Blackadder Installation

# Introduction

This manual page thoroughly describes the steps to get, compile and run Blackadder, and all complementary components, like the Topology Manager (TM), in a single machine as well as in a testbed. This manual assumes a clean installation of the latest Debian distribution although it should be accurate for other Debian-like distributions.

# Compiling and Running Core Components

Note that some of the commands might need running under **sudo**!

## Installing Click

-install Git: *apt-get install git*

-get Click: *git clone git://read.cs.ucla.edu/git/click /directory\_of\_your\_choice*

-install g++ if it’s not there: *apt-get install g++*

-configure Click without kernel module support (and support for user-space multithreading):

*./configure --disable-linuxmodule --enable-user-multithread*

Blackadder compiles and runs in the user space as well. For configuring Click with Linux module support, please refer to Click’s documentation.

-compile and install Click: *(sudo) make install*

By default many useless (for Blackadder) Click packages will be compiled and linked with Click resulting to a large library (and Click module if supported). To avoid that 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.

## Installing Blackadder

-download the latest version of Blackadder (from github.com or from the svn repository).

-install autoconf if it’s not there: *apt-get install autoconf.*

-run *autoconf* in the Blackadder /src directory.

-run *./configure (use –prefix=/path/to/click if Click is installed somewhere else)*

-run *make install*

Blackadder is now compiled and installed. By default Click and all packages are installed in */usr/local/.* To change that, configure Click with *–prefix=/path/to/click* and all packages like Blackadder with *--with-click=/path/to/click*.

## File Structure

*/usr/local/bin/*: all user-space Click-related tools as well as Click executable.

*/usr/local/sbin/*: all kernel-space tools for starting (*click-install*) and stopping (*click-uninstall*) Click.

*/usr/local/lib/*: all Click-related libraries and all user (*.uo*) and kernel (*.ko*) objects for the installed packages, like Blackadder.

## Running Blackadder

To run Blackadder just run Click with a configuration file describing a valid Blackadder node architecture. Check the sample.conf for an example Blackadder configuration. Running this instance will result in a Blackadder node that is capable of Inter-Process Information-centric communication (if you use the NODE\_LOCAL strategy in the API). LINK\_LOCAL strategy requires at least two nodes and DOMAIN\_LOCAL strategy requires the Topology Manager to run.

# Setting Up and Running the Topology Manager (Python version)

In order to properly run the python version of the TM some python packages must be installed first.

-apt-get install python-impacket

-Download igraph library at <http://igraph.sourceforge.net/>.

-untar and *./configure*

*-*install dependencies: *apt-get install libxml2-dev zlib1g-dev*

-make install

-Download python extensions for igraph

-*apt-get install python-dev*

-untar extensions and run: *python ./setup.py install*

-Download the config python package from <http://www.red-dove.com/config-0.3.9.tar.gz>

-untar and run: *python ./setup.py install*

-Download the bitvector python package from <http://RVL4.ecn.purdue.edu/~kak/dist/BitVector-3.0.tar.gz?download>

-untar and run: *python ./setup.py install*

-untar the baddernetlink-0.1.tar.gz in the packages folder in the TM directory

-install python setup-tools: *apt-get install python-setuptools*

-run *python ./setup.py install*

Finally, run the topology manager:

*python ./main.py <topology>*

<topology> is the topology in a .graphML graph file that is produced and copied to the TM node by the deployment script (see below). It contains the topology along with all the necessary properties for the TM to run the network (Link IDs, internal Link IDs, node labels). In the topology there are also auto created properties regarding the length of the LIPSIN identifiers, the node label of the TM and whether Blackadder at the TM network node runs in user or kernel space.

# Deploying Blackadder

The deployment utility runs as a C++ application. To install run *make* in the respective folder. The utility requires the *C* version of iGraph (<http://igraph.sourceforge.net/>) as well as libconfig (<http://www.hyperrealm.com/libconfig/>). Install both of them and then install the deployment utility.

Deploy Blackadder by running: *./deploy <configuration>*

## Configuration Options

### Mandatory Global Options

*BLACKADDER\_ID\_LENGTH*: the length of Scope IDs and Information IDs supported by Blackadder. Currently this parameter has to be configured separately in Blackadder at compile time (see *helper.h*).

*LIPSIN\_ID\_LENGTH:* the length of LinkIDs, internal Link IDs, and LIPSIN identifiers in bytes.

*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 at the testbed. Currently, Blackadder can run on top of Ethernet (“mac”) or Raw IP Sockets (“ip”).

### Defining a Network

A network is defined in the configuration file as follows:

*network* = {

*nodes* = (

{ ….node1

},

{ …node2

}

);

};

A configuration file can currently store a single network. The abovementioned global parameters are valid for the whole network (including all nodes and all connections).

### Defining a Network Node

A network node is defined in the configuration file as follows:

{

*testbed\_ip* = “”;

*running\_mode* = "";

*label* = "";

*role* = [“”,””];

*connections* = (

{

… connection 1

},

{

… connection 2

}

);

}

*testbed\_ip:* The IP address (in dotted decimal format) to which the deployment utility will copy the configuration files and remotely execute commands.

*running\_mode*:The mode in which Blackadder will run in this node. Use “user” for user-space (as user-space process) or “kernel” for kernel space (as a Linux module). Note that in a testbed some nodes may run in user-space while others run in kernel-space.

*label:* The Label of that network node. 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.

### Defining a Network Connection

A network connection is always unidirectional and is defined within the context of a network node as follows:

{

to = "00000002";

src\_ip = "10.0.1.18";

dst\_ip = "10.0.1.19";

}

*to:* the destination node (its label)

If the network runs on top of 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 network runs on top of Ethernet:

src\_if: the network interface from which publications will be sent. The deployment will use this and remotely acquire the respective MAC address. E.g. use “eth0” or “tap0” and Blackadder will use a physical interface or a virtual one (probably over a VPN), respectively.

*dst\_if:* the network interface to which publications will be sent. The deployment will use this and remotely acquire the respective MAC address.

*src\_mac:* the MAC address of the source network interface. This parameter is optional and if given, the deployment script will not acquire the MAC address remotely. Helpful when the network contains nodes that will not be online at the time the deployment utility will run.

*dst\_mac:* the MAC address of the destination network interface. This parameter is optional and if given, the deployment script will not acquire the MAC address remotely.

For the exact syntax check the sample\_topology.cfg file in the deployment folder as well as the configuration file grammar in <http://www.hyperrealm.com/libconfig/.>

## Using a GraphML file as Input (not supported yet)

The deployment utility will not read the configuration file describing the network topology. Instead, it will directly read a GraphML file describing the topology. Very helpful for automatically creating large topologies for PlanetLab or for drawing topologies in visualization tools, like yED or Gephi.

More soon…