COPSS Instructions

Georg-August-Universität Göttingen

GreenICN project

July 5th, 2014

{jiachen.chen, bruno.ricci}@informatik.uni-goettingen.de

COPSS Preparation

- Operating system:
  - Tested on Ubuntu 12.04 LTS, Ubuntu 13.10 and Ubuntu 14.04
  - Exact distribution/version should not matter
  - Commands in this guide: Ubuntu 13.10

- Java:
  - openjdk-7-jdk

- CCNx:
  - CCNx 0.8.0 (provided with the package)
  - Wireshark 1.8.6 (provided with the package), with CCNx plugin

- COPSS binary:
  - Provided with the package

Java

- Java 7 installation:
  - `sudo apt-get install openjdk-7-jdk openjdk-7-jre`
  - Please make sure that you are using openjdk 7 (not 6). Test:
    - `java -version`

- Sometimes, you will need to switch to java 7 from 6. Two options for Java 6/7 switch:
  - Removal, but might break/not work with other apps
    - `sudo apt-get remove openjdk-6-jdk openjdk-6-jre`
  - Updating java/javac alternatives, running *both* the commands
    - `sudo update-alternatives --config javac`
    - `sudo update-alternatives --config java`

From now on, “switch to Java *” refers to one of the two solutions above

CCNx

- Prerequisites (taken from https://www.ccnx.org/wiki/CCNx/InstallingCCNx)
  - `sudo apt-get install build-essential ant autoconf automake libssl-dev libexpat1-dev libpcap-dev libecryptfs0 libxml2-utils gawk gcc g++ git-core pkg-config libpcre3-dev`

- Compilation and installation steps:
  1. Unpack ccnx-0.8.0.tar.gz and move into its directory
  2. Compile and install in the classical way:
     - `./configure`
     - `make`
     - `sudo make install`

- Checkpoint: check the following commands
  - `ccndstart`
  - `ccndstop`
  - `ccndc add`
  - `ccnr`
  - `ccnputfile`
  - `ccngetfile`

Wireshark

- Prerequisites
  - `sudo apt-get build-dep wireshark`

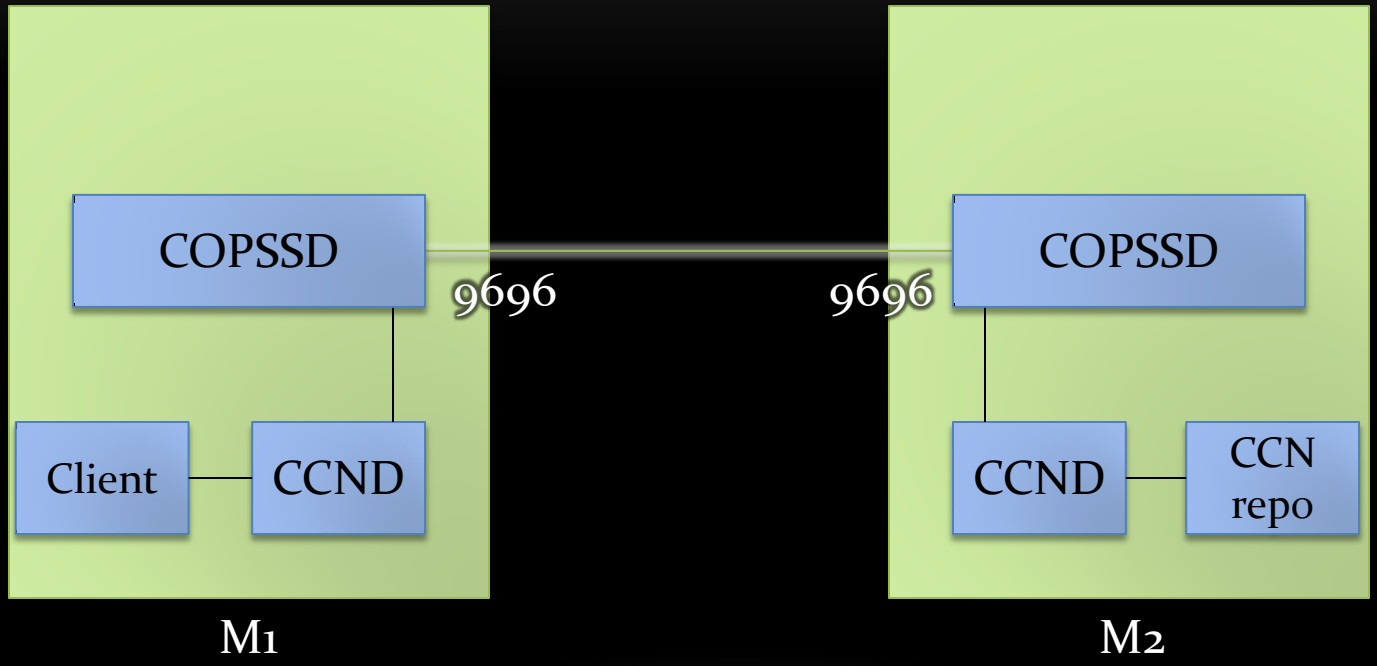
- Compilation and installation steps:
  1. Unpack wireshark-1.8.6.tar.bz2 and move into its directory
  2. Go to wireshark folder
  3. Copy all the CCNx plugin files in the plugins/ccn directory
     - `cp -R ../ccnx-0.8.0/apps/wireshark/ccn plugins/ccn`
  4. Apply patch from CCNx.
     - `patch -p1 < ../ccnx-0.8.0/apps/wireshark/wireshark-1.8.6.patch`
  5. Run autogen.sh and then compile and install in the classical way:
     - `./autogen.sh`
     - `./configure`
     - `make`
     - `sudo make install`
- Checkpoint: launch a Wireshark capture and do some CCNx related operations (e.g. `ccngetfile`/`ccnputfile`)

