Technical University Berlin

Telecommunication Networks Group

CMAP HOW TO
A short guide to install, configure and start up CMAP

Andreas Festag

festag@ee.tu-berlin.de

Berlin, June 2000

TKN Technical Report TKN-00-04

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

This document describes the usage of CMAP - the ***C**onnection **M**anagement **A**ccess **P**rotocol* - as part of the WU Gigabit Switch kit software distribution. Since running CMAP requires other protocols belonging the WUGS software environment, the details of configuration and start up of this software instances is also described. Moreover the **MOMBASA** testbed at TKN, Technical University Berlin, [10] is described.**MOMBASA** stands for **Mo**bility Support - A **M**ulticast **Bas**ed **A**pproach. In this approach the utilization of Multicast for handover is studied and an experimental testbed, which is based on the WU Gigabit Switch kit is set up. In this document describes how to install the CMAP extensions, and how to configure and start up the MOMBASA testbed components including CMAP.

This document bases mainly on a document about an out-of-date version of the WUGS Software environment [3], on information from the WUGS mailing list *gigabitk-its@arl.wustl.edu* and own experiences with CMAP in the MOMBASA testbed. As a matter of course it does not replace the study of the CMAP specification and other documents, such as [2, 3, 4, 5, 6, 7, 8].

Please send comments about this document and the MOMBASA software to

festag@ee.tu-berlin.de

Contents

Chapter 1

CMAP Installation

The WUGS software including CMAP can be downloaded from

http://www.arl.wustl.edu/gigabitkits/secure/download.html

For download an user name and password is required. In the following examples it is assumed that the software is installed on *ada.ee.tu-berlin.de* (*ada-atm.ee.tu-berlin.de*) in

/usr/wugs/src/

In the MOMBASA testbed the software was compiled

- on Linux 2.2.10 with egcs-2.91.66
- on NetBSD 1.4.1 with egcs-2.91.60

To compile CMAP it is required that the standard WUGS software is already installed. CMAP makes use of the connection manager (CM), therefore the installation of CMAP includes also the CM. It is recommended to modify some library paths in

```
/usr/wugs/src/src/make.defs.common
```

The modifications are listed in the appendix A. For installation of the CM and CMAP execute the following steps:

```
make CM_CLEAN CM_DEPEND CM_INSTALL
make CMAP_CELAN CMAP_DEPEND CMAP_INSTALL
```

For a better understanding of the CMAP and CM operations debug output is very helpful. Therefore compilation flags can be set in

```
/usr/wugs/src/src/Makefile
```

and the line DEBUG should be modified:

```
DEBUG = -g -D_ERR_DEBUG
```

The CM and CMAP software needs to be recompiled. The software instances generate debug output according to the configured debug level. Usually the debug output can be found in temporary debug files (see 2.6).

To measure the duration of particular operations in the switch controller (e.g. How long does it take to add an endpoint to the call?) the TIMING functionality can be used. Therefore the compilation flag

```
DEBUG = -g -DTIMING
```

can be used instead or additionally to the D_ERR_DEBUG flag.

To run the software environment some environment variables should be set (example: for bash modify ~user/.profile):

```
export OSTYPE=Linux
export WUGSETCPATH=/usr/wugs/etc/
export ARLBINPATH=/usr/wugs/bin/
export WUGSCONFPATH=/usr/wugs/conf/
export MOMBASA=/usr/wugs/mombasa/
```

For machines running other operation systems than Linux the variable OSTYPE must be set appropriately.

It is recommended to mount /usr/wugs/src on the other machines. Therefore on *ada* (*ada-atm* an entry to export the file system must be in /etc/exports and on all machines using the file system the entry

```
ada:/usr/wugs /usr/wugs nfs noauto,user,nosuid,rw,exec,soft
```

must exist in

```
/etc/fstab
```

Chapter 2

CMAP Configuration and Start Up

In this section the configuration and start up of the particular software instances are described. It includes:

- Node Simulator (NodeSim)
- Switch Controller (GBNSC)
- Connection Manager (CM)
- CMAP Session Manager (CMAP_SM) and
- CMAP Clients (CMAP_Cl)

2.1 Node Simulator

A node simulator (NodeSim) simulates the control behavior of an WUGS-20 cell switch. The node simulator is only required when no switching hardware is used.

A node simulator (NodeSim) is started as follows:

NodeSim [-h] [-d|-D] -f|-F *conffile* [-s|-S *num*] [-f *peerfile*]

h	Help message
d D	Debug information
f F *conffile*	Configuration file name
s S *num*	Number of simulated switch
p P *peerfile*	Peer file name (Not used)

The configuration file is required, the other parameters are optional. An example configuration file is shown below.

```
node 0 {
        SW 1 {
                SW_size 8;
                Control_port    0;
                Input_buf_size  128;
                Recycling_size  32;
                VXT_size        1024;
        }
        LINK {
```

```
        }

        CP_port : SW 1, 0;
}
```

The first and second line indicate that the (simulated) switch belongs to node 0 and is switch #1 within the node. Both values can be modified. The CP_port section indicates that the control processor is connected to switch 1 and port 0. The LINK section can be used for the case that a node consists of several switches controlled by a single switch controller.

An example peer file for two node simulators is shown below (possibly for future use, since it it not applied in the recent version of the node simulator).

```
"ada-atm" 1 2 1 2
"milano-atm" 1 4 1 4
```

The entries in the lines are set according to the GBNSC configuration file (LINK section) and should be interpreted as:

```
Hostname sndSw sndPort rcvSw rcvPort
```

A node simulator uses TCP port 4500 to communicate with other NodeSim instances by default. (See /usr/wugs/src/common_code/ATM_card/ATMCard.h)

Example:

```
$ARLBINPATH/$OSTYPE/NodeSim -d 100
-f $MOMBASA/CONFIGS/NodeSim.config.ada
-p $MOMBASA/CONFIGS/NodeSim.peer_file -s 1
```

2.2 Switch Controller

The switch controller (SC) controls a single switch. It hides details of the hardware from higher layers and monitors the state of the switch. Access to the switch hardware is given through the Jammer, a low level script tool.

