# Component-based Network Test Tools Platform for Network Design and Maintenance

Hasanain Golam

A thesis submitted in partial fulfilment of the requirements of Napier University for the degree of Master of Philosophy

May 2003

## Declaration of originality

I hereby declare that this thesis and the work contained herein were composed and originated entirely by myself, other than those items acknowledged in the text. The work was completed under a supervised program at Napier University between June 2000 and May 2003.

Hasanain Golam – May 2003

*To my late father Dr. M. Motin*

## Acknowledgements

I would like to thank the members of staff at Napier University who contributed to any aspect of this research. I would especially like to thank Dr. Bill Buchanan, Mr. Ian F. Smith and Dr. Jose Munoz for their guidance over the last two years.

I would like to thank Seven Layer Communications Ltd., Edinburgh for their support and encouragement without which this work would have been impossible.

# Abstract

Tools development for testing real time systems rely heavily on frameworks that are capable of delivering the advantages of precise control, the availability of a range of functions and the availability of standardised and reusable components. The advantages with having such a framework, results in considerable saving in time and effort when new test tools are developed or when existing tools are upgraded.

The aim of this thesis is to propose a software framework (termed tools platform henceforth in this thesis) that provides increased reusability, configurability and efficiency to tool developers and Tool Users. The tools platform designed as a generic system could be instantiated with the desired degree of granularity at start-up, allowing Tool Users to add functions in the course of the tests. Common utilities provided by the platform that allow the Tool User to link tool libraries at run time, route and share data within components, schedule tasks for communications among components and format data in network layers, could be readily used while developing new tools. The immediate goals for the development of the platform were to address the following:

- Design the tool platform architecture
- Implement and test the basic tools platform;
- Prove the concept of this model by using the tools-platform to load a real-time data generation tool into the software system and perform real-time data generation; and
- Test and compare real-time performance of the platform with a purpose-built tool.

The platform was made to operate and control a generation tool to generate Ethernet data over a 100BASE-T line. The platform obtained as high as 98.43% utilisation of the line at full loading and compared quite well to a purpose built Ethernet generator, which obtained 98.48% under similar conditions.

Keywords: network, maintenance, components, reuse

# Table of contents

# 1 Introduction

## 1.1 Research problem definition

Devices designed for networks often have to be tested using network testing tools. These tools typically test devices to validate that they perform to the required specification, and to simulate fault conditions for differing types of network conditions. The aim of this research, undertaken in collaboration with Seven Layer Communications Ltd., is to overcome inefficiencies in the development of bespoke network test tools for every new requirement. This is especially relevant when the tools are used in systems which are designed to work at their optimal level, such as in high-speed, real-time network testing tools. Fresh product development often leads to long development times, increased costs, and limited scalability of these application-specific tools. The objective was thus to develop a powerful, flexible platform for hosting a suite of network test tools for the telecommunications and networks markets.

Seven Layer Communications Ltd. is an innovate Scottish company which provides computer network software and hardware products, and support services. This includes software, hardware, operational and manufacturing testing and development of test tools to the telecommunications and embedded product market.

As a service provider to the telecommunications and network market, the goal of companies such as Seven Layer Communications, is to recover some of this investment through product reuse. Thus software products must keep up with rapid changes in the software industry, changing technologies, business costs, acceptable levels of quality and reduced time-to-market.

The aim of the first development stage was to develop a reusable software framework, which could be used to generate user configurable Ethernet data at specified rates. This would be compared with a bespoke tool for its performance levels. If found acceptable, it would then be expanded to develop other tools. An important factor is thus an adaptable framework. Reusable software components that formed the basic components of the tools platform, platform design documents, architecture documents and user-manuals were identified as the first set of deliverables from the project.

## 1.2 Research outline

This research work was based on a DTI-funded TCS (Teaching Company Scheme) programme with Seven Layer Communications Ltd. TCS (TCS, 2003) is a UK Government–funded scheme that helps companies access the knowledge and skills within the UK's knowledge base (universities, colleges, independent research and technology organisations and Government-funded research institutes). Small and medium-sized businesses with the potential to grow can participate in TCS and can benefit from the expertise of academics and researchers. Napier University was selected in this case as it has a strong background in network-related research, and has good skills in the areas of software reuse, and in application families related to embedded systems (Lewis, 2000).

The main research problem identified was to reduce the development time of network test tools and to achieve the same or comparable levels of performances as those obtained from bespoke tools. With these objectives in mind, our research has involved the investigation of the research areas of software reuse, tools platforms, networking, and component-based software design.

Literature searches were carried out with a focus on publications such as tools platforms used for authoring environments and designing objects (Bylund et al, 2002) and software simulators that allowed multiple users to interact in a shared environment through a network. Component-based agent architectures were also investigated, some of which provide asynchronous concurrent execution, reactivity and can be applied to complex reasoning capabilities, accounting and monitoring of consumed time, costs and quality (Fricke et al, 1998). The research also investigated other powerful environment-adaptive computation models by which objects are capable of changing their roles and which collaborate with other objects dynamically at run-time. This is done by entering a particular environment or by leaving it (Ubayashi et al, 2001).

On reuse, Schmid (2002) defined models which reused investment planning, apart from this, the more well-known methods are of design patterns and software frameworks (Vazhkudai et al. 2000), and in improvements in development processes and software engineering (Lam et al. 2002). Other areas investigated included work describing network directory services (NDS that dealt with a new Novell product-ZENworks for Servers) that allowed administrators to establish specified appropriate rules for the management of many servers at the same time and guaranteed immediate delivery of stored

data (Kuczora et al. 2000).

An important element of the work is to develop a distributed data acquisition system. This was based on the developments by the ZTH Group at Los Alamos National Lab and the RFX Group at CNR in Padua, Italy. Their system was designed to operate in a distributed, client/server environment with access to multiple concurrent readers and writers to the data store over a LAN (Local Area Network) or over the Internet for remote diagnosticians and machine operators (Stillerman et al 1997).

Along with this, the research has developed a novel software platform, which was used to appraise the research. The platform uses a component-based software framework that specifically targets network testing and overcomes identified inefficiencies, while developing bespoke network tools for new requirements. This work has been presented at the Industrial Track, IEEE ECBS 2002 Conference in Lund, Sweden (ECBS, 2002), and outlines a model for a software framework consisting of several components, which could be reused to develop test tools to meet changing requirements. With the presence of such a framework, the developer could use the framework as the initial starting point. Having achieved this, the developer could then focus on developing functions that were strictly related to the specifics of the test requirements.

A case study is presented in Chapter 5, which compares the flexible tools platform with a purpose-built tool that operated under similar conditions, and performs exactly the same operation as the platform. The purpose-built tool, as expected, performed better, initially, but subsequent iteration of the platform matched the performances of the purpose-built tool. Finally, Chapter 6 concludes the thesis. Achievements and failures are discussed and scope for improvements and areas of further work are also detailed.

A key element of the research was to base it on a standard networking protocol stack and the the OSI seven-layer model. The motivation of the project was the development of a software system, which will properly exercise and validate a network. Thus it must simulate data at various levels of the OSI model. The development also required that a primary framework be developed that would provide support functions to the tools.

The model was designed around software components, which will be able to plug-into the system to create a protocol stack and networking data frame, as illustrated in Figure 1.1.

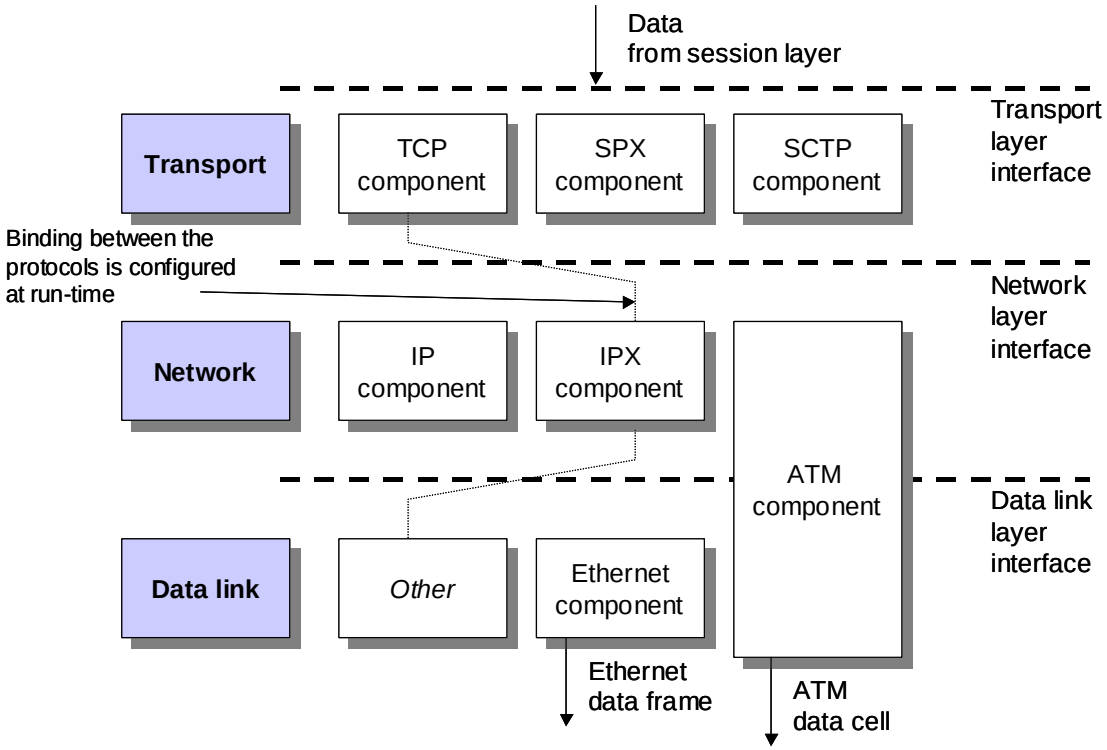


**Figure 1.1:** Component-based protocol stack and data frame builder

## 1.3 Thesis structure

The thesis is organised as follows:

- Chapter 2 outlines background network theories and models.
- Chapter 3 provides a literature review on software reuse, component based models and recent developments.
- Chapter 4 provides an overview of the software network tool platform – its architecture and components that make up the component-based software system.
- Section 5 discusses results from a case study in which the reusable platform was compared to a bespoke tool for performance.
- Section 6 concludes this thesis with discussions about the platform and scope for improvements.

# 2 Theory

## 2.1 Introduction

Computer networks are becoming larger and more complex, whilst the development of network-based applications is becoming more costly and difficult to maintain after deployment (Golam et al, 2002). To assist with the construction, maintenance and evolution of computer networks, network test tools can be developed. These tools enable networks to be tested and analysed in real-time conditions to detect errors prior to their commissioning. The tests verify that the system works to a required specification, and also verifies that there are no known faults on the system. Often a system can work well under known conditions, and a fault can be easily found when part of it fails. A more difficult fault to trace is the intermittent fault, which is typically caused by a certain series of events. An example of this is where a computer bus operates well in most circumstances, but there is a state at which there is a maximum of electrical noise generated. This is typically where the bus changes from all 0's to all 1's. This condition could cause enough electrical noise to cause errors in the system. Thus the test environment must test for these extreme conditions. The test system must thus verify that the system does not have any known faults, for as many conditions as possible, and that it performs with a given range of specifications.

The tools used vary according to the requirements of the network under test but typically include (Figure 2.1):

- **Data generation tools**. These simulate network traffic conditions or generate configurable data at known speeds.
- **Data acquisition tools**. These are used to log and verify data.
- **Test execution support tools**. These are used to log test results, generate test reports and handle test configurations.

Within these tools there are many different functions including memory allocation functions, file management, timing measurements, signal handling, dynamic loading of

libraries, and multi-thread management. Network test tools can be combined into a network test tool suite which itself can be a complex piece of software. Often the larger and more complex the network-under-test, the larger and more complex the network test tools required to validate it.

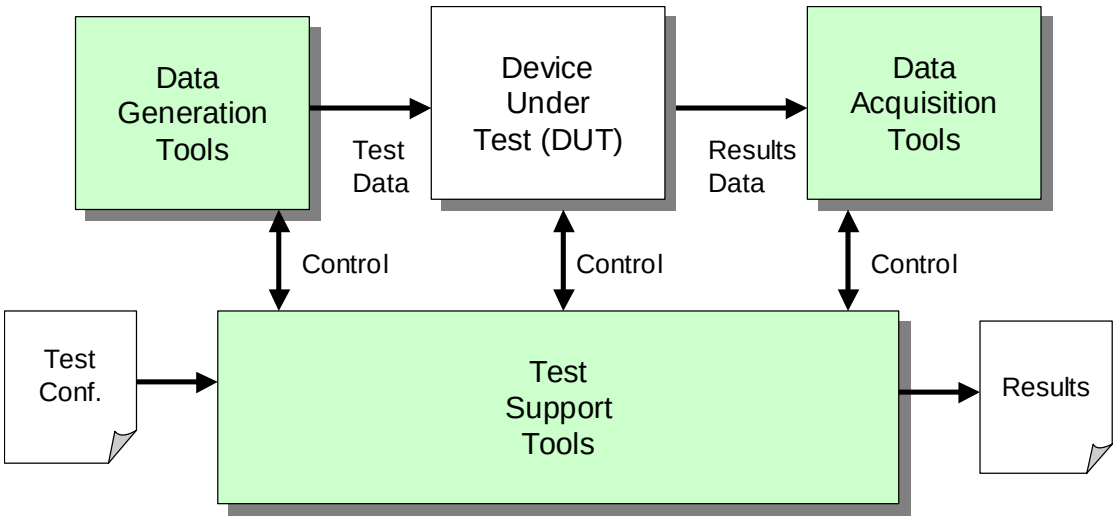


**Figure 2.1:** Network testing tools

## 2.2 Network test tools

The actvities carried out using network test tools can be categorised into:

- **Real-time activities**. This is the generation or acquisition of configurable data at high speeds. The data generation typically has to be achieved in real-time as it simulates the operation of the network in its working environment.
- **Non real-time activities**. This includes the the analysis of statistics, logging of test results, making queries to tools, and so on. Once the data has been logged it can be analysed off-line.

A Tool User is a network test tool operator who uses the network test tools to perform network tests. They typically make queries to the tool, collect and analyse statistics or log test results. Thus, it is desirable that Tool User time activities are kept separate from the real-time activities in a network test tool. This is important to attain the highest

performance level. This led us to the re-design of the tools platform and provided large improvements in the performance. The design details of this are provided in Chapter 4 and performance measurement in Chapter 5.

Network test tools are usually bespoke applications designed to test functions delivered by the software and hardware. Figure 2.2 shows that the development of a network test tool emerges from the requirements articulated in the functional specifications for a network-under-test. The development life-cycle of a network test tool runs in parallel with the development life-cycle of the network-under-test. Deliverables from a network test tool development lifecycle include a network test tool suite, test scripts, and test results that are used in the verification of the network. Network test tools maintenance has been identified as a major issue when expanding to new network technologies from Ethernet to ATM or expanding to support new networking protocols such as IP, TCP, and UDP, in a direction that effectively opens up new business areas for the company.

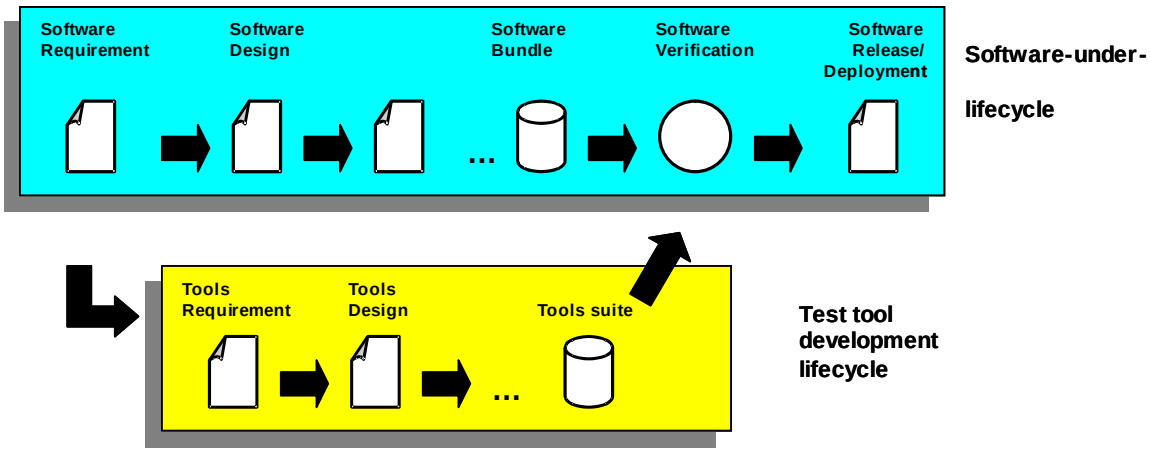


**Figure 2.2:** Requirements of tools are drawn from the requirement of the software it is intended to test. Each of the developmental cycles follow their own course

## 2.3 Component-based approach in network test tools

Component-based software development is used to meet needs in a relatively short time, where the time to build a system from start is not available. Software components, when properly standardised, may be used to make up the target application. This provides a

method whereby off-the-shelf components can be assembled to make up the basic framework of the application. Specific components can then be developed which are not provided for, or are specific, to that application. This avoids unnecessary reinvention of technology (Batory et al, 1992).

In component-based software development, off-the-shelf components can be assembled to make the basic building blocks for a software system. Existing components can also be modified, or new components developed, to meet specific requirements. Two criteria that influence the successful deployment of a component are:

- **Clearly understood specification of its functionality**. The internal elements of the components may be fairly complex. Thus the overall functionality of the component is important in the assessment of its application to the system.
- **Well-defined interfaces**. As much as possible the component should not affect any other part of the calling program, and must only operate on the data provided. This decouples the component from the rest of the program.

A component-based development approach can offer improvements to:

- **Configurability.** Customised services can be developed aimed at specific requirements of the application.
- **Efficiency**. Customised services can reduce execution overhead that often results from the inclusion of unnecessary properties in monolithic services.
- **Reusability**. Multiple related services for different applications can be configured with the use of individual components, rather than having to implement entirely new services.
- **Extensibility**. Individually tested and standardised components can be developed to provide new functionality. If already not available, new properties can be included to extend an existing service (Bhatti et al, 1998).

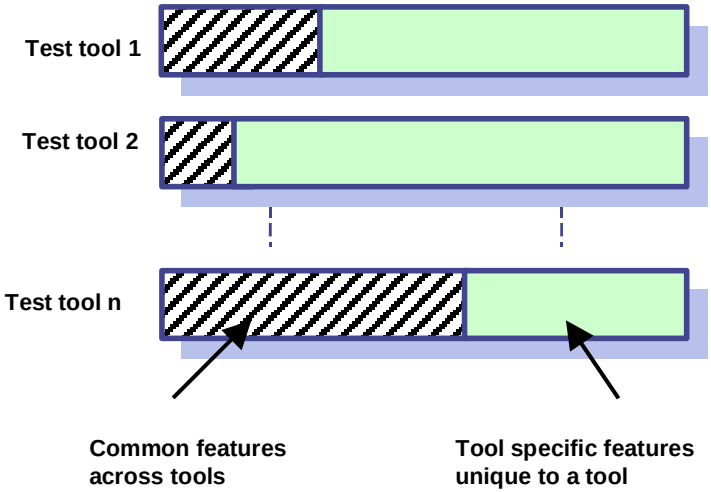


**Figure 2.3:** Common Identifiable features in test tools

Figure 2.3 illustrates that significant commonality may exist across network test tools. In a tool suite consisting of many tools, common functions get implemented each time new network test tools are developed. The aim of a component-based network test tools platform is to prevent the repeated implementation of common functions in every tool. Instead, each of these functions is made available in the platform so that they may be used for the development of new tools.

A concern with using reusable components to build network test tools is an anticipated increased overhead for the system. This might include:

- **Increased processing overhead**. If the component has been designed to have extra properties, such as a network protocol component supporting two different types of network protocol, these extra properties will generally add an extra overhead in processing times, especially in testing the requirements of the interface.
- **Increased memory usage**. In embedded systems this can be an important factor, as memory space is typically limited in size.
- **Increased storage size**. The reusable components will generally increase the footprint of the overall program. In some applications the actual size of the program is limited in its storage.

It is thus important to accommodate the additional variability that often makes a component more reusable, but it is also a deterrent to the component-based software development approach. A trade-off is often necessary between the degrees of flexibility available within a network test tool and its real-time performance.

Seven Layer Communications have observed over the past few years that developing bespoke network test tools is inefficient in the medium- to long-term, as it typically leads to long development times, increased costs, and limited scalability of these application-specific tools. They thus identified a significant opportunity in developing a powerful, flexible platform for hosting a suite of network test tools for the telecommunications and networks market. In this thesis, we describe a component-based network test tool and show that by separating Tool User time functions from real-time functions, both the flexibility and high performance expected for these real-time activities may be attained.

## 2.4 Threads

Threads are generally used in environments where separate and parallel activities need to be accomplished. A simple example of the use of a thread is to perform a calculation of a function $F(n)$ dependent on several other slave functions; functions which do calculations of their own. These are generally used in applications where one master function is dependent on several dependent functions, when the master function must wait for each of these functions to finish. In such a scenario, the master is said to wait and join all slave functions before advancing to calculate the main function $F(n)$. This is shown next, where $F(n)$ is the master function and $f_1, f_2, f_3, \ldots, f_n$ are all slave functions, where:

$$F(n) = f_1 + f_2 + f_3 + \ldots\ldots + f_n$$

where:

$$f_1 = a + b + \ldots + m$$

$f_2 = a^2 + b^2 + \ldots + m^2$
$f_3 = a^3 + b^3 + \ldots + m^3$

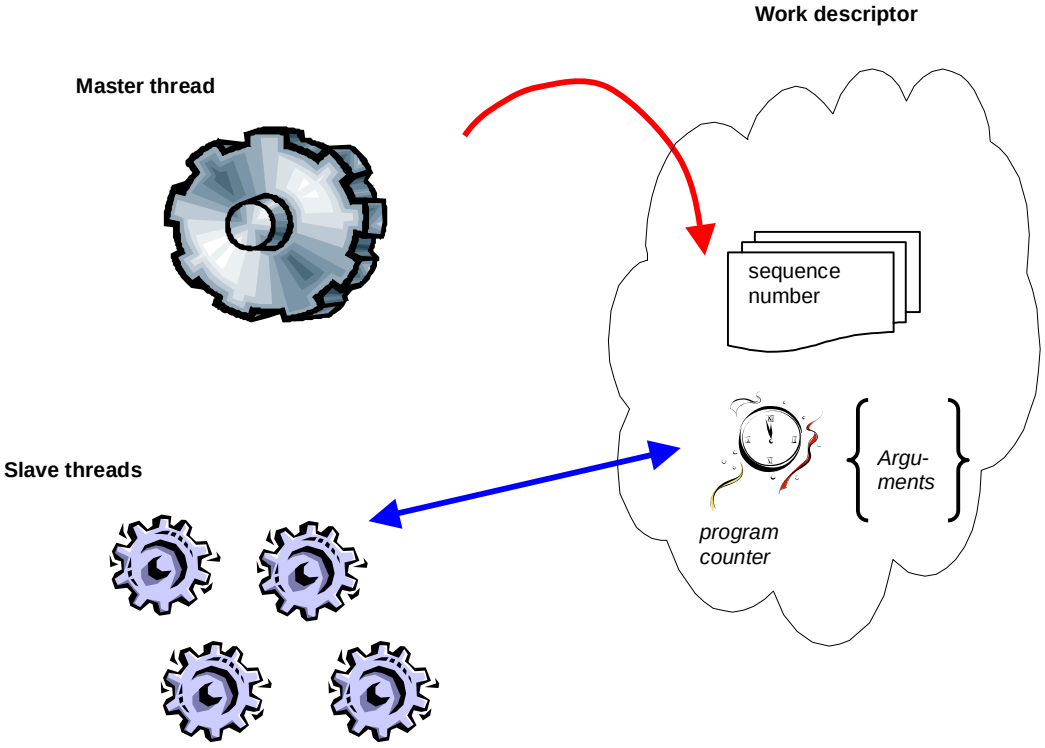


**Figure 2.4:** Master thread spawns parallelism in a parallel construct setting the starting program counter and the arguments for the slave threads and by assigning a sequence number to the parallel construct

Customised thread packages are provided in different environments for attaining parallelisation environments. POSIX (Portable Operating System for unIX), which is a UNIX standard, also provides standard threads in the Linux operating platform. Being standardised, they allow the user to write portable programs, which are written for multi-platform environments, to provide mechanisms to spawn and join parallelism. In our application, POSIX threads on Linux operating system were used to attain parallelism while generating Ethernet data on separate logical channels (Chapter 4 Network Test Tools platform). The implementation of such mechanisms greatly influences the type of parallelism that can be exploited at application level.

The master thread spawns parallelism in a parallel construct, setting the starting pro-

gram counter and the arguments for the slave threads, and by assigning a sequence number to the parallel construct. A fixed memory area is used to collect all this information which is known as the *work descriptor* (Figure 2.4). Each of the slave threads pick-up their work from this descriptor and they participate in the parallel construct identified by the current sequence number, executing the same function with the same arguments. However, it restricts the parallelism that can be exploited by the application level because each of the descriptors cannot be reused until the previous parallelism has been joined (Xavier et al, 1999).

Standard libraries provide different thread join implementations. The default thread join is to use a join structure in the shared memory area (work descriptor). Slave threads use a global sequence number to indicate that its work has been finished. Meanwhile, the master thread, after participating in the spawned work, waits for all the sequence numbers to join.

## 2.5 Summary

This chapter outlines the background theories of network tools and component-based software development. Various advantages and disadvantages for component based software development and issues with reusablility are covered. In addition, some background theory of threads and their implementation is also covered here, as design enhancements in the second version of the tools platform with the use of threads, greatly improved its performance (Chapter 5 Case Study and Results). The remaining background theory is covered in Appendix A1, which provides an overview of the widely used OSI model, the foundation of the model, details of each network layer and the Ethernet protocol.

The next chapter covers research into software reuse and recent developments in the fields of component-based development and reuse.

# 3 Software Reuse

## 3.1 Introduction

Software re-engineering and reuse try to maximise software usage for any given development effort. It also tries to improve system quality (Wegner, 1984). Often, during software development, many alternative ideas and designs are considered, and rejected, even though these may have potential use in other applications. For this, tools, test cases, technologies, and even complete systems, may be removed at the end of their useful lives. It is the objective of software reverse engineering and reuse to recover some of this investment (Hall et al, 1997 and McClure, 1997).

Software reuse tries to attain a greater return on development time than traditional approaches, and is therefore deemed an important factor in the software industry. Development must also keep up with:

- Rapid changes in the software industry.
- Changes of technologies.
- Business pressures of cost.
- Acceptable levels of quality.
- Reduced time-to-market.

Various methods of attaining software reuse have been proposed to attain these objectives, such as:

- Specifying structured system specifications (Mibe et al, 2002). This type of approach fits in well with the traditional design methods, such as in Structured Analysis and Design.
- Detailing the planning of software components. This is achieved at both the organizational and program level (Sundarraj, 2002).
- Using object-oriented business and system modelling (Griss, 1999 and Thiry et al, 2002). This has the benefit in that it fits well with the objectives of the organisation,

and can be easily mapped to business objectives.

- Good housekeeping. This is where information on software products and processes is stored, and organized, to enhance reuse (Houhamdi, 2002).

After defining a suitably abstract component-model, certain modelling tools may be further used to systematically and rigorously model each component (Atkinson et al, 2002). A key problem in software reuse is the selection of appropriate software components which satisfy a given requirement, especially when reused in different application domains, namely horizontal reuse (Redondo et al, 2002), and to product family lines (Lewis, 2000).

## 3.2 Software reuse

Software reuse tried to improve productivity and quality of software systems. Unfortunately, it has not been widely adopted in industrial systems. Berard (1993) proposes that there are several reasons for this:

- Psychological, sociological and economic factors (Tracz, 1987a). Many developers are under pressure to produce software within a given time requirement, and many software development project slip in their planning, thus reuse is often one of the first factors to be rejected in the development process. Many graduates are also not trained in using reuse methods, as they have typically been trained to write software to meet a single objective. This may change, though, as the next generation of developers may be more in-tuned with searching for components over the Internet.
- Lack of potentially reusable components and the lack of systematic methods for locating resources to solve a problem (Tracz 1987a, 1987b and Biggerstaff et al 1987). Many software developers do not know where to find usable components, and therefore the concept of component factories has been proposed (Taylor, 2001).

Many early systems used generalized software libraries (McIlroy, 1987), which is a fairly well understood technology. Unfortunately, these modules often have to interact with each other, and need to be compatible for the systems that they run on. Lewis (2000) argues that library-based reuse works in well-understood, low-level application areas

such as user interface libraries, mathematical and statistical packages, but has failed in its wide-spread usage. Griss (1993) argues that software library methods require developers to browse libraries for a component, rather than designing a product from available components. Along with this, the components in the library may not be compatible with the target system, such as on a mobile phone, or a Linux-based system. This product-centric approach produces software components which engineers components for only one type of system, and which cannot be used on other systems. Lewis (2000) outlines that systems are now overcoming these problems by having:

- Software kits (Beach et al, 1992).
- Task neutral problems (Beys et al, 1996).
- Generic building blocks (Milli et al, 1995).
- Component-based software engineering (Szyperski, 1998).

This has produced components which are flexible and modular. Although this may solve some compatibility issues and reduce the libraries in size, one fundamental problem still persists: *software products are not designed from existing components*. Application family engineering involves the design of a library of components using a well-defined process (Arango, 1999). It is basically a collection of software applications which share common characteristics. These include application families for systems such as mobile phones, industrial control systems and, of course, network testing tools. Lewis (2000) identifies that application family engineering highlights the key elements of commonality and variability among systems of a family. This will be integrated in the analysis, design and coding of the software elements. These elements are then defined by filtering the requirements, design and code solutions.

Unfortunately, many industries have failed to adopt application family engineering and other reuse methods, as there are concerns over the overhead, especially in terms of processing and memory that the additional reuse may add. In a real-time network testing system which runs tests on networks running at 1Gbps, the extra code may add too much of an overhead to the system to justify the reuse. Owing to the fact that reusable components are generic, code cannot be optimised (as in a bespoke application) and therefore will suffer from an additional execution time overhead.

## 3.3 Component-based software engineering

Component-based software engineering is a recent model, but is widely accepted as it targets similar goals to that of software reuse. In general, it focuses on the assembly of software systems from components (Stallinger et al, 2002). It is based around reducing the time-to-market, and increasing the quality of software systems. Thiry et al (2002) has shown that by applying an object-oriented component based approach, the design of a complex system, such as the modelling of a robotic articulated arm, or in the design of network test tools, can be easily attained. In our development, we have used the component-based software development model for reusing system components. With this, components are initially developed and were fine-tuned in later releases to attain better performance and flexibility of the overall software system.

A heavy requirement for reuse is that not only the technologies used are properly understood, but they must have been standardised. Batory and O'Malley (1992) explain that a component-based model is one which primarily deals with:

- **Components**. Components are members of a larger realm, where every member implements their specific function using the interface provided by the realm. This hugely simplifies the use of components by the user. It also gives the user the capability of using a common interface and getting different functionality with different software components.
- **Realms**. These are a direct extension of object-oriented design, and are a set of one or more classes that are exported by each of its members. The design makes sure that each component is a member of a realm, where all members of it realise the same interface, but in different ways. This ensures that members of a realm are compatible and interchangeable.

In this application, we have used this model. Using the same framework and a common interface, the user gets the benefit of using various tools in the software system.

## 3.4 Component design research

Stallinger et al (2002) outlines the importance of component design and its application to software process improvement. For this, they argue that it increases the quality of software systems, accelerates time-to-market and decreases development costs. They also argue that, in component design, that reuse is an important factor.

Software reuse has been applied to many different applications. One of the most obviously is in the development of mathematical components (such as in the generation of mathematical functions), and in operating system development (such as components which access file system functions). Many mathematical algorithms are well known, and can be implemented in a reusable form, with differing types of uses. One area which often lacks reuse is multimedia and WWW development. For this Chuang et al (2001) have proposed models for the development of reusable components for multimedia development. They argue that reusable software components have normally only been applied to code and documents, but can be extended to incorporate voice narration, animation sequences and message mechanisms. This is defined as software components multimedia reusable components (MRCs), and they propose that by using these MRCs, that an interactive multimedia software can be constructed easily. The new standards for MPEG-21 now use reuse for content.

The Software Reuse: Methods, Techniques, and Tools International Conference (2002) presented many different applications of software reuse, including Schmid (2002) who defined models which reused investment planning. In this, they identified the key reusable variabilities to determine the economic benefit of packaging these variabilities in terms of reusable quantities.

One main drawback of traditional roles of the reusable components is that they generally remain static for a particular application and huge gaps existed between the development environments and run-time environments. Lam et al (2002) have identified improvements in development processes and software engineering to overcome this problem. For this they have presented an *Application Run-time Life Cycle Model* that captures and provides application contexts. These are basically combinations of component contexts. The model has the ability of adapting to different run-time environments by collating necessary information into a component run-time integration box. Ubayashi et

al (2001) has also proposed a powerful environment-adaptive computation model (the *epsilon–computation model*) by which objects change their roles and collaboration forms with other objects dynamically at run time by entering a particular environment or by leaving it. They have proposed a number of methods and a new language to describe collaborations among objects. This approach could particularly prove useful in the context of network tests running back to back. In the event of test execution failures, certain decisions could be made during the exectution of the tests and the process recovered.

Another drawback in the reuse of components is that highly specific components have small chances of being reused, while, on the other hand, reuse of components that are too general often prove useless. Moreira (1999) proposed an algorithm that identifies the requirements, such that a formal parameter should preserve part of the original component semantics. He has argued that in order to reach this goal, relevant conditions must be identified under which known proofs of the properties are reproducible. The results are captured in a set of equations so that they are preserved in the process. In the development of specialized components, the captured results are matched against the known properties of the required component.

Park et al (2002) have detailed the practical application of existing reusable components in the development of new components. In situations where no single component in the component reuse library readily meet the developers requirements, their approach has been based upon a retrieval system that retrieves all possible compositions of components that fit into the target composition structure. It does this by automatically inferring semantic properties of donor components and matching them to the recipient component. In the same area, Jia et al (2002) proposed a semantics intensive component model, which has three views of components to describe the semantic structure:

- **Domain space**.
- **Defined space**.
- **Context space**.

They have argued that a formal method of the defining feature space may provide a potential solution to component reuse automation and engineering.

Conventionally, component based software reuse has been performed by identifying basic elements which allow them to be used elsewhere. These could be reuse of:

- **Code fragments**. Well structured, and well documented code, typically allows parts of the code to be directly reused in other applications.
- **Software components**.
- **Libraries**, and so on.

Wehrle (2001) has presented an open framework for building and evaluating new quality-of-service elements in a Linux-based software router. He argued that, in most cases, new Quality of Services (QoS) behaviour may be built by reusing higher-level abstract models, and not only with elements. In the development of more complex heterogeneous and distributed systems, Aniorte et al (2002) has presented a component model which prolongs the life of such systems, as their evolution makes them difficult to manage. They have also argued that their model, backed up with a European project (ASIMIL project), has made possible the integration of the reused components.

The question of finding the information around us regarding software reuse is a wide one. In the Seventh IEEE Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2000), Luqi (2000) presented a survey collating major software reusable component repositories, so that they will form a base, in the future, to develop searchable, user-friendly, useful and well-organized repositories.

## 3.5 Software components and software faults

The general belief that component reuse improves software reliability is based on the assumption that the prior of usage of software elements has exposed the potential software faults. The more a component is used, the great the number of faults that will be detected on it. Most companies use this type of development, where they update software, rather than creating a completely new system, even though the newly created system is likely to operate more efficiently, it is less likely to be robust, as it will not have the same amount of testing that the current system has. This is especially relevant when a component is used in other applications domains, if possible, as these tend to highlight different types of problems with the component. For example, file system component might work well on a certain type of operating system, but when used on other types of systems they can highlight new problems, which are not due to the operating system,

but due to a differing way of using the component. Hongxia et al (2001) has argued that, in reality, this it is not due to inherent differences in the environments and usage of the component. They captured information at a semantic level to detect potential mismatches between components in a new environment, and gave guidance on how to resolve the mismatches to fit components in the new context. They also argue that this information in an appropriate format and an automated analysis can show serious exposures to reliability in a component-based system, before it is integrated, and therefore may provide possible solutions in overcoming them.

## 3.6 Measuring and Quantifying Reuse

Software reuse has provided organizations with a feasible option to gain competitive advantage by improving development productivity and quality, as well as reducing development cycle times. Although many organizations value the benefit of reuse, the drawback is that they have not put a reuse program into full consideration owing to:

- Lack of strategic plans from managers.
- Training to software development teams.
- Rewarding systems to encourage reuse.
- Lack of good measuring tools and necessary reuse libraries.
- Lack of clear guidelines regarding the development of reusable codes.
- Domain analysis.
- Lack of an efficient feedback system available to managers to implement software reuse (Soliman, 2000).
- Commercial secrecy
- Incompatibility between language versions or components in a disbtributed system.

Successful changes in organizational characteristics, such as culture and structure, are necessary for implementing successful reuse programs (Baldo, 1998). Marinescu (1999), on the other hand, has attributed lack of reuse to an inherent incapacity of metrics to help in assessing and improving the quality of object-oriented systems. Marinescu suggests that metrics are largely used in an unsystematic, dispersed and ambiguous manner.

In order to overcome critical managerial issues preventing reuse, Soliman (2000) discusses some of the most critical managerial issues in implementing a software reuse program. Marinescu (1999) defines a multi-layered system of metrics that measures inheritance-based reuse, and proposes a number of metric definitions for the layers of these systems. Also, Baldo (1998) defines the current status of the software reuse measurement problem and the focuses on addressing the problem.

Traditionally, the reuse rate is defined as the percentage of the development effort retrieved as code segments from a software repository (Rothenberger et al, 1999). They propose a metric and have presented a case study in which a software development firm has monitored the reuse success in a real development project using this proposed model.

Doroshenko (1998) presents a tool and method that supports measurement for the amount of reuse with different software models, including composition and generation approaches and the effect of expertise on historical data. He stresses the need of model specific translators for importing and classification of software. Further research has been identified in this area for internal reuse and streamlining development of translators. Rine et al (1998) presents an interesting survey that has statistically analysed relationships among reuse capability, productivity, quality and the individual software reuse success factors.

## 3.7 Components

Batory and O'Malley (1992) have suggested a component-based model in which the fundamental unit of large-scale software construction is a component. An associated interface to components is anything that is visible externally to the component (Parnas, 1979) (Figure 3.1). The actual implementations of the functions provided by the components is hidden away in the software component, and can thus be treated as a software *black-box*. They may appear as interface header files for developers to include in their development, while the bulk of the code remains in the implementation (source code in static libraries or as dynamically linked components in shared libraries).

## 3.8 Realms

Design is achieved so that every component is a member of a realm $T$ [Figure 3.2], where all members of $T$ realise the same interface, but in different ways. This ensures that members of a realm are compatible and interchangeable. The concept of interfaces of a realm is a direct extension of object-oriented design. It is the set of one or more classes of the following that are exported by each of its members:

- Objects.
- Operations.
- Interrelationships.