Start COPSS

- Run COPSS binary:
  1. Move to `COPSSBinary` directory
  2. `java -jar COPSSD.jar`
  - You can see a “FileNotFoundException” (see Create an initial COPSS setting) .
- Commands available:
  - `link %address% %port% %isRouter%`
    - link to a node on address:port and tells if the node is a router.
  - `FIB %name% %address% %port%`
    - add an FIB entry name->address:port
  - `RP %RPName%`
    - starts an RP module using RPName
  - `status`
    - show the status of the COPSSD
  - `help`
    - show help message.
  - `stop`
    - stop COPSSD.

Test 1: Query/response using COPSS wrapper

- Architecture

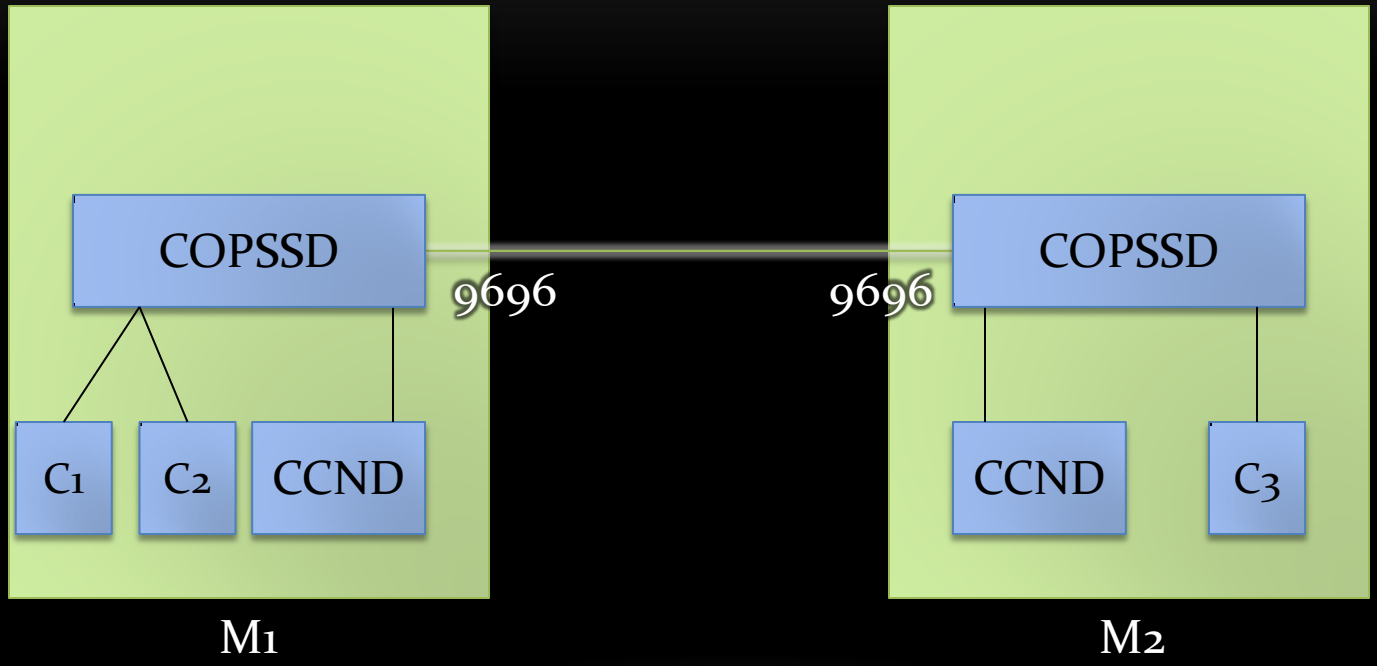


Test 1: Query/response using COPSS wrapper

- Steps:
  - Start `ccnd` and `COPSSD` on M1 and M2
  - In COPSSD on M1:
    - `link M2.IP 9696 true`
    - COPSSD will listen on a random port (M1.P1) and create a face in CCNx on that port
  - In COPSSD on M2:
    - `link M1.IP 9696 true`
    - COPSSD will listen on a random port (M2.P1) and create a face in CCNx on that port
    - If you see "Cannot find face…", ignore the messages
  - In COPSSD on M1
    - `FIB /test M2.IP 9696`
      - COPSSD on M1 will add an FIB entry in CCNX: "/test" → M1.P1
    - `FIB /ccnx.org M2.IP 9696`
      - This command is for key exchange when sending file.
  - On M2 start `ccnr` and put a file into the repo using `ccnputfile` and providing ContentName "`/test/file1`"
  - On M1:
    - `ccngetfile /test/file1 test`

Test 1: Pub/Sub using COPSS

- Architecture



Test 2: Pub/Sub using COPSS

- Steps:
  - Start `ccnd` and `COPSSD` on M1 and M2
  - In COPSSD on M1:
    - `link M2.IP 9696 true`
    - COPSSD will listen on a random port (M1.P1) and create a face in CCNx on that port
