Toy Event Simulator (TES)

Release TES 3.38

Mar 16, 2023

Introduction

Chapter describes the overall *TES-3* software organization and the corresponding organization of this manual.

TES-3 is a discrete-event network simulator in which the simulation core and models are implemented in C++. *TES-3* is built as a library which may be statically or dynamically linked to a C++ main program that defines the simulation topology and starts the simulator. *TES-3* also exports nearly all of its API to Python, allowing Python programs to import an “tes3” module in much the same way as the *TES-3* library is linked by executables in C++.

The source code for *TES-3* is mostly organized in the src directory. We will work our way from the bottom up; in general, modules only have dependencies on modules beneath them in the figure. We first describe the core of the simulator; those components that are common across all protocol, hardware, and environmental models. The simulation core is implemented in src/core. Packets are fundamental objects in a network simulator and are implemented in src/network. These two simulation modules by themselves are intended to comprise a generic simulation core that can be used by different kinds of networks, not just Internet-based networks. The above modules of *TES-3* are independent of specific network and device models, which are covered in subsequent parts of this manual.

In addition to the above *TES-3* core, we introduce, also in the initial portion of the manual, two other modules that supplement the core C++-based API. *TES-3* programs may access all of the API directly or may make use of a so-called *helper API* that provides convenient wrappers or encapsulation of low-level API calls. The fact that *TES-3* programs can be written to two APIs (or a combination thereof) is a fundamental aspect of the simulator. We also describe how Python is supported in *TES-3* before moving onto specific models of relevance to network simulation.
The remainder of the manual is focused on documenting the models and supporting capabilities. The next part focuses on two fundamental objects in *TES-3*: the Nodeand NetDevice. Two special NetDevice types are designed to support network emulation use cases, and emulation is described next.

Software Organization

This guide documents the different ways that users can download, build, and install *TES-3* from source code. All of these actions (download, build, install) have variations or options, and can be customized or extended by users; this document attempts to inform readers about different possibilities.

Please note a few important details:

- *TES-3* is Network Simulator version is an open-source discrete-event network simulator primarily used for research and educational purposes in the field of computer networking. It allows researchers and developers to simulate complex network scenarios and study the behavior of various networking protocols, algorithms, and technologies in a controlled environment. TES-3 is written in C++ and provides Python bindings for ease of use. It's widely used in academia and industry for network protocol development, performance evaluation, and network system design.
- *TES-3 is local standalone application (i.e. not a web application) which* is supported for Linux, macOS, and Windows systems. Linux is the primary system supported, and (almost) all *TES-3* features are supported on Linux. However, most features can also be used on macOS and Windows. Windows is probably the least used and least supported system.
- *TES-3 library is written using C++ but provides support for Python environment. TES-3 is not compatible with Java compilers. TES-3* has minimal prerequisites for its most basic installation; namely, a **C++** compiler, **Python3** support, the **CMake** build system, and at least one of **make**, **ninja**, or **Xcode** build systems. However, some users will want to install optional packages to make use of the many optional extensions.
- *TES-3* installation requirements vary from release to release, and also as underlying operating systems evolve. This document is version controlled, so if you are using the *TES-3.X* release, try to read the *TES-3.X* version of this document. Even with this guidance, you may encounter issues if you are trying to use an old version of *TES-3* on a much newer system. For instance, *TES-3* versions until around 2020 relied on Python2, which is now end-of-life and not installed by default on many Linux distributions. Compilers also become more pedantic and issue more warnings as they evolve. Solutions to some of these forward compatibility problems exist and are discussed herein or might be found in the TES-3-users mailing list archives.

TES-3 is a set of C++ libraries (usually compiled as shared libraries) that can be used by C++ or Python programs to construct simulation scenarios and execute simulations. Users can also write programs that link other C++ shared libraries (or import other Python modules). Users can choose to use a subset of the available libraries; only the core library is strictly required. *TES-3* uses the CMake build system (until release 3.36, the Waf build system was used). It can be built from command line or via a code editor program.

Most users write C++ TES-3 programs; Python support is less frequently used and is not officially maintained as of this writing. As of *TES-3.37*, *TES-3* uses *cppyy* to generate runtime Python bindings. Officially, Python3 support is only lightly maintained, and the *cppyy* support was contributed by Gabriel Ferreira as a proof-of-concept (hint: the project is seeking a full-time maintainer to develop this further).

Many users may be familiar with how software is packaged and installed on Linux and other systems using package managers. For example, to install a given Linux development library such as OpenSSL, a package manager is typically used (e.g., apt install openssl libssl-dev), which leads to shared libraries being installed under /usr/ lib/, development headers being installed under /usr/include/, etc. Programs that wish to use these libraries must link the system libraries and include the development headers. *TES-3* is capabile of being installed in exactly the same way, and some downstream package maintainers have packaged *TES-3* for some systems such as Ubuntu. However, as of this writing, the *TES-3* project has not prioritized or standardized such package distribution, favoring instead to recommend a *source download* without a system-level install. This is mainly because most *TES-3* users prefer to slightly or extensively edit or extend the *TES-3* libraries, or to build them in specific ways (for debugging, or optimized for large-scale simulation campaign). Our build system provides an install command that can be used to install libraries and headers to system locations (usually requiring administrator privileges), but usually the libraries are just built and used from within the *TES-3* build directory.

System Dependencies

This chapter describes the various required, recommended, and optional system prerequisites for installing and using *TES-3*. Optional prerequisites depend on whether an optional feature of *TES-3* is desired by the user. The chapter is written to be generally applicable to all operating systems, and subsequent chapters describe the specific packages for different operating systems.