The switch controller can be started separately *or* with higher layer protocols (e.g. CM and CMAP). In the second case the connection manager requires, that the connection manager is started first and after then the switch controller is called by the connection manager (see CM section).

A switch controller is started as follows:

GBNSC [-h|-H] [-r|-R] [-s|-S *simHostName*] [-P *tcpPort1*] [-p *tcpPort2*] [-n|-N *ncmoNumber*] [-d|-D *debugLevel*] -[i|-I *nodeIdentifier*] [-m|-M *nodeIdMask*] [-Q *inputPriqueID*] [-q *outputPriqueID*] [*filename*]

h H	Help message
r R	Use raw mode. SC only sends one ping cell through switch at startup
s S *simHostName*	Hostname where NodeSim is running.
P *tcpPort1*	TCP port GBNSC should listen on.
p *tcpPort2*	TCP port on which NodeSim is listening, if any.
n N *ncmoNumber*	Parent sets NCMO index for synchronization with child GBNSC. (Unknown option)
d D *debugLevel*	Controls how much debug output is printed.
i I *nodeIdentifier*	Node ID. Helpful in building trees of NCs and GBNSCs. (Unknown option)
m M *nodeIdMask*	Node ID Mask. Helpful in building trees of NCs and GBNSCs. (Unknown option)
Q *inputPriqueID*	Input (from parent) PRIQUE shared memory ID. (Unknown option)
q	*outputPriqueID* Output (to parent) PRIQUE shared memory ID. (Unknown option)
filename	GBNSC configuration file name.

For the usage of the switch controller with the APIC NIC a patch have to be installed (see URL http://www.arl.wustl.edu/gigabitkits/secure/download). Then the executable

```
$ARLBINPATH/NetBSD/GBNSC.apic
```

should be used instead of GBNSC with two additional parameters

a	use the APIC NIC instead of the ENI NIC
b	Size of the allocated buffer for the control connection (Recommended: 64000)

Note that the recent version of GBNSC.apic is for NetBSD only.

The switch controller is configured by parsing a text file. This configuration file specifies the control interface of the switch and the other ends of the links, which are necessary for operation of the connection management. A sample GBNSC configuration file is shown below:

```
switch 1 GBN
TCPPORT 3550 3560
PORTS 8
CHIPS 1 2
        IPP CHIP 3
                6
                7
END

PARAMS
        VPT 1
        CONFIGURATION TIMEOUT 60
        POLLING TIMEOUT 60
END

CONTROL
        0 0
        0/32
        0/32
END

LINKS
```

```
        0 <=> UNI @  1550000 @ 12000000 "ada-atm" 1
        1 <=> UNI @  1550000 @ 12000000 "milano-atm" 1
        2 <=> UNI @  1550000 @ 12000000 "127.0.0.2" 1
        3 <=> UNI @  1550000 @ 12000000 "asterix-atm" 1
        4 <=> UNI @  1550000 @ 12000000 "obelix-atm" 1
        5 <=> UNI @  1550000 @ 12000000 "roma-atm" 1
        6 <=> UNI @  1550000 @ 12000000 "turino-atm" 1
        7 <=> NNI @ 12200000 @ 12000000  3 2 "127.0.0.2"
END

INIT
        IPP 0 VPCOUNT 255
        IPP 1 VPCOUNT 255
        IPP 2 VPCOUNT 255
        IPP 3 VPCOUNT 255
        IPP 4 VPCOUNT 255
        IPP 5 VPCOUNT 255
        IPP 6 VPCOUNT 255
        IPP 7 VPCOUNT 255
END
```

In the configuration file above the first line indicates the number of the switch within the node. Then it is specified that the GBNSC uses the TCP port 3550 for communication. The second entry in the line TCPPORT is for GSMP (Not used yet). The CONTROL section indicates, that the control processor is connected to port 0 and VPI/VCI 0/32 is used for sending and receiving signaling messages. The LINKS section specifies whether it is an UNI or a NNI link. An UNI line should be read as:

Port Direction Type Speed Client SM-Owner-Local-ID

NNI line should be read as:

Port Direction Type Speed NodeID SwitchID PortID Other-End-CM-Machine

2.3 Connection Manager

The connection manager (CM) implements a distributed control for a network with multiple nodes. Connection managers communicate with one another using CMNP (***C**onnection **M**anagement **N**etwork **P**rotocol*) [5].

A connection manager (CM) is started as follows:

CM [-h] [-n *N*] [-t *T*] [-b N] [-c *C*] [-r *R*] [-p *P*] [-m *D_N*] [-x *D_C*] [-v *N*] [-y *N*] [-n *SC executable*] [-s *SC options*]

h	Help message
n *N*	N=NodeController Name
t *T*	T=Timer filename (Unknown option)
b *N*	b=bin directory (Unknown option)
c *C*	C=Configuration filename
r *R*	R=Router Configuration filename
p *P*	P=Number of Processors in Machine, def = 1 (Unknown option
m *D_N*	NCMO debug level D_N
x *D_C*	CMMO debug level D_C
v *N*	Use machine N as the Hand Simulation Engine (Unknown option)
y *N*	Use machine N as the Hand Simulation Engine for all CMNP traffic (Unknown option)
n *SC executable*	Switch controller executable including path
s *SC options*	Switch controller options; only with the -n option; should be repeated if several options have to be set.