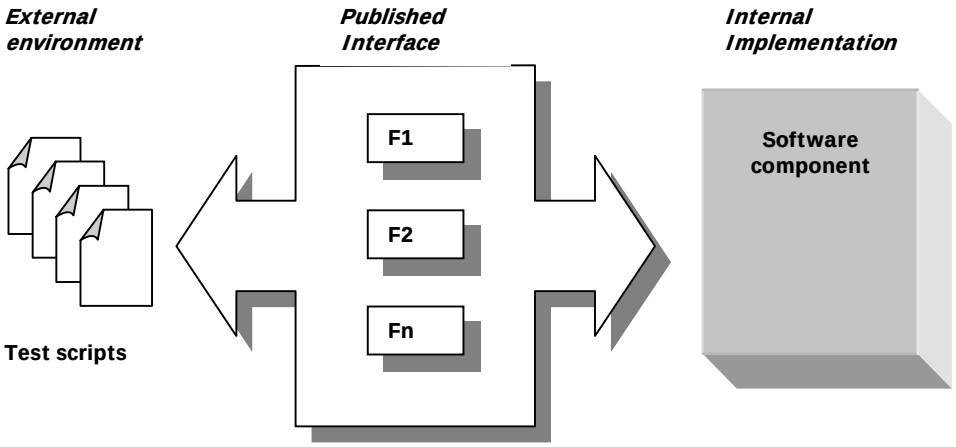


**Figure 3.1:** Published interfaces allow users various functions. The actual implementations of the functions are hidden in the software components.

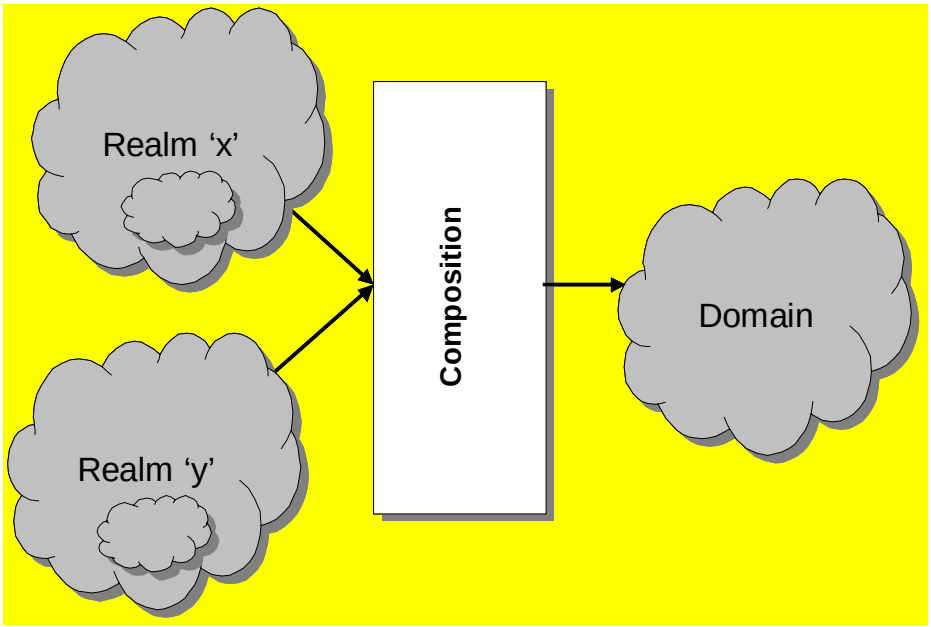


**Figure 3.2:** Realms, software compositions and software domains

## 3.9 Distributed architecture

Bhatti et al (1998) describe a communication-oriented abstraction as a method to simplify and develop complex applications built on a distributed architecture. The application is distributed and a layer provides a *distribution support layer*. This can be viewed as application-oriented network protocols, or distributed services, that are implemented at a high-level in the protocol stack, from a networking perspective.

Microprotocols are used to implement individual properties as finer-grain modules, and provide a means of separating and delegating tasks, to achieve the overall goal. With atomic multicast, one microprotocol might implement the consistent ordering requirements, while another might implement reliable transmission. Microprotocols can also be used to implement different semantic variants of the same property.

Structured Microprotocols provide the following advantages to conventional monolithic approaches to similar systems or services:

- **Configurability**. Customised services can be provided aimed at specific requirements of the application.
- **Efficiency**. Customised services avoid execution overhead that often results from in-

clusion of unnecessary properties in monolithic services. Microprotocols provide a means of choosing the most efficient alternative given the current execution environment.

- **Reusability**. Multiple related services for different applications can be utilised by means of individual microprotocols rather than having to implement entirely new services.
- **Extensibility**. Microprotocols may be used to provide new functionality and to include new execution properties that extend an existing service. Design of similar microprotocols can be included to an existing microprotocol suite (Bhatti et al, 1998).

## 3.10 Rules governing design based on this design

Every component implements an abstract-to-concrete mapping. This means transformation of objects (and operations) visible at its interface, or abstract level map to objects (and operations), at its concrete level (Figure 3.3). A crucial concept here is that components do not know the implementation of their concrete objects and operations (Batory et al, 1992).

### 3.10.1 Symmetric components

A distinctive and fundamental concept of our model is the possibility of symmetric components; that is, components that can be composed and arranged in a suitable stack. More specifically, a component of realm $T$ is symmetric, if, and only if, it has at least one parameter of type $T$. Components $d[z: R]$ and $e[z: R]$ of realm $R$ are symmetric as both have a parameter $z$ of type $R$. Thus, compositions $d[e[z: R]]$ and $e[d[z: R]]$ are possible.

The scheme also opens up possibilities to access various components of the same types through the same interface. Development of protocol layers with each layer providing general functionality could have the implementation shown next.

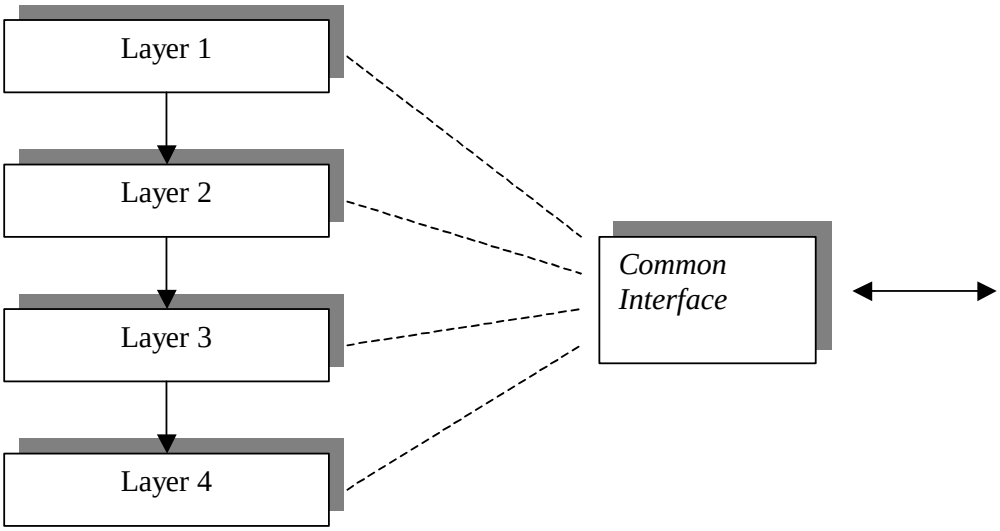


**Figure 3.3:** Stacked Microprotocols accessed through a well defined published interface

## 3.11 Summary

This chapter provided an overview of the various methods of attaining reuse, its justification, sociological and economic factors preventing software reuse and some of the traditional approaches to it. It has also looked at some of the recent developments in these fields. Other areas covered were software components and faults and an overview of measuring and quantifying reuse. Finally, some component-based designs and architectures were discussed.

The next chapter provides an overview of the network tools platform, its architecture, main components and its state transition model.

# 4 Network Test Tools Platform

## 4.1 Introduction

The motivation for the development of this component-based software system was mainly to overcome problems identified while developing purpose-built tools from scratch. The main problems identified were:

- Long development times.
- Increased costs.
- Limited reusability of existing software tools.
- Limited scalability of these application specific tools.

Once developed, these tools could only be used for the specific purpose that they were primarily developed for. New requirements normally meant development of entirely new tools. The collaborator for this research project, Seven Layer Communications Ltd., identified a significant opportunity to develop a powerful, flexible platform for hosting a suite of network test tools for the telecommunications and networks market.

Network design tools are generally used to plan networks, improve network efficiency, reduce costs, improve performance, and may also be used to predict required bandwidth for new applications (Mann 1998). Complex test harnesses, implemented using network test tools, must be capable of running multiple structured tests simultaneously. In actual cases, test tools that are based on a single thread are not sufficient and are not capable of managing multiple simultaneous requests. Amaranth (1998) presented a system capable of running multiple simultaneous tests using C and Tcl, and attained a system which is capable of running multiple structured tests, simultaneously.

The proposed development was to deliver a software framework consisting of several components, which could be reused to develop test tools to meet changing requirements. With the presence of such a framework, the developer could use it as a starting point for the initial development of the system. Having achieved this, the developer could then focus on developing functions that were strictly specific to the test require-

ments. Several functions like memory allocation functions, file handlers, signal handlers, timing functions, and so on, are housed in the framework as standard components and therefore avoided the need to rewrite these functions for every tool.

## 4.2 Problems in developing software tools for network testing

Test tools are generally custom applications designed to test functions delivered by the target *Software-Under-Test* (SUT). By virtue of the nature of the test tools being application-specific, they are designed and implemented from requirements drawn from the SUT requirements (Figure 2.2). The aim of the test tool is thus to test, through pass/fail criteria, whether the software meets the functional specifications. For the results, the test outcome is fed back into improving the software-under-test and/or into fixing bugs. The designer can also get an extensive viewpoint on the actual implementation of the system. Often bottlenecks for hardware and software can be identified only by stress testing each element of the design.

The deliverables from the test tool development lifecycle - tool suites, test scripts, results, etc - are supplied back to the SUT development life-cycle before its final release and deployment. Typical examples of software tools used in network testing (Golam et al, 2002) are:

- Data generation tools used to simulate network traffic conditions or generate configurable data at known speeds, such as software tools to generate telecommunications signals to simulate conditions of telecom traffic.
- Data acquisition tools used for data logging and verification, such as tools used to check sanity of data generated from protocol stacks.

Some of the problems associated with the development of purpose-built tools from scratch are decreased scope of extensibility of the existing class of tool and reduced efficiency. During the tool development lifecycle, it was observed that certain common features across tools were rewritten every time new tools were developed.

## 4.3 Network Test Tools Platform Architecture

The proposed architecture was based around reusing common functions that allowed developers to focus on developing compact tools that were strictly specific to requirements. Figure 4.1 shows that the Network Test Tools Platform Architecture consists of three layers:

- **Tools layer**. This contains tools that are used for generation, acquisition or verification of test data. The Tools layer used available driver functions to communicate to the hardware.
- **Drivers layer**. This makes use of third party drivers, such as Ethernet drivers and I/O card drivers, to communicate to the underlying hardware layer. Drivers were used for driving Ethernet interfaces without any modifications.
- **Hardware layer**. In this research, this consists of an Intel x86 processor based system, which was capable of loading Windows or Linux operating systems and Ethernet hardware interface modules (Mod 1 – n).

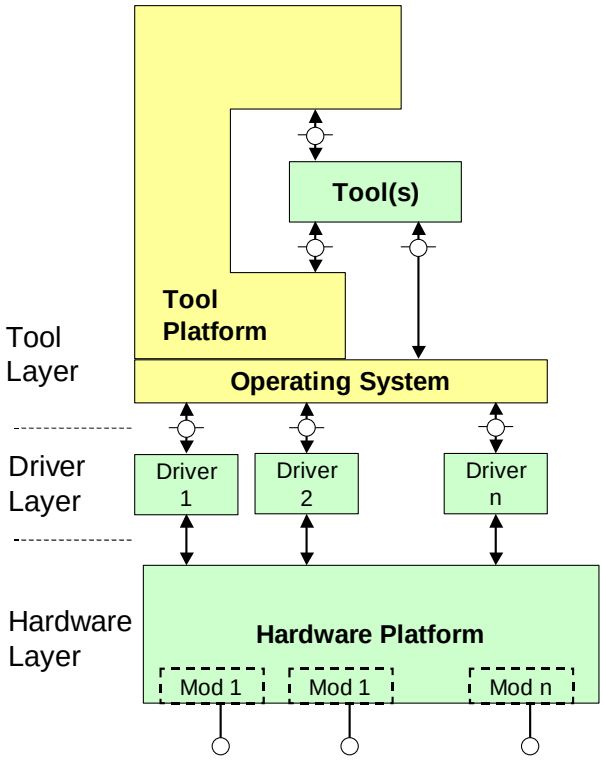


**Figure 4.1:** Network Tools Platform Architecture

Functions available on Linux, like *gettimeofday* for timing measurements and *ptheads* POSIX (Portable Operating System for unIX) threads, were used in the development. Thread functions implemented in the Tools layer were used for the generation activity while timing measurements were used to measure Ethernet data transmission times.

The network test tools platform could be used to validate any network-under-test that used any underlying transmission technology. In the current version of the platform, Ethernet technology was to generate user configurable 100 BASE-T data[1]. Ethernet generators are normally used in tests to simulate traffic conditions or to provide Tool User-configurable data at known speeds to test software. Only one Ethernet generator was used in the Tools layer to interact with the Ethernet hardware through drivers.

Ethernet conforms to the data link and physical layers of the OSI model and conform to the IEEE 802.3 standard for data frames. Typical tests on the data frame would be:

- Start and end delimiter test. Ethernet contains a start and end delimiter for the frame, which is identified with a consecutive sequence of 10101010. A test system would verify that the system could respond to this, and also simulate incorrect start and end delimiters.
- Source MAC address test. On a network each node has a unique source 48-bit MAC address. The testing device must be able to simulate the required range of source addresses, and also for incorrect MAC addresses, so that the device-under-test can be tested for incorrect operation.
- Destination MAC address test. The device-under-test should only respond to its own MAC address, and also to a broadcast network address (which is identified with the special address of FF-FF-FF-FF-FF-FF).
- Data frame payload tests. The testing system must be able to generate differing sizes of data frame, to see if the Device Under Test (DUT) can respond correctly to these. In this research the payload is generated using IP packet, which have been generated from TCP segments.

[1] Ethernet is the most widely used Local Area Network (LAN) technology that allowed data transfer rates of 10 Mbits per second to 1 Gbits per second.

- Error Detection test. Each Ethernet frame contains a CRC for error detection. The testing system must thus be able to generate correctly formed CRC, and incorrect ones (so that the DUT can detect an error).

The three layers of the architecture are shown in Figure 4.2. This contains a set of core components and a set of standard components. In this case it is an example of the components in the context of the Ethernet example.

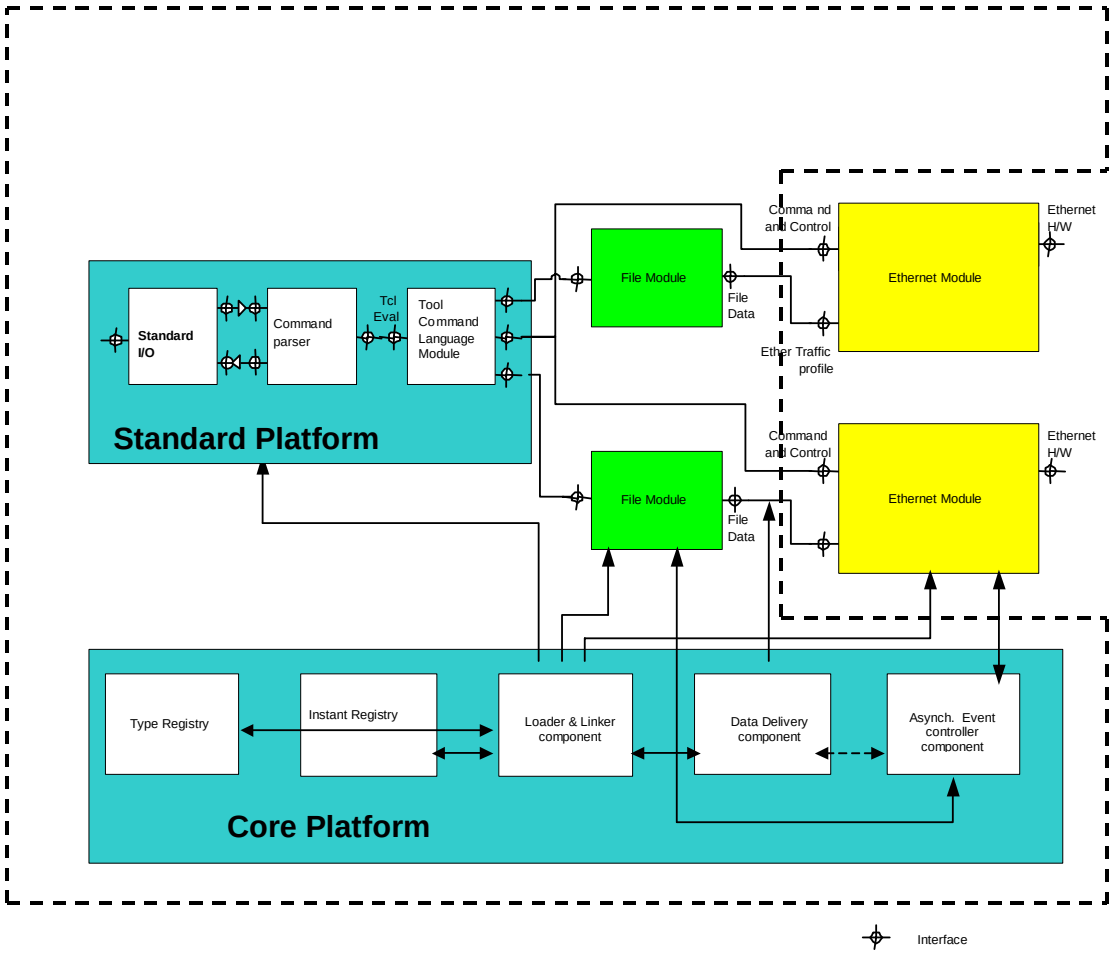


**Figure 4.2:** Ethernet Data generator using two Ethernet modules. Interactive User control is provided through the 'Standard I/O' component and File modules provide Ethernet data to the tools.

## 4.4 Core Platform Components

Core components are components that are generally present in the platform and are used in every network test tool. Various components in the platform are used to implement the generator. They include:

- **Loader and Linker component**. These are used to load shared library modules to create individual instances of all the components in the platform and individual instances of all Ethernet modules.
- **Data delivery component**. This is used for steering blocks of data from one part of the software system to another, in a known and controlled manner.
- **Asynchronous event controller**. This component provides a common mechanism for controlling and responding to asynchronous events, which can be any or all of the following:

    - File Input/Output.
    - Socket Input/Output.
    - Timing.
    - Signal.
    - Alarms.
    - Network events.
    - Tool User interactions.

- **Type Registry**. This records each instance of a particular type of component present in the platform. Type and Instance Registries are used to track and find active instances of components within a platform.
- **Instance Registry**. This keeps a count of instances of a particular type of component in the platform. For example, if the platform uses two instances of the Ethernet component to generate data, then the instance register records these with different names under the Ethernet generator type.

## 4.5 Standard Platform Components

Standard platform components depend upon the choice of interpretive language support provided. Various components in the platform are used to implement the generator. These include:

- **Standard Input/Output**. The 'StdIo' component provides the gateway for Tool User control over the platform.
- **Command Parser**. This translator component parses user commands to meaningful commands for the command interpreter.
- **Tcl Command Interface**. This provides a common programmatic interface for controlling the platform using an interpretive language.

## 4.6 Tool User Control Over The Platform

Command line control was introduced in order to provide a better programming environment, and was used to configure the network test tool platform to test the network-under-test. This method represents the communication medium between the Tool User and the environment. Normally, command interpretation is done in two distinct stages:

1. The interpreter layer interprets the Tool User command and invokes the relevant software system component.
2. The software system component then performs the requested operation.

The command line interface is implemented using Tcl (Tool Command Language), which is an interpreted language with programming features, available across platforms running the Unix, Windows or Apple Macintosh operating system. Tk, the associated toolkit, is an easy and efficient way of developing Windows-based applications. Application tasks at the lowest level were developed as C/C++ modules, and integrated with Tcl. The modules were then exported as new Tcl commands. The Tool User executes individual modules using this new command. A collection of Tcl commands in a script was composed to make the overall application. The scripting language, much like any shell language, has the ability to access and execute other programs. Several Tcl based applications can be made to work together to create a new application or extend an existing

one.

## 4.7 State Transition Model

The operation of the network test tools platform follows a state transition model from power up to shutdown. Figure 4.3 shows the state transitions based on Tool User commands, which allow a Tool User to configure the generator and prepare the operating environment before actually generating data. Appropriate queries can be made to determine the state of the software system at any time. A typical test run for the Ethernet generation consists of:

- **Power on**.
- **Initialisation**. The network test tools platform, including test tools, are created and initialised. This would bring the state of the Ethernet generator to 'not configured'.
- **Basic Configure**. The tool is configured, communication to the Ethernet driver is established, and a socket is opened for communication. A Tool User defines the contents of an Ethernet frame.
- **Start**. This action from the Tool User triggers the start of the Ethernet data generation.
- **Pre-Run Check**. After a primary *pre-run check*, the Ethernet data generation starts in a separate thread.
- **Stop**. On completion of the generation thread, the state of the generating engine would advance to halt again. If a trigger from the Tool User is received, generation would start again.

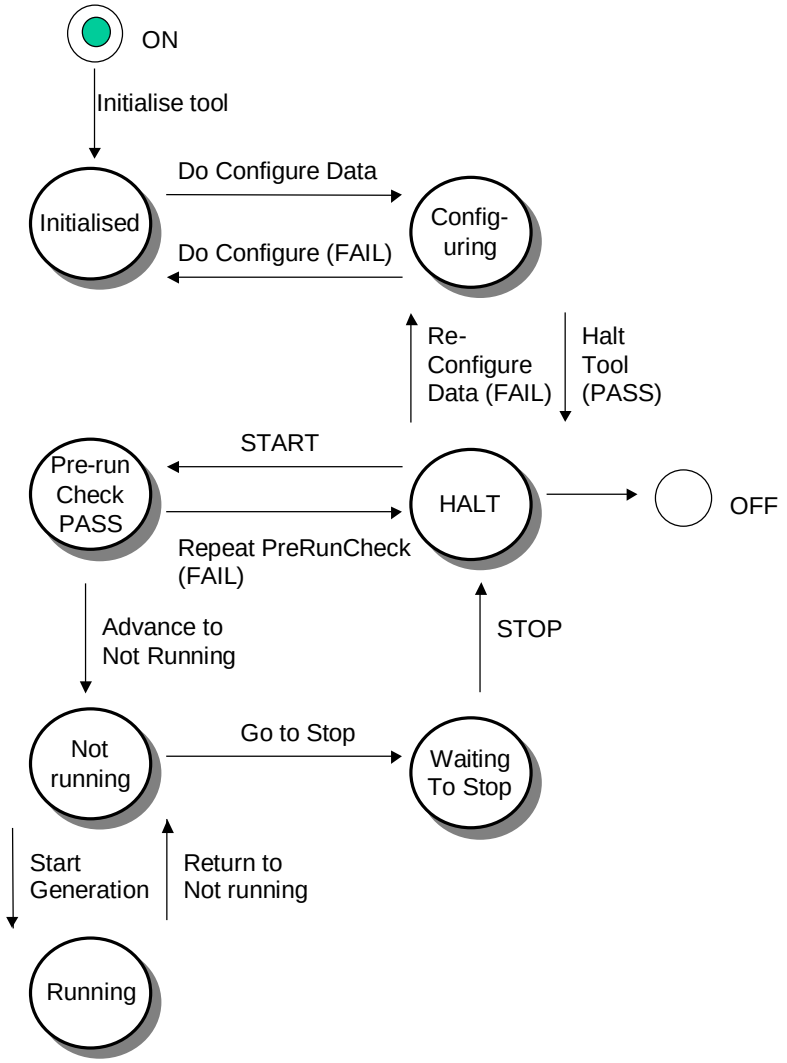


**Figure 4.3:** State transition based on incoming commands (config, start, stop) from Command handler. Data generation is managed as a single and separate thread

## 4.8 Summary

This chapter presented an introduction to the Tools platform and the problems that led to its design and development. An overview of the tools platform architecture was presented which depicted its use in the context of network testing. Components that made

up the tools platform were also presented in this chapter. The final section outlined the sequence of states and operations of the tools platform, during a typical generation process, through the use of a state transition diagram. The next chapter presents a case study involving this platform and discusses the results obtained from test results.

# 5 Case Study and Results

## 5.1 Introduction

The reason for reusing software is to attain a greater return on development time than traditional approaches. Some proposed methods of attaining software reuse are to:

- Specify structured system specifications.
- Use smaller logical components to build a complex system.
- Detailed planning of software components through the use of object-oriented business and system modelling.

Several component based modelling techniques are used to construct a basic framework with reusable components (Chuang et al, 2001, Stallinger et al, 2002, Milli et al, 1995, and Szyperski, 1998). Having done this, each component is then rigorously modelled and potential mismatches between components are resolved to adapt to the new environment (Hongxia et al, 2001). A key problem in software reuse is the selection of appropriate components for satisfying a given requirement, which must be applied to different domains of application (Redondo et al, 2002). Since this is not possible in the very first iteration, components are used to make up the basic framework of the software system, and then fine-tuned in several iterations in order to attain the level of performance that one would expect from a bespoke software system.

In this application, the network test tools platform was composed of several components which are used to control a tool. This tool generates 100BASE-T Ethernet frames in quick successions. Data frames are configured so that the actual payload in frames could be altered and re-sent. This is typically used to test network software under different payload conditions and error conditions for which errors are actually introduced by the Tool User into the Ethernet frames.

In order to assess the usefulness of the platform, a purpose-built tool was used to operate under similar conditions and perform exactly the same operation as the platform. This purpose-built tool existed before the implementation of the tools platform and

used the same third-party drivers, and had a similar architecture. The difference was that the purpose-built tool did not use any of the prefabricated components from the platform in the Tools layer, but did use the underlying driver and the hardware layer.

## 5.2 Objectives

The objectives of testing the network test tools platform were to verify the following ideas:

1. The generic component-based model attains a greater return on development time than traditional approaches (Hall et al, 1997 and McClure, 1997).

   *Test method:* This could be tested by developing a network test tool using two approaches – one using the reusable component-based model and the other using traditional methods to develop a bespoke monolithic tool – and then individually testing them for savings in development times.

   *Implementation details:* This test could not be performed owing to limited project timelines and lack of developers who could individually verify the two approaches. New product development in the company, however indicated considerable savings in development time when core components (Section 4.4 Core platform components) were used from the platform.

2. The network test tools platform could achieve the same or comparable levels of performance as those obtained from bespoke tools. The main research problem identified was to reduce the development time of network test tools and to achieve the same or comparable levels of performances as those obtained from bespoke tools (Chapter 1, Introduction, Section 1.2 Research outline.)

   *Test method:* Subject the reusable Network test tools platform to the same operation, as that of a bespoke tool. The task selected was to generate user-configurable 100 BASE-T Ethernet data at specified rates.

*Implementation details:* This approach was tested using the new network test tools platform and a bespoke tool that existed prior to the implementation of the tools platform. The case study is presented in the following sections.

## 5.3 Test set-up and test procedure

The aim was to use the component-based model to attain similar performance levels of the bespoke tool. They generated 1000 Ethernet frames with varying payload sizes (120-1514 bytes) over a 100BASE-T line, at the maximum given speed. As shown in Figure 5.1, Ethernet frames are generated from a transmitting computer on a 100 BASE-T line and were received by another system, and checked for correct formats and sizes.

To assess performances, the following were measured;

- Bit transfer rate of data in Mbps, and
- Percentage utilisation of the 100BASE-T line.

The effects of the component-based software model were studied in two stages. The first stage deployed the individual components and studied the desired level of performance and system flexibility. The second stage fine-tuned each of the components to attain the desired degree of performance as that of the purpose built tool and studied the performance of the network tools platform with respect to the purpose built tool.

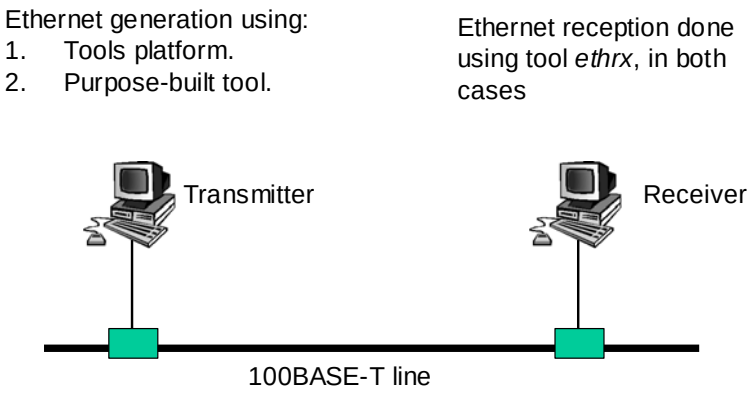


**Figure 5.1:** Test setup using two hosts to generate and collect Ethernet frames. Frames generated were verified for correctness at the receiving host

## 5.4 Sample calculations for maximum theoretical throughput

The software system used Ethernet frames that are units of Ethernet data containing a source and destination address. An Ethernet frame conforming to IEEE 802.3 has:

- **Preamble** (7 bytes)[1]. This informs all the other nodes on the segment that the node wishes to communicate, and should be enough time to detect that another node is trying to communicate at the same time.
- **Start of frame delimiter** (1 byte)[2]. This identifies that the next part of the frame contains the destination address.
- **Destination address** (6 bytes)[3]. This is a 48-bit MAC (Media Access Control) address[4], and is the physical address of the destination node.
- **Source address** (6 bytes). This is the MAC address of the originator of the data frame.
- **Type information** (2 bytes). This typically contains information which defines the length of the data frame. There are several different types of Ethernet data frames, such as Ethernet SAP, Ethernet SNAP, Ethernet II and Ethernet 802_2, which have slightly different implements of the Ethernet standard.
- **Data payload** (46 to 1500 bytes)[5]. The minimum frame payload is 46 Bytes.
- **Frame check sequence** (4 bytes). This is an error detection scheme, and uses the standard CRC-32 conversion.

---

1 The Preamble bit patterns is 10101010…1010.

2 The standard start-of-frame bit pattern is 10101011.

3 A special address of FF-FF-FF-FF-FF-FF can be used as a broadcast address, in which all the nodes on the segment listen to the data frame. The broadcast address is typically used to determine the MAC address of a host, which has a known IP address. This process is known as ARP (Address Resolution Protocol).

4 The first 24 bits identifies the manufacturer of the network card, and the second 24 bits identifies the serial number of the NIC (Network Interface Card).

5 A data frame which is less than 46 Bytes is known as a runt, and a data frame which is greater than 1514 Bytes is known as a giant.

A single transmitting node that does not suffer any collisions achieves the maximum frame rate. This implies a frame consisting of 72 bytes with a 9.6 μs inter-frame gap (corresponding to 12 Bytes at 10 Mbps) corresponds to 84 bytes (Fairhurst, 2001).

| *Frame Part* | *Minimum Size Frame (Bytes)* |
|---|---|
| Inter Frame Gap (9.6μs) | 12 |
| MAC Preamble (+ SFD) | 8 |
| MAC Destination Address | 6 |
| MAC Source Address | 6 |
| MAC Type (or Length) | 2 |
| Payload (Network PDU) | 46 |
| Check Sequence (CRC) | 4 |
| **Minimum Frame Physical Size** | **84** |

The maximum number of frames per second is:

$$Frames_{\max} = \frac{Ethernet\_Data\_Rate}{Total\_Frame\_Physical\_Size(bits)} \quad (5.1)$$

For a maximum payload of 1500 bytes, the entire frame has the following content:

$$Frames_{\max} = \frac{10{,}000{,}000}{84 \times 8)} = 14{,}800\, frames/\sec \quad (5.1)$$

| *Frame Part* | *Maximum Size Frame (Bytes)* |
|---|---|
| Inter Frame Gap (9.6μs) | 12 |
| MAC Preamble (+ SFD) | 8 |
| MAC Destination Address | 6 |
| MAC Source Address | 6 |
| MAC Type (or Length) | 2 |
| Payload (Network PDU) | 1500 |
| Check Sequence (CRC) | 4 |
| **Maximum Frame Physical Size** | **1538** |

The largest frame consists of 1526 Bytes with a 9.6 μs inter-frame gap (corresponding to 12 bytes at 10 Mbps). The total utilised period (measured in bits) therefore corresponds to1538 bytes. The maximum frame rate is:

$$Frames_{\max} = \frac{Ethernet\_Data\_Rate}{Total\_Frame\_Physical\_Size(bits)} \tag{5.3}$$

$$Frames_{\max} = 812.74 \frac{frames}{\sec} \tag{5.4}$$

The link layer throughput (i.e. number of payload bits transferred per second) is:

$$\begin{aligned} \text{Throughput} &= \text{Frame Rate} \times \text{Size of Frame Payload (bits)} \\ &= 812.74 \times (1500 \times 8) \\ &= 9{,}752{,}880 \text{ bps} \end{aligned}$$

This represents a throughput efficiency of 97.5 %.

Now, considering parts of the physical frame over which the user has control, we find that the user may generate a frame of maximum size of 1514 bytes. The part of the physical frame over which the user has no control (shown shaded in the table below) adds up to 24 bytes.

| *Frame Part* | *Bytes under control of user* | *Bytes not under user-control* |
|---|---|---|
| Inter Frame Gap (9.6μs) | | 12 |
| MAC Preamble (+ SFD) | | 8 |
| MAC Destination Address | 6 | |
| MAC Source Address | 6 | |
| MAC Type (or Length) | 2 | |
| Payload (Network PDU) | 1500 | |
| Check Sequence (CRC) | | 4 |
| **Total bytes** | **1514** | **24** |

Therefore the percentage utilisation for a maximum sized frame, is given by:

$$\begin{aligned} \text{Maximum percentage utilisation} &= \text{Bytes in use / Total bytes} \times 100\% \\ &= 1514/1538 \times 100\ \% \\ &= 98.4\% \end{aligned}$$

## 5.5 Case Study Results

All the frames sent out from the transmitting machine were received in both cases (using tools platform and purpose built tool) by the receiving machine. Performance was measured by sending frames at the maximum possible speed with variable payload sizes of *x* bytes. A fixed number of frames were sent out and the corresponding times for sending out this number of frames were recorded (*t* seconds). The actual number of bytes transmitted was obtained by multiplying the number of frames sent out with the payload sizes in bytes. This is the value *y* bytes shown in Table 5.1. Rate of bit-transfer on a 100 BASE-T line was calculated as follows.

### 5.1.1 Results before optimisation

This section outlines the first set of results that were obtained from the tools platform against the purpose-built tool, both operating under similar conditions. Table 5.1 shows bit transfer rate results values of frames with sizes from 120 bytes to 1514 bytes. Results obtained for the platform and the purpose built tool for maximum payload sizes of 1514 bytes were as follows (Figure 5.2):

| | |
|---|---|
| **Platform** | 23.11% |
| **Purpose-built tool** | 69.92 % |

**Table 5.1** - Performance of platform against purpose built tool

| Sr. No | Frame Len (x bytes) | Total bytes sent (×1000 b) (*y* bits) | Time (*t* sec) | Rate (Mbps) | Total bits sent (×1000 b) (*y* bits) | Time (*t* sec) | Rate (Mbps) |
|---|---|---|---|---|---|---|---|
| | | Purpose-built tool | | | Platform | | |
| 1 | 120 | 960 | 0.08335 | 11.52 | 960 | 0.72 | 1.33 |
| 2 | 240 | 1920 | 0.08735 | 21.98 | 1920 | 0.734 | 2.62 |
| 3 | 360 | 2278 | 0.07441 | 30.61 | 2278 | 0.721 | 3.99 |
| 4 | 480 | 2822.4 | 0.0748 | 37.73 | 2822.4 | 0.539 | 7.12 |
| 5 | 600 | 4800 | 0.10559 | 45.46 | 4800 | 0.507 | 9.47 |
| 6 | 720 | 5760 | 0.11171 | 51.56 | 5760 | 0.512 | 11.25 |
| 7 | 840 | 6720 | 0.10432 | 64.42 | 6720 | 0.512 | 13.13 |
| 8 | 960 | 7680 | 0.10125 | 75.85 | 7680 | 0.516 | 14.88 |
| 9 | 1080 | 8640 | 0.12453 | 69.38 | 8640 | 0.52 | 16.62 |
| 10 | 1200 | 9600 | 0.10189 | 94.22 | 9600 | 0.521 | 18.43 |
| 11 | 1320 | 10560 | 0.12421 | 85.01 | 10560 | 0.522 | 20.23 |
| 12 | 1440 | 11520 | 0.16218 | 71.03 | 11520 | 0.525 | 21.94 |
| 13 | 1514 | 12112 | 0.17322 | 69.92 | 12112 | 0.524 | 23.11 |

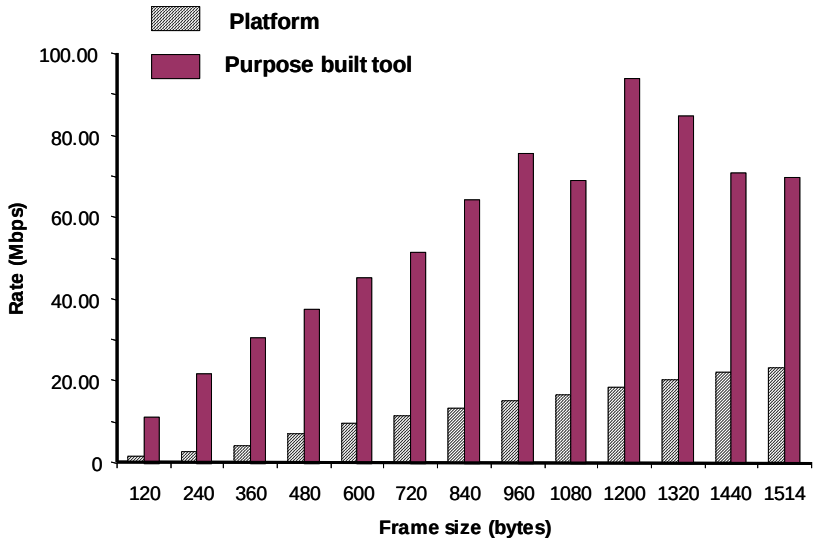


**Figure 5.2:** Performance of platform against purpose-built tool

## 5.6 Reasons for poor performance of platform

The test results demonstrated the expected results that the purpose-built tool was far superior to the general-purpose component based model. The main reasons for the poor performance of the platform were due to:

- Activity overheads like command parsing, copying data between various components slowed down real-time data generation in the platform.
- Performance was reduced as data generation by the platform was carried out asynchronously, that is, the user could execute other activities, such as configure the next set of data to be generated or log results, while generation was in progress. This was a great advantage over the bespoke tool, but slowed down the real time data generation activity. In contrast, the bespoke tool generated Ethernet data synchronously,

and therefore performed better.

- The platform used several components to construct the basic framework of the system. This was the first iteration of the components, which worked together to achieve a common goal, but required further improvements.

## 5.7 Optimisation techniques

The second iteration was mainly based around optimising several components in the platform to improve the overall performance. The Ethernet component, the File handler component, and the Tcl component (Figure 4.2: Ethernet data generator) underwent major changes so that real time data generation could be separated from the user time activities. These include changes in the design and code level, particularly:

- An Ethernet data generation thread was used in the application to greatly improve the performance of the generation activity. The network test tool now simulated real-time conditions available in the bespoke tool by using a dedicated thread to manage the data generation activity. The master thread was still used, but was mainly used for user time activities in the test tool (command parsing and querying)[6].
- As an improvement over the previous version of the tools platform, data was now stored in a common area. All components that required access to the data were provided with a pointer to the data. This avoided unnecessary copying of data between components and therefore improved real-time performance.
- The State transition model for data generation was remodelled (Figure 4.3: State transitions based on incoming commands). The operation of the network test tools platform followed the state transition model from power up to shutdown, which allowed the Tool User to configure the generator and prepare the operating environment before actually generating data. In such a system, generation threads were created to generate data and then destroyed after the real-time generation activity had been successfully completed.