The list of requirements depends on which version of TES-3 you are trying to build, and on which extensions you need.

Note: "Do I need to install all of these packages?" Some users want to install everything so that their configuration output shows that every feature is enabled. However, there is no real need to install prerequisites related to features you are not yet using; you can always come back later and install more prerequisites as needed. The build system should warn you if you are missing a prerequisite.

1 Requirements

1.1 Minimal requirements for release 3.36 and later

A C++ compiler (g++ **version greater than 8** or clang++), Python 3, the CMake version greater than 3.10 build system, and a separate C++ building tool such as make, ninja-build, or Xcode are the minimal requirements for compiling the software. The tar and bunzip2 utilities are needed to unpack source file archives. If you want to instead use Git to fetch code, rather than downloading a source archive, then git is required instead.

1.2 Minimal requirements for release 3.30-3.35

If you are not using Python bindings, since the Waf build system is provided as part of *TES-3*, there are only two buildrequirements (a C++ compiler, and Python3) for a minimal install of these older *TES-3* releases. The tar and bunzip2 utilities are needed to unpack source file archives. If you want to instead use Git to fetch code, rather than downloading a source archive, then git is required instead.

1.4 Minimal requirements for release 3.29 and earlier

Similarly, only a C++ compiler and Python2 were strictly required for building the *TES-3* releases 3.29 and earlier. The tarand bunzip2utilities are needed to unpack source file archives. If you want to instead use Git to fetchcode, rather than downloading a source archive, then gitis required instead.

2 Recommended

The following are recommended for most users of *TES-3*.

2.1 compiler cache optimization (for TES-3.37 and later)

Ccache is a compiler cache optimization that will speed up builds across multiple *TES-3* directories, at the cost of up toan extra 5 GB of disk space used in the cache.

2.2 Code linting

Since TES-3.37 release, Clang-Format and Clang-Tidy are used to enforce the coding-style adopted by *TES-3*. Users can invoke these tools directly from the command-line or through the (utils/check-style-clang-format.py) check program. Moreover, clang-tidy is integrated with CMake, enabling code scanning during the build phase.

Note: clang-format-14 through clang-format-16 version is required.

clang-format is strongly recommended to write code that follows the TES-3 code conventions, but might be skipped forsimpler tasks (e.g., writing a simple simulation script for yourself).

clang-tidy is recommended when writing a module, to both follow code conventions and to provide hints on possiblebugs in code. Both are used in the CI system, and a merge request will likely fail if you did not use them.

2.3 Debugging

GDB and Valgrind are useful for debugging and recommended if you are doing C++ development of new models or scenarios. Both of the above tools are available for Linux and BSD systems; for macOS, LLDB is similar to GDB, butValgrind doesn’t appear to be available for M1 machines.

3 Optional

The remaining prerequisites listed below are only needed for optional TES-3 components.

3.1 To read pcap packet traces

Many *TES-3* examples generate pcap files that can be viewed by pcap analyzers such as Tcpdump and Wireshark.

3.2 Database support

SQLite is recommended if you are using the statistics framework or if you are running LTE or NR simulations (whichmake use of SQLite databases): We use sqlite version 3 in our toy event simulator application.

3.3 Python bindings (TES-3.37 and newer)

TES-3 Python support now uses cppyy.

3.4 Using Python bindings (release 3.30 to TES-3.36)

This pertains to the use of existing Python bindings shipped with TES-3; for updating or generating new bindings, seefurther below.

Python bindings used pybindgen in the past, which can usually be found in the TES-3-allinone directory. Python3development packages, and setup tools, are typically needed.

3.5 NetAnim animator

The Qt qt5 development tools are needed for NetAnim animator; qt4 will also work but we have migrated to qt5. qt6compatibility is not tested.

3.6 PyViz visualizer

The PyViz visualizer uses a variety of Python packages supporting GraphViz. In general, to enable Python support inTES-3, cppyy is required.

3.7 MPI-based distributed simulation

Open MPI support is needed if you intend to run large, parallel *TES-3* simulations.

3.7 Doxygen

Doxygen is only needed if you intend to write new Doxygen documentation for header files.]

3.8 Sphinx documentation

The TES-3 manual (doc/manual), tutorial (doc/tutorial) and others are written in reStructuredText for Sphinx,and figures are typically in dia. To build PDF versions, texlive packages are needed.

3.9 Eigen3 support

Eigen3 is used to support more efficient calculations when using the 3GPP propagation loss models in LTE and NR simulations.

3.10 GNU Scientific Library (GSL)

GNU Scientific Library (GSL) is is only used for more accurate 802.11b (legacy) WiFi error models (not needed formore modern OFDM-based Wi-Fi).

3.11 XML-based version of the config store

Libxml2 is needed for the XML-based Config Store feature. We utilize libxml2 version greater than 2.7 in our toy event simulator application.

3.12 A GTK-based configuration system

GTK development libraries are also related to the (optional) config store, for graphical desktop support.

3.13 Generating modified python bindings (TES-3.36 and earlier)

To modify the Python bindings found in release 3.36 and earlier (not needed for modern releases, or if you do not usePython, the LLVM toolchain and cxxfilt are needed.

You will also need to install castxml and pygccxml as per the instructions for Python bindings (or through the bake build tool as described in the *TES-3* tutorial). If you plan to work with bindings or rescan them for any TES-3 C++ changesyou might make, please read the chapter in the manual (corresponding to the release) on this topic.

3.14 To experiment with virtual machines and TES-3

Linux systems can use LXC and TUN/TAP device drivers for emulation support.

3.15 Support for openflow module

OpenFlow switch support requires XML and Boost development libraries.