The router configuration file gives information so that the connection manager and call manager processes find each other. This is an example for a configuration with two switch controllers (Router.conf):

```
CMPORT  4534
SMPORT  4535

NODE

CM ARL1 1 ada-atm
CM ARL2 2 roma-atm
SM ARL1 ada-atm
SM ARL2 roma-atm

Route

1 2 2
2 1 1
```

The connection managers use the TCP Port `4534`, session managers `4535`. There are two connection managers and two session managers defined. The connection manager for a node called `ARL1` is running with a node identifier of `1` and is on a machine called `ada-atm`. The connection manager for a node called `ARL2` is running with a node identifier of `2` and is on a machine called `roma-atm`. The SM for `ARL1` is running on `ada-atm`. The SM for `ARL2-atm` is running on `roma-atm`.

The lines in the Route section should be interpreted as:

x y z

On node x, to get to node y go through node z.

Example:

```
$ARLBINPATH/$OSTYPE/CM -d 200
-c WUGSCONFPATH/CONFIGS/GBNSC.config.ada-atm
-r $WUGSCONFPATH/Router.config
-n $ARLBINPATH/$OSTYPE/GBNSC -d 100 -s '-r' -s '-i1' -s '-sada-atm' -s '-p4500' ARL1
```

It is recommended to start the GBNSC (or GBNSC.apic) indirectly by the CM. Therefore the option *-n* can be used to indicate the executable including the path. The *-s* option gives the required parameters for the GBNSC.

2.4 CMAP Session Manager

The CMAP session manager (CMAP_SM) realizes a session abstraction (*CALL*) used by signaling clients to request and manage connections in a network with WUGS-20 cell switches. It allows general, dynamic, multi-connection, multicast sessions. The session manager uses CMAP (***C**onnection **M**anagement **A**ccess **P**rotocol*).

A CMAP session manager (CMAP_SM) is started as follows:

CMAP_SM *highId lowId launcherhostname launcherTCPPort* -+ [-d|-D *N*] [-c|-C *N*] [-r|-R *Router-Filename*]

-CL *highId lowId*	CMAP Identifier of the call manager lowId is always 0
launcherhostname	Name of host running the launcher
launcherTCPPort	TCP port the laucher is listening (The CMAP session manager handshakes with the launcher)
d D *N*	Debug Information of level N Recommended values: 0: Startup and Fatal Stuff 10: CMAP Message Type, source and destination address 20: CMMO operations 30: Non-data debugging information (for tracing flow of control) 40: Data debugging information (for extra help) 50: All other (for now)
c C *N*	Level of base debug
r R *RouterFilename*	Name of the router file

Example:

```
$ARLBINPATH/$OSTYPE/CMAP_SM 1 0 ada-atm 1568 -+ -d 100 -c 0 -r
   $WUGSETCPATH/Router.config
```

2.5 CMAP Client

A CMAP client (CMAP_Cl) requests and manages network resources. Therefore it uses CMAP to communicate with the CMAP session manager. In particular a CMAP client can

- Open, modify and close a call
- Add, modify and drop a connection to/from a call
- Add, modify and drop an end point to/from a call
- Trace a call or an end point to/from a call

In general a CMAP client (CMAP_Cl) is started as follows:

executable -CL *highId lowId hostname TCPPort* -OT highId lowId -+

-CL *highId lowId*	CMAP Identifier of the CMAP client
hostname	Name of host running the CMAP client
TCPPort	TCP port to communicate with the CMAP session manager (usually established by the launcher)
-OT highId lowId	CMAP Identifier of other CMAP client(s)
-+	Indicates that some additional parameters follow.

Example:

```
$ARLBINPATH/$OSTYPE/fh -CL 1 1 asterix-atm 4881 -OT 1 2 -OT 1 3 -OT 1 4 -OT 1 5 -+ -SOFT
   -3 -5
```

2.6 Launcher

The launcher distributes instances of the signaling software to the individual switch controllers, base stations and end systems. This includes CMAP Clients (CMAP_Cl), CMAP session manager (CMAP_SM), connection manager (CM) and switch controller (SC). Alternatively to the switching hardware a single or several Node Simulators (NodeSim) can be started. The launcher takes two arguments: Configuration parameters are usually in *configuration*-files. They are used to set hardware specific parameters and for communication among the signaling protocols instances itself. *Topology*-files are used to assign the signaling protocols instances to the hardware.

A launcher is started as follows:

launcher *topology-file*

topology-file Used to construct command lines. It may contain the following keywords:

RSH	Remote shell
RUN	
SM	Session Manager
CL	CMAP Client
-UR	You are (used for Clients)
-OTHER	Other clients
-AND	For additional arguments (e.g. for session manager and clients
[0-9].[0-9]	Used for CMAP addresses (n.0 is for session managers, other for clients).
SLEEP	Sleep
SHOW	
ECHO	

Below is a typical topology file, for starting a node simulator, a connection manager (which in turn starts a switch controller), a CMAP session manager and several CMAP clients.

```
RSH RUN "$ARLBINPATH/$OSTYPE/NodeSim" "ada-atm" "-d 100 -f
$WUGSCONFPATH/NodeSim.config.ada-atm -s 1 >
/tmp/NodeSim-LOG.ada-atm 2>&1" ;

RSH RUN "$ARLBINPATH/$OSTYPE/CM" "ada" "-d 200 -c
$WUGSCONFPATH/GBNSC.config.asterix -r
$WUGSCONFPATH/Router.config -n
$ARLBINPATH/$OSTYPE/GBNSC -d 100 -s '-r' -s '-sada-atm' -s
'-p4500' ARL1 > /tmp/CM-LOG.ada-atm 2>&1" ;

SM RUN "$ARLBINPATH/$OSTYPE/CMAP_SM" "ada-atm" -UR 1.0
-AND " -d 100 -c 0 -r $WUGSCONFPATH/Router.config
1> /tmp/SM-LOG_1.ada-atm 2>&1";

CL RUN "$ARLBINPATH/Linux/fh" "roma-atm" -UR 1.1 -OTHER
1.2 -AND "-SOFT -3 -5 1> /tmp/testbed_logs/CL-LOG_fh_11.roma-atm 2>&1";
```