---

[6] We used POSIX (Portable Operating System for UNIX) compliant threads under Red Hat Linux 7.0 for our application

## 5.8 Results after optimisation

Table 5.2 shows bit transfer rate results values of frames with sizes from 120 bytes to 1514 bytes. Results obtained for the platform and the purpose built tool for maximum payload sizes of 1514 bytes were as follows:

| | |
|---|---|
| Platform | 98.43% |
| Purpose-built tool | 98.48% |

Results obtained from the platform were found to be consistent with the purpose built tool thereby proving the fact that there was insignificant loss of performance of the platform when compared to a purpose built tool and was therefore a good Ethernet generator. Figure 5.3 shows the comparison the percentage utilization of the tools platform with respect to the purpose built tool. The tools platform was observed to closely follow the behaviour of the purpose built tool and therefore proved that there was no or minimal loss of performance for the platform to carry out the generation activity.

**Table 5.2:** Performance of platform against purpose-built tool

| Sr. No | Frame Len ($x$ bytes) | Total bits sent (×1000 bits) ($y$ bits) | Time ($t$ sec) | Rate (Mbps) | Total bits sent (×1000 bits) ($y$ bits) | Time ($t$ sec) | Rate (Mbps) |
|---|---|---|---|---|---|---|---|
| | | **Purpose-built tool** | | | **Platform** | | |
| 1 | 120 | 199904 | 3.325 | 60.12 | 199904 | 5.152 | 38.80 |
| 2 | 240 | 396480 | 5.000 | 79.30 | 541544 | 7.322 | 73.96 |
| 3 | 360 | 450504 | 5.136 | 87.71 | 450504 | 4.823 | 93.40 |
| 4 | 480 | 1366056 | 14.348 | 95.21 | 1366056 | 14.382 | 94.98 |
| 5 | 600 | 616352 | 6.410 | 96.15 | 616352 | 6.497 | 94.87 |
| 6 | 720 | 725880 | 7.505 | 96.72 | 725880 | 7.514 | 96.60 |
| 7 | 840 | 784120 | 8.064 | 97.24 | 784120 | 8.072 | 97.14 |
| 8 | 960 | 550312 | 5.636 | 97.65 | 550312 | 5.640 | 97.57 |
| 9 | 1080 | 921824 | 9.420 | 97.86 | 921824 | 9.419 | 97.87 |
| 10 | 1200 | 635600 | 6.478 | 98.12 | 635600 | 6.488 | 97.96 |
| 11 | 1320 | 913304 | 9.305 | 98.16 | 913304 | 9.297 | 98.24 |
| 12 | 1440 | 1082928 | 11.004 | 98.41 | 1082928 | 11.001 | 98.44 |
| 13 | 1514 | 771320 | 7.832 | 98.48 | 771320 | 7.836 | 98.43 |

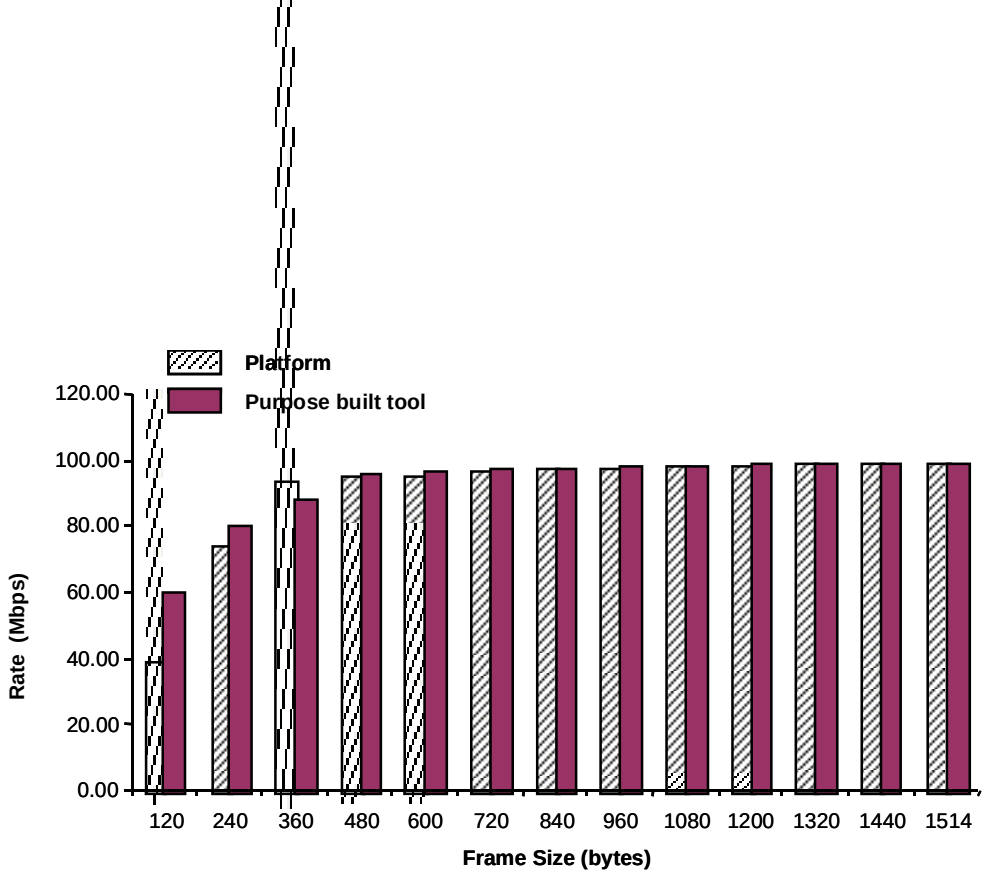


**Figure 5.3:** Performance of platform against purpose-built tool

## 5.9 Sources of errors

The main source of error is the effect of:

- Varying CPU usage. Times recorded for complete data transfer are not consistent over a series of test runs. This is due to other processes claiming CPU usage at any time and was beyond the control of the generation process.
- Timing measurements. This was greatly influenced by the use of time functions provided under Linux 7.0. Though the time functions provide time intervals of 1µs, they are not accurate to that degree.

These errors apply to both the platform and the purpose-built tool.

## 5.10 Discussion of results

The high performance obtained by the network test tools platform was due to the use of a separate child thread for generation that replicated dedicated test conditions of a purpose-built tool. A child thread was required in this case because activities not concerned

with the generation process (file I/O, data logging, querying, etc) were separated out in the master thread and the child thread was concerned only with the generation activity. The purpose built tool did not require a separate thread as no user time activity was implemented.

During the data generation process in the platform, a child thread was created when the state reached *'Not Running'* (Figure 4.3). When the state of the platform progressed to *'Running'*, the platform simulated the exact running condiitons as the bespoke tool and hence produced almost identical results.

The use of an Ethernet data generation thread influenced the performance of the generation activity. By logically separating the Tool User time activity (command parsing, querying, etc) from the real-time generation activity, the network test tool closely simulated real-time conditions available in a purpose-built tool. Customised thread packages are provided in different environments for attaining parallelism. Being standardised, they allow the Tool User to write portable programs[7] (McCarthy et al, 1997). Each package provides mechanisms to spawn and join parallelism. The implementation of such mechanisms greatly influences the type of parallelism, which can be exploited at the application level (Xavier et al, 1999). Each of the child threads must wait until the child thread with the longest execution time finishes before joining to the master thread. Hence, at the application level, the master process is always dictated by the child thread with the longest execution time.

Parallelism was achieved in the platform by spawning a child thread at run-time and delegating all data generation tasks to it. The child thread logically separated itself from the slow activities of the parent thread and therefore obtained better results. The generating thread on completion joined itself to the main thread.

The main advantages of a Network Test Tools platform were:

- The provision of an operating environment to configure and control all tools. Tool Users could choose from a family of commands like configure, reconfigure, start, stop, load, unload, delete, and so on, through the use of a published command line interface.

---

[7] These are programs that are written for programs to be used across different multi-platform environments

- Communication between components inside the platform was manageable and well understood with the use of published interfaces. This allows Tool User access to components already available within the platform, or to develop new components to add to the platform suite.
- A programmable Tool User control over the platform allowed automatic and complete test executions through the use of test scripts.
- The provision of multi-tool support.

The result are monitoring and data logging tools to capture test results and monitor test progress. Problems identified during tool development were the lack of precise control over tools, the lack of understanding of system states usually important for debugging and logging test results and the increase of performance overheads for individual tools.

A network test tools platform could reduce the testing time of network maintenance and evolution, as the development and configuration of the networks test tool itself is made much shorter through the reuse of core and standard components. Reusable testing components could also bring consistency and reliability to the testing process. The platform supported scalability by allowing Tool Users to add new components.

A concern with using reusable components to build network test tools is the anticipated increased processing overhead, such as, increased embedded memory or processing time. This is incurred to accommodate the additional variability that often makes a component more reusable and is often a deterrent to a component-based software development approach.

In this research, we have demonstrated that the performance of the network test tools platform compares favourably with the performance of the purpose-built network test environment.

# 6 Conclusions

## 6.1 Outline

While Chapter 5 details our findings with respect to the case study, this chapter outlines our main objectives, the extent to which these have been met, the scope for improvements and future work. This research was identified as a significant opportunity to develop a reusable tools platform, as the development of bespoke network test tools was inefficient owing to typically long development times, increased costs, and limited scalability of these application specific tools (Chapter 4, Network test tools platform). The objectives were to:

- Research the various options available for developing a reusable framework.
- Compare such a framework with a bespoke tool.
- Reduce the general development time of new tools through this reuse and to recover some of this investment (Hall et al, 1997 and McClure, 1997).

The advantages from the developed system were:

- Extensibility. The tools platform could be used to provide new functionality and to include new execution properties that extended an existing service. Design of similar function could be included to an existing tool suite. This, along with reusability, gave the approach of tools platform its strength.
- Reusability. Multiple related services for different applications could be used with the use of individual functions rather than having to implement entirely new services. Development was based around standardised components that had been tested. These components served as standard building blocks and integrated well into specific development with minimum rework.
- Configurability. Customised services could be provided by adding new functionality to the platform. The platform provided the basic software framework for such additions. Developers could make use of the available interfaces available within the

platform or define their own when developing new components.

## 6.2 Review of objectives

Our main research problem was to reduce the development time of network test tools and to achieve the same or comparable levels of performances as those obtained from purpose-built tools. Our research involved investigation into areas of software reuse, tools platforms, networking, and component-based software design and techniques. This led to literature searches to particularly focus on publications such as tools platforms used for authoring environments and designing objects, software simulators in a shared environment, component-based agent architectures, environment-adaptive computation models and models which reused investment planning and improvements in development processes and software engineering (Chapter 3 Software Reuse).

Along with this, the research has developed a novel software platform, which was used to appraise the research (Chapter 4 Network test tools platform). The platform is a component-based software framework that could specifically target network testing and overcome inefficiencies identified, while developing bespoke network tools for new requirements. The work has been presented at the Industrial Track, IEEE ECBS 2002 Conference in Lund, Sweden (ECBS, 2002). The model was tested against a bespoke tool and levels of performance of the platform were found to closely follow that of the bespoke tool (Chapter 5 Case Study and Results).

The test results obtained achieved our objective of researching and developing a reusable software platform that obtained acceptable levels of performances. Our second objective of testing whether this generic component-based model attained a greater return on development time than traditional approaches could not be performed owing to lack of time and manpower (Section 5.2 Objectives, Chapter 5 Case study and results).

## 6.3 Discussions

A concern with using reusable components to build network test tools is:

- Anticipated increased processing overhead.
- Increased memory (particularly embedded memory) or processing time
- The additional variability for making a component more reusable

These reasons above may act as deterrents to the component-based software development approach. However, the use of a reusable software system could:

- Reduce the testing time of network maintenance and evolution.
- Add consistency with reusable testing components.
- Add reliability to the testing process.
- Make development times shorter. The development and configuration of the networks test tool itself is made much shorter as they are strictly specific to the test requirements. All other generic functions are derived from the reusable framework.

## 6.4 Further work

In our research, we have focused on generic building blocks (Milli et al, 1995) and component-based software engineering (Szyperski, 1998). Our approach has been the development of reusable components from available libraries. In contrast, Griss (1993) describes methods by which developers could browse libraries for a component, rather than designing a product from available components. This approach could potentially find application when the network tools platform hosts a fair number of components and/or attains a certain degree of maturity.

Our proposed application was developed with a target application domain in mind, namely, the network test tool domain. We have developed a generating engine, which was capable of generating user-configurable 100 BASE-T Ethernet data. The component-based network test tool could be potentially expanded to include other network test tools. A possible future development would be the design and development of an acquisition engine, which would be capable of accepting network traffic and checking the correctness of data and looking for error conditions. In addition, the platform framework could be expanded to include other components in order to cater to both network and non-network testing requirements.

We have found in our journal research that a considerable amount of work has been done in application run-time models in which models adapt to different run-time environments (Lam et al, 2002 and Ubayashi et al, 2002). Objects in the software system usually have a predefined role throughout their lifetimes. Ubayashi et al. (2002) have

proposed an environment-adaptive computation model (*epsilon computational model*), which derives its run-time information from the language side and allow objects to change their roles dynamically along with the progression of computation. This approach could be added as a separate higher-level functionality to our network test tools platform, especially in cases where several network tests must be carried out by the same platform. Complex Regression testing would typically require several tests to run back-to-back or several of the these chains in parallel. In cases, where sub-tests fail or error conditions appear in the execution of the sub-test, the master process must decide the progression of the test. This could mean that the chain of tests may need to continue, stop, wrap-up or halt depending upon the structure of the whole testing mechanism.

A key factor of the measurement of reuse would be to use reuse rate measurements (Rothenberger et al. 1999).

# References


Amaranth P (1998) A Tcl-based multithreaded test harness. *Proceedings of the Sixth Annual Tcl/Tk Conference. USENIX Assoc.* pp69–77.

Aniorte P, Seyler F (2002) Distributed software: from component model to software architecture. *ITI 2002. Proceedings of the 24th International Conference on Information Technology Interfaces. Univ. Zagreb. Part vol.1.* pp455–64.

Arango J, McKinley P (2000) Vguide: Design and performance evaluation of a synchronous collaborative virtual reality application. *Proceedings of the IEEE International Conference on Multimedia and Expo, July 2000. http://citeseer.nj.nec.com/arango00vguide.html*

Atkinson C, Bunse C, Groß H-G, Kühne T, (2002) Towards a general component model for Web-based applications. *Annals of Software Engineering, vol.13, 2002.* pp35-69.

Baldo J Jr (1998) A measurement framework for organizational software reuse attributes. *Joint Conference on Intelligent Systems 1999 (JCIS'98). Assoc. for Intell. Machinery. Part vol.* pp523-526.

Batory D, O'Malley S (1992) The design and implementation of hierarchical software systems with reusable components. *ACM Transactions on Software Engineering and Methodology, Vol 1, No 4, October 1992*. pp355–398.

Berard E (1993) *Life Cycle Approaches, Chapter 4 in Essays on Object-Oriented Software Engineering*. Prentice Hall.

Beys P, Benjamins R, Heijst G (1996) Remedying the Reusability -- Usability Tradeoff for Problem-Solving Methods. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Alberta, Canada.* pp2–1,2–20.

Bhatti T, Hiltunen A, Schlichting D, Chiu W (1998) Coyote: a system for constructing fine-grain configurable communication services. *ACM Transactions on Computer Sys-*

*tems, Volume 16, Issue 4.* pp321–366.

Biggerstaff T, Richter C (1987) Reusability framework, assessment, and directions. *Proceedings of the Twentieth Hawaii International Conference on System Sciences 1987. Hawaii Int. Conference Syst. Sci. 1987 Honolulu, HI, USA. vol 2.* pp502–12

Bylund M, Espinoza F (2002) Testing and demonstrating context-aware services with Quake III Arena. *Communications of the ACM, vol.45, no.1, Jan 2002.* pp46–8.

Chen H, Tse T, Chen T (2001) TACCLE: a methodology for object-oriented software testing at the class and cluster levels. *ACM Transactions on Software Engineering and Methodology, 10, 1 Jan. 2001.* pp56 – 109.

Chuang S, Chen C, Deng-Jyi C, Chen D (2001) The design and implementation of multimedia reusable components. *Proceedings of the ISCA 3rd International Conference Information Reuse and Integration. Int. Soc. Comput. and their Applications – ISCA, 2001.* pp97-101.

Crispen R, Stuckey L (1994) Structural model: Architecture for Software Designers. *Boeing Defence and Space Group ACM Press Series-Proceeding.* pp272-281.

Doroshenko E (1998) Toward a method for deriving measures of reuse. *Proceedings 1998 Australian Software Engineering Conference, IEEE Comput. Soc.* pp170-83.

Fairhurst G (2001) Ethernet frame calculations, *Last updated Oct 2001. Retrieved September 2002 from http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/enet-calc.html.*

Ferrero A (1999) *The Eternal Ethernet, Second Edition*. Addison-Wesley ISBN 0-201-36056-X.

Fricke S, Albayrak S, Meyer U, Bamberg B, Tobben H (1998) A development and test environment for agent-based telematic services. *Intelligent Agents for Telecommunications Applications Basics, Tools, Languages and Applications, IOS Press. 1998, Amsterdam, Netherlands.* pp225-51.

Golam H, Ian S, Buchanan W, Munoz J, Mannion M (2002) Development Model for a

Component-based Tools Platform. *Workshop On Component-Based Software Engineering: Composing Systems from Components, April 10-11 2002, Lund University, Lund, SWEDEN.*

Griss M (1999) Architecting for large-scale systematic component reuse. *Proceedings of the 1999 International Conference on Software Engineering, ACM. New York, USA.* pp615-16.

Hall P, Lingzi J (1997) The Re-engineering and reuse of software. *Software Engineering, The Open University, IEEE Computer Society, The Institute of Electrical and Electronics Engineers, Inc. 1997.*

Hongxia J, Santhanam P (2001) An approach to higher reliability using software components. *Proceedings 12th International Symposium on Software Reliability Engineering, IEEE Comput. Soc. 2001.* pp2-11.

Houhamdi Z (2002) Integrated support for software reuse in CASE. *Modelling and Simulation 2002. 16th European Simulation Multiconference. ESM'2002. SCS Europe.* pp108-14.

Jia Y, Gu Y (2002) Domain feature space based semantic representation of component. *Ruan Jian Xue Bao/Journal of Software, vol.13, no.2, Feb. 2002.* pp311-16.

Khorshid W (1988) Generating Environments for Programming-in-the-large. *Computer Science Department; Wayne State University; Detroit, MI 48202. Proceedings of the 1988 ACM sixteenth annual conference on Computer science.*

Kuczora M (2000) NDS and ZENworks of Novell Proposal of network administration tools. *Wydawnictwo Politech. Slaskiej. Studia Informatica, vol.21, no.1, 2000.* pp731-9.

Lakshmi S, Venkat V (1996) Tcl Tk cookbook. *Advanced Interactive Systems Division, Department for Computing and Information Systems Department Rutherford Appleton Laboratory, Chilton, Didcot. OX11 0QX.*

Lam T V, Lixin T (2002) Customizable software component run-time model.

*MASPLAS'02. Proceedings of the 8th Annual Mid-Atlantic Student Workshop on Programming Languages and Systems. Pace Univ.* 7.1-7.12.

Lewis O, Mannion M, Buchanan W (2000) Performance Issues of Variability Design for Embedded System Product Lines. *22nd International Conference on Software Engineering (ICSE), Limerick.*

Lowell J (1988) *Software Evolution, The Software Maintenance Challenge Chapter 1.* John Wiley & Sons, Inc. 1988, ISBN 0-471-62871-9.

Luqi G (2000) A survey of software reuse repositories. *Proceedings Seventh IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2000). IEEE Comput. Soc.* pp92-100.

Mann K (1998) Modelling goes out of fashion network design tools. *Network News (UK Edition), 28 Oct. 1998, 38. Publisher: VNU Business Publications, UK.* pp35-36.

Marinescu R (1999) A multi-layered system of metrics for the measurement of reuse by inheritance. *Proceedings Technology of Object-Oriented Languages and Systems. IEEE Comput. Soc.* pp146-55.

Martorell X, Ayguadé E, Navarro N, Corbalán J, González M, Labarta J (1999) Thread fork/join techniques for multi-level parallelism exploitation in NUMA multiprocessors. *Proceedings of the 1999 international conference on Supercomputing.* pp294 – 301.

McCarthy M, Linux J (1997) What is Multi-Threading? *ACM 1997, ISSN:1075-3583, Article 3. http://www.acm.org/pubs/citations/journals/linux/1997-1997-34es/a3-mccarthy/#abstract*.

McCarthy M, Linux J (1997) Thread-Specific Data and Signal Handling in Multi-Threaded Applications. *ACM 1997, 36es (Apr. 1997), Article, 3 http://www.acm.org/pubs/citations/journals/linux/1997-1997-36es/a3-mccarthy/*

McClure C (1997) Harvesting components for reuse. *Object Magazine, vol.7, no.7, Sept. 1997.* pp58-9, 62-4.

McIlroy D (1968) Mass-produced software components. In P. Naur and B. Randell, editors, *Software Engineering, NATO Science Committee report*. pp138-155.

McKinley P, Malenfant A, Arango J (1999) Pavilion: A Middleware Framework for Collaborative Web-Based Applications. *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, Phoenix, Arizona, November 1999.* pp179-188.

Mibe R, Takahashi S (2002) A method for software reuse in information and control software systems development. *Transactions of the Institute of Electrical Engineers of Japan, Part C, vol.122-C, no.5, May 2002.* pp851-9.

Milli H, Mili F, Mili A (1995) Reusing Software: Issues and Research Directions. *IEEE Transactions on Software Engineering, Vol. 21, No. 6: June 1995.* pp528-562.

Moreira A (1999) Proof preservation in component generalization. *FM'99 - Formal Methods. World Congress on Formal Methods in the Development of Computing Systems. Proceedings, Vol.II. (Lecture Notes in Computer Science Vol.1709). Springer-Verlag. Part vol.2.*

Olarnsakul M, Batanov D (2000) A component co-ordination model for customization and composition of component-based system design. *Seventh IEEE International Conference and workshop on the Engineering of Computer based systems. 3-7 April 2000.*

Taylor P (2001) Process Support for Component Factories on the Internet. *PhD, Feb 2001. University of Ulster.*

Park Y, Wu L (2002) Software component retrieval by composition using semantic properties. *International Journal of Computers & Applications, vol.24, no.1, 2002.* pp8-13.

Parnas D (1979) Designing software for ease of extension and contraction. *IEEE Transaction on Software. Eng. Mar. 1979.*

Prieto-Diaz R, Arango G (1991) Domain Analysis and Software Systems Modelling *IEEE Computer Society Press*, *1991.*

Redondo R, Arias J, Vilas A, Martinez B (2002) Approximate Retrieval of Incomplete and Formal Specifications applied to horizontal reuse. *Proceedings 28th Euromicro Conference, IEEE Comput. Soc.* pp90-97.

Rine D, Sonnemann R (1998) Investments in reusable software. A study of software reuse investment success factors. *Journal of Systems & Software, vol.41, no.1, April 1998.* pp17-32.

Rothenberger M, Hershauer J (1999) A software reuse measure: monitoring an enterprise-level model driven development process. *Information Management, vol.35, no.5, May 1999.* pp283-93.

Schmid K (2002) Integrating reference architecture definition and reuse investment planning. Software Reuse: Methods, Techniques, and Tools. *7th International Conference, ICSR-7. Proceedings (Lecture Notes in Computer Science Vol.2319). Springer-Verlag. 2002.* pp137-52.

Soliman K (2000) Critical success factors in implementing software reuse: a managerial prospective. *Challenges of Information Technology Management in the 21st Century. 2000 Information Resources Management Association International Conference. Idea Group Publishing.* pp1174-5.

Sommerville I (1998) *Software Engineering. Chapter 32, Software Maintenance, Section 32.1, Fifth Edition 1998.* Addison-Wesley. ISBN 0-201-42756-6

Spurgeon C (2000a) *Ethernet: The Definitive Guide*. O'Reilly and Associates. ISBN: 1565926609. Published Feb 2000.

Spurgeon C (2000b) Quick Reference Guides to 10 Mbps Ethernet. *http://www.ethermanage.com/ethernet/descript-10quickref.html. Last updated on January 31, 2000. Retrieved December 15, 2002.*

Stallinger F, Dorling A, Rout T, Henderson-Sellers B, Lefever B (2002) Software process improvement for component-based software engineering: an introduction to the OOSPICE project. *Proceedings 28th Euromicro Conference. IEEE Comput. Soc.* pp318-23.

Stillerman J, Fredian T, Klare K, Manduchi G (1997) MDSplus data acquisition system [for pulsed experiments]. *Review of Scientific Instruments, vol.68, no.1, pt.2, Jan. 1997*. pp939-42.

Sundarraj R (2002) An optimization approach to plan for reusable software components. *European Journal of Operational Research, vol.142, no.1, 1 Oct. 2002.* pp128-37.

Szyperski C (1999) *Component Software: Beyond Object-Oriented Programming.* Addison-Wesley.

Teitelbaum T, Reps T (1988) The Cornell Program Synthesizer: A syntax Directed Programming Environment. *Proceedings of the 1988 ACM sixteenth annual conference on Computer science.*

Thiry L, Thirion B (2002) Object-oriented modelling and simulation of complex control systems. *Modelling and Simulation 2002. 16th European Simulation Multiconference 2002, ESM'2002. SCS Europe.* pp115-19.

Thorne F (1999) Costs and benefits of reuse. *Australian Computer Journal, vol.31, no.1, Feb. 1999.* pp1-8.

Tracz W (1987a) Software Reuse: Motivators and Inhibitors. *Digest of Papers COMPCON, Spring 1987, Computer Society Order Number 764, Computer Society Press of the IEEE, Washington, D.C.* pp358 - 363.

Tracz W (1987b) Ada Reusability Efforts: A Survey of the State of the Practice. *Proceedings of the Joint Ada Conference, Fifth National Conference on Ada Technology and Washington Ada Symposium, U.S. Army Communications-Electronics Command, Fort Monmouth, New Jersey.* pp35 - 44.

Ubayashi N, Tamai T (2002) Modeling collaborations among objects that change their roles dynamically and its modularization mechanism. *Systems & Computers in Japan, vol.33, no.5, May 2002*. pp51-63.

Vazhkudai S, Cunningham H (2000) A reusable software framework for distributed de-

cision-making protocols. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. PDPTA'2000. CSREA Press. Part vol.2.* pp867-73

Wegner O (1984) Capital-intensive software technology. *[Journal Paper] IEEE Software, vol.1, no.3, July 1984.* pp7-45

Wehrle K (2001) An open architecture for evaluating arbitrary quality of service mechanisms in software routers. *Networking - ICN 2001. First International Conference on Networking. Proceedings, Part II (Lecture Notes in Computer Science Vol.2094). Springer-Verlag.* pp117-26

# Bibliography

Bianchi A, Fasolino A, Visaggio G (2000) An Exploratory Case Study of the Maintenance Effectiveness of Traceability Models. *IEEE 2000.*

Britton K, Parket R, Parnas D (1981) A procedure for designing abstract interfaces for device interface modules. *International Conference on Software Engineering*. pp195-204.

Clinger M (1990) Very high performance Networking for Supercomputing, *Conference on High Performance Networking and Computing. Proceedings of the 1990 ACM/IEEE conference on Supercomputing.*

Felice L, Riesco D (2002) Applying a reusable component model in RAISE formal method. *Issues and Trends of Information Technology Management in Contemporary Organizations. 2002 Information Resources Management Association International Conference. Idea Group Publishing. Part vol.1.* pp589-92.

Ford R (1990) A Generic Embedded Real-Time Monitor Subsystem. *Proceedings of the 1990 ACM annual conference on Cooperation 1990, Washington, D.C., United States ACM; ISBN 089791-348-5/90/0002/0312.*

Gutknecht O, Ferber J, Michel F (2001) Integrating Tools and Infrastructure for Generic Multi-Agent Systems. *Proceedings of the fifth international conference on Autonomous agents 2001, Montreal, Quebec, Canada*.

Kutten S, Peleg D, Vishkin U (2001) Deterministic Resource Discovery in Distributed Networks. *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures 2001, Crete Island, Greece ISBN 1-58113-409-6/01/07.*

Marques J, Guedes P (1989) Extending the Operating System to Support an Object Oriented Environment. *OOPSLA 1989 Proceedings.*

Simmonds R, Bradford R, Unger B (2000) Applying Parallel Discrete Event Simulation of Network Emulation. *University of Calgary, Canada and University of Bath, UK.*

*IEEE 2000.*

Tsasakou S, Voros N, Koziotis M, Verkest D, Prayati A, Birbas A (2001) Hardware-Software Co-design of embedded systems using CoWare's N2C methodology for application development. *Project performed in the framework of the ESPRIT project 24129 CODAC.*

Vishkin U (2000) A no-busy-wait balanced tree parallel algorithmic paradigm. *Proceedings of the Twelfth annual ACM symposium on Parallel algorithms and architectures.* pp147 – 155.

# A1 Appendix

## A1.1 Background Theory

Networks have rapidly grown from providing basic point-to-point connections in offices (the invention of Ethernet in Xerox PARC) to providing immense e-commerce services to allow customers the ability to shop, bank, collect and publish information, communicate and advertise among a host of other services. It has therefore become very important for networks to become robust, reliable and secure.

Networking involves the interconnection of workstations, terminals and other networked devices that allows computers of different types to intercommunicate using a network protocol. In order to make networks more manageable and robust, the OSI reference model provides a framework that eases the problem of moving information between computers by dividing the problem into seven smaller and more manageable tasks. Similarly, security measures must be implemented so that they do not inhibit or dissuade the intended e-commerce operation. These threats originate from both hackers as well as the e-commerce site itself. The vulnerable spots are at the endpoints - the customer's computer/network and the business' servers/network. Privacy issues are amongst the major drivers for improved network security along with the elimination of theft, fraud and vandalism (Marchany et al, 2002).

Some of the problems in the electronics industry are the interconnection of equipments, compatibility of software and connection of electronic equipments in one part of the world to another. The International Standards Organization (ISO) developed a model known as the OSI (Open Systems Interconnection) model to address these problems. Its main objects were to:

- Allow manufacturers of different systems to interconnect their equipment through standard interfaces.
- Allow software and hardware to integrate well and be portable on different systems.
- Create a model for all countries of the world to use.

Figure A.1 shows the OSI model in the context of a transmitter and a receiver system communicating with each other. In a typical transaction of data, data passes from the top layer of the sender to the bottom and then up from the bottom layer to the top on the recipient. Each layer on the sender, though, communicates directly to the recipient's corresponding layer, which creates a virtual data flow between layers. The seven layers of the model solves each of problem areas, which are: the physical layer; the data link layer; the network layer; the transport layer; the session layer; the presentation layer and the application layer.

The top layer, known as the application layer, initially gets data from an application and appends it with data that the recipient's application layer reads. This appended data passes to the next layer, known as the presentation layer whereby again it appends it with its own data, and so on, down to the physical layer. The physical layer is then responsible for transmitting the data to the recipient. The data sent can be termed as a data frame, whereas data sent by the network and the transport layers are typically referred to as a data packet and a data segment, respectively.

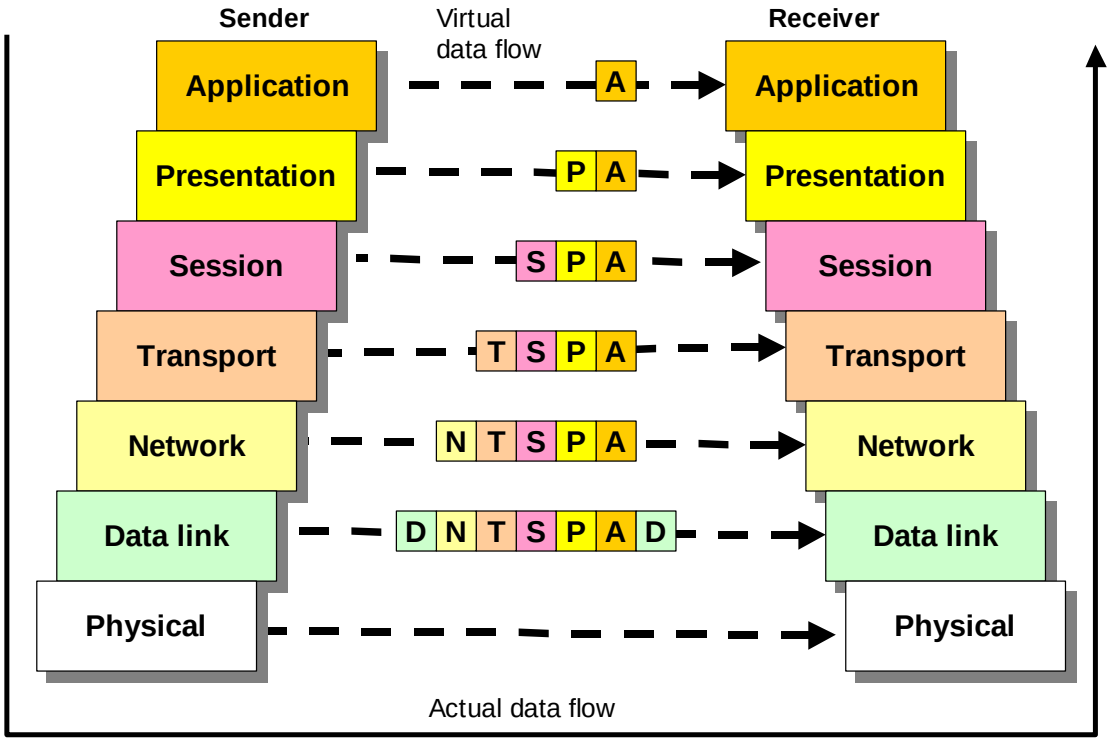


**Figure A.1:** Seven-layer OSI model

The basic functions of each of the layers are:

- **Physical.** This layer is concerned with the transmission of binary data. It defines the electrical characteristics of the communications channel and the transmitted signals, such as voltage levels, connector types, cabling, etc.
- **Data link.** This layer ensures that the transmitted bits are received in a reliable way. Extra bits are added to define the start and end of the data frame, error detection/correction bits for ensuring that multiple nodes do not try to access a common communication channel simultaneously.
- **Network.** This layer is concerned with addressing and determining the best path by routing data packets through a network. If data packets require going out of a network, then the transport layer routes them through interconnected networks. Its task may involve, for example, splitting data for transmission and re-assembling it upon reception. The IP part of TCP/IP is involved with the network layer (or IPX in Novell NetWare).
- **Transport.** The transport layer is concerned with the end-to-end connection and ensures reliability of connection. Network transmission protocol supports the transmission of multiple streams from a single computer. For example, the TCP part of TCP/IP is involved with the transport layer (or SPX in Novell NetWare).
- **Session.** This layer provides an open communications path with the other system. It involves the setting up, maintaining and closing down of a session. The communication channel and the internetworking of the data should be transparent to the session layer. For example, a typical session protocol is telnet, which allows for remote login over a network.
- **Presentation.** This layer provides a set of translations that allows the data to be interpreted properly. For example it may have to translate between two systems if they use different presentation standards, such as different character sets or differing character codes. The presentation layer can also add data encryption for security purposes.
- **Application.** This layer provides network services to application programs, such as file transfer and electronic mail.

## A1.2 Foundations of the OSI model

The OSI reference model is purely an abstract model, and provides a conceptual framework, which defines the network functions at each layer. It thus defines how data from the source - the network device that is sending data - is transmitted to the destination, which is the network device that is receiving data. This data is transmitted in the form of data packets. At the source, the data is passed through all of the layers of the OSI model, with each layer adding its own information. The process of adding the extra information is known as encapsulation. The data packet is thus wrapped in a particular protocol header. For example, Ethernet networks require an Ethernet protocol header before transmitting onto the Ethernet network.

Figure A.2 shows how the data link, network and transport layers are responsible for transporting data between applications basically covering the session, presentation and application layers. The data link layer delivers data between devices on a network segment, and the network layer is responsible for passing it between network segments and delivers the data at the destination using routers. The transport layer multiplexes the data into a single data stream for transmission, and demultiplexes it at the destination.

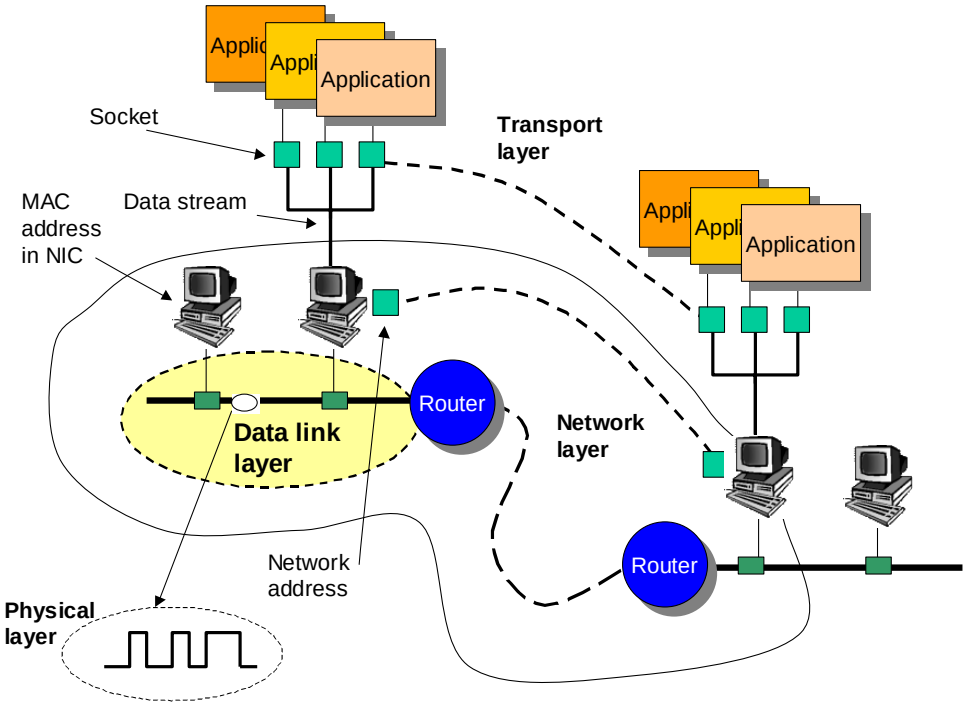


**Figure A.2:** Networking showing lower-level layers

### A1.2.1Data link layer – MAC addresses and Network Interface Cards

Computers connect to the physical media using an NIC (network interface card). The data link layer provides for the access to the network media and thus builds on the physical layer. It takes data packets from the upper levels and frames them so that they can be transmitted from one node to another. The data link layer provides for:

- **Error control.** This provides for the addition of binary digits that can be used to identify if there has been an error in the transmission of one of more bits. Generally, some mechanism exists for the destination to tell the source that it has received bits in error, and to request a retransmission.
- **Flow control.** This is where there is an orderly flow of transmitted data between the source and the destination, so that the source does not send data more than the destination is equipped to handle. Typically the destination sends back messages that indicate whether the destination can receive data, or not.
- **Line discipline.** This provides for the orderly access to the network media. This is to prevent multiple nodes to gain access to the common network at the same time. Typically only one node is allowed access to the network at a time, with techniques that allow an orderly access through collision detection and token passing. Collision detection involves detection of other nodes trying to transmit at the same time while token passing involves nodes passing an electronic token from one node to the next, so that nodes can only transmit when they capture the token.
- **Network topology.** Physical arrangement of network nodes and media within an enterprise networking structure.
- **Ordered delivery of frames.** This provides for sequencing of the data frames in the correct order, and allows the recipient to determine if there are any gaps in the sequence of the received data frames.
- **Physical addressing.** Each node on a network has a unique hardware address, which is normally known as a Media Access Control (MAC) address. This MAC address must be used if a node is to receive the transmitted data frame. The only other data frame that a node can receive is when the destination address is a broadcast, which is also received by all the nodes on the network. On Ethernet networks, the MAC address has six bytes, which is allocated by the IEEE.

