Implementation of a Layer 2 Bridge in ns-3

Eduard Bonada, Darko Cavic, Dolors Sala
Universitat Pompeu Fabra
Pg. Circumval-lació 8
Barcelona, Spain
{eduard.bonada,darko.cavic,dolors.sala}@upf.edu

1. INTRODUCTION

The main objective of this poster is to show the key points of the ns-3 software architecture in order to modify its internal functionality (e.g. creation of a new module). It is intended for those researchers that want to use *ns-3* for their simulations and need more than a simple user level platform.

The work presented in here is the first stage of a larger project which objective is to build a layer-2 simulation platform (concretely IEEE 802.1). This first step includes the implementation of the basic functionalities of a layer-2 bridge. The simulation platform requirements are: (a) *extensibility* to improve the simulator with added functionalities; (b) *scalability* to simulate large network topologies; (c) *flexibility* to execute massive simulation runs in order to deal with a complete sensitivity analysis; and (d) *reliability* that will be achieved with an exhaustive control of single simulation events.

ns-3 has been selected as the base of the simulation. It is a discrete-event simulator based on a modular object-oriented architecture with very intuituve classes for networking researchers (e.g. a node works with the usual OSI layers).

2. BRIDGE MODEL

A bridge is a layer-2 device that aims at interconnecting LAN segments in order to build a bridged (and larger) network.

Fig.1 shows the model structure. The *Port* modules include physical and MAC level functionalities (framing and queueing system) and act as network interface connecting the channel with the *relay function* and the *STP* module. Note that they also include the demultiplexing to separate management and data frames. The *relay function* in the data path is in charge of the frame forwarding and address learning. When a frame arrives to an input port, this function decides to which output port it must be sent (solid line). Moreover, the *Filtering Database* is updated at every frame reception. Finally, the *STP* is a management module that builds the active topology by processing messages that each bridge receives from its neighbors. When one of this messages is received it is directly delivered to the STP module who will take the appropriate decisions about the active topology (dashed line). Note that the example frames in the figure only go from *port 1* to *port 2* but ports must be considered bidirectional interfaces.

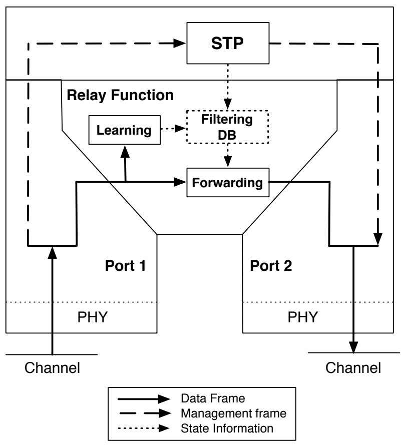

Figure 1: Node bridge architecture

3. IMPLEMENTATION

ns-3 uses a scripting language (c++ or python) that acts as the interface with the core of the simulator. It is concretely in these internals where this work is focused.

ns-3 has a modular structure based on object-oriented programming. One of the keys to understand the software structure is the *Node* class. It includes a list of applications, a customizable L3/L4 stack, and a list of network devices. Each one of these last elements is in turn associated with a communication medium and to the corresponding L3 object (lowest layer of the stack). In the L2 *bridge* model the application and stack have been removed and the Ports act as ns-3 Network Devices. Besides, additional modules that perform the bridge functionalities have been added (*relay function* and *STP*).

How modules are connected is another key aspect to understand the development of a new module. The default communication between internal modules in ns-3 is done using *callback functions*. These are object methods that are called from external instances. Their objective is to avoid direct linking to a concrete module. For instance, a L3 protocol could be associated to several L2 network devices. It is enough having a unique callback function in the L3 object that would be called from each L2 object.

SIMUTOOLS 2008, March 03-07, Marseille, France
Copyright © 2008 ICST 978-963-9799-20-2
DOI 10.4108/ICST.SIMUTOOLS2008.3082