The launcher parses the topology file above. As a general rule the topology file consists of:

```
RSH RUN Program Hostname Other_arguments
```

(for NodeSim and CM) or

```
PROGRAM_ID RUN Program Hostname Other_arguments
```

(for CMAP_SM or CMAP_Cl). Note that in both cases the programs are started via RSH . This requires that password checks have to be bypassed (See man rhosts for details)

The launcher constructs a command line as required and starts the node simulator, connection manager, CMAP session manager and CMAP clients remotely via rsh . As explained before the switch controller GBNSC is started indirectly by the connection manager.

Example:

```
$ARLBINPATH/$OSTYPE/launcher $WUGSCONFPATH/topo.1
```

2.7 Jammer

The Jammer is a low level script language, which offers access to switch tables and registers within the switch. It supports interactive or batch-oriented input and simple programming constructs. Before running Jammer, it is necessary to start NodeSim (if no switch should be used) and the switch controller.
The Jammer is started as follows:

Jammer [-h|-H] [-g|-G] -f -s|S -D*value* -d*value* -T*num* -t*num* *host_name* *tcp_port* [*include_filename*]

h H	Help message
g G	forces core dump for SIGBUS and SIGSEGV. Debugging aid.
f	Handle include files through buffer file instead of memory.
s S	Keep Jammer small, don't buffer at all.
D *value*	Set Debug_ON to value.
d *value*	Set JAM_base_debug to value. Larger value means less debug output.
T *num*	Set maximum number of outstanding transactions before Jammer pauses.
t *num*	Set maximum number of outstanding transactions before Jammer resumes after pausing. This is for hysteresis.
switch_address	switch_address (e.g. 1.1). This is not the IP address)
host_name	Hostname the switch controller is running on.
tcp_port	TCP port on which the switch controller is listening.
include_filename	Jammer script include file to process.

See other documents (e.g. Jammer HOW TO) for details of Jammer and for predefined procedures for easy usage.

Example:

```
$ARLBINPATH/$OSTYPE/Jammer 1.0 ada 3550
```

The Jammer can also be used simultaneously with the CM. Then the Jammer and CM accesses the GBNSC in parallel. This is an important feature in order to track the operations of the CMAP_SM.

Chapter 3

MOMBASA Installation, Configuration and Start up

MOMBASA stands for ***Mo**bility Support - a **M**ulticast **Bas**ed **A**pproach*. In this approach the utilization of multicast for handover in mobile cellular networks is investigated and an experimental testbed is set up [Figure 3.1].

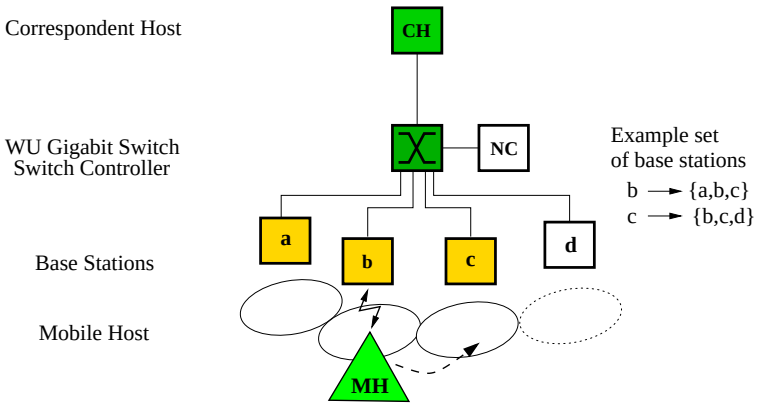

Figure 3.1: MOMBASA testbed

One of the studied multicast schemes is based on CMAP as an access protocol, which realizes a dynamic multipoint-to-multipoint communication in switched network. The MOMBASA testbed demonstrates a number of CMAP operations, such as dynamic *open-call* and *close-call* operations, *add-endpoint* and *drop-endpoint* operations, *trace-call* and *trace-endpoint* operations. Moreover in the testbed some important CMAP features, such as *surrogate signaling*, the setup of multiple *parallel* connections with different QoS parameters, usage of the *call monitoring* feature [4]. This section describes the MOMBASA testbed setup as far as it is relevant for usage of CMAP.

3.1 MOMBASA Installation

The software which is required to run the overall setup can be downloaded from

http://www-tkn.ee.tu-berlin.de/research/mombasa.tgz

The software includes:

- Source code of the CMAP protocol extensions
- Source code of the CMAP clients
- Configuration files of the MOMBASA setup at TKN
- Source code of a lightweight signaling protocol for handover control
- Shells to run the overall setup

The software can be installed in /usr/wugs with

```
cp mombasa.tgz /usr/wugs
cd /usr/wugs
tar xvzf mombasa.tgz
cd $MOMBASA
make all
```

3.1.1 CMAP Protocol Extensions

The extensions to the CMAP protocols include mainly the following functionality:

- Tracing calls with the *trace_call* command
- Tracing endpoints with the *trace_ep* command
- Suppression of announcements of the call owner for the appropriate monitoring field