**Physical and network addresses**

The MAC address identifies the physical address of the NIC, and differs from the network address (which is also known as a protocol address), which is used by the network layer. An Ethernet address takes the form of a hexadecimal number, such as:

`0000.0E64.5432` or `00-00-0E-64-54-32`

The network address for IP takes the form of a dot address, such as:

`168.176.155.130`

All computers that connect onto the Internet must have a unique IP address.

### A1.2.2 Network layer protocols for reliable delivery

The network layer defines protocols that are responsible for data delivery at the required destination, and requires Network addresses and Routing.

- **Network addresses.** This identifies the logical location of the node, the address within the network, and the actual node, which is the physical address. The form of the network address depends on the actual protocol used. For example, IP uses a dot address, such as 168.176.2.130 that identifies the network and the host. On the other hand, Novell Netware uses IPX address, which are eight-digit hexadecimal address to identify the network address and the node portion with a 12-digit MAC address, such as F5332B10: 00000E645432. Unlike network addresses that are set-up in software and are loaded into the computer when it starts, MAC address are unique and are set-up in the hardware.
- **Routing.** Routing refers to passing of data packets from one network segment to another. A router is responsible for routing and it does so by reading the network address and deciding on which of its connections it should pass the data packet on to. Routing information is not static and must change as the conditions on the network change. Thus each route must maintain a routing table, which is used to determine the route that the data packet takes. These routing tables are updated by each of the routers talking to each other using a routing protocol. Two typical rout-

ing protocols are Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). RIP uses the least number of hops which relates to the number of routers between the destination and the current router, whereas OSPF uses other metrics to determine the best route, such as latency and bandwidth capacity.

### A1.2.3 Transport layer

The transport layer provides for reliable end-to-end error and flow control. This is required because the network layer does not validate that any data packets have been successfully received and is therefore up to the transport layer to detect error and provide flow control.

- **Connection type.** This defines the method of handshaking between the source and the destination, and can be connection-oriented or connectionless. In the former, there are no acknowledgements and responses when the data is transmitted from the source to the destination. In the latter, however, a virtual connection is set up, and data is acknowledged by the destination, by sending acknowledgement data to the source. With this information, the source is able to determine whether the data has been received correctly. In order to detect if data segments have been lost or are in error, each data segment has a sequence number. The destination sends back the acknowledgement with the data packet segment that it expects to receive from the source, thus acknowledging all previously transmitted data segment to the acknowledged data segment number. Figure A.3 shows an example flow of information. The transport layer creates a connection by negotiation, where both the source and destination pass the details of their connection, for a connection-oriented connection. Passing details of their socket and the data segment number attains this.
- **Name resolution.** This allows for the resolution of logical names to logical network addresses. It is often easier to access networked devices using a logical name, rather than their logical address, as these are more user friendly. Domain Name Service (DNS) resolves domain names to IP addresses in a TCP/IP network. For example, a domain name of www.slayer.com could be resolved to the network address of 216.92.22.229.

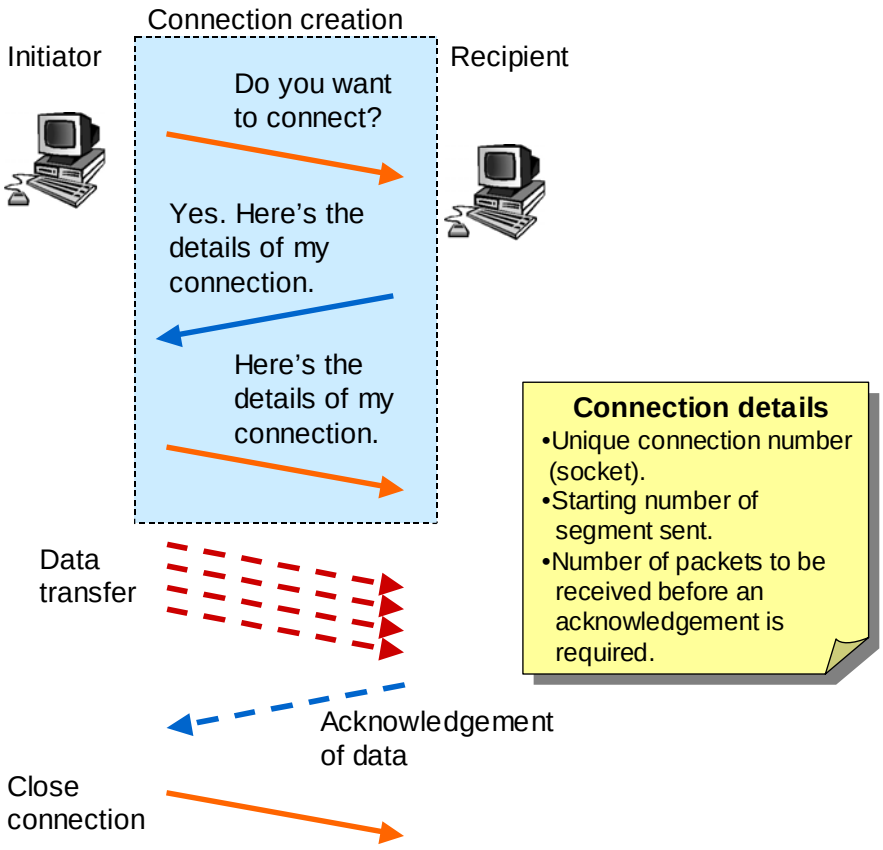


**Figure A.3**: Basic transport layer connection-oriented protocol

## A1.3 Ethernet

Ethernet is a local area network (LAN) technology that transmits information between computers at speeds of 10 and 100 million bits per second and most recently 1000 million bits per second (Gigabit Ethernet). The most widely and commonly used version of Ethernet technology is the 10-Mbps twisted-pair variety. Varieties of the 10-Mbps Ethernet media include the thick coaxial system, thin coaxial, twisted-pair, and fiber optic systems. 100-Mbps Fast Ethernet systems operate over twisted-pair and fiber optic media. This sub-section provides an overview of the Ethernet system, background history, elements of the Ethernet system and related protocols.

The first experimental Ethernet system was developed in Xerox PARC to interconnect the Xerox Alto, a personal workstation with a graphical user interface in late 1972. The first Ethernet was also used to link servers and laser printers. The signal clock was

derived from the Alto's system clock and resulted in the data transmission rate of 2.94 Mbps. It was named Ethernet with the name “ether” to describe the way of describing the physical medium (the cable), which was an essential feature of the system, much the same way that the old "luminiferous ether" was once thought to propagate electromagnetic waves through space (Spurgeon, 2000b).

### A1.3.1 Elements of the Ethernet System

The Ethernet system consists of three basic elements. These are:

- The physical medium which is used to carry Ethernet signals between computers,
- A set of access control rules that is embedded in each Ethernet interface so as to allow multiple computers to negotiate access to the shared Ethernet channel, and
- The Ethernet frame shown in Figure A.5, which has a structured set of bits used to carry data over the system.

### A1.3.2 Physical medium for Ethernet transmission

Ethernet-equipped computers, known as stations, operate independently of all other stations on the network by sharing a signalling medium. Each station follows the simple rule of waiting for a silence period by listening to the network before transmitting data in the form of an Ethernet frame.

At the end of each frame transmission, all stations on the network must contend equally for the next frame transmission opportunity, thereby ensuring that access to the network channel is fair, and that no single station locks out the network. This is achieved by the medium access control (MAC) mechanism embedded in the Ethernet interface located in each station.

This mechanism is based on a system called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This protocol operates by determining silence periods in the media before allowing transmission of data. If transmission from any other interface is detected (collision detection), the interface in question simply waits for its turn. Multiple access refers to all Ethernet interfaces that are equal in their ability to send frames onto the network without any interface getting a higher priority than anyone else.

Collision detection specifically refers to an algorithm of rescheduling transmission in the event of signal collision from different sources. All stations are notified of this event,

and instantly each reschedules their transmission using a specially designed backoff algorithm (Spurgeon, 2000b). As part of this algorithm the stations involved each choose a random time interval to schedule the retransmission of the frame, which keeps the stations from making transmission attempts in lock step. In the event of a collision, data is automatically retransmitted in a few microseconds.

High-level network protocols can ensure that the data is correctly received at the destination computer by establishing a reliable data transport service using sequence numbers and acknowledgment mechanisms in the packets.

The heart of the Ethernet system is the Ethernet frame, which is used to deliver data between computers. The frame consists of a set of bits organized into several fields, which include address fields, a variable size data field that carries from 46 to 1,500 bytes of data, and an error checking field that checks the integrity of the bits in the frame to make sure that the frame has arrived intact.

### A1.3.3High-Level Protocols and Ethernet Addresses

Computers attached to an Ethernet generally use high-level protocol software, such as the TCP/IP protocol suite, to communicate. High-level protocols such as IP use a 32-bit addressing system. The high-level IP-based networking software in a given station is aware of its own 32-bit IP address and can read the 48-bit Ethernet address of its network interface, but may only establish communication with other networks by requesting this data from its environment.

IP resolves this by using yet another high-level protocol called the Address Resolution Protocol (ARP) in order to discover the Ethernet addresses of other IP-based stations on the network.

For example, if an IP-based station (station “A”) with IP address 192.168.2.1 wishes to send data over the Ethernet channel to another IP-based station (station “B”) with IP address 192.168.2.2. Station “A” sends a packet to the broadcast address containing an ARP request. The ARP request basically says “Will the station on this Ethernet channel that has the IP address of 192.168.2.2 please tell me what the address of its Ethernet interface is?”

Since the ARP request is sent in a broadcast frame, every Ethernet interface on the network reads it in and hands the ARP request to the networking software running on the station. Only station “B” with IP address 192.168.2.2 will respond, by sending a

packet containing the Ethernet address of station “B” back to the requesting station. Now station “A” has an Ethernet address to which it can send data destined for station “B”, the high-level protocol communication can proceed. (Spurgeon, 2000a and 2000b)

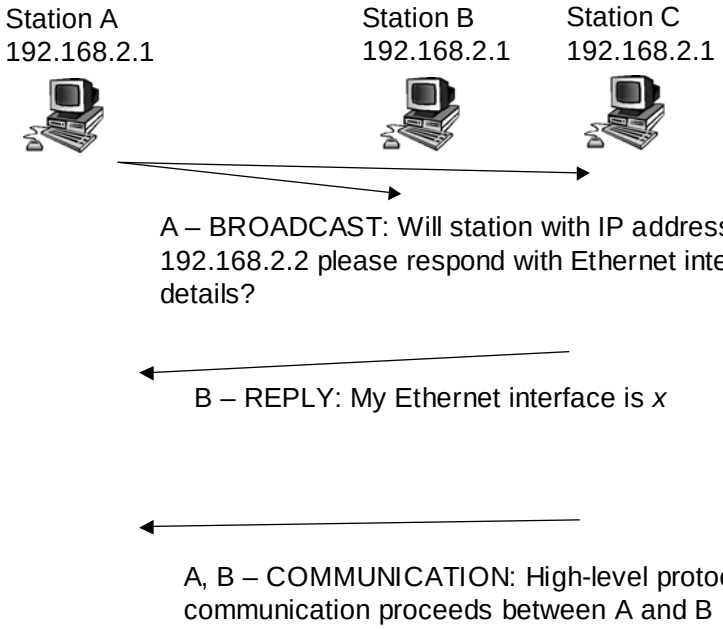


**Figure A.4:** Address Resolution Protocol (ARP) in order to discover the Ethernet address for a given IP address. Station B replies to the ARP request while station C ignores the request

## The Ethernet Frame

The Ethernet frame is at the heart of the Ethernet system, which is used to deliver data between computers. The Ethernet frame consists of several fields, which include address fields, a variable size data field that carries from 46 to 1,500 bytes of data, and an error checking field that checks the integrity of the bits in the frame to make sure that the frame has arrived intact. (Figure A.5)

The first two fields in the frame are each 48-bit addresses, called the destination and source addresses. The IEEE controls the assignment of these addresses by administering a portion of the address field, by providing 24-bit identifiers called "Organizationally Unique Identifiers" (OUIs) to each organisation that wishes to build Ethernet interfaces. The organisation essentially retains the first 24 bits assigned to it in the 48-bit address of the Ethernet hardware. This 48-bit address is also known as the physical address, hard-

ware address, or MAC address.

All Ethernet interfaces look at the first 48-bit field of the frame, which contains the destination address whenever an Ethernet frame is sent onto the shared signal channel. The interfaces compares the destination address of the frame with their own address and reads in the entire frame and delivers it to networking software, whenever a match is found. Otherwise, the network interfaces ignore the frame.

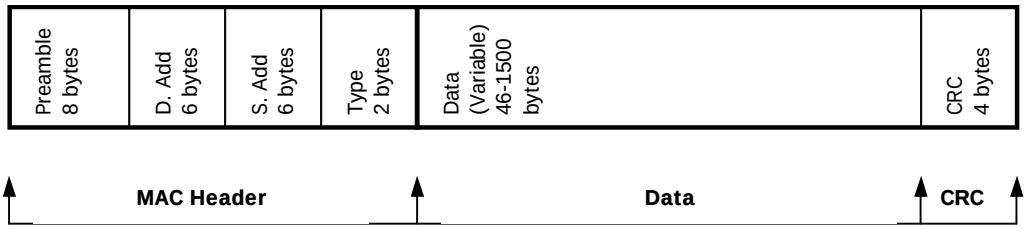


**Figure A.5:** Fields in the Ethernet frame. The main components are the MAC header (22 bytes), a variable data field (46-1500 bytes)

# Source code

## Component Loader

```
/* ********************************************************************
 *
 *   Source File Name   : CmpLoader.cc
 *
 *   Module Name  : CmpLoader
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c) 2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:  Component Loader


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes   (normal system)
 * ======================================================== */
#include <fstream>
#include <dlfcn.h>
#include <string>
#include <map>


/* ========================================================
 *  External Includes        (external toolkits)
 * ======================================================== */


/* ========================================================
 *  Project-wide Includes    (project only)
 * ======================================================== */


/* ========================================================
 *  Module Includes       (module only)
 * ======================================================== */
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/Iface/IfSched.hh"
#include "../../../inc/Iface/IfRouter.hh"


/* ========================================================
 *  Module #DEFINEs
 * ======================================================== */
#define SCHED_SHLIBPATH "../../../lib/libCmpSched.so"
#define ROUTER_SHLIBPATH "../../../lib/libCmpRouter.so"
```

```
#define SCHED_SYMNAME "CreateCmp_Scheduler";
#define ROUTER_SYMNAME "CreateCmp_Router";

/* ------------------------------------------------------
 *  Enumerations & Other Typedefs (defn)
 * ------------------------------------------------------ */
class tTypeRegEnt;
typedef tComponent * (*tCreateCmpSym)();
typedef map<string, tTypeRegEnt> tTypeRegistry;
typedef map<string, tComponent *>  tInstRegistry;


/* ------------------------------------------------------
 *  Classes (forward decl) & Structures (defn)
 * ------------------------------------------------------ */


/* ------------------------------------------------------
 *  Module Functions (decl.)  tIfLoader   (static local only)
 * ------------------------------------------------------ */


/* ------------------------------------------------------
 *  Global Variables (defn.)    (used externally)
 * ------------------------------------------------------ */


/* ------------------------------------------------------
 *  Local Module Variables (defn.)  (static local only)
 * ------------------------------------------------------ */
static string mCmpNameBase("CreateCmp_");



/* ----------------------------------------------------------------------
 * ----------------------------------------------------------------------
 *  Functions & class definitions visible externally
 * ---------------------------------------------------------------------- */


/* ----------------------------------------------------------------------
 * ----------------------------------------------------------------------
 *  Functions & class definitions internal to module
 * ---------------------------------------------------------------------- */
/* Function and class header prototypes */

/* ******************************************************
 * Class : tTypeRegEnt
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class tTypeRegEnt
{
public:
  tTypeRegEnt(string aShlibPath)
    : sShlibPath(aShlibPath),
      sCreateCmpSym(0),
      sInstances(0)
  { }
  ~tTypeRegEnt()
  { }

  tComponent * Create(const string & aTypeName);

private:
```

```
  string    sShlibPath;
  tCreateCmpSym      sCreateCmpSym;
  unsigned int    sInstances;
  void *    sHdl;
};

/* ******************************************************
 * Function : Create()
 * Description : Create instance of CmpType
 * Parameters  : const string & aTypeName
 * Returns  : pointer tComponent *
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tComponent *
tTypeRegEnt::Create(const string & aTypeName)
{
  tComponent * lpCmp;

  // if Instances=0, no shared Lib in mem
  if (sInstances == 0)
    {
      // open dlinker file
      sHdl = dlopen(sShlibPath.c_str(), RTLD_NOW | RTLD_GLOBAL);
      if (sHdl == NULL)
   {
     DBG("oops, bad dlopen for " << sShlibPath << ": " << dlerror());
     throw (string("bad : ") + string(dlerror()) );
   }

      sCreateCmpSym =
   (tCreateCmpSym)dlsym(sHdl, (mCmpNameBase + aTypeName).c_str() );
      // DBG("sCreateCmpSym: " << sCreateCmpSym );
      if (sCreateCmpSym == NULL)
   {
//      DBG("oops, no symbol: " << dlerror());
     throw (string("no symbol: ") + string(dlerror()) );
   }
    }

  lpCmp = sCreateCmpSym();
  ++sInstances;
  return (lpCmp);
}

/* ******************************************************
 * Class : tCmpLoader
 * Description : created from main, loader controls
 *               creation of all other components
 * ******************************************************
 * NOTES:         see tIfLoader.hh
 * ****************************************************** */
class tCmpLoader
  : public tComponent,
    public tIfLoader
{
public:
  tCmpLoader();
  ~tCmpLoader();
  void      Invoke();
  void      ReadTypeRegFile(string aRegFile);
  tComponent *    InstCreate(string aTypeName, string aInstName);
  bool      CreateDefCmp();
  tComponent * CheckAgainstDefCmp(string aTypeName, string aInstName);
  tComponent *    InstQuery(string aInstName);
  void      InstDelete(string aInstName);
```

```
  bool      InternalTask (char * apBuffer, unsigned int aCount);

private:
  tTypeRegistry      sTypeRegister;
  tInstRegistry      sInstRegister;
  string    lRegFile;
  bool         sCreateDefCmp_Flag;

};

/* ******************************************************
 * Function : tCmpLoader()
 * Description : create instance of tCmpLoader
 * Parameters  : none
 * Returns  : CTOR tCmpLoader
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tCmpLoader::tCmpLoader()
  : sCreateDefCmp_Flag(0)
{
  string   lDefLoaderName("loader");

//   cout << "CTOR: tCmpLoader created" << endl;
  // default name for loader instance "loader"
  sInstRegister.insert(make_pair(lDefLoaderName, this));
}


/* ******************************************************
 * Function : ~tCmpLoader()
 * Description : remove instance of tCmpLoader
 * Parameters  : none
 * Returns  : DTOR tCmpLoader
 * ******************************************************
 * NOTES:   THIS REMOVES ALL COMPONENTS FROM sInstRegister
 * ****************************************************** */
tCmpLoader::~tCmpLoader()
{
  for (tInstRegistry::iterator  lInstIter = sInstRegister.begin();
       lInstIter != sInstRegister.end();
       ++lInstIter)
    {
      delete (lInstIter->second);
    }

}


/* ******************************************************
 * Function : Invoke()
 * Description : invoke tCmpLoader
 * Parameters  : none
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpLoader::Invoke()
{
  DBG("hello from tCmpLoader");
}


/* ******************************************************
 * Function : ReadTypeRegFile()
```

```
 * Description : readS local file to populate sTypeRegister
 * Parameters  : string aRegFile
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpLoader::ReadTypeRegFile(string aRegFile)
{
  string lName;
  string lPath;
  ifstream  lReg(aRegFile.c_str());

  while(lReg >> lName >> lPath)
    {
//       cout << "inserting " << lName << ", " << lPath << endl;
      sTypeRegister.insert(make_pair(lName, tTypeRegEnt(lPath)));

    }

}


/* ******************************************************
 * Function : InstCreate()
 * Description : Create an instances of a given Cmp
 * Parameters  : string aTypeName, string aInstName
 * Returns  : tComponent *
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tComponent *
tCmpLoader::InstCreate(string aTypeName, string aInstName)
{
  tTypeRegistry::iterator  lTypeIter;
  tInstRegistry::iterator  lInstIter;
  tComponent *       lpComponent;

//   if (sCreateDefCmp_Flag != 1)
//     {
//       CreateDefCmp();
//     }

//   // expand to include other default components
//   if ((aTypeName == "Scheduler") ||
//       (aTypeName == "Router") )
//     {
// //       lpComponent = CheckAgainstDefCmp(aTypeName, aInstName);
//     }
//   else
//     {

      // ensure no duplicate instance is created
      lInstIter = sInstRegister.find(aInstName);
      if (lInstIter != sInstRegister.end())
   {
     // Found duplicate
     DBG("Duplicate entry for " << aInstName);
     return (NULL);
   }

      lTypeIter = sTypeRegister.find(aTypeName);
      if (lTypeIter == sTypeRegister.end())
   {
     // Not found
     DBG("didn't find entry for " << aTypeName);
```

```
      return (NULL);
    }

       //update sInstRegister
       lpComponent = lTypeIter->second.Create(aTypeName);
       if (lpComponent == NULL)
    {
      throw;
    }
       sInstRegister.insert(make_pair(aInstName, lpComponent));

//     }

  return (lpComponent);
}

/* ******************************************************
 * Function : CreateDefCmp
 * Description : creates default components
 * Parameters  : -
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpLoader::CreateDefCmp()
{
  tComponent *       lpSched;
  tComponent *       lpRouter;

  string       lDefSchedInst("sched");
  string       lDefRouterInst("router");

  string          lSchedCmp("Scheduler");
  string          lRouterCmp("Router");

  string          lSchedPath("/view/hasanain_tcs01_00.01/lib/libCmpSched.so");
  string          lRouterPath("/view/hasanain_tcs01_00.01/lib/libCmpRouter.so");

  tComponent *       (*lpCreateSchedSym)();
  tComponent *       (*lpCreateRouterSym)();

  string          lCreateSchedSymName("CreateCmp_Scheduler");
  string          lCreateRouterSymName("CreateCmp_Router");

  void *       lpSchedHdl;
  void *       lpRouterHdl;

  bool            Sched_CreateFlag = 0;
  bool            Router_CreateFlag = 0;

  tTypeRegistry::iterator  lTypeIter;
  tTypeRegistry::iterator  lTypeIter2;


  // create 'Scheduler'

  DBG("creating default Scheduler ");
  lpSchedHdl = dlopen(lSchedPath.c_str(), RTLD_NOW | RTLD_GLOBAL);
  if ( lpSchedHdl != NULL)
    {
      lpCreateSchedSym =
   (tComponent * (*)())dlsym(lpSchedHdl, lCreateSchedSymName.c_str() );

      if (lpCreateSchedSym == NULL)
   {
```

```
      DBG(" bad symbol: " << dlerror());
    }

       lpSched = lpCreateSchedSym();
       if (lpSched == NULL)
    {
      throw (string("couldn't create scheduler component"));
    }

       // now update sInstRegister and sTypeRegister with new entry
       sInstRegister.insert(make_pair(lDefSchedInst, lpSched));
       lTypeIter = sTypeRegister.find(lSchedCmp);
       if (lTypeIter == sTypeRegister.end())
    {
      // Not found
     sTypeRegister.insert(make_pair(lSchedCmp, lSchedPath));
    }

       DBG("made sched");
       Sched_CreateFlag = 1;
     }
   else
     {
       DBG(" bad dlopen: " << dlerror());
     }

   // repeat the same for 'Router'

   DBG("creating default Router ");
   lpRouterHdl = dlopen(lRouterPath.c_str(), RTLD_NOW | RTLD_GLOBAL);

   if ( lpRouterHdl != NULL)
     {
       lpCreateRouterSym = (tComponent * (*)())dlsym(lpRouterHdl, lCreateRouterSym-
Name.c_str() );
       if (lpCreateRouterSym == NULL)
    {
      DBG("could not create router symbol");
    }

       lpRouter = lpCreateRouterSym();
       if (lpRouter == NULL)
    {
      throw (string("couldn't create router component"));
    }

       DBG("made router");

       // now update sInstRegister with new entry
       sInstRegister.insert(make_pair(lDefRouterInst, lpRouter));
       lTypeIter2 = sTypeRegister.find(lRouterCmp);
       if (lTypeIter == sTypeRegister.end())
    {
      // Not found
     sTypeRegister.insert(make_pair(lRouterCmp, lRouterPath));
    }
       Router_CreateFlag = 1;
     }
   else
     {
       DBG("could not open router component lib");
     }

 if ( (Sched_CreateFlag) && (Router_CreateFlag))
   {
     sCreateDefCmp_Flag = 1;
```

```
    return (true);
   }
 else
   {

    DBG("failed creating default components");
    return (false);
   }
}

/* ******************************************************
 * Function : CheckAgainstDefCmp
 * Description : avoids duplication of defaults
 * Parameters  : string aTypeName, string aInstName
 * Returns  : tComponent *
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tComponent *
tCmpLoader::CheckAgainstDefCmp(string aTypeName, string aInstName)
{
  string lDefSchedInst("sched");
  string lDefRouterInst("router");

  string    lSchedCmp("Scheduler");
  string    lRouterCmp("Router");

  tComponent * lpComponent;


  if (aTypeName == lRouterCmp)
    {
      lpComponent = InstQuery(lDefRouterInst);
      if (lpComponent != NULL)
   DBG("handle to default 'router' returned, " << aInstName << " ignored");
    }
  else
    {
      if (aTypeName == lSchedCmp)
   {
     lpComponent = InstQuery(lDefSchedInst);
     if (lpComponent != NULL)
       DBG("handle to default 'sched' returned, " << aInstName << " ignored");
   }
    }


  return lpComponent;

}

/* ******************************************************
 * Function : InstQuery
 * Description : Queries loader to find pointer to components
 * Parameters  : string aInstName
 * Returns  : tComponent *
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tComponent *
tCmpLoader::InstQuery(string aInstName)
{
  tInstRegistry::iterator  lInstIter;

  lInstIter = sInstRegister.find(aInstName);
  if (lInstIter == sInstRegister.end())
```

```
    {
      // none found in InstRegister
      DBG("No entry for " << aInstName << " found");
      return (NULL);
    }
  else
    {
      return (lInstIter->second);
    }
}

/* ******************************************************
 * Function : InstDelete
 * Description : Deletes the instance of the component
 * Parameters  : string aInstName
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpLoader::InstDelete(string aInstName)
{
  tInstRegistry::iterator  lInstIter;

  // find instance to delete
  lInstIter = sInstRegister.find(aInstName);
  if (lInstIter == sInstRegister.end())
    {
      // none found
      DBG("Found no instances of " << aInstName << "to delete");
    }
  else
    {
      // delete component & remove InstRegistry entry
      delete (lInstIter->second);
      sInstRegister.erase(lInstIter);
    }
}

/* ******************************************************
 * Function : InstDelete
 * Description : Deletes the instance of the component
 * Parameters  : string aInstName
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpLoader::InternalTask (char * apBuffer, unsigned int aCount)
{
  // do nothing, for now
  return (true);
}

/* ======================================================================
 * ======================================================================
 *  Entry Points
 * ====================================================================== */


extern "C"
tComponent *
CreateCmp_Loader()
{
  return (new tCmpLoader);
}
```

```
/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

# Component EthGen

```
/* ********************************************************************
 *
 *   Source File Name   : CmpEthGen.cc
 *
 *   Module Name  : CmpEthGen
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:   Sequence of events for copying data into memory
 *    A temporary memory is used to make a copy just before
 *    generation.
 *
 * Config() builds the memory
 * PreRunCheck() makes a temporary map of the above memory before gen
 *
 * Config()
 * {
 *    sSeqStoreArray[8] is used to store arrays of FrameVectors
 *    sFrameVector[8]   stores pointers to frames as a vector
 *    sEthGenSeq[8]     is used to contain sequence specific
 *             parameters (like timeInt) as this
 *             information is not stored in sSeqStoreArray
 *             Additionally sEthGenSeq also contains
 *             sFrameVector[] as ptr
 *
```

```
 * }
 *
 * PreRunCheck()
 * {
 *    sChannelArray[8]  copies everything from sEthGenSeq[8]
 *
 *             sChannelArray[] is used for generation
 * }
 *
 *  END OF NOTES
 * ========================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes   (normal system)
 * ======================================================== */
#include <pthread.h>
#include <dlfcn.h>
#include <string>
#include <map>
#include <vector>
#include <fstream>
#include <unistd.h>

#include <sys/time.h>

#include <linux/if_ether.h>   /* The L2 protocols */
#include <linux/if_packet.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/if.h>
#include <netinet/in.h>
#include <errno.h>
#include <string>

#include <iomanip>

/* ========================================================
 *  External Includes        (external toolkits)
 * ======================================================== */


/* ========================================================
 *  Project-wide Includes    (project only)
 * ======================================================== */


/* ========================================================
 *  Module Includes       (module only)
 * ======================================================== */
#include "../../inc/Iface/IfEthGen.hh"
#include "../../inc/EthGenBuf.hh"
#include "../../inc/Component.hh"
#include "../../inc/ThreadBase.hh"

#include "../../inc/EthGenFrame.hh"
#include "../../inc/EthGenSeq.hh"

/* ========================================================
 *  Module #DEFINEs
 * ======================================================== */
#define TRANSITIONS  12
#define MAX_FRAME_RATE  8127
```

```
/* ========================================================
 *  Enumerations & Other Typedefs (defn)
 * ======================================================== */
// typedef long   ChannelTimeDelay[NO_OF_CHANNELS];

/* ========================================================
 *  Classes (forward decl) & Structures (defn)
 * ======================================================== */


/* ========================================================
 *  Module Functions (decl.)    (static local only)
 * ======================================================== */


/* ========================================================
 *  Global Variables (defn.)    (used externally)
 * ======================================================== */


/* ========================================================
 *  Local Module Variables (defn.) (static local only)
 * ======================================================== */


/* ========================================================================
 * ========================================================================
 *  Functions & class definitions visible externally
 * ======================================================================== */




/* ========================================================================
 * ========================================================================
 *  Functions & class definitions internal to module
 * ======================================================================== */

/* Function and class header prototypes */

/* ******************************************************
 * Class : tThreadData
 * Description : thread data class
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class tThreadData_SeqTimer
  :public tThreadData
{
public:
  tThreadData_SeqTimer(tEthGenSeq * apSeq)
    : spSeq(apSeq)
  { }

  tEthGenSeq *    GetPtrToSeq()
    {
      return (spSeq);
    }

private:
  tEthGenSeq *    spSeq;
};
```

```
/* ******************************************************
 * Class : tCmpEthGen
 * Description : EthGen class
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class tCmpEthGen
  : public tComponent,
    public tIfEthGen,
    private tThreadBase
{
public:
  tCmpEthGen()
    : sSock(-1),
      sIfEthernet("eth1"),
      sGenThreadLoop(false),
      sFrameCount_All(0)
  {
    // Create socket
    sSock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if (sSock == -1)
      {
   throw (string ("Could not create socket - requires root access "));
      }

    // set initial state as 'Not configured' at creation time
    sContext.sState = eDssNotInit;

    ChangeState(eDseInit);

    // Init mutex
    pthread_mutex_init(&sContext.sStateMutex, NULL);

    DBG("created CmpEthGen" );

  }

  ~tCmpEthGen()
  {

    // bring system state to halt

    // Destroy mutex and close socket
    pthread_mutex_destroy(&sContext.sStateMutex);

    close(sSock);

    DBG("deleted CmpEthGen" );
  }

  const string       QueryState() const;

  bool         SetInterface(string aInterfaceName);
  const string    QueryInterface() const;

  bool         Config(const tChannelArray * apChannelArray);
  bool         PreRunCheck();
  bool         StartEngine();
  bool         StopEngine();

  bool               Generate();
  bool         StopGenerate();

  bool         InsertChannel(unsigned int    aChannelNo,
                 tEthGenSeq *   aEthNewSeq);
```

```
bool             ModifyChannel(unsigned int  aChannelNum,
                   tEthGenSeq *   aEthNewSeq);
bool         RemoveChannel(unsigned int    aChannelNum);

bool         InsertSeq();
bool         ModifySeq();
bool         RemoveSeq();

bool         InsertFrame();
bool         ModifyFrame();
bool         RemoveFrame();

bool         EraseIntMemory();
bool         EraseTempMemory();


// set and query Pdu len
bool         SetPDU_Len(unsigned int aBufLen,
             unsigned int   aChannelNum,
             unsigned int   aPduNum);

const unsigned int QueryPDU_Len(unsigned int aChannelNum,
               unsigned int aPduNum);


// assign rate to specific channel
bool         SetRate_FramePerSec(unsigned int aFrameRate,
                unsigned int aChannelNum);
bool         SetRate_Mbps(unsigned int aRate_Mb,
               unsigned int aChannelNum);

// assign time/loop limit to specific channel
bool         SetLimit_Time(long      aTime_Sec,
                unsigned int   aChannelNum);
bool         SetLimit_Loops(long     aLoops,
                unsigned int  aChannelNum);

const unsigned int QueryRate_FramePerSec(unsigned int aChannelNum) const;
const unsigned int QueryRate_BitsPerSec(unsigned int aChannelNum) const;

// query stats from channel
const long long QueryStats_FrameSent(unsigned int aChannelNum) const;
const timeval   QueryStats_TimeInterval(unsigned int aChannelNum) const;

// modify destination buffer, given src buffer, offset and len
int       ModifyBuffer(unsigned int aOffSet,
               unsigned int aBufLen,
               unsigned char * apDestBuf,
               unsigned char * apSrcBuf);

bool         ModifyFrameBuffer(unsigned int   aOffSet,
               unsigned int    aBufLen,
               unsigned char * apSrcBuf,
               unsigned int aChannelNum,
               unsigned int aFrameNum);

bool         QueryFrameBuffer(unsigned int    aOffSet,
              unsigned int  aBufLen,
              unsigned char * apBuf,
              unsigned int  aChannelNum,
              unsigned int  aFrameNum);

void         Invoke()
{ DBG("invoked CmpEthGen");  }

unsigned int ConvertHexToInt(char * aInb, int aLen, char * aOutb);
```

```
  unsigned int ConvertIntToHex(char * aInb, int aLen,
            char * aOutb);


private:

  // Socket state
  int    sSock;

  // default ethernet interface
  string    sIfEthernet;

  // channel array[] is an array of ptrs to Seqs (temp mem)
  tChannelArray      sChannelArray;

  // array for sequences
  tEthGenSeq      sEthGenSeq[NO_OF_CHANNELS];

  // array of vectors (of tEthGenFrame ptrs) to be used in memory
  tEthGenFrameVec sFrameVector[NO_OF_CHANNELS];

  // actual storage of all channel/seq/frame internal to EthGen
  tSeqStoreArray  sSeqStoreArray;

  // flag for looping for gen thread
  bool         sGenThreadLoop;

  // number of frames to be sent out
  unsigned int       sFrameCount_All;

  // generation thread pointer to avoid mem conflicts
  tChannelArray * spActiveArrayPtr;

  // GENERATION TIME/LOOPS/INF VARIABLES
  long         sGenLoopLimit[NO_OF_CHANNELS];
  long         sGenTimeLimit[NO_OF_CHANNELS];

  // VARIABLES, STATS ON GEN

  // count of frames sent out from each channel
  long long    sFramesSent_Channel[NO_OF_CHANNELS];

  // generator start time
  long      sGenStartTime_sec;
  long      sGenStartTime_usec;

  // generator finish times
  long      sGenFinTime_sec[NO_OF_CHANNELS];
  long      sGenFinTime_usec[NO_OF_CHANNELS];

  // ChannelRate actual values to which they are set -
  // many rate in fps == one TimeDelay in usecs
  // hence this arrangement
  unsigned int sChannelRate_fps[NO_OF_CHANNELS];


  // count for Ethernet interface failure to send data - returns (-1)
  long   long  sRtlDriverFailCount;

  // Component states
  typedef enum {
    eDssNotInit,
    eDssInit,
    eDssNotCfg,
    eDssCfging,
    eDssHalt,
```

```
  eDssPreRC,
  eDssNotRunning,
  eDssRunning,
  eDssWaitStop
} tEthGenStates;

// Component events
typedef enum {
  eDseInit,
  eDseConfig,
  eDseBadCfg,
  eDseGoodCfg,
  eDseStartPreRC,
  eDsePreRCFail,
  eDsePreRCPass,
  eDseStop,
  eDseGen,
  eDsePostGen,
  eDseEndGen
} tEthGenEvents;


// Component context across threads

struct {
  // State stuff
  tEthGenStates sState;
  pthread_mutex_t  sStateMutex;
} sContext;

// generation thread main
int    sGenThrCount;

tThreadData * ThrMain(tThreadData * apData);

// Main thread event ops
bool      DoInit();
bool      DoConfig();
bool      DoBadCfg();
bool      DoGoodCfg();
bool      DoReConfig();
bool      DoStart();

bool      DoBadPreRC();
bool      DoGoodPreRC();

bool      DoGenerate();
bool      DoRGStop();
bool      DoNRStop();
bool      DoRGEndGen();
bool      DoWSEndGen();
bool      DoNothing();


typedef struct {
  // state transition lookup
  tEthGenStates sCurState;
  tEthGenEvents sEvent;
  tEthGenStates sNewState;
  bool (tCmpEthGen::*sFunctionCall)();
} tStateTrParmEnt;

static tStateTrParmEnt sStateTrParmTbl[TRANSITIONS];

typedef   bool (tCmpEthGen::*sFunctionCall)();
```

```
  // changing states
  bool   ChangeState(tEthGenEvents aEthGenEvent);

};

tCmpEthGen::tStateTrParmEnt tCmpEthGen::sStateTrParmTbl[TRANSITIONS] = {
  {eDssNotInit,      eDseInit,   eDssNotCfg, &tCmpEthGen::DoInit},
  {eDssNotCfg,    eDseConfig, eDssCfging, &tCmpEthGen::DoConfig},
  {eDssCfging,    eDseBadCfg, eDssNotCfg, &tCmpEthGen::DoBadCfg},
  {eDssCfging,    eDseGoodCfg,   eDssHalt,   &tCmpEthGen::DoGoodCfg},
  {eDssHalt,      eDseConfig,    eDssCfging, &tCmpEthGen::DoReConfig},

  {eDssHalt,      eDseStartPreRC, eDssPreRC,    &tCmpEthGen::DoStart},
  {eDssPreRC,           eDsePreRCFail,  eDssHalt,       &tCmpEthGen::DoBadPreRC},
  {eDssPreRC,           eDsePreRCPass,  eDssNotRunning, &tCmpEthGen::DoGoodPreRC},

