1. Top level (and COMMON)

Author: Juhana Räsänen et al

Top level is the part of the TOVE switch controller software that ties together all other modules and produces an executable signalling binary. Switch configuration and software setup functionality is implemented here, as well as the main() functions for both user side signalling daemon and switch_controller.

COMMON module includes some miscellaneous classes that are used (or potentially usable) in several other modules, but do not really fit into OVOPS++ or any other module. Example of these are the socket device classes for UDP and ATM sockets which are used to interface the signalling software to the real network.

1.1 1 Introduction

TOVE project has implemented a number of different protocol entities, but as they don't operate by themselves in vacuum, there must be a higher level control and orchestration of the switch controller software as a whole. Protocols must be instantiated and organised into stacks that reflect the configuration of the real, physical switch. There must be a part of program that knows this configuration in a way or another and is able to set up the software accordingly. This has been implemented in the top level of the TOVE software, near main() functions for the signalling binaries.

Also a trivial routing function is needed for the initial tests of TOVE software. Because everything happens within one switch (no Network Node Interface is implemented yet), routing needs no real functionality, only a simple lookup of output port by its E.164 address is enough.

1.2 2 Architecture

In current implementation the configuration is statically defined, ie. if the switch configuration changes (for example when a port is added or dropped), the configuration file must be changed and switch_controller software restarted. This is not practical for real-life systems, but for the purpose of TOVE project at this stage this was considered adequate.

There are two configurable entities of the TOVE software, VE and signalling. This separation has been made, because both are more or less independent of each other. Signalling sees only a virtual representation provided by VE that hides the physical switching fabrics from the rest of the software. VE maps the physical ports of possibly heterogeneous fabrics into general representation of ports, between which connections can be opened. Also the user of VE might be something else than signalling (in another application than TOVE switch controller), so the above separation was considered feasible.

Configuration data is stored in a configuration file (by default switch.conf in current directory), which is a textual representation of the switching fabric(s) and their ports. The syntax of this file is described in the example file that is included in TOVE Code Release 1/97, so it is not repeated here. The configuration file is organised into two sections, one of which contains the virtual port definitions and the other physical port definitions of VE.

After the switch_controller main() function has parsed the command line options, it opens the configuration file and passes it to the initialisation routine in tvSwitch (singleton) class. This routine in turn calls VE initialisation routine, which sets up the physical ports of the controlled switching fabric. After the VE initialisation is complete, the virtual ports visible for the signalling software are initialised. After completion of this initialisation the protocol instances are created and connected according to the configuration given in configuration file and the software enters the main loop of OVOPS++ and starts waiting for incoming connections.

1.3 3 Future development

A lot. This part of the software is under constant development, as the capabilities of the signalling software increase and more needs to be taken into account. An important development would be a better user interface than command-line options, but no actual plans have been made for this. This might become necessary, however, when management functionality is implemented into TOVE software so that the configuration can be changed dynamically at run-time.

It also might be good to move configuration functionality into a module of its own, so it can be developed more independently of the rest of the software. Also the routing function is a good candidate for a separate module.

1.4 4 Statistics

These parts of the software have evolved during the whole project, so these figures are only crude estimates.

Activity	Research	Design	Coding	Reviews	Total
Duration (h)	0	20	50	0	70

Table 1 Duration of activities

Lines Of Code (LOC)	Number of files	Number of classes
1517	18	6

Table 2 Metrics