost identical to the ones described in Section 3.1.1, with 2 modifications:

1. For any new node, *label* value **must be** set to the well-known nodeID **"0000000x"**, with 'x' representing that the node is un-identified as an ICN node at the moment
2. For any connection, *to* value must be set to the static IP address of the attachment node (instead of label which was used in the past)

3.1.2.2 Detach an Existing Node

To detach one of the existing nodes, the previous configuration file can be altered and renamed to delete_node.cfg, with the following information:

```
BLACKADDER_ID_LENGTH = 8;
LIPSIN_ID_LENGTH = 32;
CLICK_HOME = "/usr/local/";
WRITE_CONF = "/tmp/";
USER = "point";
SUDO = true;
OVERLAY_MODE = "mac";

network = {
    nodes = (
    {
        testbed_ip = "10.1.1.2";
        running_mode = "user";
        label = "xxxxxxxx"; //node to be deleted
        role = [];
        connections = (
        );
    });
};
```

The only difference with the configuration file of node addition, is that the label value "xxxxxxxx" indicates that the node 10.1.1.2 is to be deleted.

3.1.3 Opendaylight-configured Topology

Create a deployment tool configuration file, following the structure of the provided example: https://github.com/point-h2020/point-3.0.0/tree/master/deployment/examples/odl_sample.cfg.

To enable deployment using Opendaylight SDN controller, and assuming that Opendaylight is already up and running, execute the typical deployment tool command, using the `-o` parameter:

```
~ ./deploy -c examples/odl_sample.cfg -o
```

3.2 Network Attachment Point

The NAP configuration template is placed in /usr/share/doc/nap/nap.cfg and allows the user to configure the NAP software.

For a full list and description of all configuration parameters please read the LaTeX-based NAP documentation. In order to compile the PDF please follow the steps described in the README.md file located in the NAP directory (apps/nap/README.md.

3.3 Mininet Cluster

Mininet clusters in POINT can be defined with a JSON based configuration file, which can be accessed at: ~/`Blackadder/deployment/mininet/doc/mn_config.json`. The format of the file is described below.

`"READ"` specifies where to locate the: mininet executable script, mininet exit script, NAP identification files and end point identification files. These files holds IP information of the respective mininet host.

`"MANAGEMENT_NET"` defines the management network to be used for remote access to the mininet hosts.

`"DATA_NET"` is the IP data network to be connected over ICN

`"EXTERNAL_NET"` is the network of the mininet hosting machine (not the mininet hosts). This network connects mininet hosts to the underlying machine and to other machines reached via the same network.

`"clusters"` specifies:

1. The set of machines to be use
2. The topology
3. Number of client-NAP pairs
4. Main switch connecting the hosts (either `ovs` or `linux`)
5. Controller IP address
6. Management interface of the mininet host
7. Data interface of the mininet host

3.4 Mininet ICN

The ICN topology of a mininet cluster is defined in another JSON file that can be accessed at: ~/`Blackadder/deployment/mininet/doc/mn_icn_icn_config.json`. The file describes first a single topology that will be applied to every mininet cluster (i.e. set of mininet hosts on a machine). Second it specifies the list of machines (hosting clusters) that will have the topology configured in them.

3.5 Monitoring Server

Similarly to the NAP MOOSE is configured via a libconfig-based configuration file which is placed to /use/share/doc/moose/moose.cfg. The full documentation on the various configuration variables can be found in apps/monitoring/server/doc/text/moose.pdf. In order to generate the PDF please follow the steps described in apps/monitoring/server/README.md

4 Network Deployment

Deploy Blackadder in your network:

d. Create a network configuration file for your network topology using the libconfig template provided in deployment/ directory (see "Deploying Blackadder")

e. In the *deployment* directory, run:

```
~$  ~/blackadder/deploy -c YOUR_NETWORK.cfg
```

f. Start the topology manager in the TM node (if the deployment tool has not done that already, check with `pgrep tm`):

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

Try some of the applications in the *examples* directory.

5 Mininet Deployment

To Deploy mininet clusters, run the tool providing the respective json config file:

```
~$ ~/blackadder/deployment/mininet/deploy.py -m MN_FILE_NAME.json
```

The tool may also be used to generate the ICN config file that can be used with the ICN deployment tool. To do so, run:

```
~$ ~/blackadder/deployment/mininet/deploy.py -i ICN_FILE_NAME.json
```

For a comprehensive description of example deployment scenarios, please refer to the Examples document