  {eDssNotRunning,   eDseGen, eDssRunning,   &tCmpEthGen::DoGenerate},
  {eDssRunning,      eDsePostGen,   eDssNotRunning,   &tCmpEthGen::DoNothing},
  {eDssNotRunning,   eDseStop,   eDssWaitStop,  &tCmpEthGen::DoRGStop},
  {eDssWaitStop,  eDseEndGen, eDssHalt,   &tCmpEthGen::DoWSEndGen}
};


/* *******************************************************
 * Function : QueryState
 * Description : query the present system state
 * Parameters  : none
 * Returns  : string
 * *******************************************************
 * NOTES:
 * ******************************************************* */
const string
tCmpEthGen::QueryState() const
{
  string    lDsState;

  switch(sContext.sState)
    {
        case eDssNotInit:
    lDsState = "Not Initialised";
    break;

   case eDssNotCfg:
     lDsState = "Not Configured";
     break;

   case eDssCfging:
     lDsState = "Configuring";
     break;

   case eDssHalt:
     lDsState = "Halt";
     break;

   case eDssPreRC:
     lDsState = "PreRun Check";
     break;

        case eDssNotRunning:
     lDsState = "Not Running";
     break;

   case eDssRunning:
     lDsState = "Running";
     break;
```

```
   case eDssWaitStop:
     lDsState = "Waiting to Stop";
     break;

        default:
          lDsState = "unknown";
           break;

    }

  // DBG("Current Engine state = " << lDsState);

  return lDsState;
}


/* ******************************************************
 * Function : SetInterface
 * Description : sets ethernet interface to <name>
 * Parameters  : string
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::SetInterface(string aInterfaceName)
{
  // make a copy of interface name in case call fails
  string lInterfaceOld = QueryInterface();

  if (lInterfaceOld == aInterfaceName)
    {
      DBG("'" << aInterfaceName << "' already set");
      return (true);
    }
  else
    {
      sIfEthernet = aInterfaceName;

      // DoInit() again to remove previous interface
      if (!DoInit())
   {
     // reset old interface name to sIfEthernet
     sIfEthernet = lInterfaceOld;

     return (false);
   }
    }

  return (true);
}

/* ******************************************************
 * Function : QueryInterface
 * Description : gets ethernet interface name currently active
 * Parameters  : none
 * Returns  : string
 * ******************************************************
 * NOTES:
 * ****************************************************** */
const string
tCmpEthGen::QueryInterface() const
{
  if (sIfEthernet == "eth0")
    {
```

```
      DBG("eth0 must not be used");
    }

  return (sIfEthernet);
}



/* ******************************************************
 * Function : Config
 * Description : configure
 * Parameters  : arNetNames
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::Config(const tChannelArray * apChannelArray)
{
  tEthGenFrameVec *     lpFrameVec = NULL;

//   tEthGenFrameVec *     lpFrameVec2 = NULL;

  long            lTimeDelay = 0;

  // lock out state to 'Configuring' before doing anything
  if (!ChangeState(eDseConfig))
    {
      DBG("could change state to configuring");
    }

  // erase all in internal memory before proceeding
  if (!EraseIntMemory())
    {
      DBG("could not erase internal memory");
      return (false);
    }

  // set all channels to generate for infinite time by default
  for (unsigned int lChannelNum = 0; lChannelNum < NO_OF_CHANNELS; ++lChannelNum)
    {
      sGenLoopLimit[lChannelNum] = -1;
      sGenTimeLimit[lChannelNum] = -1;
    }

  // copy all into internal memory

  // find channel 0 - 7 and populate Frame Vectors
  for (unsigned int lChannelNo = 0; lChannelNo < NO_OF_CHANNELS;
       ++lChannelNo)
    {
      // get frameVec for non-null entries
      if ((*apChannelArray)[lChannelNo] != NULL)
   {
     lpFrameVec = (*apChannelArray)[lChannelNo]->GetFrameVec();

     if (lpFrameVec != NULL)
       {
         DBG("Stored channel[" << lChannelNo << "] into internal memory");

         // for each frame in vector, copy into internal mem
         for (tEthGenFrameVec::iterator   lFrame = lpFrameVec->begin();
         lFrame != lpFrameVec->end(); ++lFrame)
      {
        // internal storage config
        (sSeqStoreArray[lChannelNo]).push_back(tEthGenFrame((**lFrame).GetFrameName(),
```

```
                              (**lFrame).GetEthGenBuf_Len(),
                              (**lFrame).GetEthGenBuf_Buf(),
                              (**lFrame).GetEthGenBuf_LoopCount()));

          DBG("(**lFrame).GetFrameName() = " << (**lFrame).GetFrameName());
        }
         }
     }
      }

  // copy all sequence specific parameters, required by sChannelArray[] later in PreConfig
  for (unsigned int lGenChannel = 0; lGenChannel < NO_OF_CHANNELS; ++lGenChannel)
    {
      if ((*apChannelArray)[lGenChannel] != NULL)
     {
       // for each frame in sSeqStoreArray vector, copy into internal mem
       for (tSeqStore_Vec::iterator   lStoredFrame = sSeqStoreArray[lGenChannel].begin();
            lStoredFrame != sSeqStoreArray[lGenChannel].end(); ++lStoredFrame)
         {
           DBG("Copying '" << lStoredFrame->GetFrameName() << "' ");
           sFrameVector[lGenChannel].push_back(lStoredFrame);
         }

       lTimeDelay = (*apChannelArray)[lGenChannel]->GetTimeDelay();

       sEthGenSeq[lGenChannel].SetFrameVec(&sFrameVector[lGenChannel]);
       sEthGenSeq[lGenChannel].SetTimeDelay(lTimeDelay);

       DBG("Time delay " << sEthGenSeq[lGenChannel].GetTimeDelay()<< " into mem");
       DBG("------------- Copied Channel[" << lGenChannel << "] -----------------");
     }
      }

  // now copy all non-null pointers to temporary buffer

      // now done in PreRunCheck() stage

  // if not ready, set state to 'not configured'
  if(!ChangeState(eDseGoodCfg))
    {
      DBG("could not advance state to halt");
      ChangeState(eDseBadCfg);
      return (false);
    }

  return (true);
}


/* ******************************************************
 * Function : PreRunCheck
 * Description : do prerun check and advance to state to
 *                'NotRunning'
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES: invoke thread and provide input data to process on
 * ****************************************************** */
bool
tCmpEthGen::PreRunCheck()
{
  tEthGenFrameVec *  lpFrameVec = NULL;

  DBG("into PreRunCheck ");

  // set state to 'PreRunCheck'
```

```
  if (!ChangeState(eDseStartPreRC))
    {
      DBG("could not advance state to 'PreRC'");
      return (false);
    }

  // erase all in temp memory before proceeding
  if (!EraseTempMemory())
    {
      DBG("could not erase temporary memory");
      return (false);
    }


  // copy all PDUs/Seq to temp buffer
  for (unsigned int lGenChannel = 0; lGenChannel < NO_OF_CHANNELS; ++lGenChannel)
    {
      DBG("Checking store seq[" << lGenChannel << "] time delay = '"
     << sEthGenSeq[lGenChannel].GetTimeDelay() << " usecs");

      // copy only valid channels to temp memory
      lpFrameVec = sEthGenSeq[lGenChannel].GetFrameVec();

      if (lpFrameVec != NULL)
   {
     if (lpFrameVec->size() > 0)
       {
         DBG("Vector size is " << lpFrameVec->size() << " frames");
         sChannelArray[lGenChannel] = &sEthGenSeq[lGenChannel];

         // keep count of total number of frames to generate
         lpFrameVec = sChannelArray[lGenChannel]->GetFrameVec();

         sFrameCount_All += lpFrameVec->size();

       }
   }
      else
   {
     // explicitly set to null - may have already been done in EraseTempMem()
     sChannelArray[lGenChannel] = NULL;
   }
    }

  DBG("sFrameCount_All = " << sFrameCount_All);

  // point to new array
  spActiveArrayPtr = &sChannelArray;



  // advance state to 'NotRunning'
  if (!ChangeState(eDsePreRCPass))
    {
      DBG("could not advance state to 'NotRunning'");
      ChangeState(eDsePreRCFail);
      DBG("resetting to halt state");
      return (false);
    }

  return (true);
}
```

```
/* ******************************************************
 * Function : StartEngine
 * Description : start generating engine
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES: invoke thread and provide input data to process on
 * ****************************************************** */
bool
tCmpEthGen::StartEngine()
{
  // Start engine after checking for correct state
  if (QueryState() == "Not Running")
    {
      // advance state to 'eDseGen' to kick off generation
      if (ChangeState(eDseGen))
   {
     // now kick of generation thread
     if (!ThrGo(NULL))
       {
         DBG(":-P Could not create thread");
         return (false);
       }
     else
       {
         DBG("Thread running ..... ");
       }
   }
      else
   {
     DBG("could not advance state to 'Running'");
     return (false);
   }
    }
  else
    {
      DBG("! Incorrect state to start Engine");
      return (false);
    }

  return (true);
}


/* ******************************************************
 * Function : StopEngine
 * Description : stop generating engine
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::StopEngine()
{
  tEthGenFrameVec *     lpFrameVec = NULL;

  timeval   lGetTime;
//   long      lFinishTime_sec;
//   long      lFinishTime_usec;

  long      lDeltaTime_sec;
  long      lDeltaTime_usec;

  float     lTimeInterval;
```

```
  float     lBwUtilisation;

  unsigned int  lBytesInSeq_Average[NO_OF_CHANNELS];
  unsigned int  lBytesInSeq[NO_OF_CHANNELS];

  int    lSeqSize[NO_OF_CHANNELS];

  long      lTotalBits[NO_OF_CHANNELS];

//   // stop engine only if running
//   if (QueryState() != "Running")
//     {
//       DBG("Engine not running");
//       return (false);
//     }

  gettimeofday(&lGetTime, 0);

  DBG("END:     To --> " << lGetTime.tv_sec << "." << lGetTime.tv_usec);

  for (int lStatsCount = 0; lStatsCount < NO_OF_CHANNELS; ++lStatsCount)
    {
      DBG("pre  stop engine finish time [" << lStatsCount << "] = " << sGenFin-
Time_sec[lStatsCount]
     << "." << sGenFinTime_usec[lStatsCount]);
    }

  // provision to complete whole sequence when engine is stopped by user

  // case 1: loop limit not reached
  //     forced stop, hence set loop limit to 1 to allow complete generation
  //     ideally must be set for sequences, not PDUs
  for (unsigned int lChannelNo = 0; lChannelNo < NO_OF_CHANNELS;
      ++lChannelNo)
    {
      if (sChannelArray[lChannelNo] != NULL)
   {
     // for all channels set loop limit to 1
     if (sChannelArray[lChannelNo]->GetFrameVec()->size() > 1)
       {
         // let ThrMain handle the rest
         sGenLoopLimit[lChannelNo] = 1;
       }
     else
       {
         // for PDUs
           // check thread existence
         if (sGenThreadLoop)
      {
        sGenThreadLoop = false;

        // allow thread to cleanup
        usleep(200000);
      }
          else
      {
        DBG("generation loop already finished");
      }
       }
   }
    }

  // sChannel[] is made NULL in ThrMain to deactivate
  while (sGenThreadLoop)
    {
      // wait until ThrMain terminates
```

```
      //cout << ". ";
    }

  // case 2: time limit not reached
  //     forced stop, ignore time


  // reset all channels (generate for infinite time by default)
  for (unsigned int lChannel = 0; lChannel < NO_OF_CHANNELS; ++lChannel)
    {
      sGenLoopLimit[lChannel] = -1;
    }

  // go to state 'waittostop'
  if (!ChangeState(eDseStop))
    {
      DBG("could not advance state to 'waiting to stop'");
      return (false);
    }

  if (!ThrJoin())
    {
      DBG("! Could not join thread");
    }
  else
    {
      for (int lStatsCount = 0; lStatsCount < NO_OF_CHANNELS; ++lStatsCount)
   {
     if ((sGenFinTime_sec[lStatsCount] == 0) && (sGenFinTime_usec[lStatsCount] == 0))
       {
         // store finish times for stats later
         sGenFinTime_sec[lStatsCount] = lGetTime.tv_sec;
         sGenFinTime_usec[lStatsCount] = lGetTime.tv_usec;
       }
   }

      // get stats on all channels
      for (unsigned int lStatsCount = 0; lStatsCount < NO_OF_CHANNELS; ++lStatsCount)
   {
     if (sChannelArray[lStatsCount])
       {
         // DBG(setw(10) << lGetTime.tv_sec << "." << setw(6) <<   lGetTime.tv_usec
         // << " [" << lStatsCount << "] = " << sFramesSent_Channel[lStatsCount]);

         lpFrameVec = sChannelArray[lStatsCount]->GetFrameVec();

         lSeqSize[lStatsCount] = lpFrameVec->size();
         DBG("SeqSize[" << lStatsCount << "] " << lSeqSize[lStatsCount]);


         // for each frame in vector, copy into internal mem
         for (tEthGenFrameVec::iterator   lFrame = lpFrameVec->begin();
         lFrame != lpFrameVec->end(); ++lFrame)
      {
        lBytesInSeq[lStatsCount] += (**lFrame).GetEthGenBuf_Len();
      }

         lBytesInSeq_Average[lStatsCount] = lBytesInSeq[lStatsCount]/lSeqSize[lStatsCount];
         DBG("Bytes in seq[" << lStatsCount << "] " << lBytesInSeq[lStatsCount]);
         DBG("Average size in seq[" << lStatsCount << "] " << lBytesIn-
Seq_Average[lStatsCount]);

         if (sGenFinTime_usec[lStatsCount] < sGenStartTime_usec)
      {
        // DBG("** delay time adjustments");
        lDeltaTime_usec = (sGenFinTime_usec[lStatsCount] + 1000000) - sGenStartTime_usec;
```

```
        lDeltaTime_sec = sGenFinTime_sec[lStatsCount] - sGenStartTime_sec - 1;

      }
         else
      {
        lDeltaTime_usec = sGenFinTime_usec[lStatsCount] - sGenStartTime_usec;
        lDeltaTime_sec = sGenFinTime_sec[lStatsCount] - sGenStartTime_sec;
      }

         lTimeInterval = lDeltaTime_sec*1000000 + lDeltaTime_usec;

         // calculate total bytes from average
         lTotalBits[lStatsCount] = (lBytesInSeq_Average[lStatsCount] * 8);

         DBG("bits out " << (lBytesIn-
Seq_Average[lStatsCount]*sFramesSent_Channel[lStatsCount]*8));
         DBG("time " << lTimeInterval);

         lBwUtilisation = ((lBytesIn-
Seq_Average[lStatsCount]*sFramesSent_Channel[lStatsCount]*8)/lTimeInterval);

         DBG("Frames sent [" << sFramesSent_Channel[lStatsCount] << "] in " << lTimeInter-
val/1000000 << " secs, Channel '" << lStatsCount << "' rate = "
        << (sFramesSent_Channel[lStatsCount]*1000000)/lTimeInterval << " fps ");

         DBG("BandWidth utilisation (100BASE-T) Channel[" << lStatsCount << "] " <<
lBwUtilisation << " % ");
       }
   }

      for (int lStatsCount = 0; lStatsCount < NO_OF_CHANNELS; ++lStatsCount)
   {
     DBG("post stop engine finish time [" << lStatsCount << "] = " << sGenFin-
Time_sec[lStatsCount]
         << "." << sGenFinTime_usec[lStatsCount]);
   }

      if (sRtlDriverFailCount != 0)
   {
     DBG("**** Rtl driver failed " << sRtlDriverFailCount << " times ");
   }
    }

  // bring state back to Halt
  if (!ChangeState(eDseEndGen))
    {
      DBG("Could not return state back to 'halt'");
      return (false);
    }

  return (true);
}


/* ******************************************************
 * Function : Generate
 * Description : start generating frames
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::Generate()
{
  if (!ChangeState(eDseGen))
```

```
    {
      return (false);
    }
  return (true);
}

/* *******************************************************
 * Function : InsertChannel
 * Description : introduce a new channel to system
 * Parameters  : none
 * Returns  : bool
 * *******************************************************
 * NOTES:
 * ******************************************************* */
bool
tCmpEthGen::InsertChannel(unsigned int   aChannelNo,
          tEthGenSeq * aEthNewSeq)
{

  // lock out state to 'Configuring' before doing anything
  if (!ChangeState(eDseConfig))
    {
      DBG("could change state for Reconfig");
    }

  // check for existing duplicate channels

  // if gen thread is not running

      // remove duplicate

      // insert new channel element to 'sChannel'
      sChannelArray[aChannelNo] = aEthNewSeq;

      if (sChannelArray[aChannelNo] == NULL)
   {
     DBG("! Could not insert channel '"
         << aChannelNo << "'");

     return (false);
   }


  // else

     // copy contents to new location and insert element


     // update flags of old location


  // set state back to halt
  if (!ChangeState(eDseGoodCfg))
    {
      DBG("could not reset to 'halt'");
      return (false);
    }

  // return
      return (true);
}

/* *******************************************************
 * Function : ModifyChannel
 * Description :
 * Parameters  : none
```

```
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::ModifyChannel(unsigned int   aChannelNum,
          tEthGenSeq * aEthNewSeq)
{

  // lock out state to 'Configuring' before doing anything
  if (!ChangeState(eDseConfig))
    {
      DBG("could change state for Reconfig");
    }

  // check for for the existing channel

  // if it exists, remove previous and insert this one

  // if it does not exist, create new channel

  // set state back to halt
  if (!ChangeState(eDseGoodCfg))
    {
      DBG("could not reset to 'halt'");
      return (false);
    }

  return (true);
}

/* ******************************************************
 * Function : SetPDU_Len
 * Description : set pdu length
 *  check <= current length performed; data not modified
 * Parameters  : buffer length, channel num and PduNum
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::SetPDU_Len(unsigned int   aBufLen,
             unsigned int  aChannelNum,
             unsigned int  aPduNum)
{

  unsigned int lSeqStoreSize = 0;
  unsigned int lNewLength = 0;

  // do usual checks on parameters
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports " << NO_OF_CHANNELS << " channels only");
      return (false);
    }

  if (aBufLen > ETH_MAX_LEN)
    {
      // buf overflow - beyond max PDU size
      DBG("Buffer length (" << aBufLen << " is more than permissible " << ETH_MAX_LEN << "
bytes");
      return (false);
    }

  // ensure frame exists in sSeqStoreArray
  tSeqStore_Vec::iterator   lFrameVector = sSeqStoreArray[aChannelNum].begin();
```

```
  lSeqStoreSize = sSeqStoreArray[aChannelNum].size();

  // if sSeqStoreArray[].size() is zero, the channel does not exist
  if (lSeqStoreSize == 0)
    {
      DBG("!Channel[" << aChannelNum << "] does not exist");
      return (false);
    }

    // note aFrameNum starts from 0
  if ((lSeqStoreSize > 0) && (lSeqStoreSize >= (aPduNum+1)))
    {
      // go to particular frame
      for (unsigned int lIndex = 0; lIndex < aPduNum; ++lIndex)
   {
     ++lFrameVector;
   }

      if (aBufLen == lFrameVector->GetEthGenBuf_Len())
   {
     DBG("Requested buffer len is equal to existing PDU len");
     DBG("Nothing to modify");
     return (true);
   }

      // if greater
      if (aBufLen > lFrameVector->GetEthGenBuf_Len())
   {
     DBG("PDU is only " << lFrameVector->GetEthGenBuf_Len() << " bytes long");
     return (false);
   }
      else
   {
     // valid case
     lNewLength = lFrameVector->SetEthGenBuf_Len(aBufLen);

     if (lNewLength != aBufLen)
       {
         DBG("Buffer length was set to " << lNewLength << "instead of "
        << aBufLen << " bytes");
         return (false);
       }

   }
    }
  else
    {
      DBG("!Channel[" << aChannelNum << "] contains " << lSeqStoreSize << " PDU(s)");
      return (false);
    }



  return (true);
}

/* ******************************************************
 * Function : QueryPDU_Len
 * Description : query pdu length
 * Parameters  : channel num and PduNum
 * Returns  : unsigned int
 * ******************************************************
 * NOTES:
 * ****************************************************** */
const unsigned int
```

```
tCmpEthGen::QueryPDU_Len(unsigned int aChannelNum,
         unsigned int aPduNum)
{
  unsigned int lSeqStoreSize = 0;
  unsigned int lPDULen = 0;

  // do usual checks on parameters
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports " << NO_OF_CHANNELS << " channels only");
      return (0);
    }

  // ensure frame exists in sSeqStoreArray
  tSeqStore_Vec::iterator   lFrameVector = sSeqStoreArray[aChannelNum].begin();

  lSeqStoreSize = sSeqStoreArray[aChannelNum].size();

  // if sSeqStoreArray[].size() is zero, the channel does not exist
  if (lSeqStoreSize == 0)
    {
      DBG("!Channel[" << aChannelNum << "] does not exist");
      return (0);
    }

  // note aFrameNum starts from 0
  if ((lSeqStoreSize > 0) && (lSeqStoreSize >= (aPduNum+1)))
    {
      // go to particular frame
      for (unsigned int lIndex = 0; lIndex < aPduNum; ++lIndex)
   {
     ++lFrameVector;
   }
      // get PDU length
      lPDULen = lFrameVector->GetEthGenBuf_Len();
    }
    else
    {
      DBG("!Channel[" << aChannelNum << "] contains " << lSeqStoreSize << " PDU(s)");
      return (0);
    }

  return (lPDULen);
}


/* ******************************************************
 * Function : SetRate_FramePerSec
 * Description : set tx rate for channel
 * Parameters  : frame rate and channel num
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::SetRate_FramePerSec(unsigned int aFrameRate,
           unsigned int aChannelNum)
{
  long      lTimeInterval_usec;

  tEthGenFrameVec *  lpFrameVec = NULL;

  // check valid frame rate
  if (aFrameRate == 0)
    {
      DBG("cannot set framerate = 0 fps");
```

```
      return (false);
    }

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      return (false);
    }

  // warn if requested rate is higher than permissible
  if (aFrameRate > 8127)
    {
      DBG("! Permissible frame rate is " << MAX_FRAME_RATE << " max " << endl
     << "! Unknown behaviour with set time limit and set loop limit");
    }

  // change at the source sEthGenSeq[]
  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  if (lpFrameVec != NULL)
    {
      if (lpFrameVec->size() > 0)
   {
     if (aFrameRate <= 1000000)
       {
         lTimeInterval_usec = 1000000/aFrameRate;
       }
     else
       {
         DBG("! invalid frame rate request, greater than 1e6");
         return (false);
       }

     DBG("calculated time interval = " << lTimeInterval_usec << " usecs" );

     // set time delay for sequence
     if (!(sEthGenSeq[aChannelNum].SetTimeDelay(lTimeInterval_usec)))
       {
         DBG("could not set time delay");
         return (false);
       }
     else
       {
         // now update local copy
         sChannelRate_fps[aChannelNum] = aFrameRate;
       }
   }
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");
      return (false);
    }