The extensions are in line wih the CMAP specification [4]. The following files were modified:

```
/usr/wugs/src/CMAP/common/CMAP_MessageTraceCall.h
/usr/wugs/src/CMAP/common/CMAP_MessageTraceCall.C
/usr/wugs/src/CMAP/common/CMAP_MessageTraceEp.h
/usr/wugs/src/CMAP/common/CMAP_MessageTraceEp.C
/usr/wugs/src/CMAP/client/Makefile
/usr/wugs/src/CMAP/client/CMAP_HandleTraceCall.C (new file)
/usr/wugs/src/CMAP/client/CMAP_HandleTraceEp.C (new file)
/usr/wugs/src/CMAP/client/CMAP_MessageHandler.C
/usr/wugs/src/CMAP/client/CMAP_OperationHandler.h
/usr/wugs/src/CMAP/client/CMAP_OperationHandler.C
/usr/wugs/src/CMAP/network/CMAP_SM_AddEpHandler.C
/usr/wugs/src/CMAP/network/Makefile
/usr/wugs/src/CMAP/network/CMAP_SM_Call.h
/usr/wugs/src/CMAP/network/CMAP_SM_Call.C
/usr/wugs/src/CMAP/network/CMAP_SM_CloseCallHandler.C
/usr/wugs/src/CMAP/network/CMAP_SM_CloseCallHandler.h
/usr/wugs/src/CMAP/network/CMAP_SM_HandlersInclude.h
/usr/wugs/src/CMAP/network/CMAP_SM_MessageDistributor.h
/usr/wugs/src/CMAP/network/CMAP_SM_MessageHandlerManager.h
```

```
/usr/wugs/src/CMAP/network/CMAP_SM_TraceCallHandler.C (new file)
/usr/wugs/src/CMAP/network/CMAP_SM_TraceCallHandler.h (new file)
/usr/wugs/src/CMAP/network/CMAP_SM_TraceEpHandler.C (new file)
/usr/wugs/src/CMAP/network/CMAP_SM_TraceEpHandler.h (new file)
```

Up to now it is required to copy the files manually from $MOMBASA/ to /usr/wugs/src . Note that it is recommended to backup the original files before and it is required to recompile and reinstall CM and CMAP.

3.1.2 CMAP Clients

In the MOMBASA testbed the CMAP clients reroute connections for handover. Rerouting consists of the following CMAP operations:

- Trace the call
- Trace the end point of an other CMAP client
- Add the own end point to the call
- Drop the end point of the other CMAP client from the call

The source code of the CMAP clients consists of:

```
$MOMBASA/base_io.H
$MOMBASA/base_io.C
$MOMBASA/bs_shared.h
$MOMBASA/bs_shared.C
$MOMBASA/bs.C
$MOMBASA/bs_inactive.C
$MOMBASA/fh.C
```

The CMAP operations to perform rerouting are implemented in bs . The CMAP client fh performs an *open_call* operation only. The CMAP client bs_inactive does nothing than accepting announcements in the case that a bs client performs surrogate signaling. All CMAP clients are based on bs_shared which implements data structures and procedures common to all CMAP clients (call structures, callback functions, etc.). In base_io the functionality to communicate with the mobile host is realized (see 3.1.3).

Three kinds of rerouting are considered:

- *Hard rerouting:* At first the *add-endpoint* operation and then the *drop-endpoint* operation is performed.
- *Soft rerouting:* At first the *add-endpoint* operation and then the *drop-endpoint* operation is performed. The execution order of the operations is vice versa to the hard rerouting.
- *Predictive rerouting:* The *call* to potential new base stations is pre-established. Therefore the CMAP clients of this potential base stations (which form a set) are added to the *call* in advance. The set of potential new base stations has to be updated after handover: some CMAP clients have to be added, some dropped.

The MOMBASA implementation of the clients utilizes some arguments using the *additional* arguments (see 2.5):

executable -CL *highId lowId hostname TCPPort* -OT highId lowId -+ -[HARD | SOFT | MC] [-C] [-H]

-[*HARD* | *SOFT* | *MC*]	Indicates the rerouting scheme to be applied.
-C	Number of parallel connections, which belong to the call
-H	Overall number of operations, which will be executed.

In the MOMBASA scenario the *other id* is used as follows:the first other client acts as a correspondent host, which opens a call, the second other client is the other base station and the following clients belong to the group of base station, which are involved in rerouting operations.

3.1.3 Lightweight wireless signaling protocol

The lightweight signaling protocol implements a communication protocol between the mobile host and the base stations. This simple protocol bases mainly a *ho_request* and *ho_response* message exchange, but can be modified easily with already pre-defined messages. It is realized in Java (actual version jdk1.2.2 for Linux). The mobile (Java) client communicate with the CMAP clients in the base stations via TCP port 7777. It is required to have JAVA installed on the mobile host. For compilation do

```
cd $MOMBASA/mobile
javac *.java
```

3.2 MOMBASA Configuration

The MOMBASA testbed consists of a single WU Gigabit Switch, a switch controller, four base stations, a correspondent host and a mobile host [Figure 3.2]. The correspondent host remains fixed, whereas the mobile hosts moves through the coverage of the (virtual) cellular network. The movement is *simulated* and the mobile registers alternating with two of the base stations. In that way the mobile performs a *Ping-Pong* handover between the both base station.

The following typical scenario is considered: First the correspondent host opens a *call* with one or several connections. In the call the correspondent host is the *owner*. Then the mobile host registers with the base station *obelix*. Base station *obelix* in turn adds itself to the *call*. When a handover occurs (this is initiated frequently by a time out in the mobile host) the mobile registers with the new base station.

To abstract from an error prone wireless channel the wireless links between the mobile host and base stations are replaced with Ethernet.

The switch controller and correspondent host are equipped with an ATM NIC Efficient Networks 155P-MF1. The base stations are equipped with an APIC NIC. All devices carry additionally an Ethernet NIC.

The mobile host runs Linux 2.2.10; the correspondent host and switch controller Linux 2.2.10 with *atm-on-linux* Version 0.59 (3-JUN-1999) [9]. The base station-s's OS is NetBSD 1.4.1 (with APIC kernel extensions from Washington University).