  - In COPSSD on M2:
    - `link M1.IP 9696 true`
    - COPSSD will listen on a random port (M2.P1) and create a face in CCNx on that port
  - In COPSSD on M1
    - `FIB /RP M2.IP 9696`
  - In COPSSD on M2
    - `RP /RP`
  - Start `SimpleCOPSSClients` (with the package in `SimpleCOPSSClientBinary`)
    - `java -jar SimpleCOPSSClient.jar %listenPort%`
    - Cn listens on Cn.P

Test 2: Pub/Sub using COPSS

- Steps:
  - In COPSSD on M1:
    - `link 127.0.0.1 C1.P false`
    - `link 127.0.0.1 C2.P false`
    - This links COPSSD with the clients
  - In COPSSD on M2:
    - `link 127.0.0.1 C3.P false`
  - If you see “Invalid packet type” on clients, ignore.
  - Commands available in SimpleCOPSSClient:
    - `sub`: subscribe to a set of CDs
    - `unsub`: unsubscribe from a set of CDs
    - `pub`: publish a message
    - `help`: print this message
    - `stop`: cleanup the states and exit the program
    - Examples (see next slide)
  - Use different combinations of subscription and publication to send/receive data

Test 2: Pub/Sub using COPSS

- SimpleCOPSSClient commands example:

```
> sub
Please input CDs to subscribe (1 CD per line), end with an empty line
? /sports/football
Subscription to /sports/football done.
? /sports/basketball
Subscription to /sports/basketball done.
?
> unsub
Please input CDs to unsubscribe (1 CD per line), end with an empty line
? /sports/football
Subscription to /sports/football removed.
?
> pub
Please input CDs to publish (1 CD per line), end with an empty line
? /sports/football
CDs: [/sports/football]
? /news/bbc
CDs: [/sports/football, /news/bbc]
?
Please input content publish, end with an empty line
? This is a test news
? This is the second line
?
Message sent.
>
```