  return (true);
}

/* ******************************************************
 * Function : SetRate_Mbps
 * Description : set tx rate for channel in Mbps
 * Parameters  : frame rate and channel num
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

```
bool
tCmpEthGen::SetRate_Mbps(unsigned int aRate_Mb,
          unsigned int aChannelNum)
{
  tEthGenFrameVec *  lpFrameVec = NULL;

  unsigned int  lBytesInSeq = 0;
  unsigned int  lBytesInSeq_Average = 0;
  unsigned int lRate_fps = 0;

  int    lSeqSize = 0;

  // check valid frame rate
  if (aRate_Mb == 0)
    {
      DBG("cannot set frame rate = 0 Mbps");
      return (false);
    }

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      return (false);
    }

  // warn if requested rate is higher than permissible
  // already done in Tcl stage

  // get frame vec
  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  if (lpFrameVec != NULL)
    {
      // sequence exists
      if (lpFrameVec->size() > 0)
   {
      // get size of sequence
      lSeqSize = lpFrameVec->size();

      DBG("sequence size = " << lSeqSize);

      // look through each vector and get length in bytes
      for (tEthGenFrameVec::iterator   lFrame = lpFrameVec->begin();
           lFrame != lpFrameVec->end(); ++lFrame)
        {
          lBytesInSeq += (**lFrame).GetEthGenBuf_Len();
        }

      // convert to average bits and then to Mbits
      lBytesInSeq_Average = lBytesInSeq/lSeqSize;
      DBG("average bytes in sequence = " << lBytesInSeq_Average);

      // convert to fps
      lRate_fps = (aRate_Mb*1000000)/(lBytesInSeq_Average*8);

//      if (lRate_fps <= MAX_FRAME_RATE)
//        {
//          DBG("calculated fps = " << lRate_fps);

          if (!SetRate_FramePerSec(lRate_fps, aChannelNum))
       {
         DBG("Could not set rate ");
         return (false);
       }
//        }
```

```
//      else
//        {
//          DBG("calculated fps = " << lRate_fps);
         DBG("fps read out as " << QueryRate_FramePerSec(aChannelNum));
//        }
   }
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");
      return (false);
    }

  return (true);
}


/* ******************************************************
 * Function : SetLimit_Time
 * Description : set time limit for channel
 * Parameters  : time in seconds and channel num
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::SetLimit_Time(long     aTime_Sec,
           unsigned int    aChannelNum)
{

  tEthGenFrameVec *  lpFrameVec = NULL;

//   long         lTotPDU_Count = 0;

  unsigned int    lPDU_InSeq = 0;

  long         lSeq_Loops = 0;

  // check valid time rate
  if (aTime_Sec <= 0)
    {
      DBG("cannot set 0 or -ve time");
      return (false);
    }

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      return (false);
    }

  // get frame vec
  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  // get total number of frames in seq ('n' frames)
  lPDU_InSeq = lpFrameVec->size();

  if (lpFrameVec != NULL)
    {
      if (lpFrameVec->size() > 0)
   {

     // if dealing with sequences, convert to loops to be executed
     if (lPDU_InSeq > 1)
       {
```

```
        DBG("PDUs in sequence[" << aChannelNum << "] is " << lPDU_InSeq << " PDU(s)");

        // total number of sequences to be sent out in 1 second * total time
        lSeq_Loops = (QueryRate_FramePerSec(aChannelNum) * aTime_Sec)/ lPDU_InSeq;

        if (!((QueryRate_FramePerSec(aChannelNum)*aTime_Sec)%lPDU_InSeq))
      {
        // exact match to total number of loops
        DBG("convert to loop count (exact match) " << lSeq_Loops);
        SetLimit_Loops(lSeq_Loops, aChannelNum);
      }
         else
      {
        // allow for loop to complete
        DBG("converting to loop count = " << lSeq_Loops+1);
        SetLimit_Loops(lSeq_Loops+1, aChannelNum);
      }
       }
     else
       {
        // dealing with PDUs (lPDU_InSeq =  1)

        DBG("time limit set to " << aTime_Sec << " secs" );

        sGenTimeLimit[aChannelNum] = aTime_Sec;

        // set loop to -1 - they are mutually exclusive
        sGenLoopLimit[aChannelNum] = -1;
       }
   }
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");
      return (false);
    }


  return (true);
}

/* *******************************************************
 * Function : SetLimit_Loops
 * Description : set number of loops
 * Parameters  : loops and channel num
 * Returns  : bool
 * *******************************************************
 * NOTES:
 * ******************************************************* */
bool
tCmpEthGen::SetLimit_Loops(long    aLoops,
           unsigned int   aChannelNum)
{
  tEthGenFrameVec *  lpFrameVec = NULL;

  // check valid time rate
  if (aLoops <= 0)
    {
      DBG("cannot set 0 or -ve loops");
      return (false);
    }

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
```

```
      return (false);
    }

  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  if (lpFrameVec != NULL)
    {
      if (lpFrameVec->size() > 0)
   {
     DBG("limit set to " << aLoops << " loops" );

     sGenLoopLimit[aChannelNum] = aLoops;

     // set time to -1 - they are mutually exclusive
     sGenTimeLimit[aChannelNum] = -1;
   }
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");
      return (false);
    }

  return (true);
}


/* ******************************************************
 * Function : QueryRate_FramePerSec
 * Description : query tx rate for channel
 * Parameters  : channel num
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
const unsigned int
tCmpEthGen::QueryRate_FramePerSec(unsigned int aChannelNum) const

{
  unsigned int       lFrameRate;

  tEthGenFrameVec *  lpFrameVec = NULL;

  long         lTimeDelay_usec = 0;

  long         lCalcTimeDelay_usec = 0;

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      return (0);
    }

  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  if (lpFrameVec != NULL)
    {
      if (lpFrameVec->size() > 0)
   {
     lTimeDelay_usec = sEthGenSeq[aChannelNum].GetTimeDelay();

     if (lTimeDelay_usec != 0)
       {
         lFrameRate = 1000000/lTimeDelay_usec;
```

```
         // check this value against local value
         lCalcTimeDelay_usec = 1000000/sChannelRate_fps[aChannelNum];

         if (lTimeDelay_usec != lCalcTimeDelay_usec)
      {
        // return lFrameRate
        DBG("local FrameRate value is not updated");
        return (lFrameRate);
      }
       }
     else
       {
         DBG("found time delay for channel[" << aChannelNum << " set to 0");
         return (0);
       }
   }
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");
      return (0);
    }

  return (sChannelRate_fps[aChannelNum]);
}


/* ******************************************************
 * Function : QueryStats_FrameSent
 * Description : find how many frames were sent on channel x
 * Parameters  : Channel Number
 * Returns  : long long
 * ****************************************************** */
const long long
tCmpEthGen::QueryStats_FrameSent(unsigned int aChannelNum) const
{

  // check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      return (-1);
    }

  DBG("frames sent on Channel[" << aChannelNum << "] = "
      << sFramesSent_Channel[aChannelNum]);

  return (sFramesSent_Channel[aChannelNum]);

}


/* ******************************************************
 * Function : QueryStats_TimeInterval
 * Description : get generation time interval
 * Parameters  : none
 * Returns  : timeval
 * ****************************************************** */
const timeval
tCmpEthGen::QueryStats_TimeInterval(unsigned int aChannelNum) const
{
  timeval      lTimeInterval;
  tEthGenFrameVec *  lpFrameVec = NULL;

//   lTimeInterval.tv_usec = sGenFinTime_usec[aChannelNum];
//   lTimeInterval.tv_sec  = sGenFinTime_sec[aChannelNum];
```

```
// check for valid channel num
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports 0-" << NO_OF_CHANNELS-1 << " channels only");
      lTimeInterval.tv_sec  = -1;
      lTimeInterval.tv_usec = -1;

      return (lTimeInterval);
    }

  // check if channel is not empty
  lpFrameVec = sEthGenSeq[aChannelNum].GetFrameVec();

  if (lpFrameVec != NULL)
    {
//       DBG("finding time " << lTimeInterval.tv_sec << "." << lTimeInterval.tv_usec);

      if (sGenFinTime_usec[aChannelNum] < sGenStartTime_usec)
   {
//      DBG("** time adjustments");
     lTimeInterval.tv_usec = (sGenFinTime_usec[aChannelNum] + 1000000) - sGenStart-
Time_usec;
     lTimeInterval.tv_sec  = sGenFinTime_sec[aChannelNum] - sGenStartTime_sec - 1;

   }
      else
   {
     lTimeInterval.tv_usec = sGenFinTime_usec[aChannelNum] - sGenStartTime_usec;
     lTimeInterval.tv_sec  = sGenFinTime_sec[aChannelNum] - sGenStartTime_sec;
   }

//       DBG("finish time " << sGenFinTime_sec[aChannelNum] << "." << sGenFin-
Time_usec[aChannelNum]);

      DBG("time interval " << lTimeInterval.tv_sec << "." << lTimeInterval.tv_usec);
    }
  else
    {
      DBG("Channel[" << aChannelNum << "] is empty");

      lTimeInterval.tv_sec  = -1;
      lTimeInterval.tv_usec = -1;
    }

  return lTimeInterval;
}

/* *****************************************************
 * Function : EraseIntMemory
 * Description : erase internal memory
 * Parameters  : none
 * Returns  : bool
 * ***************************************************** */
bool
tCmpEthGen::EraseIntMemory()
{

    for (unsigned int lIndex = 0; lIndex < NO_OF_CHANNELS; ++lIndex)
    {
      // erase sSeqStoreArray
      if (sSeqStoreArray[lIndex].size() != 0)
   {
     cout << "Channel[" << lIndex << "] frame size reduced from "
          << sSeqStoreArray[lIndex].size() << " to ";
```

```
      sSeqStoreArray[lIndex].clear();

      cout << sSeqStoreArray[lIndex].size() << "." << endl;

      // array could not be cut down
      if (sSeqStoreArray[lIndex].size() != 0)
        {
          return (false);
        }
    }
       else
    {
//       DBG("Channel [" << lIndex << "] frame vec size " << sSeqStoreArray[lIndex].size());
    }

       // erase sFrameVector
       for (unsigned int lChannel = 0; lChannel < NO_OF_CHANNELS; ++lChannel)
    {
      if (sFrameVector[lChannel].size() != 0)
        {
          cout << "EraseIntMemory: sFrameVector[" <<  lChannel << "] size reduced from "
          << sFrameVector[lChannel].size() << " to ";

          sFrameVector[lChannel].clear();

          cout << sFrameVector[lChannel].size() << endl;
        }
    }

       // erase sEthGenSeq
       for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
    {
      try
        {
          sEthGenSeq[lCount].SetFrameVec(NULL);
        }
      catch (...)
        {
          DBG("could not remove Frame vector from sEthGenSeq[" << lCount << "]");
        }

    }

     }

  return (true);

}


/* ******************************************************
 * Function : EraseTempMemory
 * Description : erase temporary memory
 * Parameters  : none
 * Returns  : bool
 * ****************************************************** */
bool
tCmpEthGen::EraseTempMemory()
{
  // clear out all non-null frame vectors
  for (unsigned int lIndex = 0; lIndex < NO_OF_CHANNELS; ++lIndex)
    {
//       if (sFrameVector[lIndex].size() != 0)
//    {
//      DBG("clearing temp mem sFrameVector[" << lIndex << "]");
//      sFrameVector[lIndex].clear();
```

```
//    }

      sChannelArray[lIndex] = NULL;

      DBG("setting sChannelArray[" << lIndex << "] to NULL");
    }

  // sFrameCount for all sequences at start to 0
  sFrameCount_All = 0;

  // check drive

  return (true);
}


/* ******************************************************
 * Function : ModifyBuffer
 * Description : modify destination buffer, given src
 *       buffer, offset and len
 * Parameters  : offset, len, dest buf and src hex string
 * Returns  : 2 bufdata < len, 0, 1 buf truncated,
 *       -3 zero buflen, -4 buf overflow,
 *       -5 memcpy error
 * ****************************************************** */
int
tCmpEthGen::ModifyBuffer(unsigned int    aOffSet,
          unsigned int     aBufLen,
          unsigned char *  apDestBuf,
          unsigned char *  apSrcBuf)
{
  // removing buffer check - check for buffer length
  // must be done at higher levels

  unsigned char lTempBuffer[ETH_MAX_LEN];

  // clear out temp buffer
  for (unsigned int lCount = 0; lCount < ETH_MAX_LEN; ++lCount)
    {
      lTempBuffer[lCount] = 0;
    }


//   DBG("apSrc buffer contents");
//   for (unsigned int i = 1; i < 20; ++i)
//     {
//       cout << apSrcBuf[i];
//     }
//   cout << endl;


  if (aBufLen <= 0)
    {
      // buflen is zero, nothing to modify
      return (-3);
    }

  // check offset + len is not beyond limit
  if ((aOffSet + aBufLen) > ETH_MAX_LEN)
    {
      // buf overflow - beyond max PDU size
      return (-4);
    }

  // convert hex string to int
  if (!ConvertHexToInt((char *)apSrcBuf, aBufLen, (char *)lTempBuffer))
```

```
    {
      DBG("could not convert hex string");
    }

  // copy to dest
  if (memcpy(apDestBuf+aOffSet, lTempBuffer, aBufLen) == NULL)
    {
      return (-5);
    }


  // return 0 otherwise
  return (0);
}


/* ******************************************************
 * Function : ModifyFrameBuffer
 * Description : modify contents of PDU
 * Parameters  : offset, len, src buf, ChannelNum, FrNum
 * Returns  : bool
 * ****************************************************** */
bool
tCmpEthGen::ModifyFrameBuffer(unsigned int  aOffSet,
              unsigned int   aBufLen,
              unsigned char *   apSrcBuf,
              unsigned int   aChannelNum,
              unsigned int   aFrameNum)
{
  int lSeqStoreSize = 0;

  unsigned int    lModifyVal;
  unsigned int  lNewFrameLen;

  // do usual checks on parameters
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports " << NO_OF_CHANNELS << " channels only");
      return (false);
    }

  if ((aOffSet + aBufLen) > ETH_MAX_LEN)
    {
      // buf overflow - beyond max PDU size
      DBG("Offset(" << aOffSet << ") + aBufLen(" << aBufLen << ") exceeds "
     << ETH_MAX_LEN);

      return (false);
    }

  // check for existing channel
  //  check size of sSeqStoreArray[aChannelNum]
  tSeqStore_Vec::iterator   lFrameVector = sSeqStoreArray[aChannelNum].begin();

  lSeqStoreSize = sSeqStoreArray[aChannelNum].size();

  // if sSeqStoreArray[].size() is zero, the channel does not exist
  if (lSeqStoreSize == 0)
    {
      DBG("!Channel[" << aChannelNum << "] does not exist");
      return (false);
    }

  // note aFrameNum starts from 0
  if ((lSeqStoreSize > 0) && (lSeqStoreSize >= (int)(aFrameNum+1)))
    {
```

```
      // go to the particular frame
      for (unsigned int lIndex = 0; lIndex < aFrameNum; ++lIndex)
    {
      ++lFrameVector;
    }

      if ((aOffSet+aBufLen) >  lFrameVector->GetEthGenBuf_Len())
    {
      DBG("PDU is only " << lFrameVector->GetEthGenBuf_Len() << " bytes long");
      return (false);
    }

      // get buffer and modify
      lModifyVal = ModifyBuffer(aOffSet, aBufLen, lFrameVector->GetEthGenBuf_Buf(),
            apSrcBuf);

      //interpret return values
      switch (lModifyVal)
    {
    case -3: DBG("Zero buflen - nothing to modify");
      break;

    case -2: DBG("Offset+len is beyond PDU size(" << ETH_MAX_LEN << ") limit");
      break;

    case -5: DBG("Could not memcpy");
      break;

    case 0: DBG("Buffer copied");
      break;

    case 1: DBG("Buffer was truncated at end");
      break;

    case 2: DBG("Supplied data falls short of requested length, extra spaces filled with
0s");
      break;

    default: DBG("unknown behaviour ");
      break;

    }


      // change buffer length
      lNewFrameLen = (lFrameVector->GetEthGenBuf_Len() > (aOffSet + aBufLen)) ?
    lFrameVector->GetEthGenBuf_Len() : (aOffSet + aBufLen) ;

      if (lFrameVector->SetEthGenBuf_Len(lNewFrameLen) != lNewFrameLen)
    {
      DBG("!!Could not modify Length");
    }
      else
    {
      DBG("modified length of buffer is " << lNewFrameLen);
    }


      // change sequence specific parameters - sFrameVector and sEthGenSeq
      sFrameVector[aChannelNum].clear();

      for (tSeqStore_Vec::iterator   lStoredFrame = sSeqStoreArray[aChannelNum].begin();
      lStoredFrame != sSeqStoreArray[aChannelNum].end(); ++lStoredFrame)
    {
      DBG("Copying modified '" << lStoredFrame->GetFrameName() << "' into generator");
      sFrameVector[aChannelNum].push_back(lStoredFrame);
```

```
   }

      sEthGenSeq[aChannelNum].SetFrameVec(&sFrameVector[aChannelNum]);

    }
  else
    {
      DBG("!Channel[" << aChannelNum << "] contains " << lSeqStoreSize << " PDU(s)");
      return (false);
    }

  // return
  return (true);
}

/* ******************************************************
 * Function : QueryFrameBuffer
 * Description : supply a buffer to read out contents from
 *       a PDU (starting 0 - n) from
 *       a Channel (starting 0 - x)
 * Parameters  : offset, len, buffer, ChannelNum, FrameNum
 * Returns  : bool
 * ****************************************************** */
bool
tCmpEthGen::QueryFrameBuffer(unsigned int   aOffSet,
             unsigned int    aBufLen,
             unsigned char * apBuf,
             unsigned int aChannelNum,
             unsigned int aFrameNum)
{
    unsigned char lTempBuffer[ETH_MAX_LEN];

    unsigned int lSeqStoreSize = 0;

//     char          lDebugBuf[1514*2];

  // do usual checks on parameters
  if ((aChannelNum < 0) || (aChannelNum >= NO_OF_CHANNELS))
    {
      DBG("Engine supports " << NO_OF_CHANNELS << " channels only");
      return (false);
    }

  if ((aOffSet + aBufLen) > ETH_MAX_LEN)
    {
      // buf overflow - beyond max PDU size
      DBG("Offset(" << aOffSet << ") + aBufLen(" << aBufLen << ") exceeds "
     << ETH_MAX_LEN);

      return (false);
    }

  // ensure frame exists in sSeqStoreArray
  tSeqStore_Vec::iterator   lFrameVector = sSeqStoreArray[aChannelNum].begin();

  lSeqStoreSize = sSeqStoreArray[aChannelNum].size();

  // if sSeqStoreArray[].size() is zero, the channel does not exist
  if (lSeqStoreSize == 0)
    {
      DBG("!Channel[" << aChannelNum << "] does not exist");
      return (false);
    }

  // note aFrameNum starts from 0
  if ((lSeqStoreSize > 0) && (lSeqStoreSize >= (aFrameNum+1)))
```

```
    {
      // go to particular frame
      for (unsigned int lIndex = 0; lIndex < aFrameNum; ++lIndex)
   {
     ++lFrameVector;
   }

      // copy contents from offset upto length into temp buffer
      if ((aOffSet+aBufLen) > lFrameVector->GetEthGenBuf_Len())
   {
     DBG("PDU is only " << lFrameVector->GetEthGenBuf_Len() << " bytes long");
     return (false);
   }

      if (memcpy(lTempBuffer, (lFrameVector->GetEthGenBuf_Buf()) + aOffSet,
       aBufLen) != NULL)
   {
     for (unsigned int i = 0; i < aBufLen; ++i)
       {
         cout << (int)lTempBuffer[i];
       }

     // Convert contents to hex string
     if (ConvertIntToHex((char *)lTempBuffer, (int)(aBufLen), (char *)apBuf) != 1)
       {
         DBG("Could not convert to hex string");
       }

     // Display hex string for now
//      for (unsigned int i = 0; i < aBufLen; ++i)
//        {
//          cout << (int)lDebugBuf[i];
//        }
   }
      else
   {
     DBG("memcpy error");
     return (false);
   }
    }
  else
    {
      DBG("!Channel[" << aChannelNum << "] contains " << lSeqStoreSize << " PDU(s)");
      return (false);
    }

  return (true);

}


/* ******************************************************
 * Function : all other seq and frame and rate fns
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ****************************************************** */

bool     tCmpEthGen::RemoveChannel(unsigned int   aChannelNum){return (true);}
bool     tCmpEthGen::InsertSeq(){return (true);}
bool     tCmpEthGen::ModifySeq(){return (true);}
bool     tCmpEthGen::RemoveSeq(){return (true);}
bool     tCmpEthGen::InsertFrame(){return (true);}
bool     tCmpEthGen::ModifyFrame(){return (true);}
bool     tCmpEthGen::RemoveFrame(){return (true);}
```

```
// bool     tCmpEthGen::SetRate_BitsPerSec(unsigned int aBitRate,
//                       unsigned int aChannelNum){return (true);}


const unsigned int tCmpEthGen::QueryRate_BitsPerSec(unsigned int aChannelNum) const
{return (1);}


/* ******************************************************
 * Function : StopGenerate
 * Description : does not do anything at this point
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::StopGenerate()
{
  return (true);
}


/* ******************************************************
 * Function : private method
 * Description : thread main
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 *       time measurement is done as follows
 *    start time is set to current time gettimeofday
 *
 *         get timestamp from each sequence
 *              every time a frame is sent, start time is reset
 *               to a new time which is equal to previous time
 *     plus timestamp
 * ****************************************************** */
tThreadData *
tCmpEthGen::ThrMain(tThreadData * apData)
{
  timeval         lGetTime;
  long            lNextTime_sec[NO_OF_CHANNELS];
  long            lNextTime_usec[NO_OF_CHANNELS];
  long            lSeqTimeStamp[NO_OF_CHANNELS];
  long            lSeqTimeStamp_sec[NO_OF_CHANNELS];
  long            lSeqTimeStamp_usec[NO_OF_CHANNELS];

  long            lLoopLimit[NO_OF_CHANNELS];

  int             lFrameLen;

  int             lBufSize;

  int             lActiveChCounter = 0;

  // running counter to keep track of next frame in vec
  tEthGenFrameVec::const_iterator     lFrameIterator[NO_OF_CHANNELS];

  tEthGenFrameVec *        lpFrameVec[NO_OF_CHANNELS];

  // Don't care about any args
  if (apData != NULL)
    {
      delete (apData);
    }
```

```
  // find number of active channels and bump counter lActiveChannelCounter
  for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
    {
      if (sChannelArray[lCount] != NULL)
   {
     ++lActiveChCounter;
   }
    }
  DBG("found " << lActiveChCounter << " active channels for generation");


  // sRtlDriverFailCount set to 0
  sRtlDriverFailCount = 0;

  // clear out all stats from all channels, set to 0
  for (unsigned int lStatsCount = 0; lStatsCount < NO_OF_CHANNELS; ++lStatsCount)
    {
      sFramesSent_Channel[lStatsCount] = 0;
    }


  // debug print out for all channels
    for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
    {
      if (sChannelArray[lCount] != NULL)
   {
     DBG("------------- SEQ[" << lCount << "] ----------------");

     lpFrameVec[lCount] = sChannelArray[lCount]->GetFrameVec();

     for (tEthGenFrameVec::const_iterator lFrameIter = lpFrameVec[lCount]->begin();
          lFrameIter != lpFrameVec[lCount]->end(); ++lFrameIter)
       {
         DBG("-> " << (*lFrameIter)->GetFrameName() << " - " << (*lFrameIter)-
>GetEthGenBuf_Len());
       }
   }

      DBG("sGenTimeLimit[" << lCount << "] = " << sGenTimeLimit[lCount]);
      DBG("sGenLoopLimit[" << lCount << "] = " << sGenLoopLimit[lCount]);
   }


  // look all channels
  sGenThreadLoop = true;

  // get all time stamps from channels and store locally
  for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
    {
      if (sChannelArray[lCount] != NULL)
   {
     lSeqTimeStamp[lCount] = sChannelArray[lCount]->GetTimeDelay();

     lSeqTimeStamp_sec[lCount] = lSeqTimeStamp[lCount]/1000000;
     lSeqTimeStamp_usec[lCount] = lSeqTimeStamp[lCount]%1000000;
   }
    }

  for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
    {
       // get all frame vectors and make iterators point to beginning
      if (sChannelArray[lCount] != NULL)
   {
     // get vector
     lpFrameVec[lCount] = sChannelArray[lCount]->GetFrameVec();
```

```
      // point iterator to beginning
      lFrameIterator[lCount] = (lpFrameVec[lCount])->begin();

      // clear out loop limit
      if (sGenLoopLimit[lCount] != -1)
        {
          lLoopLimit[lCount] = 0;
        }
     }
      }

    // get initial time set t=to for all channels and save start time
    gettimeofday(&lGetTime, 0);

    sGenStartTime_sec = lGetTime.tv_sec;
    sGenStartTime_usec = lGetTime.tv_usec;

    for (unsigned int lLimitCount = 0; lLimitCount < NO_OF_CHANNELS; ++lLimitCount)
      {
        // clear out finish times to begin with
        sGenFinTime_sec[lLimitCount] = 0;
        sGenFinTime_usec[lLimitCount] = 0;
      }


    for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
      {
        lNextTime_usec[lCount] = lGetTime.tv_usec;
        lNextTime_sec[lCount] = lGetTime.tv_sec;
      }

    for (unsigned int lCount = 0; lCount < NO_OF_CHANNELS; ++lCount)
      {
        if (sChannelArray[lCount] != NULL)
     {
        if ((lGetTime.tv_usec + lSeqTimeStamp_usec[lCount]) > 999999)
          {
            DBG("adjusted overflow before while()");
            lNextTime_usec[lCount] = (lNextTime_usec[lCount] + lSeqTime-
Stamp_usec[lCount])%1000000;
            lNextTime_sec[lCount]  = lNextTime_sec[lCount] + lSeqTimeStamp_sec[lCount] + 1;
          }
        else
          {
            lNextTime_usec[lCount] = lNextTime_usec[lCount] + lSeqTimeStamp_usec[lCount];
            lNextTime_sec[lCount]  = lNextTime_sec[lCount] + lSeqTimeStamp_sec[lCount];
          }
     }
      }

      DBG("   START: From --> " << lGetTime.tv_sec << "." << lGetTime.tv_usec);

      while (sGenThreadLoop)
      {
        for (unsigned int lChannelNo = 0; lChannelNo < NO_OF_CHANNELS;
        ++lChannelNo)
     {
        gettimeofday(&lGetTime, 0);

        if (sChannelArray[lChannelNo] != NULL)
          {
            // look for next frame in line
            // this loops takes around 1.5 usecs to complete

            // generation with delay
```

```
   if ((lGetTime.tv_sec > lNextTime_sec[lChannelNo]) ||
  ((lGetTime.tv_sec == lNextTime_sec[lChannelNo]) &&
   (lGetTime.tv_usec >= lNextTime_usec[lChannelNo])) )
{
  // get BufSize
  lBufSize = (*lFrameIterator[lChannelNo])->GetEthGenBuf_Len();


  if (lBufSize <= ETH_MAX_LEN)
    {
      lFrameLen = send(sSock, (*lFrameIterator[lChannelNo])->GetEthGenBuf_Buf(),
         lBufSize, 0);
    }
  else
    {
      DBG("oooops!!!! invalid buffer size(" << lBufSize << "), contents[" <<
      ((*lFrameIterator)[lChannelNo])->GetEthGenBuf_Buf() << "]");

    }

  if (lFrameLen == -1)
    {
      // Something bad happened
      ++sRtlDriverFailCount;

      // DBG("Could not send buffer: " << strerror(errno));
      continue;
    }

  if (lFrameLen != lBufSize)
    {
      DBG("Could not send out req buffer size ");
      continue;
    }

  // keep count of frame sent
  ++sFramesSent_Channel[lChannelNo];

  // adjust start time and overflow
  lNextTime_usec[lChannelNo] += lSeqTimeStamp_usec[lChannelNo];
  lNextTime_sec[lChannelNo]  += lSeqTimeStamp_sec[lChannelNo];

  if (lNextTime_usec[lChannelNo] > 999999)
    {
      lNextTime_usec[lChannelNo] -= 1000000;
      lNextTime_sec[lChannelNo]  += 1;
    }


  // point to the next frame in line, reset if end is reached
  ++(lFrameIterator[lChannelNo]);

  if (lFrameIterator[lChannelNo] == (lpFrameVec[lChannelNo])->end())
    {
      lFrameIterator[lChannelNo] = (lpFrameVec[lChannelNo])->begin();

      // abort if loop limit is requested
      if ((sGenLoopLimit[lChannelNo] != -1) &&
     (++lLoopLimit[lChannelNo] >= sGenLoopLimit[lChannelNo]))
   {
     DBG(lLoopLimit[lChannelNo] <<
         " coming out of seq[" << lChannelNo << "] at end of "
         << lLoopLimit[lChannelNo] << " loops "
         << "lActiveCounter " << lActiveChCounter);

     // store finish times for stats later
```

```
            sGenFinTime_sec[lChannelNo] = lGetTime.tv_sec;
            sGenFinTime_usec[lChannelNo] = lGetTime.tv_usec;

            sChannelArray[lChannelNo] = NULL;

            //decrement active channel count
            --lActiveChCounter;
          }
           }

         // check time limit if requested
         if ((sGenTimeLimit[lChannelNo] != -1) &&
             ((lGetTime.tv_sec >  (sGenStartTime_sec + sGenTimeLimit[lChannelNo])) ||
              ((lGetTime.tv_sec == (sGenStartTime_sec + sGenTimeLimit[lChannelNo])) &&
          (lGetTime.tv_usec >= sGenStartTime_usec )) ) )
           {

             DBG((sGenStartTime_sec + sGenTimeLimit[lChannelNo])
            << " coming out of seq[" << lChannelNo << "] at end of "
            << (lGetTime.tv_sec - sGenStartTime_sec)  << " seconds, "
            << (lGetTime.tv_usec - sGenStartTime_usec) << " micro seconds");

             // store finish times for stats later
             sGenFinTime_sec[lChannelNo] = lGetTime.tv_sec;
             sGenFinTime_usec[lChannelNo] = lGetTime.tv_usec;

             sChannelArray[lChannelNo] = NULL;

             //decrement active channel count
             --lActiveChCounter;
           }

         // for every time limit or loop limit completion decrement counter
         // for counter == 0, come out of loop
         if (lActiveChCounter == 0)
           {
             DBG("no active channels left to generate");

             // go out of while(1) loop
             sGenThreadLoop = false;

           }

       }
        }
    }
     }

   // do post gen
   if (!ChangeState(eDsePostGen))
     {
       DBG("! Could not return state to 'Not Running'");
     }

   return (0);
}


   ///
/// State transition functions
///


/* ******************************************************
 * Function : ChangeState
 * Description : change states depending upon action and
```

```
 *       existing and final states
 * Parameters  : Event that happened
 * Returns  : bool
 * ******************************************************
 * NOTES: looks up table and takes action based on
 *    event requested and on existing s/m state
 * ****************************************************** */
bool
tCmpEthGen::ChangeState(tEthGenEvents aEthGenEvent)
{
  // lock mutex
  pthread_mutex_lock(&sContext.sStateMutex);

  // Hunt the state/event pair
  int lIdx;
  for (lIdx = 0; lIdx < TRANSITIONS; ++lIdx)
    {
//       DBG("into ChangeState before function call ");

      if ( (sStateTrParmTbl[lIdx].sCurState == sContext.sState) &&
      (sStateTrParmTbl[lIdx].sEvent == aEthGenEvent) )
   {
     // Found a match - break
     break;
   }
    }

  // Check if found a match
  if (lIdx == TRANSITIONS)
    {
      // No match
      pthread_mutex_unlock(&sContext.sStateMutex);
      DBG(QueryState() << ": haven't found match in state transition table " );
      return (false);
    }


  if ((this->*sStateTrParmTbl[lIdx].sFunctionCall) != NULL)
    {
      if (!( (this->*sStateTrParmTbl[lIdx].sFunctionCall)()) )
   {
     DBG("error from function call");

     // change state irrespective of fail - reconfigure type
     // function require the new state to work on
     sContext.sState = sStateTrParmTbl[lIdx].sNewState;

     // unlock mutex
     pthread_mutex_unlock(&sContext.sStateMutex);

     return (false);
   }
    }

  // change state
  sContext.sState = sStateTrParmTbl[lIdx].sNewState;

  // unlock mutex
  pthread_mutex_unlock(&sContext.sStateMutex);

  return (true);

}


/* ******************************************************
```

```
 * Function : DoInit
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoInit()
{
  // Find interface index
  ifreq  lIfReq;
  strncpy(lIfReq.ifr_name, sIfEthernet.c_str(), IFNAMSIZ);

  if (ioctl(sSock, SIOCGIFINDEX, &lIfReq) == -1 )
    {
      DBG("Could not get interface index: " << strerror(errno));
    }

  // bind socket to interface
  sockaddr_ll   lAddr;
  memset(&lAddr, 0, sizeof(lAddr));
  lAddr.sll_family = AF_PACKET;
  lAddr.sll_protocol = htons(ETH_P_ALL);
  lAddr.sll_ifindex = lIfReq.ifr_ifindex;
  if (bind(sSock, (sockaddr *)&lAddr, sizeof(lAddr)) == -1)
    {
      DBG("Could not bind socket: " << strerror(errno));
      return (false);
    }
  else
    {
      DBG("Bound '" << sIfEthernet.c_str() << "'");
    }

  return (true);
}

/* ******************************************************
 * Function : DoConfig
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoConfig()
{
  return (true);
}

/* ******************************************************
 * Function : DoReConfig
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoReConfig()
{
  return (true);
}
```

```
/* ******************************************************
 * Function : DoBadCfg
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoBadCfg()
{
  return (true);
}


/* ******************************************************
 * Function : DoGoodCfg
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoGoodCfg()
{
//   DBG("into DoGoodCfg");

  return (true);
}

/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoStart()
{
  return (true);
}


/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoBadPreRC()
{
//   DBG("DoBadPreRC ");

  // set flag to false and return to halt

  return (true);
}

/* ******************************************************
 * Function : DoGoodPreRC private method
```

```
 * Description : create thread after good pre-run check
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoGoodPreRC()
{
  return (true);
}

/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoGenerate()
{
//   DBG("DoGenerate executed ");
  return (true);
}


/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool// State transitions

 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoRGStop()
{

  return (true);
}


/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool// State transitions

tCmpEthGen::DoNRStop()
{

  return (true);
}


/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
```

```
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoRGEndGen()
{
//   DBG("into DoRGEndGen() ");

  return (true);
}


/* ******************************************************
 * Function : private method
 * Description :
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoWSEndGen()
{

  return (true);
}


/* ******************************************************
 * Function : private method
 * Description : cosmetic subroutine - does nothing
 * Parameters  : none
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpEthGen::DoNothing()
{
  return (true);
}

/* ******************************************************
 * Function : ConvertHexToInt
 * Description :
 * Parameters  : char * aInb, int aLen, char * aOutb
 * Returns  : unsigned int
 * ******************************************************
 * NOTES:
 * ****************************************************** */
unsigned int
tCmpEthGen::ConvertHexToInt(char * aInb, int aLen, char * aOutb)
{
  int inidx = 0;
  int outidx = 0;

  bool lHexChar = false;

  // higher nibble
  bool nib = true;
  int num = 0;

//   DBG("into ConverHexToInt");

  while (inidx < aLen*2)
```

```
    {
      if (aInb[inidx] >= '0' && aInb[inidx] <= '9')
    {
      num = aInb[inidx] - '0';
      lHexChar = true;
    }
       else if ((aInb[inidx] >= 'A') && (aInb[inidx] <= 'F'))
    {
      num = aInb[inidx] - 'A' + 10;
      lHexChar = true;
    }
       else if ((aInb[inidx] >= 'a') && (aInb[inidx] <= 'f'))
    {
      num = aInb[inidx] - 'a' + 10;
      lHexChar = true;
    }

       // return if char is not hex
       if (!lHexChar)
    {
      DBG("invalid hex char '" << (char)aInb[inidx] << "'");
      return (0);
    }


       ++inidx;

       // reset lHexChar to false for incoming chars
       lHexChar = false;

       if (nib)
    {
      aOutb[outidx] = num << 4;
      nib = false;
    }
       else
    {
      aOutb[outidx] = aOutb[outidx] | num;
      nib = true;
      ++outidx;
    }

//       cout << (int)aOutb[outidx];

    }

  return (1);
}

/* ******************************************************
 * Function : ConvertIntToHex
 * Description : return outbuffer to represent hex data
 * Parameters  : char * aInb, int aLen, char * aOutb
 * Returns  : unsigned int
 * ******************************************************
 * NOTES: Outb size >= 2*Inb
 * ****************************************************** */
unsigned int
tCmpEthGen::ConvertIntToHex(char * aInb, int aLen, char * aOutb)
{
  int inidx = 0;
  int outidx = 0;

  unsigned int lNum = 0;

  char lTable[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8',
```

```
          '9', 'A', 'B', 'C', 'D', 'E', 'F'};

  while (inidx < aLen)
    {
//       DBG(" ---------> aInb[" << inidx << "] = " << (int)aInb[inidx]);

      // correction for signed int
      if (aInb[inidx] < 0)
   {
     lNum = aInb[inidx] + 256;
//      DBG(" lNum = " << lNum);
//      DBG(" lNum >> 4 = " << (lNum >> 4));

     aOutb[outidx] = lTable[lNum >> 4];
   }
      else
   {
     aOutb[outidx] = lTable[aInb[inidx] >> 4];
   }

//       DBG("tbl(" << (int)(aInb[inidx] >> 4) << ") " << "aOutb[" << outidx << "] = " <<
aOutb[outidx]);
      ++outidx;

      aOutb[outidx] = lTable[aInb[inidx] & 0xf];
//       DBG("tbl(" << (int)(aInb[inidx] & 0xf) << ") " << "aOutb[" << outidx << "] = " <<
aOutb[outidx]);
      ++outidx;

      ++inidx;
    }

  return (1);

}


/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ==================================================================== */

extern "C"
tComponent *
CreateCmp_EthGen()
{
  return (new tCmpEthGen);
}


/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

```
/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */


//  LocalWords:  TCS UDP

/* ********************************************************************
 *
 *   Source File Name   : CmpThrdMgr.cc
 *
 *   Module Name  : CmpThrdMgr
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS01
 *
 * ********************************************************************
 * (c)
 * ********************************************************************
 *  NOTES:


 *  END OF NOTES
 * ==================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ======================================================
 *  Standard Library Includes   (normal system)
 * ====================================================== */


/* ======================================================
 *  External Includes        (external toolkits)
 * ====================================================== */


/* ======================================================
 *  Project-wide Includes     (project only)
 * ====================================================== */


/* ======================================================
 *  Module Includes       (module only)
 * ====================================================== */
#include "../../inc/Component.hh"
#include "../../inc/Iface/IfThrdMgr.hh"
#include "../../inc/ThreadBase.hh"


/* ======================================================
 *  Module #DEFINEs
 * ====================================================== */
#include <iostream>

#include <sys/time.h>


/* ======================================================
 *  Enumerations & Other Typedefs (defn)
```

```
 * ======================================================== */


/* ========================================================
 *  Classes (forward decl) & Structures (defn)
 * ======================================================== */


/* ========================================================
 *  Module Functions (decl.)    (static local only)
 * ======================================================== */


/* ========================================================
 *  Global Variables (defn.)    (used externally)
 * ======================================================== */


/* ========================================================
 *  Local Module Variables (defn.) (static local only)
 * ======================================================== */



/* ========================================================================
 * ========================================================================
 *  Classes visible internally only
 * ======================================================================== */

/* ******************************************************
 * Class : tThreadEntryArg
 * Description : Hold the argument data for a thread
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class tThreadEntryArg
{
public:
  tThreadEntryArg(tThreadBase * apObj,
        tThreadData * apData)
    : spObj(apObj),
      spData(apData)
  {}
  tThreadBase *      spObj;
  tThreadData *      spData;
private:
  tThreadEntryArg();
};



/* ========================================================================
 * ========================================================================
 *  Functions visible internally only (static, defn)
 * ======================================================================== */



/* ========================================================================
 * ========================================================================
 *  Classes visible externally
 * ======================================================================== */

/* ******************************************************
 * Function : tThreadBase()
 * Description : Ctor
```

```
 * Parameters  : N/A
 * Returns  : N/AlpThrArg->spObj->Main(lpThrArg->spData)
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tThreadBase::tThreadBase()
{
}

/* ******************************************************
 * Function : tThreadBase::Go()
 * Description : Starts the new thread
 * Parameters  : Context data for new thread
 * Returns  : true = created thread, false = failed
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tThreadBase::ThrGo(tThreadData * apData)
{
  int       lRtn;

  lRtn = pthread_create(&sThrHdl, 0, tThreadBase::ThreadEntry,
         new tThreadEntryArg(this, apData));
//   cout << "ThrGo:lRtn = " << lRtn << endl;
  if (lRtn != 0)
    {
      return (false);
    }
  return (true);
}

/* ******************************************************
 * Function : tThreadBase::Detach()
 * Description : Waits for thread to terminate
 * Parameters  : Context data for new thread
 * Returns  : true = created thread, false = failed
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tThreadBase::ThrDetach()
{
  int       lRtn;

  lRtn = pthread_detach(sThrHdl);
  if (lRtn != 0)
    {
      return (false);
    }
  return (true);
}

/* ******************************************************
 * Function : tThreadBase::Join()
 * Description : Waits for thread to terminate
 * Parameters  : Context data for new thread
 * Returns  : true = created thread, false = failed
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tThreadBase::ThrJoin()
{
  int       lRtn;
  void *    lThrRtn;
```

```
//   timeval      lGetTime;

  lRtn = pthread_join(sThrHdl, &lThrRtn);
//   cout << "ThrJoin:lRtn = " << lRtn << endl;
  if (lRtn != 0)
    {
      return (false);
    }
//   gettimeofday(&lGetTime, 0);

//   cout << "ThrJoin: " << lGetTime.tv_sec << ": " << lGetTime.tv_usec << endl;

  return (true);
}

/* ******************************************************
 * Function : tThreadBase::ThreadEntry
 * Description : Function called as entry point for thread
 * Parameters  : Pointer to arbitrary data
 * Returns  : Pointer to arbitrary data
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void *
tThreadBase::ThreadEntry(void * apArg)
{
  tThreadEntryArg *  lpThrArg = static_cast<tThreadEntryArg *>(apArg);
  tThreadBase *      lpObj = lpThrArg->spObj;
  tThreadData *      lpData = lpThrArg->spData;
  void *    lThrRtn;

  delete (lpThrArg);
  lThrRtn = lpObj->ThrMain(lpData);
  return (lThrRtn);
}


/* ====================================================================
 * ====================================================================
 *  Functions visible externally
 * ==================================================================== */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
// class tCmpThrdMgr
//   : public tComponent,
//     public tIfThrdMgr
// {
// public:

// };




/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ==================================================================== */
// extern "C"  tComponent *
// CreateCmp_ThreadManager()
// {
```

```
//   return (new tCmpThrdMgr);
// }
```

```
/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

# 1.1 Component Router

```
/* ********************************************************************
 *
 *   Source File Name   : CmpRouter.cc
 *
 *   Module Name  : CmpRouter
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS01
 *
 * ********************************************************************
 * (c) 2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:  Component Router


 *  END OF NOTES
 * ==================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */
```

```
/* ============================================================
 *  Standard Library Includes   (normal system)
 * ============================================================ */
#include <string>
#include <set>

/* ============================================================
 *  External Includes        (external toolkits)
 * ============================================================ */


/* ============================================================
 *  Project-wide Includes    (project only)
 * ============================================================ */


/* ============================================================
 *  Module Includes       (module only)
 * ============================================================ */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfRouter.hh"
#include "../../../inc/Iface/IfLoader.hh"

#include "../../../inc/Conduit.hh"


/* ============================================================
 *  Module #DEFINEs
 * ============================================================ */


/* ============================================================
 *  Enumerations & Other Typedefs (defn)
 * ============================================================ */

/* ============================================================
 *  Classes (forward decl) & Structures (defn)
 * ============================================================ */


/* ============================================================
 *  Module Functions (decl.)    (static local only)
 * ============================================================ */


/* ============================================================
 *  Global Variables (defn.)    (used externally)
 * ============================================================ */


/* ============================================================
 *  Local Module Variables (defn.) (static local only)
 * ============================================================ */



/* ====================================================================
 * ====================================================================
 *  Functions & class definitions visible externally
 * ==================================================================== */



/* ====================================================================
```

```
 * ====================================================================
 *  Functions & class definitions internal to module
 * ==================================================================== */

/* ******************************************************
 * Class : tCmpRouter
 * Description : Component Router with defn for Bind fns
 * ******************************************************
 * NOTES:   inherits tComponent and interface tIfRouter
 * ****************************************************** */

class tCmpRouter
  : public tComponent,
    public tIfRouter
{
public:
  tCmpRouter()
  {
     DBG("Creating Router");
  }

  ~tCmpRouter()
  {
    DBG("Deleting Router");
  }

  void Invoke()
  {
    DBG("hello from tCmpRouter");
  }

  bool AddConduit(tConduit * apConduit);

  bool Bind(const string & arCmpName1, const string & arPortName1,
       const string & arCmpName2, const string & arPortName2);
  bool Bind(tComponent * apCmpHdl1, const string & arPortName1,
       tComponent * apCmpHdl2, const string & arPortName2);
  bool Bind(tIfDataPort * apPortHdl1, tIfDataPort * apPortHdl2);

  bool Bind(const string & aCmpName,
       const string & arPortName, tConduit * apConduit);
  bool Bind(tComponent * apCmpHdl,
       const string & arPortName, tConduit * apConduit);
  bool Bind(tIfDataPort * apPortHdl, tConduit * apConduit);

private:
  typedef set<tConduit *>  tConduitList;
  tConduitList sConduitList;
};

/* ******************************************************
 * Function : AddConduit( * Conduit)
 * Description : add conduits to list
 * Parameters  : tConduit * apConduit
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpRouter::AddConduit(tConduit * apConduit)
 {
   tConduitList::iterator  lpIter;

   lpIter = sConduitList.find(apConduit);
   if (lpIter == sConduitList.end())
     {
```

```
      sConduitList.insert(apConduit);
       return (true);
     }
   else
     {
       DBG("duplicate conduit, cannot add: " << apConduit);
       return (false);
     }

}

/* ******************************************************
 * Function : [1]Bind(str Cmp, str Port, str Cmp, str Port)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpRouter::Bind(const string & arCmpName1,
       const string & arPortName1,
       const string & arCmpName2,
       const string & arPortName2)
{
  tComponent * lpCmpHdl1;
  tComponent * lpCmpHdl2;
  tIfLoader *  lpLoader = tIfLoader::GetLoader();

  // convert string aCmpName1 & arCmpName2 to instance handles
//   DBG("(1) finding " << arCmpName1);
  lpCmpHdl1 = lpLoader->InstQuery(arCmpName1);
//   DBG("(2) finding " << arCmpName2);
  lpCmpHdl2 = lpLoader->InstQuery(arCmpName2);
//   DBG("(3) found " << arCmpName1 << "and " << arCmpName2);

  Bind(lpCmpHdl1, arPortName1, lpCmpHdl2, arPortName2);
//   DBG("(8) called Bind[2] ");

  return (true);
}

/* ******************************************************
 * Function : [2]Bind(* Cmp, str Port, *Cmp, str Port)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpRouter::Bind(tComponent * apCmpHdl1, const string & arPortName1,
       tComponent * apCmpHdl2, const string & arPortName2)
{

  tIfDataPort *      lpPortHdl1;
  tIfDataPort *      lpPortHdl2;

  // query component for its port and get pointer to port
//   DBG("(4) Bind[2]: CmpPortMapGet " << arPortName1);
  lpPortHdl1 = dynamic_cast<tIfCmpPorts *>(apCmpHdl1)->CmpPortMapGet().find(arPortName1)-
>second;

//   DBG("(5) Bind[2]: CmpPortMapGet " << arPortName2);
  lpPortHdl2 = dynamic_cast<tIfCmpPorts *>(apCmpHdl2)->CmpPortMapGet().find(arPortName2)-
>second;
```

```
//   DBG("(6) Bind[2]: bind portHdl '" << arPortName1 << "' and portHdl '" << arPortName2
<< "'");
  Bind(lpPortHdl1, lpPortHdl2);
//   DBG("(11) done port to port bind");
  return (true);
}

/* ******************************************************
 * Function : [3]Bind( * PortHdl, str Conduit)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpRouter::Bind(tIfDataPort * apPortHdl1, tIfDataPort * apPortHdl2)
{
  // make conduit
  tConduit *   lpConduit = new tConduit;

//    DBG("(7) created new conduit");

  // bind if conduit successfully created
  if (AddConduit(lpConduit))
    {
//       DBG("(8) Bind[3] handle 2 to new conduit");
      Bind( apPortHdl2, lpConduit); // port2 to conduit
//       DBG("(9) Bind[3] handle 1 to new conduit");
      Bind( apPortHdl1, lpConduit); // port1 to conduit

//       DBG("(12) ports bound to both ends of conduit");
      return (true);
    }
  else
    {
//       DBG("Could not bind[3]");
      delete (lpConduit);
      return (false);
    }
}

/* ******************************************************
 * Function : [4]Bind(str Cmp, str Port, * Conduit)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES: Cmp1, Port2, conduit is not valid
 *      This functions must be used when one
 *        component has more than one port.
 * ****************************************************** */
bool
tCmpRouter::Bind(const string & arCmpName,
       const string & arPortName, tConduit * apConduit)
{
  tComponent * lpCmpHdl;
  tIfLoader *  lpLoader = tIfLoader::GetLoader();

  // query loader to get component pointer
  lpCmpHdl = lpLoader->InstQuery(arCmpName);

  // use next function
  Bind(lpCmpHdl, arPortName, apConduit);
  return (true);
```

```
}

/* ******************************************************
 * Function : [5]Bind(* CmpHdl, str Port, * Conduit)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES: CmpHdl1, Port2, conduit is not valid
 *      This functions is particularly useful when one
 *        component has more than one port.
 * ****************************************************** */
bool
tCmpRouter::Bind(tComponent * apCmpHdl,
       const string & arPortName, tConduit * apConduit)

{
  tIfDataPort * lpPortHdl;

  // query component to get pointer to its port
  //  lpPortHdl = apCmpHdl->CmpPortMapGet().find(arPortName)->second;
  lpPortHdl = dynamic_cast<tIfCmpPorts *>(apCmpHdl)->CmpPortMapGet().find(arPortName)-
>second;
  Bind(lpPortHdl, apConduit);

  return (true);
}

/* ******************************************************
 * Function : [6]Bind(* PortHdl, * Conduit)
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpRouter::Bind(tIfDataPort * apPortHdl, tConduit * apConduit)
{
//   DBG("(9) Bind[6]: handle binding to conduit ");
  apPortHdl->BindToConduit(apConduit); // returns void

//   DBG("(10) Bind[6]: conduit binding to port ");

  if (apConduit->BindToPort(apPortHdl))
    {
//       DBG("(11) successful return from conduit binding to port");
      return (true);
    }
  else
    {
//       DBG("Conduit could not be bound ");
      return (false);
    }
}

/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ================================================================== */
extern "C"
tComponent *
CreateCmp_Router()
{
  return (new tCmpRouter);
}
```

```
/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */



/* ********************************************************************
 *
 *   Source File Name   : Conduit.cc
 *
 *   Module Name  : Conduit
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:  see Conduit.hh


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ======================================================
 *  Standard Library Includes   (normal system)
 * ====================================================== */


/* ======================================================
 *  External Includes        (external toolkits)
 * ====================================================== */


/* ======================================================
 *  Project-wide Includes    (project only)
 * ====================================================== */


/* ======================================================
 *  Module Includes       (module only)
```

```
 * ========================================================== */
#include "../../../inc/Conduit.hh"


/* ==========================================================
 *  Module #DEFINEs
 * ========================================================== */


/* ==========================================================
 *  Enumerations & Other Typedefs (defn)
 * ========================================================== */


/* ==========================================================
 *  Classes (forward decl) & Structures (defn)
 * ========================================================== */


/* ==========================================================
 *  Module Functions (decl.)    (static local only)
 * ========================================================== */


/* ==========================================================
 *  Global Variables (defn.)    (used externally)
 * ========================================================== */


/* ==========================================================
 *  Local Module Variables (defn.) (static local only)
 * ========================================================== */



/* ======================================================================
 * ======================================================================
 *  Functions & class definitions visible externally
 * ====================================================================== */

/* ******************************************************
 * Function : BindToPort
 * Description : binds conduit to port
 * Parameters  : tIfDataPort * apPort
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tConduit::BindToPort(tIfDataPort * apPort)
{
  const string &  lrType  = apPort->DataPortType();
  tPortList *     lpList;
  tPortList::iterator   lIter;

  if (lrType == "ByteIn")
    {
      lpList = &sOut;
    }
  else if (lrType == "ByteOut")
    {
      lpList = &sIn;
    }
  else
    {
      return (false);
```

```
    }

  lIter = lpList->find(apPort);
  if (lIter != lpList->end())
    {
      return (true);
    }
  lpList->insert(apPort);
  DBG("BindToPort successful");
  return (true);
}

/* ******************************************************
 * Function : UnBindFromPort
 * Description : unbinds conduit from port
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tConduit::UnBindFromPort(tIfDataPort * apPort)
{
  sIn.erase(apPort);
  sOut.erase(apPort);

  DBG("UnBound from port ");
  return (true);
}

/* ******************************************************
 * Function : Transfer
 * Description : transfers bytes from buffer
 * Parameters  : char * apBuffer, unsigned int  aCount
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tConduit::Transfer(char *  apBuffer,
         unsigned int   aCount)
{
  for (tPortList::iterator lIter = sOut.begin();
       lIter != sOut.end();
       ++lIter)
    {
//       DBG("Transfering");
      (*lIter)->Transfer(apBuffer, aCount);
    }
  return (true);
}



/* ====================================================================
 * ====================================================================
 *  Functions & class definitions internal to module
 * ==================================================================== */




/* ====================================================================
 * ====================================================================
```

```
 *  Entry Points
 * ================================================================== */









/* ============ *
 *     EOF      *
 * ============ */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */



/* ********************************************************************
 *
 *   Source File Name   : ByteOut.cc
 *
 *   Module Name  : ByteOut
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:    Specific type of data port
 *    outport of bytes stream
 *
 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes   (normal system)
 * ======================================================== */
```

```
/* ========================================================
 *  External Includes        (external toolkits)
 * ======================================================== */


/* ========================================================
 *  Project-wide Includes    (project only)
 * ======================================================== */


/* ========================================================
 *  Module Includes       (module only)
 * ======================================================== */
#include "../../../inc/ByteOut.hh"


/* ========================================================
 *  Module #DEFINEs
 * ======================================================== */


/* ========================================================
 *  Enumerations & Other Typedefs (defn)
 * ======================================================== */


/* ========================================================
 *  Classes (forward decl) & Structures (defn)
 * ======================================================== */


/* ========================================================
 *  Module Functions (decl.)    (static local only)
 * ======================================================== */


/* ========================================================
 *  Global Variables (defn.)    (used externally)
 * ======================================================== */


/* ========================================================
 *  Local Module Variables (defn.) (static local only)
 * ======================================================== */
const string   tByteOut::sPortType("ByteOut");



/* ====================================================================
 * ====================================================================
 *  Functions & class definitions visible externally
 * ==================================================================== */







/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ==================================================================== */
```

```
/* =========== *
 *     EOF     *
 * =========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */



/* ********************************************************************
 *
 *   Source File Name   : ByteIn.cc
 *
 *   Module Name  : ByteIn
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:    Specific type of data port
 *    outport of bytes stream
 *
 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes   (normal system)
 * ======================================================== */
#include <fstream>

/* ========================================================
 *  External Includes        (external toolkits)
 * ======================================================== */


/* ========================================================
 *  Project-wide Includes    (project only)
```

```
 * ======================================================== */


/* ========================================================
 *  Module Includes       (module only)
 * ======================================================== */
#include "../../../inc/ByteIn.hh"


/* ========================================================
 *  Module #DEFINEs
 * ======================================================== */


/* ========================================================
 *  Enumerations & Other Typedefs (defn)
 * ======================================================== */


/* ========================================================
 *  Classes (forward decl) & Structures (defn)
 * ======================================================== */


/* ========================================================
 *  Module Functions (decl.)    (static local only)
 * ======================================================== */


/* ========================================================
 *  Global Variables (defn.)    (used externally)
 * ======================================================== */


/* ========================================================
 *  Local Module Variables (defn.) (static local only)
 * ======================================================== */
const string   tByteIn::sPortType("ByteIn");


/* ======================================================================
 * ======================================================================
 *  Functions & class definitions visible externally
 * ====================================================================== */







/* ======================================================================
 * ======================================================================
 *  Entry Points
 * ====================================================================== */




/* =========== *
```

```
 *     EOF     *
 * =========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */



Component Scheduler
/* ********************************************************************
 *
 *   Source File Name   : CmpSched.cc
 *
 *   Module Name  : CmpSched
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c) 2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:  component Scheduler


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ======================================================
 *  Standard Library Includes    (normal system)
 * ====================================================== */
#include <fstream>
#include <list>
#include <string>
#include <sys/poll.h>
#include <errno.h>
#include <unistd.h>

#include <signal.h>

/* ======================================================
 *  External Includes        (external toolkits)
 * ====================================================== */


/* ======================================================
```

```
 *  Project-wide Includes     (project only)
 * ========================================================== */


/* ==========================================================
 *  Module Includes        (module only)
 * ========================================================== */
#include "../../../inc/Iface/IfSched.hh"
#include "../../../inc/Iface/IfDataPort.hh"



/* ==========================================================
 *  Module #DEFINEs
 * ========================================================== */


/* ==========================================================
 *  Enumerations & Other Typedefs (defn)
 * ========================================================== */
typedef list<tComponent *> tCmpList;
typedef list<tIfDataPort *>   tPortList;
typedef list<tEvDetails>   tEvDetTbl;


/* ==========================================================
 *  Classes (forward decl) & Structures (defn)
 * ========================================================== */


/* ==========================================================
 *  Module Functions (decl.)     (static local only)
 * ========================================================== */


/* ==========================================================
 *  Global Variables (defn.)     (used externally)
 * ========================================================== */


/* ==========================================================
 *  Local Module Variables (defn.)  (static local only)
 * ========================================================== */



/* ======================================================================
 * ======================================================================
 *  Functions & class definitions visible externally
 * ====================================================================== */



/* ======================================================================
 * ======================================================================
 *  Functions & class definitions internal to module
 * ====================================================================== */
/* Function and class header prototypes */
void  IntrHdlr (int aSignal);
void  TermHdlr (int aSignal);

/* ******************************************************
 * Function : IntrHdlr
 * Description : interrupt handler function
 * Parameters  :
 * Returns  :
```

```
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
IntrHdlr(int aSignal)
{
  // ^c interrupts polling in Invoke()
  // poll returns -1 and hence breaks out
  // nothing else needs to be done here
  DBG(endl << "received signal ^c ....." << endl);
}

/* ******************************************************
 * Function : TermHdlr
 * Description : term handler function
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
TermHdlr(int aSignal)
{
  DBG(endl << "received term signal ....." << endl);
}

/* ******************************************************
 * Class : tCmpSched
 * Description :
 * ******************************************************
 * NOTES:   inherits tComponent and interface tIfSched
 * ****************************************************** */
class tCmpSched
  : public tComponent,
    public tIfSched
{
public:
  tCmpSched()
    : saPollTbl(0),
      sPollTblSize(0),
      sDefInstName("sched")
  {
    DBG("Creating Scheduler");
  }

  ~tCmpSched()
  {
    DBG("Deleting Scheduler");
  }

  void AddCmp(tComponent * apCmp);
  void AddPort(tIfDataPort * apPort);

  void RunCmp();
  void RunPort();

  void Invoke();
  bool RegEvHdl(tEvDetails &  arDetails);
  string GetDefName();
  tComponent * GetDefInst();
  bool   InternalTask(char * apBuffer, unsigned int aCount);


private:
  tCmpList      sCmpList;
  tPortList     sPortList;
```

```
  string        sName;
  tComponent *  spCmp;

  // File descriptor stuff
  tEvDetTbl sFdWatch;
  pollfd *  saPollTbl;
  int    sPollTblSize;
  string        sDefInstName;
};


/* ******************************************************
 * Function : AddCmp
 * Description : Adds one component at a time to list
 * Parameters  : tComponent * apCmp
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpSched::AddCmp(tComponent * apCmp)
{
//   DBG("adding component to list");
  //add to end of list
  sCmpList.push_back(apCmp);
}

/* ******************************************************
 * Function : AddPort
 * Description : Adds one port at a time to list
 * Parameters  : tIfDataPort * apPort
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpSched::AddPort(tIfDataPort * apPort)
{
  DBG("adding port to list");
  //add to end of list
  sPortList.push_back(apPort);
}


/* ******************************************************
 * Function : RunCmp()
 * Description : runs all components from list
 * Parameters  : none
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpSched::RunCmp()
{
//   tCmpList::iterator lCmpIter;
//   tComponent *    lpComponent;
//   unsigned int          lCmpCount = 0;

  for (tCmpList::iterator  lCmpIter = sCmpList.begin();
       lCmpIter != sCmpList.end();
       lCmpIter++)
    {
//       DBG("Invoking Component" << ": " << ++lCmpCount);
      (*lCmpIter)->Invoke();
    }
```

```
}


/* ******************************************************
 * Function : RunPort()
 * Description : runs all port from list
 * Parameters  : none
 * Returns  : void
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpSched::RunPort()
{
  unsigned int          lPortCount = 0;

  for (tPortList::iterator  lPortIter = sPortList.begin();
       lPortIter != sPortList.end();
       lPortIter++)
    {
      DBG("Invoking port" << ": " << ++lPortCount);
//       (*lPortIter)->WakePort();
    }
}


/* ******************************************************
 * Function : RegEvHdl()
 * Description : registers an event handler
 * Parameters  : tEvDetails &
 * Returns  : bool
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpSched::RegEvHdl(tEvDetails &   arDetails)
{
  // call back

  tEvDetTbl::const_iterator   lIter;
  int          lIdx = 0;

  // Validate details
  if ( (arDetails.sEvType != tEvDetails::eEctFdRead)
       && (arDetails.sEvType != tEvDetails::eEctFdWrite) )
    {
      // Invalid event type
      return (false);
    }

  // Check for duplicate entry
  for (tEvDetTbl::const_iterator lIter = sFdWatch.begin();
       lIter != sFdWatch.end();
       ++lIter)
    {
      if ( (lIter->sEvType == arDetails.sEvType)
      && (lIter->sFiledes == arDetails.sFiledes) )
   {
     // Existing entry
     DBG("duplicate entry");
     return (false);
   }
    }

  // Add to watch table
  sFdWatch.push_front(arDetails);
```

```
  // Rebuild poll table
  if (sPollTblSize != 0)
    {
      // Free old table
      delete (saPollTbl);
    }

  sPollTblSize = sFdWatch.size();

  cout << "sPollTblSize = " << sPollTblSize << endl;

  saPollTbl = new pollfd[sPollTblSize];
  for (lIter = sFdWatch.begin(), lIdx = 0;
       lIter != sFdWatch.end();
       ++lIter, ++lIdx)
    {
      // Setup each poll entry
      saPollTbl[lIdx].fd = lIter->sFiledes;
      switch (lIter->sEvType)
   {
   case tEvDetails::eEctFdRead:
     saPollTbl[lIdx].events = POLLIN | POLLPRI;
     break;
   case tEvDetails::eEctFdWrite:
     saPollTbl[lIdx].events = POLLOUT;
     break;
   default:
     break;
   }
      saPollTbl[lIdx].revents = 0;
    }

  DBG("Event registered ");
  return (true);
}


/* ******************************************************
 * Function : Invoke
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
void
tCmpSched::Invoke()
{
  int          lPollStatus;
//   unsigned int          lCmpCount = 0;
  unsigned int          lPortCount = 0;

  // signal handlers for ^c or term interrupts
  signal(SIGINT, IntrHdlr);
  signal(SIGTERM, TermHdlr);

  // Main loop
  while (1)
    {
      ///
      /// Service asyncronous events
      ///

      // File descriptors check
      lPollStatus = poll(saPollTbl, sPollTblSize, 10); // 10 ms max
```

```
      // check for changes in revents - invoke that component
      if (lPollStatus == 0)
   {
     // Timeout
   }
      else if (lPollStatus == -1)
   {
     // Error
     DBG("break in poll, " << strerror(errno));
     break;
   }
      else
   {
     // service components
     tEvDetTbl::const_iterator   lIter = sFdWatch.begin();
     int          lIdx = 0;

     while (lPollStatus > 0)
       {
         if ((saPollTbl[lIdx].revents != 0) )
      {
        // Activity
//         DBG("Scheduler EvCbInvoke()");
        lIter->spObj->EvCbInvoke(lIter->spData);
        --lPollStatus;

      }
         ++lIter;
         ++lIdx;
       }
   }

      // Timers

      // Signals


      // Service components

       for (tCmpList::iterator  lCmpIter = sCmpList.begin();
       lCmpIter != sCmpList.end();
       lCmpIter = sCmpList.begin())
    {
      //service each component from list
//       DBG("Invoking Component" << ": " << ++lCmpCount);
      (*lCmpIter)->Invoke();

      //remove entry when control returns
      sCmpList.erase(lCmpIter);

    }

        // Service ports

       for (tPortList::iterator  lPortIter = sPortList.begin();
       lPortIter != sPortList.end();
       lPortIter = sPortList.begin())
    {
      //service each component from list
      DBG("Invoking Port" << ": " << ++lPortCount);
//       (*lPortIter)->WakePort();

      //remove entry when control returns
      sPortList.erase(lPortIter);
```

```
    }
    }
}

/* ******************************************************
 * Function : GetDefName
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
string
tCmpSched::GetDefName()
{
  return sDefInstName;
}

 /* ******************************************************
 * Function : GetDefInst
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
tComponent *
tCmpSched::GetDefInst()
{
  return (this);
}


/* ******************************************************
 * Function : InternalTask
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
bool
tCmpSched::InternalTask(char * apBuffer, unsigned int aCount)
{
  DBG("default method");
  return (true);

}


/* ======================================================================
 * ======================================================================
 *  Entry Points
 * ====================================================================== */
extern "C"  tComponent * CreateCmp_Scheduler()
{
  return (new tCmpSched);
}
```

```
/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

## 1.2 Component StdIo