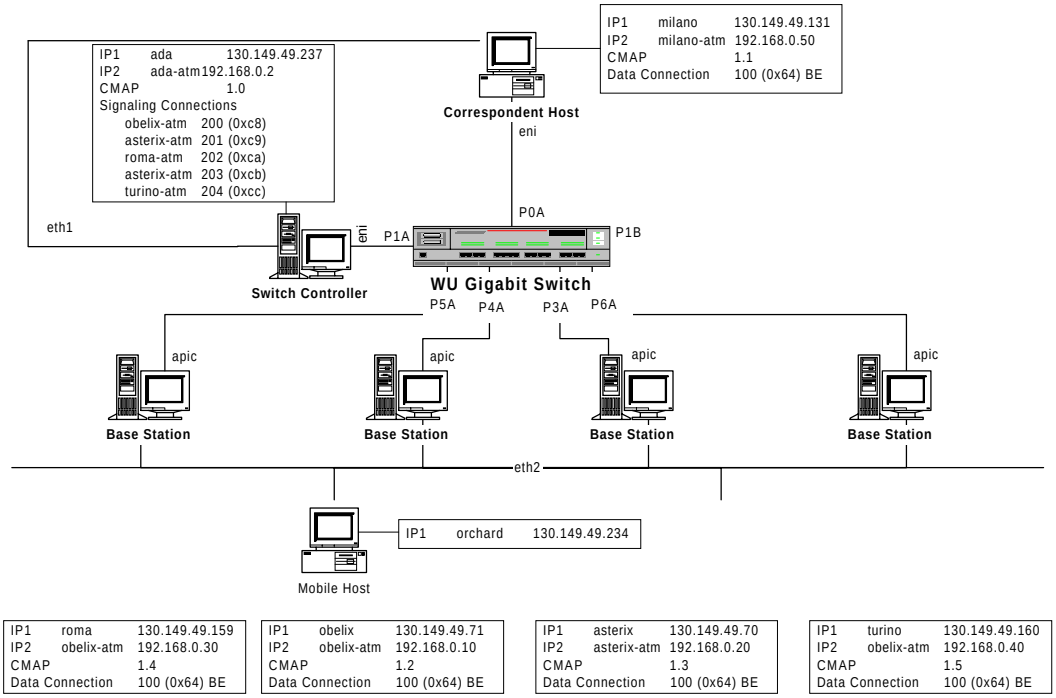

Figure 3.2: MOMBASA Setup

To enable communication between the CMAP session manager on the switch controller and the CMAP clients on the correspondent host and base station, signaling connections (bidirectional) have to be setup (VCP/VCI 0/200-204). For data communication between the correspondent host and the mobile host (via one of the base stations as an IP level gateway) the VPI/VCI 0/100 was chosen. It is required to set the appropriate connections locally in the base station, correspondent host and switch controller via *atm_ifconfig* command (for APIC NICs) or via *atm-arp* command (for the ENI NIC).

3.3 MOMBASA Startup

For easy setup of the WUGS software environment, a shell script can be used (see appendix D). It is assumed to start the script on host *ada.ee.tu-berlin.de*.

The script can be executed with the following options

all.sh [hard|soft|pred] [sim|real]

hard soft pred	Rerouting scheme {hard,soft,predictive}
sim real	Setup with NodeSim (sim) or with WUGS-20 (real)

3.4 Acknowledgement

Thanks to Theodoros Assimakopoulos for an initial testbed setup, to Lars Westerhoff (both TKN) for extensive implementation work and to John de Hardt (ARL, Washington University,

St. Louis) for numerous advices regarding the WUGS environment.

The MOMBASA testbed is in part supported by funding from the BMBF (German Ministry for Science and Technology).

Bibliography

[1] J. Turner "WUGS - Washington University Gigabit Switch Environment" Washington University, St. Louis, USA, URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[2] J. Turner "System Architecture Document for Gigabit Switching Technology" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[3] J. de Hardt "Connection Management Software (CMSS) Architecture" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[4] K. Cox, et al. "Connection Management Access Protocol (CMAP) Specification" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[5] J. deHart "Connection Management Network Protocol (CMNP) Specification" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[6] D. Wu "Node Controller Managed Object (NCMO) and Node Controller Communication Protocol (NCCP)" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[7] K. Cox, et al. "GBNSC: The GigaBit Network Switch Controller" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[8] O. Beal, et al. "Jammer Language Description" available at URL: http://boushi.arl.wustl.edu/gigabitkits/kits.html

[9] W. Almesberger "ATM on Linux Version 0.59 (3-JUN-1999)" available at URL: ftp://lrcftp.epfl.ch/pub/linux/atm/dist/atm-0.59.tar.gz

[10] A. Festag, et al. "Rerouting for Handover in Mobile Networks with Connection-Oriented Backbones - An experimental testbed" Proc. of ICATM'2000, Heidelberg, Germany, June 2000 available at URL: http://www-tkn.ee.tu-berlin.de

Appendix A

Modifications in make.defs.common

Old:

```
CM_LIBS = $(COM_LIBS) -lCM_CMMO -lCM_CMNPENG

CM_LIBS_DEPENDS = $(COM_LIB_DEPENDS) $(ROOT)/lib/$(OSTYPE)/libCM_CMMO.a $(ROOT)/lib/$(OSTYPE)/libCM_CMNPENG.a

CMAP_LIB_DEPENDS = $(COM_LIB_DEPENDS) $(ROOT)/lib/$(OSTYPE)/libSM_CMMO.a
$(ROOT)/lib/$(OSTYPE)/libSM_CMNPENG.a  $(ROOT)/lib/$(OSTYPE)/libCMAPcommon.a
$(ROOT)/lib/$(OSTYPE)/libCMAPclient.a
$(ROOT)/lib/$(OSTYPE)/libCMAPtransport.a
```