Subscribe to /sports/football /sports/basketball

Unsubscribe from /sports/football

Publish a message

Programming interface of COPSS client

- For query/response, please use the original CCNx commands
- For FIB add/removal please use the commands in COPSSD
- Library for COPSS client (with the package, in EndHostLib)
  - `package common;`
  - `abstract class NetworkListener:`
    - Listens to a UDP port and handles UDP packets
    - `public NetworkListener(int port)`
      - Initiator of the class
      - `port`: the UDP port to listen to
    - `protected abstract void handlePacket(DatagramPacket packet)`
      - Callback function, called when there is a packet on the port
      - `packet`: the packet received
    - `protected void send(InetSocketAddress target, byte[] buf)`
      - `packet`: the target to send the UDP packet to
      - `buf`: the content of the UDP packet

Programming interface of COPSS client

- Library for COPSS client (with the package, in EndHostLib)
  - `package copss.protocol;`
  - `public final class Control extends GenericXMLEncodable:`
    - Packet for subscribe/unsubscribe
    - This packet can also be extended to FIB change and others. The control messages in COPSS. Control.ControlType.FIBChange not implemented.
    - `public Control(Control.ControlType type, LinkedList<ContentName> contentNameAdd, LinkedList<ContentName> contentNameRemove, int version, int ttl)`
      - Initiator of the class
      - `type`: type of the packet (Control.ControlType.STChange for now)
      - `contentNameAdd`: CDs to subscribe to
      - `contentNameRemove`: CDs to unsubscribe from
      - `version`: version of the packet (reserved)
      - `ttl`: TTL of the packet, -1 for infinite
    - Check `org.ccnx.ccn.protocol.Interest` in CCNx to see how to convert between Control (Interest) and byte[]

Programming interface of COPSS client

- Library for COPSS client (with the package, in EndHostLib)
  - `package copss.protocol;`
  - `public final class Multicast extends GenericXMLEncodable:`
    - Packet for Publish (multicast)
    - `public Multicast(LinkedList<ContentName> contentNames, byte[] content)`
      - Initiator of the class
      - `contentNames`: CDs of the message
      - `content`: content of the message
    - Check `org.ccnx.ccn.protocol.Interest` in CCNx to see how to convert between Multicast (Interest) and byte[]

- Example COPSS client code in folder SimpleCOPSSClient
  - Only 1 Java file (simplecopssclient.SimpleCOPSSClient.java)

Programming interface of COPSSD

- To start/stop COPSSD from Java program rather than command line, you can create COPSSD object in your code directly if you use COPSSD.jar (and other jars in the COPSS binary lib) as your library.
- `package copss.protocol;`
- `public class COPSSD extends NetworkListener`
  - COPSS Daemon
  - `public COPSSD(int listenPort, int ccnPort)`
    - Create a COPSS daemon and start listening on listenPort
    - `listenPort`: the port COPSSD listens to
    - `ccnPort`: the local UDP port CCND listens to
  - `public int link(InetSocketAddress address, boolean isRouter)`
    - Same as link command
  - `public boolean addFIB(ContentName prefix, InetSocketAddress outgoingAddress)`
    - Same as FIB command
  - `public int setRP(ContentName rpName)`
    - Same as RP command

Programming interface of COPSSD

- To start/stop COPSSD from Java program rather than command line, you can create COPSSD object in your code directly if you use COPSSD.jar (and other jars in the COPSS binary lib) as your library.
- `package copss.protocol;`
- `public class COPSSD extends NetworkListener`
  - COPSS Daemon
  - `public String toString()`
    - Same as status command
  - `public void stop()`
    - Same as stop command

- Please check COPSSDsrc/Main.java to see how to control COPSSD object

Create an initial COPSSD setting

- By default, COPSSD will read “Command.txt” for startup settings
  - That’s why you see the FileNotFoundException at the beginning
- You can put COPSSD commands in the Command.txt file to avoid typing the commands repeatedly.
  - E.g., on M1, you can write in Command.txt
    - `link M2.IP 9696 true`
    - `link 127.0.0.1 C1.P false`
    - `link 127.0.0.1 C2.P false`
    - `FIB /RP M2.IP 9696`
    - `FIB /test M2.IP 9696`
    - `FIB /ccnx.org M2.IP 9696`
    - …