```
/* ********************************************************************
 *
 *   Source File Name   : CmpStdIo.cc
 *
 *   Module Name  : CmpStdIo
 *
 *   Application Name   : TP1
 *
 *   Project Name : TCS_01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes    (normal system)
 * ======================================================== */
#include <unistd.h>
#include <errno.h>
#include <string.h>

#include <iostream>


/* ========================================================
```

```
 *  External Includes        (external toolkits)
 * ========================================================= */


/* =========================================================
 *  Project-wide Includes    (project only)
 * ========================================================= */


/* =========================================================
 *  Module Includes       (module only)
 * ========================================================= */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfEvCb.hh"
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/Iface/IfSched.hh"
#include "../../../inc/Iface/IfCmpPorts.hh"
#include "../../../inc/ByteIn.hh"
#include "../../../inc/ByteOut.hh"


/* =========================================================
 *  Module #DEFINEs
 * ========================================================= */


/* =========================================================
 *  Enumerations & Other Typedefs (defn)
 * ========================================================= */


/* =========================================================
 *  Classes (forward decl) & Structures (defn)
 * ========================================================= */


/* =========================================================
 *  Module Functions (decl.)    (static local only)
 * ========================================================= */


/* =========================================================
 *  Global Variables (defn.)    (used externally)
 * ========================================================= */


/* =========================================================strerror
 *  Local Module Variables (defn.) (static local only)
 * ========================================================= */



/* =====================================================================
 * =====================================================================
 *  Functions & class definitions visible externally
 * ===================================================================== */




/* =====================================================================
 * =====================================================================
 *  Functions & class definitions internal to module
 * ===================================================================== */

/* Function and class header prototypes */
```

```
/* ******************************************************
 * Class : tCmpStdIo
 * Description :
 * ******************************************************
 * NOTES:   inherits tComponent and interface tIfStdIo
 * ****************************************************** */
class tCmpStdIo
  : public tComponent,
    public tIfEvCb,
    public tIfCmpPorts
{
public:
  tCmpStdIo()
    : sIn(this, NULL, ReadCallback),
      sOut(this),
      sInstName()
  {
    tIfLoader *      lpCmpLoader;
    tIfSched  *   lpCmpSched;
    tEvDetails    lEvDetails(tEvDetails::eEctFdRead, 0, this, (void*)0);

    DBG("Creating tCmpStdIo");

    // Setup portmap
    sPortMap.insert(make_pair(string("stdin"), &sOut));
    sPortMap.insert(make_pair(string("stdout"), &sIn));

    // get loader
    lpCmpLoader = tIfLoader::GetLoader();

    // query for Scheduler + check existence of Sched
    lpCmpSched = dynamic_cast<tIfSched  *>(lpCmpLoader->InstQuery("sched") );

    if (lpCmpSched == NULL)
    {
      throw (string("lpCmpSched: cross cast failed"));
    }

    // register itself with scheduler
    lpCmpSched->RegEvHdl(lEvDetails);
  }

  const tDataPortNameMap & CmpPortMapGet()
  {
    return (sPortMap);
  }

  void Invoke()
  {
    DBG("hello from tCmpStdIo");
  }



  bool EvCbInvoke(void * apData)
  {
    char lBuffer[256];
    size_t  lCount;

//     DBG("waiting on stdin.......");
    lCount = read(0, lBuffer, 256);
    if (lCount > 0)
      {
   // Got some bytes
   sOut.Transfer(lBuffer, lCount);
```

```
//    DBG("transferring from CmpStdIo ...");

      }
    else if (lCount == 0)
      {
   // Stdin closed
   DBG("stdin closed");
   return (false);
      }
    else
      {
   // Error - log it
   DBG("stdin error: " << strerror(errno));
   return (false);
      }

    return (true);
  }

  bool Read(char *      apBuffer,
       unsigned int  aCount)
  {
    cout << string(apBuffer, aCount) << endl;
    return (true);
  }

  bool InternalTask (char * apBuffer, unsigned int aCount)
  {
    DBG("no internal task");
    return (true);
  }

private:
  static bool ReadCallback(tComponent *   apCmp,
            char *   apBuffer,
            unsigned int   aCount)
  { return (dynamic_cast<tCmpStdIo *>(apCmp)->Read(apBuffer, aCount) ); }

  tByteIn      sIn;
  tByteOut     sOut;
  tDataPortNameMap   sPortMap;
  string    sInstName;

};

/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ==================================================================== */
extern "C"  tComponent * CreateCmp_StdIo()
{
  return (new tCmpStdIo);
}


/* ============ *
 *     EOF      *
 * ============ */


/* Function and class header prototypes */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
```

```
 * Returns  :
 * ********************************************************
 * NOTES:
 * ******************************************************** */

/* ********************************************************
 * Class :
 * Description :
 * ********************************************************
 * NOTES:
 * ******************************************************** */
```

## 1.3 Tcl-C Interface Sources

```
/* ************************************************************************
 *
 *   Source File Name   : CmpTcl_Loader.cc
 *
 *   Module Name  : CmpTcl_Loader
 *
 *   Application Name   : TP01
 *
 *   Project Name : TCS01
 *
 * ************************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ************************************************************************
 *  NOTES:


 *  END OF NOTES
 * ======================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes    (normal system)
 * ======================================================== */


/* ========================================================
 *  External Includes         (external toolkits)
 * ======================================================== */
#include <tcl.h>
// #include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

/* ========================================================
 *  Project-wide Includes     (project only)
 * ======================================================== */


/* ========================================================
 *  Module Includes        (module only)
 * ======================================================== */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfLoader.hh"
```

```
#include "../../../inc/TclCmp_LookupCmd.hh"
#include "../../../inc/TclCmp_GetLoader.hh"


/* ======================================================
 *  Module #DEFINEs
 * ====================================================== */
#define FILE_NAME_LOADER_CC   "CmpTcl_Loader.cc"

/* ======================================================
 *  Enumerations & Other Typedefs (defn)
 * ====================================================== */


/* ======================================================
 *  Classes (forward decl) & Structures (defn)
 * ====================================================== */


/* ======================================================
 *  Module Functions (decl.)    (static local only)
 * ====================================================== */


/* ======================================================
 *  Global Variables (defn.)    (used externally)
 * ====================================================== */


/* ======================================================
 *  Local Module Variables (defn.) (static local only)
 * ====================================================== */
static tTclCmpCmdMap<tIfLoader>    mCmdMap;


/* ====================================================================
 * ====================================================================
 *  Functions & class definitions visible externally
 * ==================================================================== */

/* ====================================================================
 * ====================================================================
 *  Functions & class definitions internal to module
 * ==================================================================== */

/* ******************************************************
 * Function : ReadTypeRegFile
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_ReadTypeRegFile(tIfLoader *  apIface,
             Tcl_Interp *  apInterp,
             int     aObjc,
             Tcl_Obj *CONST   aapObjv[])
{
  apIface = tIfLoader::GetLoader();
  if (apIface == NULL)
    {
      throw (string ("could not get loader interface"));
    }

  if (aObjc < 1)
```

```
    {
      throw (string (FILE_NAME_LOADER_CC) +
        string(": invalid number of parameters"));
    }

  // quick check to ensure file presence in file space
  int lFile = open((const char *)Tcl_GetStringFromObj(aapObjv[0], 0),
        0, O_RDONLY);

  if (lFile >= 0)
    { // close file and continue
       if (close(lFile) != 0)
    {
      cout << string(Tcl_GetStringFromObj(aapObjv[0], 0))
      << " file could not be closed" << endl;
    }
    }
  else
  {
    throw(string("file '") + string(Tcl_GetStringFromObj(aapObjv[0], 0))
      + string("' not found") );
  }

  apIface->ReadTypeRegFile(string(Tcl_GetStringFromObj(aapObjv[0], 0)) );

  return (TCL_OK);
}


/* ******************************************************
 * Function : TclCmp_InstCreate
 * Description : create the specified cmp instance
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_InstCreate(tIfLoader *    apIface,
        Tcl_Interp *    apInterp,
        int          aObjc,
        Tcl_Obj *CONST  aapObjv[])
{
  tComponent * lpCmp;
  apIface = tIfLoader::GetLoader();

  if (aObjc < 2)
   {
     throw (string (FILE_NAME_LOADER_CC) +
       string("invalid number of parameters"));
   }

  if (apIface == NULL)
    {
      throw (string ("could not get loader interface") );
    }

  lpCmp = apIface->InstCreate(string(Tcl_GetStringFromObj(aapObjv[0], 0)),
               string(Tcl_GetStringFromObj(aapObjv[1], 0)) );

  if (lpCmp == NULL)
    {
      throw (string("could not create '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("' : '") +
        Tcl_GetStringFromObj(aapObjv[1], 0) +
```

```
        string("' instance") );
    }

  return (TCL_OK);
}

/* ******************************************************
 * Function : TclCmp_InstQuery
 * Description : check existence of component instance
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_InstQuery(tIfLoader *  apIface,
       Tcl_Interp *  apInterp,
       int     aObjc,
       Tcl_Obj *CONST   aapObjv[])
{
  tComponent * lpCmp;
  apIface = tIfLoader::GetLoader();

  if (aObjc < 1)
   {
     throw (string (FILE_NAME_LOADER_CC) +
       string("invalid number of parameters"));
   }
//   cout << "aapObjv[0] = " << string(Tcl_GetStringFromObj(aapObjv[0], 0))
//        << "aapObjv[1] = " << string(Tcl_GetStringFromObj(aapObjv[1], 0))
//        << endl;

  lpCmp = apIface->InstQuery(string(Tcl_GetStringFromObj(aapObjv[0], 0)) );

  if (!lpCmp)
    {
      cout << "false" << endl;
      throw (string("Instance '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) + "' " +
        string("does not exist") );
    }

  cout << "true" << endl;
  return (TCL_OK);
}


/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_InstDelete(tIfLoader *    apIface,
       Tcl_Interp *    apInterp,
       int          aObjc,
       Tcl_Obj *CONST  aapObjv[])
{
  apIface = tIfLoader::GetLoader();

  if (aObjc < 1)
   {
     throw (string (FILE_NAME_LOADER_CC) +
```

```
        string("invalid number of parameters"));
    }

  apIface->InstDelete(string(Tcl_GetStringFromObj(aapObjv[0], 0)));

  return (TCL_OK);
}



/* ========================================================================
 * ========================================================================
 *  Entry Points
 * ======================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
extern "C"
int
Cmptcl_loader_Init(Tcl_Interp * apInterp)
{
  // Populate sub-command map
  mCmdMap.AddCmd((const char *)"ReadTypeRegFile", TclCmp_ReadTypeRegFile);
  mCmdMap.AddCmd((const char *)"InstCreate", TclCmp_InstCreate);
  mCmdMap.AddCmd((const char *)"InstQuery", TclCmp_InstQuery);
  mCmdMap.AddCmd((const char *)"InstDelete", TclCmp_InstDelete);

  // New interpreter commands
  TclCmp_RegCmdMap(apInterp, "IfLoader", mCmdMap);

  return (TCL_OK);
}




/* ========== *
 *     EOF    *
 * ========== */


/* Function and class header prototypes */
#if 0

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class
```

```
{
public:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

protected:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

private:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

};

#endif

Component Router
/* ********************************************************************
 *
 *   Source File Name   : CmpTcl_Router.cc
 *
 *   Module Name  : CmpTcl_Router
 *
 *   Application Name   : TP01
 *
 *   Project Name : TCS01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes   (normal system)
 * ======================================================== */


/* ========================================================
 *  External Includes        (external toolkits)
 * ======================================================== */
#include <tcl.h>


/* ========================================================
 *  Project-wide Includes    (project only)
```

```
 * ========================================================== */


/* ==========================================================
 *  Module Includes       (module only)
 * ========================================================== */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfRouter.hh"
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/TclCmp_LookupCmd.hh"



/* ==========================================================
 *  Module #DEFINEs
 * ========================================================== */
#define FILE_NAME_ROUTER_CC   "CmpTcl_Router.cc"


/* ==========================================================
 *  Enumerations & Other Typedefs (defn)
 * ========================================================== */


/* ==========================================================
 *  Classes (forward decl) & Structures (defn)
 * ========================================================== */


/* ==========================================================
 *  Module Functions (decl.)    (static local only)
 * ========================================================== */


/* ==========================================================
 *  Global Variables (defn.)    (used externally)
 * ========================================================== */


/* ==========================================================
 *  Local Module Variables (defn.) (static local only)
 * ========================================================== */
static tTclCmpCmdMap<tIfRouter>    mCmdMap;


/* ======================================================================
 * ======================================================================
 *  Functions & class definitions visible externally
 * ====================================================================== */



/* ======================================================================
 * ======================================================================
 *  Functions & class definitions internal to module
 * ====================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

```
static int
TclCmp_Bind(tIfRouter * apIface,
     Tcl_Interp * apInterp,
     int       aObjc,
     Tcl_Obj *CONST  aapObjv[])
{
    if (aObjc < 3)
    {
      throw (string (FILE_NAME_ROUTER_CC) +
        string(": invalid number of parameters"));
    }

    if (!apIface->Bind(string(Tcl_GetStringFromObj(aapObjv[0], 0)),
             string(Tcl_GetStringFromObj(aapObjv[1], 0)),
             string(Tcl_GetStringFromObj(aapObjv[2], 0)),
             string(Tcl_GetStringFromObj(aapObjv[3], 0)) ) )
      {
   throw(string("could not Bind cmps & ports") );
      }

  return (TCL_OK);
}


/* ======================================================================
 * ======================================================================
 *  Entry Points
 * ====================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
extern "C"
int
Cmptcl_router_Init(Tcl_Interp * apInterp)
{
  // Populate sub-command map
  mCmdMap.AddCmd((const char *)"Bind", TclCmp_Bind);

  // New interpreter commands
  TclCmp_RegCmdMap(apInterp, "IfRouter", mCmdMap);

  return (TCL_OK);
}



/* =========== *
 *     EOF     *
 * =========== */


/* Function and class header prototypes */
#if 0

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
```

```
 * ******************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class
{
public:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

protected:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

private:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

};

#endif
Component Scheduler
/* ********************************************************************
 *
 *   Source File Name   : CmpTcl_Sched.cc
 *
 *   Module Name  : CmpTcl_Sched
 *
 *   Application Name   : TP01
 *
 *   Project Name : TCS01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ======================================================
 *  Standard Library Includes   (normal system)
 * ====================================================== */
```

```
/* ============================================================
 *  External Includes        (external toolkits)
 * ============================================================ */
#include <tcl.h>


/* ============================================================
 *  Project-wide Includes    (project only)
 * ============================================================ */


/* ============================================================
 *  Module Includes       (module only)
 * ============================================================ */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfSched.hh"
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/TclCmp_LookupCmd.hh"



/* ============================================================
 *  Module #DEFINEs
 * ============================================================ */
#define FILE_NAME_SCHED_CC "CmpTcl_Sched.cc"


/* ============================================================
 *  Enumerations & Other Typedefs (defn)
 * ============================================================ */


/* ============================================================
 *  Classes (forward decl) & Structures (defn)
 * ============================================================ */


/* ============================================================
 *  Module Functions (decl.)    (static local only)
 * ============================================================ */


/* ============================================================
 *  Global Variables (defn.)    (used externally)
 * ============================================================ */


/* ============================================================
 *  Local Module Variables (defn.) (static local only)
 * ============================================================ */
static tTclCmpCmdMap<tIfSched>     mCmdMap;


/* ============================================================================
 * ============================================================================
 *  Functions & class definitions visible externally
 * ============================================================================ */



/* ============================================================================
 * ============================================================================
 *  Functions & class definitions internal to module
```

```
 * ========================================================================= */


/* ******************************************************
 * Function : TclCmp_AddCmp
 * Description : Add component to scheduler list
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_AddCmp(tIfSched *   apIface,
         Tcl_Interp *   apInterp,
         int      aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  tComponent * lpCmp;
  tIfLoader *  lpLoader = tIfLoader::GetLoader();
  tIfSched *   lpSchedIface;
  if (aObjc < 1)
   {
     throw (string(FILE_NAME_SCHED_CC) +
       string("invalid number of parameters"));
   }

  lpCmp = lpLoader->InstQuery(string(Tcl_GetStringFromObj(aapObjv[0], 0)) );

  if (lpCmp == NULL)
    {
      throw (string("invalid component name '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) + "'");
    }

  // do not add scheduler - test for type and throw for sched type
  if ((lpSchedIface = dynamic_cast<tIfSched *>(lpCmp)) )
    {
      throw (string("cannot add component'") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("' to itself ") );
    }

  apIface->AddCmp(lpCmp);

  return (TCL_OK);
}

/* ******************************************************
 * Function : TclCmp_RunCmp
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_RunCmp(tIfSched *   apIface,
         Tcl_Interp *   apInterp,
         int      aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  apIface->RunCmp();

  return (TCL_OK);
}
```

```
/* ******************************************************
 * Function : RegEvHdl
 * Description : register event handle to invoke event
 *       later
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_RegEvHdl(tIfSched * apIface,
      Tcl_Interp *   apInterp,
      int      aObjc,
      Tcl_Obj *CONST aapObjv[])
{
  tComponent * lpCmp;
  tIfLoader *  lpLoader = tIfLoader::GetLoader();

  if (aObjc < 1)
    {
      throw (string(FILE_NAME_SCHED_CC) +
        string("invalid number of parameters"));
    }

  Tcl_Obj *    lpPtrToList = Tcl_ObjGetVar2(apInterp,
                      aapObjv[0],
                      NULL,
                      TCL_LEAVE_ERR_MSG);
  int       lListElementCount_Argc;
  Tcl_Obj **      lpListElement_Argv;

  // EvDetails
  tEvDetails::tEvCbType lEvType;
  int       lFiledes;
  tIfEvCb *    lpObj;
  void *    lpData;

  if (!lpPtrToList)
    {
      throw (string("no list '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("' found") );
    }

  if (Tcl_ListObjGetElements(apInterp, lpPtrToList,
            &lListElementCount_Argc, &lpListElement_Argv) != TCL_OK)
    {
      throw (string("could not GetElements from list"));
    }

//   DBG("lpListElement_Argv[0] = " << Tcl_GetStringFromObj(lpListElement_Argv[0], 0) );
//   DBG("lpListElement_Argv[1] = " << Tcl_GetStringFromObj(lpListElement_Argv[1], 0) );
//   DBG("lpListElement_Argv[2] = " << Tcl_GetStringFromObj(lpListElement_Argv[2], 0));
//   DBG("lpListElement_Argv[3] = " << Tcl_GetStringFromObj(lpListElement_Argv[3], 0));


  // strip elements from list and make up arguments for tEvDetails
  // tEvDetails[arg 1, , , ]

  if (strcmp(Tcl_GetStringFromObj(lpListElement_Argv[0], 0), "None") == 0)
    {
      lEvType = tEvDetails::eEctNone;
    }

  if (strcmp(Tcl_GetStringFromObj(lpListElement_Argv[0], 0), "FdRead") == 0)
```

```
    {
      lEvType = tEvDetails::eEctFdRead;
    }

  if (strcmp(Tcl_GetStringFromObj(lpListElement_Argv[0], 0), "FdWrite") == 0)
    {
      lEvType = tEvDetails::eEctFdWrite;
    }

  // tEvDetails[ ,arg 2, , ]
  Tcl_GetIntFromObj(apInterp, lpListElement_Argv[1], &lFiledes);

  if (Tcl_GetObjResult == NULL)
    {
      throw (string("could not get int form obj ") +
        Tcl_GetStringFromObj(lpListElement_Argv[1], 0) );
    }

  // tEvDetails[ , ,arg 3, ]
  lpCmp = lpLoader->InstQuery(string(Tcl_GetStringFromObj(lpListElement_Argv[2], 0)) );
  lpObj = dynamic_cast<tIfEvCb *>(lpCmp);

  if (lpObj == NULL)
    {
      throw (string("could not cast '") +
        Tcl_GetStringFromObj(lpListElement_Argv[2], 0) + "'" +
        string(" to tIfEvCb type") );
    }

   // tEvDetails[ , , ,arg 4]
  lpData = Tcl_GetStringFromObj(lpListElement_Argv[3], 0);

 // fill up arguments
  tEvDetails   lEvDetails(lEvType, lFiledes, lpObj, (void*)lpData);

  // call Register ev handle
  if (!apIface->RegEvHdl(lEvDetails))
    {
      throw (string("could not register ") +
        string(Tcl_GetStringFromObj(lpListElement_Argv[2], 0)) +
        string(" with scheduler ") );
    }

  return (TCL_OK);
}

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_GetDefName(tIfSched *    apIface,
        Tcl_Interp *    apInterp,
        int          aObjc,
        Tcl_Obj *CONST  aapObjv[])
{
  Tcl_SetResult(apInterp, (char *)(apIface->GetDefName()).c_str(), TCL_VOLATILE);
  return (TCL_OK);
}


/* ******************************************************
```

```
 * Function : Invoke
 * Description : tComponent level invoke
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES: required to invoke main loop in Sched
 * ****************************************************** */
static int
TclCmp_Invoke(tIfSched *      apIface,
         Tcl_Interp *      apInterp,
         int         aObjc,
         Tcl_Obj *CONST    aapObjv[])
{
  tComponent * lpCmp;
  lpCmp = dynamic_cast<tComponent *>(apIface);
  if (lpCmp == NULL)
    {
      throw (string("could not downcast sched Iface to tComponent"));
    }
  lpCmp->Invoke();


  return (TCL_OK);
}
/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ==================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
extern "C"
int
Cmptcl_sched_Init(Tcl_Interp * apInterp)
{
  // Populate sub-command map
  mCmdMap.AddCmd((const char *)"AddCmp", TclCmp_AddCmp);
  mCmdMap.AddCmd((const char *)"RunCmp", TclCmp_RunCmp);
  mCmdMap.AddCmd((const char *)"RegEvHdl", TclCmp_RegEvHdl);
  mCmdMap.AddCmd((const char *)"GetDefName", TclCmp_GetDefName);
  mCmdMap.AddCmd((const char *)"Invoke", TclCmp_Invoke);

  // New interpreter commands
  TclCmp_RegCmdMap(apInterp, "IfSched", mCmdMap);

  return (TCL_OK);
}









/* ========== *
 *     EOF    *
 * ========== */
```

```
/* Function and class header prototypes */
#if 0

/* ********************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ********************************************************
 * NOTES:
 * ******************************************************** */

/* ********************************************************
 * Class :
 * Description :
 * ********************************************************
 * NOTES:
 * ******************************************************** */
class
{
public:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

protected:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

private:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

};

#endif
```