New:

```
CM_LIBS = $(COM_LIBS) -lCM_CMMO -L$(CMMO)/$(OSTYPE)/CM_VERSION -lCM_CMNPENG

CM_LIBS_DEPENDS = $(COM_LIB_DEPENDS) $(CMMO)/$(OSTYPE)/CM_VERSION/libCM_CMMO.a
$(CMNP)/$(OSTYPE)/CM_VERSION/libCM_CMNPENG.a

CMAP_LIB_DEPENDS = $(COM_LIB_DEPENDS) $(ROOT)/lib/$(OSTYPE)/libSM_CMMO.a
$(ROOT)/lib/$(OSTYPE)/libSM_CMNPENG.a  $(ROOT)/lib/$(OSTYPE)/libCMAPcommon.a
$(ROOT)/lib/$(OSTYPE)/libCMAPclient.a
$(ROOT)/lib/$(OSTYPE)/libCMAPtransport.a
```

Appendix B

MOMBASA Configuration files

Switch Controller Configuration File 'GBNSC.config.ada-atm'

```
switch 1 GBN
{

TCPPORT 3550 3560
PORTS 8
CHIPS 1 2
        IPP CHIP 3
                6
                7
END


PARAMS
        VPT 1
        CONFIGURATION TIMEOUT 60
        POLLING TIMEOUT 60
END

CONTROL
        0 0
        0/32
        0/32
END

LINKS
        0 <=> UNI @  1550000 @ 12000000 "ada-atm" 1
        1 <=> UNI @  1550000 @ 12000000 "milano-atm" 1
        2 <=> UNI @  1550000 @ 12000000 "127.0.0.2" 1
        3 <=> UNI @  1550000 @ 12000000 "asterix-atm" 1
        4 <=> UNI @  1550000 @ 12000000 "obelix-atm" 1
        5 <=> UNI @  1550000 @ 12000000 "roma-atm" 1
        6 <=> UNI @  1550000 @ 12000000 "turino-atm" 1
        7 <=> NNI @ 12200000 @ 12000000  3 2 "127.0.0.2"
END

INIT
        IPP 0 VPCOUNT 255
        IPP 1 VPCOUNT 255
        IPP 2 VPCOUNT 255
        IPP 3 VPCOUNT 255
        IPP 4 VPCOUNT 255
        IPP 5 VPCOUNT 255
        IPP 6 VPCOUNT 255
```

```
        IPP 7 VPCOUNT 255
END
}
```

Router Configuration File 'Router.config'

```
CMPORT  4534
SMPORT  4535

NODE

CM ARL1 1 ada
SM ARL1 ada

Route
```

NodeSim Configuration File 'NodeSim.config.ada-atm'

```
node 0 {
        SW 1 {
                SW_size 8;
                Control_port    0;
                Input_buf_size  128;
                Recycling_size  32;
                VXT_size        1024;
        }
        LINK {
        }

        CP_port : SW 1, 0;
}
```

Appendix C

MOMBASA Topology files

Topology File for NodeSim and CM 'topo.1a'

```
RSH RUN "$ARLBINPATH/Linux/NodeSim" "$HOSTNAME_CP"   "-d 100
-f $MOMBASA/CONFIGS/NodeSim.config.$HOSTNAME_CP
-s 1 '>' $LOGPATH/NodeSim-LOG.$HOSTNAME_CP '2>&1'" ;

SLEEP 1;

RSH RUN "$ARLBINPATH/Linux/CM" "$HOSTNAME_CP" "-d 100
-c $MOMBASA/CONFIGS/GBNSC.config_apic.$HOSTNAME_CP
-r $MOMBASA/CONFIGS/Router.config -n $ARLBINPATH/Linux/GBNSC.apic
-d 100 -s '-r'  -s '-sada' -s '-i1' -s '
-p4500' ARL1 '>' $LOGPATH/CM-LOG.$HOSTNAME_CP '2>&1'" ;

SLEEP 1;
```

Topology File for CM (No NodeSim) 'topo.1b'

```
RSH RUN "$ARLBINPATH/Linux/CM" "$HOSTNAME_CP" "-d 100
-c $SIMPATH/CONFIGS/GBNSC.config_apic.$HOSTNAME_CP
-r $SIMPATH/CONFIGS/Router.config -n $ARLBINPATH/Linux/GBNSC -s '-d100'
-s '-r' -s '-i1' -s '-p4500' ARL1 '>' $LOGPATH/CM-LOG.$HOSTNAME_CP '2>&1'" ;

SLEEP 1;
```

Topology File for CMAP_SM and CMAP_Cl (for hard and soft rerouting) 'topo.2a'

```
SM RUN "$ARLBINPATH/Linux/CMAP_SM"  "$HOSTNAME_CP"  -UR  1.0 -AND "
-d 100 -c 10 -r $MOMBASA/CONFIGS/Router.config
'1>' $LOGPATH/SM-LOG_1.$HOSTNAME_CP '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/Linux/fh"  "$HOSTNAME_FH" -UR  1.1
-OTHER 1.1 1.2 1.3 -AND "-$HO_RR -1 -1440 '1>'
$LOGPATH/CL-LOG_fh_11.$HOSTNAME_FH '2>&1'";

SLEEP 1;
```

```
CL RUN "$MOMBASA/NetBSD/bs"  "$HOSTNAME_BS1" -UR  1.2 -OTHER 1.1 1.3
-AND "-$HO_RR -1 -1440 1.2 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_12.$HOSTNAME_BS1 '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/NetBSD/bs"  "$HOSTNAME_BS2" -UR  1.3 -OTHER 1.1 1.2
-AND "-$HO_RR -1 -1440 1.3 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_13.$HOSTNAME_BS2 '2>&1'";

SLEEP 1;
```

