Chapter 3
Linkage Between OTcl and C++ in NS2

NS2 is an object-oriented simulator written in OTcl and C++ languages. While OTcl acts as the frontend (i.e., user interface), C++ acts as the backend running the actual simulation (Fig. 2.1). From Fig. 3.1 class hierarchies of both languages can be either standalone or linked together using an OTcl/C++ interface called TclCL. The OTcl and C++ classes which are linked together are referred to as the interpreted hierarchy and the compiled hierarchy, respectively.

Object construction in NS2 proceeds as follows. A programmer creates an object from an OTcl class in the interpreted hierarchy. Then, NS2 (or more precisely TclCL) automatically creates a so-called shadow object from a C++ class in the compiled hierarchy. It is important to note that no shadow object would be created when a programmer creates an object from a class in both compiled and standalone OTcl hierarchies.

Written in C++, TclCL consists of the following six main classes. First, class TclClass maps class names in the compiled hierarchy to class names in the interpreted hierarchy. Second, class InstVar binds member variables in both hierarchies together. Third, class TclCommand allows the Tcl interpreter to execute non-OOP C++ statements. Fourth, class TclObject is the base class for all C++ simulation objects in the compiled hierarchy. Fifth, class Tcl provides methods to access the interpreted hierarchy from the compiled hierarchy. Finally, class EmbeddedTcl translates OTcl scripts into C++ codes. The details of the above classes are located in files “tclcl/tclcl.h,” “tclcl/Tcl.cc,” and “tclcl/tclAppInit.cc.”

This chapter focuses on using TclCL in the following meaningful ways:

• Section 3.1 presents the motivation of having two languages in NS2.
• Section 3.2 explains class binding which maps C++ class names to OTcl class names.

1Refer to [16] for the C++ programming language.
Fig. 3.1 Two language structure of NS2 [14]. Class hierarchies in both the languages may be standalone or linked together. OTcl and C++ class hierarchies which are linked together are called the interpreted hierarchy and the compiled hierarchy, respectively.

- Section 3.3 discusses how NS2 binds a pair of member variables of two bound classes so that a change in one variable will be automatically reflected in the other.
- Section 3.4 shows a method to execute C++ statements from the OTcl domain.
- Section 3.5 walks through the shadow object construction process.
- Section 3.6 discusses various functionalities to access the Tcl interpreter from the C++ domain: Tcl statement execution, result passing between both the domains, and the TclObject reference retrieval.
- Section 3.7 briefly outlines how the OTcl codes are translated into the C++ code.

3.1 The Two-Language Concept in NS2

3.1.1 The Natures of OTcl and C++ Programming Languages

Why two languages? Loosely speaking, NS2 uses OTcl to create and configure a network (i.e., user frontend), and C++ to run simulation (i.e., internal mechanism). All C++ codes need to be compiled and linked to create an executable file. Since the body of NS2 is fairly large, the compilation time is not negligible. A typical Macbook Pro computer requires few seconds (long enough to annoy most programmers) to compile and link the codes with a small change such as including a C++ statement “int i=0;” into the program. OTcl, on the other hand, is an interpreted programming language, not a compiled one. Any change in a OTcl file can be executed without compilation. Since OTcl does not convert the codes into machine language, each line needs more execution time.2

2Although OTcl is an interpreted programming language, NS2 translates most of its OTcl codes into C++ using class EmbeddedTcl (see Sect. 3.7) to speed up the simulation. As a result, most change in OTcl also requires compilation.