Topology File for CMAP_SM and CMAP_Cl (for predictive rerouting) 'topo.2b'

```
SM RUN "$ARLBINPATH/Linux/CMAP_SM"  "$HOSTNAME_CP"  -UR  1.0 -AND "
-d 1 -c 1000 -r $MOMBASA/CONFIGS/Router.config '1>'
$LOGPATH/SM-LOG_1.$HOSTNAME_CP '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/Linux/fh"  "$HOSTNAME_FH" -UR  1.1
-OTHER 1.1 1.2 1.3 1.4 1.5 -AND "-$HO_RR -1 -100 '1>'
$LOGPATH/CL-LOG_fh_11.$HOSTNAME_FH '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/NetBSD/bs"  "$HOSTNAME_BS1" -UR  1.2
-OTHER 1.1 1.3 1.4 1.5 -AND "-$HO_RR -1 -100 1.2 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_12.$HOSTNAME_BS1 '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/NetBSD/bs"  "$HOSTNAME_BS2" -UR  1.3
-OTHER 1.1 1.2 1.5 1.4 -AND "-$HO_RR -1 -100 1.3 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_13.$HOSTNAME_BS2 '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/NetBSD/bs_inactive"  "$HOSTNAME_BS3" -UR  1.4
-OTHER 1.1 1.2 1.3 1.5 -AND "-$HO_RR -1 -100 1.4 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_14.$HOSTNAME_BS3 '2>&1'";

SLEEP 1;

CL RUN "$MOMBASA/NetBSD/bs_inactive"  "$HOSTNAME_BS4" -UR  1.5
-OTHER 1.1 1.2 1.3 1.4 -AND "-$HO_RR -1 -100 1.5 $HOSTNAME_MH '1>'
$LOGPATH/CL-LOG_bs_15.$HOSTNAME_BS4 '2>&1'";
```

Appendix D

MOMBASA Startup script (all.sh)

```
#!/bin/bash
echo " ---------------------------------------";
echo " Rerouting for Handover in co-Networks ";
echo "           A.F, T.A., L.W.             ";
echo "                 TKN                   ";
echo " ---------------------------------------";
export HOSTNAME_CP=ada-atm
export HOSTNAME_MH=orchard
export HOSTNAME_FH=milano-atm
export HOSTNAME_BS1=obelix-atm
export HOSTNAME_BS2=asterix-atm
export HOSTNAME_BS3=roma-atm
export HOSTNAME_BS4=turino-atm
if [ $# -ne 2 ]; then
    echo "$0: Wrong parameters"
    echo "Usage: all.sh hard|soft|pred sim|real"
    echo "   Rerouting scheme"
    echo "                      hard"
    echo "                      soft"
    echo "                      pred"
    echo "   Setup"
    echo "                      sim"
    echo "                      real"
    exit 1
fi
case $1 in
    hard)
        export HO_RR=hard
        ;;
    soft)
        export HO_RR=soft
        ;;
    pred)
        export HO_RR=pred
        ;;
    *)
        echo "Unknown rerouting scheme: $1"
        exit 1
        ;;
esac
case $2 in
    sim)
        SIM=1
        ;;
    real)
```

```
        SIM=0
        ;;
    *)
        echo "Unknown setup: $2"
        exit 1
        ;;
esac
if [ ada != $HOSTNAME ]
then
   echo "$0 started in $HOSTNAME"
   echo "$0 Retry in ada!"
   exit 1
fi
if [ $PWD != $MOMBASA ]
then
   echo "$0 started in $PWD"
   echo "$0 Retry in $MOMBASA!"
   exit 1
fi
if [ $LOGNAME != ibms ]
then
   echo "$0 started as $LOGNAME"
   echo "$0 Retry as user 'ibms'"
   exit 1
fi
#$MOMBASA/kill_all.sh
#$MOMBASA/free_ipc_rsrc
sleep 1
echo "$0 ... starting mobile"
rsh $HOSTNAME_MH -e "export
    DISPLAY=localhost:0.0 ;
    /usr/local/jdk1.2.2/bin/java
    -cp $MOMBASA/MOBILE HO_Mobile >
    $LOGPATH/mh.log 2>&1 &"
if [ $? ]
then
    echo "mh ... OK"
else
    echo "Could not start java. exit.."
    exit 1
fi
sleep 3
$ARLBINPATH/Linux/launcher $MOMBASA/TOPOS/topo.0
sleep 1
case $SIM in
    0)
        #Starts CM with GBNSC (no NodeSim)
        echo "$0: Starting CM with GBNSC (no NodeSim)"
        $ARLBINPATH/Linux/launcher $MOMBASA/TOPOS/topo.1b
        ;;
    1)
        #Starts NodeSim and CM with GBNSC
        echo "$0: Starting NodeSim and CM with GBNSC"
        $ARLBINPATH/Linux/launcher $MOMBASA/TOPOS/topo.1a
        ;;
esac
sleep 5
case $HO_RR in
    hard | soft)
        #Starts CMAP_SM and 3 Clients (fh, bs x 2)
        echo "$0: Starting CMAP_SM and 3 Clients (fh, bs x 2) topo.2a"
        $ARLBINPATH/Linux/launcher $MOMBASA/TOPOS/topo.2a
        ;;
    pred)
```

```
        #Starts CMAP_SM and 5 Clients (fh, bs x 2, bs_inactive x 2)
        echo "$0: Starting CMAP_SM and 5 Clients (fh, bs_1, bs_2, bs_inactive x 2) topo.2b"
        $ARLBINPATH/Linux/launcher $MOMBASA/TOPOS/topo.2b
        ;;
esac
unset HO_RR
echo " $0: ... OK ..."
exit 0
```