## 1.4 Component Ethgen

```
/* ********************************************************************
 *
 *   Source File Name   : CmpTcl_EthGen.cc
 *
 *   Module Name  : CmpTcl_EthGen
 *
 *   Application Name   : TP01
 *
 *   Project Name : TCS01
 *
```

```
 * ======================================================================
 * (c)2001 Seven Layer Communications Ltd.
 * ======================================================================
 *  NOTES:


 *  END OF NOTES
 * ====================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ========================================================
 *  Standard Library Includes    (normal system)
 * ======================================================== */


/* ========================================================
 *  External Includes         (external toolkits)
 * ======================================================== */
#include <tcl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <typeinfo>


/* ========================================================
 *  Project-wide Includes     (project only)
 * ======================================================== */


/* ========================================================
 *  Module Includes        (module only)
 * ======================================================== */
#include "../../../inc/Iface/IfEthGen.hh"
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/TclCmp_LookupCmd.hh"


/* ========================================================
 *  Module #DEFINEs
 * ======================================================== */
#define FILE_NAME_ETHGEN_CC   "CmpTcl_EthGen.cc"
#define MAX_FRAMES_PER_SEQ 64

/* ========================================================
 *  Enumerations & Other Typedefs (defn)
 * ======================================================== */
typedef map<string, tEthGenFrame>  tEthGenFrameMap;

/* ========================================================
 *  Classes (forward decl) & Structures (defn)
 * ======================================================== */


/* ========================================================
 *  Module Functions (decl.)     (static local only)
 * ======================================================== */


/* ========================================================
 *  Global Variables (defn.)     (used externally)
 * ======================================================== */
```

```
/* ========================================================
 *  Local Module Variables (defn.)  (static local only)
 * ======================================================== */
static tTclCmpCmdMap<tIfEthGen>    mCmdMap;


/* ========================================================================
 * ========================================================================
 *  Functions & class definitions visible externally
 * ======================================================================== */




/* ========================================================================
 * ========================================================================
 *  Functions & class definitions internal to module
 * ======================================================================== */
unsigned int   ConvertHexToInt(char * aInb, int aLen, char * aOutb);
unsigned int   ConvertIntToHex(char * aInb, int aLen, char * aOutb);
int      GetIntFromObj(char * aVal);

/* ******************************************************
 * Function : TclCmp_QueryState
 * Description : Query internal states of EthGen component
 * Parameters  : none
 * Returns  : TCL_OK [Interp set to state description]
 * ******************************************************
 * NOTES: TclCmd % IfEthGen ethgen QueryState
 * ****************************************************** */
static int
TclCmp_QueryState(tIfEthGen *    apIface,
        Tcl_Interp *    apInterp,
        int          aObjc,
        Tcl_Obj *CONST  aapObjv[])
{
  Tcl_SetResult(apInterp, (char *)(apIface->QueryState()).c_str(), TCL_VOLATILE);
  return (TCL_OK);
}

/* ******************************************************
 * Function : TclCmp_SetInterface
 * Description : sets ethernet interface to <name>
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES: TclCmd % IfEthGen ethgen <IfName>
 * ****************************************************** */
static int
TclCmp_SetInterface(tIfEthGen *    apIface,
         Tcl_Interp *      apInterp,
         int          aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  if (aObjc < 1)
   {
     throw (string (FILE_NAME_ETHGEN_CC) +
       string(": IfEthGen ethgen SetInterface <Name>"));
   }

  if (!(apIface->SetInterface(string(Tcl_GetStringFromObj(aapObjv[0], 0)))))
    {
      throw (string("Could not set interface '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
```

```
        string("'") );
    }
  else
    {
      Tcl_SetResult(apInterp, Tcl_GetStringFromObj(aapObjv[0], 0), TCL_VOLATILE);
    }

  return (TCL_OK);
}

/* ******************************************************
 * Function : TclCmp_QueryInterface
 * Description : query ethernet interface name
 * Parameters  : none
 * Returns  : string
 * ******************************************************
 * NOTES: TclCmd % IfEthGen ethgen QueryInterface
 * ****************************************************** */
static int
TclCmp_QueryInterface(tIfEthGen *     apIface,
            Tcl_Interp *      apInterp,
            int         aObjc,
            Tcl_Obj *CONST aapObjv[])
{
  Tcl_SetResult(apInterp, (char *)(apIface->QueryInterface()).c_str(), TCL_VOLATILE);
  return (TCL_OK);
}

/* ******************************************************
 * Function : TclCmp_Config
 * Description : Configure all channels
 * Parameters  : Sequence/Frame list
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> Config <bstsfile> <seq/frame list>
 *
 * Only buffer length of 1514 (max) can be handled
 * ****************************************************** */
static int
TclCmp_Config(tIfEthGen *  apIface,
         Tcl_Interp *   apInterp,
         int      aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  DBG("----- 1 --------");

  // create a map to avoid copying frames with same names
  tEthGenFrameMap    lFrameRegister_Map;
  tEthGenFrameMap::iterator   lFrameNameIter;
  tEthGenFrameMap::iterator   lFrameNameIter2;

  // frame vectors and indiv seqs for all channels
  tEthGenFrameVec    lFrameVector[NO_OF_CHANNELS];
  tEthGenSeq         lSeq[NO_OF_CHANNELS];

  // available procs in global scope
  string    lProc_Bsts_GetPattern = "Bsts_GetPattern";
  string    lProc_Bsts_GetFrameData = "Bsts_GetFrameData";
  string    lProc_FindFile =  "FindFile";

  // storage for Sequence elements
  Tcl_Obj *    lSeqObj[MAX_FRAMES_PER_SEQ];

  int       lSeqElementCount = 0;
  Tcl_Obj    **      lppSeqListElement;
```

```
  int          lNumOfFramesInSeq = 0;

  // storage for Frame elements
  int       lFrameElementCount = 0;
  Tcl_Obj    **      lppFrameListElement;

  tChannelArray      lChannelArray;

  char         lBuf[ETH_MAX_LEN];

  DBG("----- 2 --------");

  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen Config <bstsfile> <seq/frame> .. .. "));
    }

  // Find file containing PDUs and sequences
  if (Tcl_Eval(apInterp,
             (char *)(lProc_FindFile
           + string(" ")
           + string(Tcl_GetStringFromObj(aapObjv[0], 0))
           ).c_str())
        != TCL_OK)
    {
         throw(string("could not find file '") +
          string (Tcl_GetStringFromObj(aapObjv[0], 0)) +
          string ("'"));
    }

  // 8 channels plus "Config"
  if (aObjc > NO_OF_CHANNELS + 1)
    {
      DBG("! found more than " << NO_OF_CHANNELS << " channels");
      DBG("! ignoring trailing channels '" <<
Tcl_GetStringFromObj(aapObjv[NO_OF_CHANNELS+1], 0) << " ...");
    }

  DBG("----- 3 --------");

  // set all channels to NULL at start
    for (unsigned int i = 0; i < NO_OF_CHANNELS; ++i)
    {
      lChannelArray[i] = NULL;
    }

  for (int lChannel = 0; lChannel < (aObjc - 1); ++lChannel)
    {
      DBG("----- 3 . " << lChannel << "  --------");

      if (strcmp(Tcl_GetStringFromObj(aapObjv[lChannel+1], 0), "null") == 0)
   {
//      DBG("set channel [" << lChannel << "] = NULL");
     lChannelArray[lChannel] = NULL;
   }
      else
   {
     DBG("----- 3 . " << lChannel << " . 0 --------");

     // get list of sequences
//      if (Tcl_Eval(apInterp,
//              (char *)(lProc_Bsts_GetPattern
//            + string(" ")
//            + string(Tcl_GetStringFromObj(aapObjv[0], 0))
//            + string(" ")
```

```
//             + string(Tcl_GetStringFromObj(aapObjv[lChannel+1], 0))).c_str())
//          != TCL_OK)
//        {
//          throw(string("Could not read PDU/Sequence"));
//        }
//      else
//        {
         DBG("----- 3 . " << lChannel << " . 1 --------");

         DBG((char *)(lProc_Bsts_GetPattern
            + string(" ")
            + string(Tcl_GetStringFromObj(aapObjv[0], 0))
            + string(" ")
            + string(Tcl_GetStringFromObj(aapObjv[lChannel+1], 0))).c_str());

         Tcl_EvalEx(apInterp,
          (char *)(lProc_Bsts_GetPattern
              + string(" ")
              + string(Tcl_GetStringFromObj(aapObjv[0], 0))
              + string(" ")
              + string(Tcl_GetStringFromObj(aapObjv[lChannel+1], 0))).c_str(),
          -1,
          TCL_EVAL_GLOBAL);


         DBG("----- 3 . " << lChannel << " . 2 --------");


         // for sequence 1-8
         if (Tcl_ListObjGetElements(apInterp,  Tcl_GetObjResult(apInterp),
                &lSeqElementCount, &lppSeqListElement) != TCL_OK)
      {
        throw(string("Could not get elements from seq/pdu(s)"));
      }
         else
      {
        lNumOfFramesInSeq = (lSeqElementCount+1)/2;

        for (int i = 0; i < lNumOfFramesInSeq; ++i)
          {
            // copy all frame names to a new location
            lSeqObj[i] = Tcl_NewStringObj(Tcl_GetStringFromObj(lppSeqListElement[2*i + 1],
0), -1);
          }
      }
//        }

     DBG("----- 3 . " << lChannel << " . 3 --------");

     // for each frame in list
     for (int lFrameCount = 0; lFrameCount < lNumOfFramesInSeq; ++lFrameCount)
       {
         // get list of frame buf and len 'Bsts_GetFrameData'
         if (Tcl_Eval(apInterp,
            (char *)(lProc_Bsts_GetFrameData
                + string(" ")
                + string(Tcl_GetStringFromObj(aapObjv[0], 0))
                + string(" ")
                + string(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0))).c_str())
        != TCL_OK)
      {
        throw(string("Could not get PDU data"));
      }
         else
      {
        // get elements from frame - frame len and buffer
```

```
        if (Tcl_ListObjGetElements(apInterp,  Tcl_GetObjResult(apInterp),
                  &lFrameElementCount, &lppFrameListElement) != TCL_OK)
          {
            throw(string("Could not get frame elements"));
          }
        else
          {
            lFrameNameIter = lFrameRegis-
ter_Map.find(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0));

            if (lFrameNameIter != lFrameRegister_Map.end())
         {
//            DBG("'" << Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0) << "' found");

           // frame found - therefore insert framePtr into vector
           lFrameVector[lChannel].push_back(&(lFrameNameIter->second));
         }
            else
         {
//            DBG("FrameLen(" << (unsigned
int)atoi(Tcl_GetStringFromObj(lppFrameListElement[0], 0)) << "),Buffer["
//                << (unsigned char *)Tcl_GetStringFromObj(lppFrameListElement[1], 0) <<
"]");

           // resize frame length to max of 1514
           if (atoi(Tcl_GetStringFromObj(lppFrameListElement[0], 0)) > ETH_MAX_LEN)
             {
               // convert to int
               if (ConvertHexToInt(Tcl_GetStringFromObj(lppFrameListElement[1], 0),
                    ETH_MAX_LEN,
                    lBuf) == 0)
            {
              DBG("Error - could not convert hex buffer");
              return (TCL_ERROR);
            }

               DBG("----- 3 . " << lChannel << " . 3_1 --------");

               lFrameRegis-
ter_Map.insert(make_pair(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0),
                         tEthGenFrame(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0),
                                ETH_MAX_LEN,
                                (unsigned char *)lBuf,
                                (unsigned int)1)));

               DBG("----- 3 . " << lChannel << " . 4_1 --------");


               DBG("resetting frame length from " <<
atoi(Tcl_GetStringFromObj(lppFrameListElement[0], 0))
             << " to " << ETH_MAX_LEN);

             }
           else
             {
               // convert hex string to suitable format for storage
               if (ConvertHexToInt(Tcl_GetStringFromObj(lppFrameListElement[1], 0),
                      atoi(Tcl_GetStringFromObj(lppFrameListElement[0], 0)),
                      lBuf) == 0)
            {
              DBG("Error - could not convert hex buffer");
              return (TCL_ERROR);
            }

               DBG("----- 3 . " << lChannel << " . 3_2 --------");
```

```
              lFrameRegis-
ter_Map.insert(make_pair(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0),
                         tEthGenFrame(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0),
                                 (unsigned
int)atoi(Tcl_GetStringFromObj(lppFrameListElement[0], 0)),
                                 (unsigned char *)lBuf,
                                 (unsigned int)1)));
              DBG("----- 3 . " << lChannel << " . 4_2 --------");
            }

           DBG("Map: inserted '" << Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0) << "' ");

           // do a search again and insert

           lFrameNameIter2 = lFrameRegis-
ter_Map.find(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0));

//            DBG(Tcl_GetStringFromObj(lSeqObj[lFrameCount], 0) << " is searched again");

           lFrameVector[lChannel].push_back(&(lFrameNameIter2->second));

//            DBG("lFrameVector contents " << (lFrameNameIter2-
>second).GetEthGenBuf_Buf());
//            DBG("lFrameVector length   " << (lFrameNameIter2-
>second).GetEthGenBuf_Len());
         }
          }

      }

         // make up sequence
         // time delay set to default min possible tx time = 0
         lSeq[lChannel].SetFrameVec(&lFrameVector[lChannel]);
         lSeq[lChannel].SetTimeDelay(0);

       }

     // make up sequence array 'tChannelArray'
     lChannelArray[lChannel] = &lSeq[lChannel];
   }
    }

  DBG("----- 4 --------");

  // execute Config
  if (!(apIface->Config(&lChannelArray)) )
    {
      DBG("! could not configure EthGen");

      // set interp to 0 for failure
      Tcl_SetResult(apInterp, "0", TCL_VOLATILE);
    }
  else
    {
      // set interp to 1 for success
      Tcl_SetResult(apInterp, "1", TCL_VOLATILE);
    }

  DBG("----- 5 --------");

  return (TCL_OK);
}


/* ******************************************************
```

```
 * Function : PreRunCheck
 * Description : perform pre-run check for EthGen
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> PreRunCheck
 * ****************************************************** */
static int
TclCmp_PreRunCheck(tIfEthGen *     apIface,
         Tcl_Interp *      apInterp,
         int         aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  if (!(apIface->PreRunCheck()))
    {
      throw (string("Could not perform pre-run check "));
    }

  return (TCL_OK);
}


/* ******************************************************
 * Function : StartEngine
 * Description : start generating engine
 * Parameters  : none
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> StartEngine
 * ****************************************************** */
static int
TclCmp_StartEngine(tIfEthGen *     apIface,
         Tcl_Interp *      apInterp,
         int         aObjc,
         Tcl_Obj *CONST aapObjv[])
{
  if (!(apIface->StartEngine()))
    {
      throw (string("Could not Start generation engine "));
    }

  return (TCL_OK);

}

/* ******************************************************
 * Function : StopEngine
 * Description : stop generating engine
 * Parameters  : none
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> StopEngine
 * ****************************************************** */
static int
TclCmp_StopEngine(tIfEthGen *    apIface,
        Tcl_Interp *    apInterp,
        int          aObjc,
        Tcl_Obj *CONST  aapObjv[])
{
  if (!(apIface->StopEngine()))
    {
      throw (string("Could not Stop generation engine "));
    }

  return (TCL_OK);
```

```
}

/* ******************************************************
 * Function : TclCmp_SetPDU_Len
 * Description : reduce PDU length, does not modify buffer,
 *       does not increase existing buffer
 *       length,
 *      ModifyFrameBuffer must be used to
 *       populate data first
 * Parameters  : ChNum, PduNum, Pdu len
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> SetPDU_Len ChNum PduNum <len>
 * ****************************************************** */
static int
TclCmp_SetPDU_Len(tIfEthGen *    apIface,
        Tcl_Interp *    apInterp,
        int          aObjc,
        Tcl_Obj *CONST  aapObjv[])
{
  int    lChNum;
  int    lFrameNum;
  int       lBufLen;

  if (aObjc < 3)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string (": IfEthGen ethgen SetPDU_Len ") +
        string ("<chNum> <pduNum> <buflen>"));

    }

  // get all integer and check for errors for non-ints
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChNum) == TCL_ERROR)
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lFrameNum) == TCL_ERROR)
    {
      throw (string("PDU number '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[2], &lBufLen) == TCL_ERROR)
    {
      throw (string("Buffer len value '") +
        string(Tcl_GetStringFromObj(aapObjv[2], 0)) +
        string("' is invalid"));
    }

  if (!(apIface->SetPDU_Len(lBufLen, lChNum, lFrameNum)))
    {
      return (TCL_ERROR);
    }

  return (TCL_OK);

}

/* ******************************************************
 * Function : TclCmp_QueryPDU_Len
 * Description : queries PDU length,
```

```
 * Parameters  : ChNum, PduNum
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> QueryPDU_Len ChNum PduNum
 * ****************************************************** */
static int
TclCmp_QueryPDU_Len(tIfEthGen *    apIface,
          Tcl_Interp *     apInterp,
          int        aObjc,
          Tcl_Obj *CONST   aapObjv[])
{
  int    lChNum;
  int    lFrameNum;

  unsigned int  lPduLen = 0;

  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string (": IfEthGen ethgen QueryPDU_Len ") +
        string ("<chNum> <pduNum> "));

    }

  // get all integer and check for errors for non-ints
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChNum) == TCL_ERROR)
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lFrameNum) == TCL_ERROR)
    {
      throw (string("PDU number '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  lPduLen = apIface->QueryPDU_Len(lChNum, lFrameNum);

  if (lPduLen == 0)
    {
      throw(string("Pdu len = 0"));
    }

  Tcl_SetResult(apInterp,
      Tcl_GetStringFromObj(Tcl_NewIntObj(lPduLen), 0),
      TCL_VOLATILE);

  return (TCL_OK);

}


/* ******************************************************
 * Function : TclCmp_ModifyFrameBuffer
 * Description : modify contents of PDU
 * Parameters  :
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> ModifyFrameBuffer
 *    <chNum> <pduNum> <offset> <buflen> <data string>
 * ****************************************************** */
static int
TclCmp_ModifyFrameBuffer(tIfEthGen *   apIface,
```

```
         Tcl_Interp *  apInterp,
         int     aObjc,
         Tcl_Obj *CONST   aapObjv[])
{
  int    lChNum;
  int    lFrameNum;
  int    lOffSet;
  int       lBufLen;

  int    lHexString_Len = 0;

  char      lpHexString_Buf[ETH_MAX_LEN*2];

  char      lHexStringTestBuf[ETH_MAX_LEN*2];

  if (aObjc < 5)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string (": IfEthGen ethgen ModifyFrameBuffer ") +
        string ("<chNum> <pduNum> <offset> <buflen> <data string>"));

    }

  // get all integer and check for errors for non-ints
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChNum) == TCL_ERROR)
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lFrameNum) == TCL_ERROR)
    {
      throw (string("PDU number '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[2], &lOffSet) == TCL_ERROR)
    {
      throw (string("Offset value '") +
        string(Tcl_GetStringFromObj(aapObjv[2], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[3], &lBufLen) == TCL_ERROR)
    {
      throw (string("Buffer len value '") +
        string(Tcl_GetStringFromObj(aapObjv[3], 0)) +
        string("' is invalid"));
    }

  // check for valid hex string, throw for invalid
  // Function checks for invalid chars
  if (ConvertHexToInt(Tcl_GetStringFromObj(aapObjv[4], 0),
            (strlen(Tcl_GetStringFromObj(aapObjv[4], 0))/2),
            lHexStringTestBuf) == 0)
    {
      throw(string("Supplied hex buffer has invalid hex characters"));
    }


  lHexString_Len = strlen(Tcl_GetStringFromObj(aapObjv[4], 0));

  DBG("hex string length = " << lHexString_Len);
```

```
  // clear out lpHexString_Buf[ETH_MAX_LEN] to begin with
  for (unsigned int lCount = 0; lCount < ETH_MAX_LEN*2; ++lCount)
    {
      // strlen uses terminating char to determine length
      lpHexString_Buf[lCount] = '\0';
    }

  // get src string and compare with requested length
  if (lBufLen > lHexString_Len/2)
    {
      DBG("buffer length is greater than supplied data");

      // fill extra spaces with 0s and make up buffer
      if (memcpy(lpHexString_Buf, Tcl_GetStringFromObj(aapObjv[4], 0),
       lHexString_Len) != NULL)
   {
     for (int lCount = lHexString_Len; lCount < lBufLen*2; ++lCount)
       {
         lpHexString_Buf[lCount] = '0';
       }
   }
      else
   {
     throw (string("!memcpy error"));
   }
    }
  else
    {

      // truncate supplied data
      if (memcpy(lpHexString_Buf, Tcl_GetStringFromObj(aapObjv[4], 0),
       lBufLen*2) != NULL)
   {

   }
      else
   {
     throw (string("!memcpy error"));
   }
    }

  // debug printout
  for (int lCount = 0; lCount < lBufLen*2; ++lCount)
    {
      cout << lpHexString_Buf[lCount] << ".";
    }
  cout << endl;

  // call interface function with hex string
  if (!(apIface->ModifyFrameBuffer(lOffSet, lBufLen,
              (unsigned char *)lpHexString_Buf,
              lChNum, lFrameNum)))
    {
      return (TCL_ERROR);
    }


  // display modified value and write to Tcl interp


  return (TCL_OK);
}


/* ******************************************************
 * Function : TclCmp_QueryFrameBuffer
```

```
 * Description : view contents of PDU
 * Parameters  :
 * Returns  : hex string (hex string char len = 2*BufLen)
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> QueryFrameBuffer
 *    <ChNum> <PduNum> <offset> <BufLen>
 * ****************************************************** */
static int
TclCmp_QueryFrameBuffer(tIfEthGen * apIface,
         Tcl_Interp *   apInterp,
         int      aObjc,
         Tcl_Obj *CONST aapObjv[])
{

  // return buffer is twice in length to max len
  unsigned char      lReturnBuffer[ETH_MAX_LEN*2];
  Tcl_Obj *    lpTcl_ObjRes;

  int    lChNum;
  int    lFrameNum;
  int    lOffSet;
  int       lBufLen;


  if (aObjc < 4)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string (": IfEthGen ethgen QueryFrameBuffer ") +
        string ("<chNum> <pduNum> <offset> <buflen>"));

    }



  // get all integer values
  // conversion and error handling for non-int vals

  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChNum) == TCL_ERROR)
    {
      throw (string("Channel Num '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

    if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lFrameNum) == TCL_ERROR)
    {
      throw (string("PDU Num '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

    if (Tcl_GetIntFromObj(apInterp, aapObjv[2], &lOffSet) == TCL_ERROR)
    {
      throw (string("Offset value '") +
        string(Tcl_GetStringFromObj(aapObjv[2], 0)) +
        string("' is invalid"));
    }

        if (Tcl_GetIntFromObj(apInterp, aapObjv[3], &lBufLen) == TCL_ERROR)
    {
      throw (string("Buffer Len '") +
        string(Tcl_GetStringFromObj(aapObjv[3], 0)) +
        string("' is invalid"));
    }

  // invoke QueryFrameBuffer()
```

```
  if (!(apIface->QueryFrameBuffer(lOffSet,
              lBufLen,
              lReturnBuffer,
              lChNum,
              lFrameNum)))
    {
      throw (string("Query error"));
    }

  // package result into return string for Tcl
  lpTcl_ObjRes = Tcl_NewStringObj((const char *)lReturnBuffer, lBufLen*2);

  if (lpTcl_ObjRes != NULL)
    {
      Tcl_SetResult(apInterp, Tcl_GetStringFromObj(lpTcl_ObjRes, 0), TCL_VOLATILE);
    }

    return (TCL_OK);
}


/* ******************************************************
 * Function : SetRate_FramePerSec
 * Description : set channel rate in frames/sec
 * Parameters  : rate
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> SetRate_FramePerSec
 *    <rate> <ChannelNum>
 * ****************************************************** */
static int
TclCmp_SetRate_FramePerSec(tIfEthGen * apIface,
            Tcl_Interp *   apInterp,
            int      aObjc,
            Tcl_Obj *CONST aapObjv[])
{
  int    lRate;
  int       lChannel;


  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen SetRate_FramePerSec <rate fps> <channel>"));
    }

  // get int value and check errors for non-int values
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lRate) == TCL_ERROR)
    {
      throw (string("Rate value '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }


  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }



  // set rate value in frames per second
  if (!(apIface->SetRate_FramePerSec(lRate, lChannel)))
```

```
    {
      throw (string("Could not set rate '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("'") );
    }



  return (TCL_OK);
}

/* ******************************************************
 * Function : SetRate_Mbps
 * Description : set channel rate in Mbps
 * Parameters  : rate
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> SetRate_FramePerSec
 *    <rate> <ChannelNum>
 * ****************************************************** */
static int
TclCmp_SetRate_Mbps(tIfEthGen *  apIface,
          Tcl_Interp *  apInterp,
          int     aObjc,
          Tcl_Obj *CONST   aapObjv[])
{
  int    lRate;
  int       lChannel;

  // regular checks
  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen SetRate_Mbps <rate Mbps> <channel>"));
    }

  // get int value and check errors for non-int values
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lRate) == TCL_ERROR)
    {
      throw (string("Rate value '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }


  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  // check for rate limits
  if (lRate > 100)
    {
      DBG("100 BASE-T generation suppports a maximum of 100 Mbps");
    }

  // invoke SetRate_Mbps
  if (!(apIface->SetRate_Mbps(lRate, lChannel)))
    {
      throw (string("Could not set rate '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("' Mbps") );
    }
```

```
  return (TCL_OK);

}


/* ******************************************************
 * Function : SetLimit_Time
 * Description : set channel time limit in seconds
 * Parameters  :
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> SetLimit_Time
 *    <time> <ChannelNum>
 * ****************************************************** */
static int
TclCmp_SetLimit_Time(tIfEthGen * apIface,
          Tcl_Interp * apInterp,
          int       aObjc,
          Tcl_Obj *CONST  aapObjv[])
{
  long      lTime_secs;
  int       lChannel;

  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen SetLimit_Time <time sec> <channel>"));
    }

  // get int values and check for invalid characters
  if (Tcl_GetLongFromObj(apInterp, aapObjv[0], &lTime_secs) == TCL_ERROR)
    {
      throw (string("Time value '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  // call interface functions
  if (!(apIface->SetLimit_Time(lTime_secs, lChannel)))
    {
      throw (string("Could not set time limit '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("'") );
    }

  return (TCL_OK);
}


/* ******************************************************
 * Function : SetLimit_Loops
 * Description : set channel Loops limit
 * Parameters  :
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> SetLimit_Time
 *    <time> <ChannelNum>
 * ****************************************************** */
static int
```

```
TclCmp_SetLimit_Loops(tIfEthGen *   apIface,
           Tcl_Interp * apInterp,
           int       aObjc,
           Tcl_Obj *CONST  aapObjv[])
{
  long      lLoops;
  int       lChannel;

  if (aObjc < 2)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen SetLimit_Loops <loops> <channel>"));
    }

  // get int values and check for non-int values
  if (Tcl_GetLongFromObj(apInterp, aapObjv[0], &lLoops) == TCL_ERROR)
    {
      throw (string("Loops value '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  if (Tcl_GetIntFromObj(apInterp, aapObjv[1], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel '") +
        string(Tcl_GetStringFromObj(aapObjv[1], 0)) +
        string("' is invalid"));
    }

  // invoke interface function
  if (!(apIface->SetLimit_Loops(lLoops, lChannel)))
    {
      throw (string("Could not set loop limit '") +
        Tcl_GetStringFromObj(aapObjv[0], 0) +
        string("'") );
    }

  return (TCL_OK);
}


/* ******************************************************
 * Function : QueryRate_FramePerSec
 * Description : query channel rate in frames/sec
 * Parameters  : channel num
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> QueryRate_FramePerSec
 *    <ChannelNum>
 * ****************************************************** */
static int
TclCmp_QueryRate_FramePerSec(tIfEthGen *  apIface,
              Tcl_Interp * apInterp,
              int       aObjc,
              Tcl_Obj *CONST  aapObjv[])
{
  int          lChannel;
  unsigned int       lChResult;

  if (aObjc < 1)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen SetRate_FramePerSec <channel>"));
    }

  // get integer value and check for errors for non-ints
```

```
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  lChResult = apIface->QueryRate_FramePerSec(lChannel);

  if (lChResult == 0)
    {
      throw(string("error"));
    }

  Tcl_SetResult(apInterp,
      Tcl_GetStringFromObj(Tcl_NewIntObj(lChResult), 0),
      TCL_VOLATILE);

  return (TCL_OK);
}

/* ******************************************************
 * Function : QueryStats_FrameSent
 * Description : query channel statistics; frames sent out
 * Parameters  : channel num
 * Returns  : TCL_OK, throws otherwise
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> QueryStats_FrameSent
 *    <ChannelNum>
 * ****************************************************** */
static int
TclCmp_QueryStats_FrameSent(tIfEthGen *   apIface,
             Tcl_Interp *  apInterp,
             int     aObjc,
             Tcl_Obj *CONST   aapObjv[])
{
  int       lChannel;

  char   *     lResult;

  long long       lRetVal;

  int          lTopHalf;
  int          lBotHalf;

  char         s[80];

  if (aObjc < 1)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen QueryStats_FrameSent <channel>"));
    }

  // get integer value and check for errors for non-ints
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChannel) == TCL_ERROR)
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  lRetVal = apIface->QueryStats_FrameSent(lChannel);

  // invalid channel number
  if (lRetVal < 0)
    {
```

```
      throw (string("error"));
    }

  // return 0 if result = 0
  if (lRetVal == 0)
    {
      Tcl_SetResult(apInterp, "0", TCL_VOLATILE);
      return (TCL_OK);
    }

  lTopHalf = lRetVal/1000000;
  lBotHalf = lRetVal%1000000;

  // pad with 0's
  sprintf(s, "%06d", lBotHalf);

  // add to top half to make the whole number
  lResult = strcat(Tcl_GetStringFromObj(Tcl_NewLongObj(lTopHalf), 0), s);

  // trim out leading zeroes before returning interp
  if (Tcl_Eval(apInterp,
          (char *)(string("string trimleft")
         + string(" ")
         + string(lResult)
         + string(" ")
         + string("0")).c_str())
      != TCL_OK)
    {
      throw(string("could not trim leading zeroes from '") +
       string (lResult) +
       string ("'"));
    }

  return (TCL_OK);
}


/* ******************************************************
 * Function : QueryStats_TimeInterval
 * Description : query valid channel statistics
 * Parameters  :
 * Returns  : list of two elements (sec usec)
 * ******************************************************
 * NOTES: TclCmd % IfEthGen <ethgen> QueryStats_TimeInterval
 *       <channelnum>
 * ****************************************************** */
static int
TclCmp_QueryStats_TimeInterval(tIfEthGen *  apIface,
               Tcl_Interp *  apInterp,
               int     aObjc,
               Tcl_Obj *CONST   aapObjv[])
{
  int       lChannel;

  timeval      lTimeInterval;

  Tcl_Obj *    lpResultListObj = Tcl_NewListObj(0, NULL);

  if (aObjc < 1)
    {
      throw (string (FILE_NAME_ETHGEN_CC) +
        string(": IfEthGen ethgen QueryStats_TimeInterval <channel>"));
    }

  // get integer value and check for errors for non-ints
  if (Tcl_GetIntFromObj(apInterp, aapObjv[0], &lChannel) == TCL_ERROR)
```

```
    {
      throw (string("Channel number '") +
        string(Tcl_GetStringFromObj(aapObjv[0], 0)) +
        string("' is invalid"));
    }

  lTimeInterval = apIface->QueryStats_TimeInterval(lChannel);

  if ((lTimeInterval.tv_sec == -1) && (lTimeInterval.tv_usec == -1))
    {
      throw (string("error"));
    }

  // make up a list - sec and usec
  if (Tcl_ListObjAppendElement(apInterp, lpResultListObj,
                Tcl_NewLongObj(lTimeInterval.tv_sec))
      != TCL_OK)
    {
      throw (string("sec could not be appended to list") );
    }

  if (Tcl_ListObjAppendElement(apInterp, lpResultListObj,
                Tcl_NewLongObj(lTimeInterval.tv_usec))
      != TCL_OK)
    {
      throw (string("usec could not be appended to list") );
    }

  DBG("Tcl_GetTimeInterval " << Tcl_GetStringFromObj(Tcl_NewLongObj(lTimeInterval.tv_sec),
0)
      << Tcl_GetStringFromObj(Tcl_NewLongObj(lTimeInterval.tv_usec), 0));

   Tcl_SetObjResult(apInterp, lpResultListObj);

  return (TCL_OK);
}


/* ====================================================================
 * ====================================================================
 *  Internal functions - internal to this file
 * ==================================================================== */
/* ******************************************************
 * Function : ConvertHexToInt
 * Description :
 * Parameters  : char * aInb, int aLen, char * aOutb
 * Returns  : unsigned int
 * ******************************************************
 * NOTES:
 * ****************************************************** */
unsigned int
ConvertHexToInt(char * aInb, int aLen, char * aOutb)
{
  int inidx = 0;
  int outidx = 0;

  bool lHexChar = false;

  // higher nibble
  bool nib = true;
  int num = 0;

  while (inidx < aLen*2)
    {
      if (aInb[inidx] >= '0' && aInb[inidx] <= '9')
   {
```

```
      num = aInb[inidx] - '0';
      lHexChar = true;
    }
       else if ((aInb[inidx] >= 'A') && (aInb[inidx] <= 'F'))
    {
      num = aInb[inidx] - 'A' + 10;
      lHexChar = true;
    }
       else if ((aInb[inidx] >= 'a') && (aInb[inidx] <= 'f'))
    {
      num = aInb[inidx] - 'a' + 10;
      lHexChar = true;
    }

       // return if char is not hex
       if (!lHexChar)
    {
      DBG("invalid hex char '" << (char)aInb[inidx] << "'");
      return (0);
    }

       ++inidx;

       // reset lHexChar to false for incoming chars
       lHexChar = false;

       if (nib)
    {
      aOutb[outidx] = num << 4;
      nib = false;
    }
       else
    {
      aOutb[outidx] = aOutb[outidx] | num;
      nib = true;
      ++outidx;
    }
//       cout << (int)aOutb[outidx];

     }

  return (1);
}


/* ******************************************************
 * Function : ConvertIntToHex
 * Description : return outbuffer to represent hex data
 * Parameters  : char * aInb, int aLen, char * aOutb
 * Returns  : unsigned int
 * ******************************************************
 * NOTES:
 * ****************************************************** */
unsigned int
ConvertIntToHex(char * aInb, int aLen, char * aOutb)
{
  int inidx = 0;
  int outidx = 0;

  unsigned int lNum = 0;

  char lTable[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8',
           '9', 'A', 'B', 'C', 'D', 'E', 'F'};

  while (inidx < aLen)
    {
```

```
//       DBG(" ---------> aInb[" << inidx << "] = " << (int)aInb[inidx]);

      // correction for signed int
      if (aInb[inidx] < 0)
   {
     lNum = aInb[inidx] + 256;
//      DBG(" lNum = " << lNum);
//      DBG(" lNum >> 4 = " << (lNum >> 4));

     aOutb[outidx] = lTable[lNum >> 4];
   }
      else
   {
     aOutb[outidx] = lTable[aInb[inidx] >> 4];
   }

//       DBG("tbl(" << (int)(aInb[inidx] >> 4) << ") " << "aOutb[" << outidx << "] = " <<
aOutb[outidx]);
      ++outidx;

      aOutb[outidx] = lTable[aInb[inidx] & 0xf];
//       DBG("tbl(" << (int)(aInb[inidx] & 0xf) << ") " << "aOutb[" << outidx << "] = " <<
aOutb[outidx]);
      ++outidx;

      ++inidx;
    }

  return (1);

}



/* *******************************************************
 * Function : GetIntFromObj
 * Description : return int from TclObj
 * Parameters  : TclObj *
 * Returns  : positive int value (-ve value for error)
 * *******************************************************
 * NOTES:
 * ******************************************************* */
int GetIntFromObj(char * aVal)
{
  int lIntOut;

  lIntOut = isdigit((int)aVal);

  if (lIntOut < 0)
    {
      return (lIntOut);
    }
  else
    {
      DBG("found " << aVal << " to be "
     << lIntOut);

    }

  return (lIntOut);
}
```

```
/* ========================================================================
 * ========================================================================
 *  Entry Points
 * ======================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
extern "C"
int
Cmptcl_ethgen_Init(Tcl_Interp * apInterp)
{
  // Populate sub-command map
  mCmdMap.AddCmd((const char *)"QueryState", TclCmp_QueryState);
  mCmdMap.AddCmd((const char *)"SetInterface", TclCmp_SetInterface);
  mCmdMap.AddCmd((const char *)"QueryInterface", TclCmp_QueryInterface);
  mCmdMap.AddCmd((const char *)"Config", TclCmp_Config);
  mCmdMap.AddCmd((const char *)"PreRunCheck", TclCmp_PreRunCheck);
  mCmdMap.AddCmd((const char *)"StartEngine", TclCmp_StartEngine);
  mCmdMap.AddCmd((const char *)"StopEngine", TclCmp_StopEngine);

  mCmdMap.AddCmd((const char *)"SetPDU_Len", TclCmp_SetPDU_Len);
  mCmdMap.AddCmd((const char *)"QueryPDU_Len", TclCmp_QueryPDU_Len);
//   mCmdMap.AddCmd((const char *)"RemoveChannel", TclCmp_RemoveChannel);
//   mCmdMap.AddCmd((const char *)"SetRate_BitsPerSec", TclCmp_SetRate_BitsPerSec);

  mCmdMap.AddCmd((const char *)"ModifyFrameBuffer", TclCmp_ModifyFrameBuffer);
  mCmdMap.AddCmd((const char *)"QueryFrameBuffer", TclCmp_QueryFrameBuffer);
  mCmdMap.AddCmd((const char *)"SetRate_FramePerSec", TclCmp_SetRate_FramePerSec);
  mCmdMap.AddCmd((const char *)"SetRate_Mbps", TclCmp_SetRate_Mbps);
  mCmdMap.AddCmd((const char *)"SetLimit_Time", TclCmp_SetLimit_Time);
  mCmdMap.AddCmd((const char *)"SetLimit_Loops", TclCmp_SetLimit_Loops);
  mCmdMap.AddCmd((const char *)"QueryRate_FramePerSec", TclCmp_QueryRate_FramePerSec);
  mCmdMap.AddCmd((const char *)"QueryStats_FrameSent", TclCmp_QueryStats_FrameSent);
  mCmdMap.AddCmd((const char *)"QueryStats_TimeInterval", TclCmp_QueryStats_TimeInterval);

  // New interpreter commands
  TclCmp_RegCmdMap(apInterp, "IfEthGen", mCmdMap);

  return (TCL_OK);
}


/* ========== *
 *     EOF     *
 * ========== */


/* Function and class header prototypes */
#if 0

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
```

```
/* ******************************************************
 * Class :
 * Description :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
class
{
public:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

protected:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

private:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

};

#endif

Component StdIo
/* ********************************************************************
 *
 *   Source File Name   : CmpTcl_StdIo.cc
 *
 *   Module Name  : CmpTcl_StdIo
 *
 *   Application Name   : TP01
 *
 *   Project Name : TCS01
 *
 * ********************************************************************
 * (c)2001 Seven Layer Communications Ltd.
 * ********************************************************************
 *  NOTES:


 *  END OF NOTES
 * ================================================================== */

/* Uncomment if 'what' string is needed */
/* static char gIdent[] = "@(#)filename   Version 0.0 "; */


/* ======================================================
 *  Standard Library Includes   (normal system)
 * ====================================================== */


/* ======================================================
```

```
 *  External Includes        (external toolkits)
 * ========================================================== */
#include <tcl.h>


/* ==========================================================
 *  Project-wide Includes    (project only)
 * ========================================================== */


/* ==========================================================
 *  Module Includes       (module only)
 * ========================================================== */
#include "../../../inc/Component.hh"
#include "../../../inc/Iface/IfStdIo.hh"
#include "../../../inc/Iface/IfLoader.hh"
#include "../../../inc/TclCmp_LookupCmd.hh"



/* ==========================================================
 *  Module #DEFINEs
 * ========================================================== */
#define FILE_NAME_STDIO_CC "CmpTcl_StdIo.cc"


/* ==========================================================
 *  Enumerations & Other Typedefs (defn)
 * ========================================================== */


/* ==========================================================
 *  Classes (forward decl) & Structures (defn)
 * ========================================================== */


/* ==========================================================
 *  Module Functions (decl.)    (static local only)
 * ========================================================== */


/* ==========================================================
 *  Global Variables (defn.)    (used externally)
 * ========================================================== */


/* ==========================================================
 *  Local Module Variables (defn.) (static local only)
 * ========================================================== */
static tTclCmpCmdMap<tIfStdIo>     mCmdMap;


/* ======================================================================
 * ======================================================================
 *  Functions & class definitions visible externally
 * ====================================================================== */



/* ======================================================================
 * ======================================================================
 *  Functions & class definitions internal to module
 * ====================================================================== */

/* ******************************************************
```

```
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
static int
TclCmp_Invoke(tIfStdIo *   apIface,
         Tcl_Interp *   apInterp,
         int      aObjc,
         Tcl_Obj *CONST aapObjv[])
{
    if (aObjc < 1)
    {
      throw (string (FILE_NAME_STDIO_CC) +
        string(": invalid number of parameters"));
    }

    apIface->Invoke();

    return (TCL_OK);
}


/* ====================================================================
 * ====================================================================
 *  Entry Points
 * ================================================================== */

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
 * Returns  :
 * ******************************************************
 * NOTES:
 * ****************************************************** */
extern "C"
int
Cmptcl_stdio_Init(Tcl_Interp * apInterp)
{
  // Populate sub-command map
  mCmdMap.AddCmd((const char *)"Invoke", TclCmp_Invoke);

  // New interpreter commands
  TclCmp_RegCmdMap(apInterp, "IfStdIo", mCmdMap);

  return (TCL_OK);
}



/* ========== *
 *     EOF     *
 * ========== */


/* Function and class header prototypes */
#if 0

/* ******************************************************
 * Function :
 * Description :
 * Parameters  :
```

```
 * Returns  :
 * ****************************************************
 * NOTES:
 * ****************************************************** */

/* ******************************************************
 * Class :
 * Description :
 * ****************************************************
 * NOTES:
 * ****************************************************** */
class
{
public:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

protected:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

private:
  //// Types
  //// Ctor/Dtor
  //// Static methods
  //// Static members
  //// Instance methods
  //// Instance members

};

#endif
```

## 1.5 Tcl Sources

```
#  Test script for inputting bsts pattern sequence lists
#   to Ethernet generator.
#
#   Proc tested
#          -
#          -


# -------------- basic configure --------------------
load ../libCmpTcl_Loader.so
load ../libCmpTcl_EthGen.so
IfLoader loader ReadTypeRegFile Registry
IfLoader loader InstCreate EthGen ethgen

IfEthGen ethgen QueryState

# -------------- proc Bsts_GetFrameData --------
proc Bsts_GetFrameData {aFileName aFrameName } {
    set Channel [open $aFileName r]
```

```
    set lCount 1
    set lNoOfLinesInBstsFile 1584
    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lFrame [gets $Channel]
   if {$lFrame == $aFrameName} {
       set Layer [gets $Channel]
       set EthHdr [gets $Channel]
       set BufLen [gets $Channel]
       puts $lFrame
       break
   }

   incr lCount
    }

    set HexData 1
    while { $HexData >= 0} {
        set HexData [gets $Channel]
   lappend lList $HexData
    }
    incr lCount

    close $Channel
    set TempList [join $lList ""]
    regsub -all " " $TempList "" ContHexString
    set lList [list $BufLen $ContHexString]
    return $lList
}


# -------------- proc Bsts_GetPattern --------
proc Bsts_GetPattern {aFileName aPatternname } {
    set Channel [open $aFileName r]
    set lCount 1
    set lNoOfLinesInBstsFile 1584

    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lPattern_Name [gets $Channel]
   if {$lPattern_Name == $aPatternname} {
       set lPattern_Len [gets $Channel]
       for {set lCount 0} {$lCount < $lPattern_Len} {incr lCount} {
      set lElement [gets $Channel]
      lappend lList [split $lElement " "]
       }
       break
   }
   incr lCount
    }
    close $Channel

    set lNewList [join $lList " "]

    return $lNewList
}

# -------------- do the above ----------------
source platform.tcl

CreateEthGen2.0

# -------------- Config EthGen ----------------
#IfEthGen ethgen Config ../bsts/bsts IP_004 IP_001

IfEthGen ethgen Config ../bsts/bsts IP_001 IP_002 IP_004 IP_005 IP_006 IP_008 IP_009 IP_010

IfEthGen ethgen Config ../bsts/bsts SEQ_ETHER
```

```
IfEthGen ethgen Config ../bsts/bsts 802.2_ICMP_Max

# -------------- set rate for channel 0 -------
IfEthGen ethgen SetRate_FramePerSec 10 0
IfEthGen ethgen SetRate_FramePerSec 10 1
IfEthGen ethgen SetRate_FramePerSec 10 2
IfEthGen ethgen SetRate_FramePerSec 10 3
IfEthGen ethgen SetRate_FramePerSec 10 4
IfEthGen ethgen SetRate_FramePerSec 10 5
IfEthGen ethgen SetRate_FramePerSec 10 6
IfEthGen ethgen SetRate_FramePerSec 10 7

# -------------- query rate for channel 0 -------
IfEthGen ethgen QueryRate_FramePerSec 0

# -------------- set interface to eth2 -------
IfEthGen ethgen SetInterface eth2

IfEthGen ethgen QueryInterface


# -------------- PreRun Check ----------------
IfEthGen ethgen PreRunCheck

# -------------- Query State ----------------
IfEthGen ethgen QueryState


# -------------- SetLimit_Time ----------------
IfEthGen ethgen SetLimit_Time 2 0


# -------------- SetLimit_Loops ----------------
IfEthGen ethgen SetLimit_Loops 1000 0


# --------------  Query PDU  ----------------
IfEthGen ethgen QueryFrameBuffer 0 1 0 4

# --------------  Change PDU  ----------------
IfEthGen ethgen ModifyFrameBuffer 0 1 0 4 abcd6789

# --------------  Query PDU  ----------------
IfEthGen ethgen QueryFrameBuffer 0 1 0 4

ifconfig eth1
# -------------- Start Engine ----------------
IfEthGen ethgen StartEngine
ifconfig eth1

# -------------- Query Stats ----------------
IfEthGen ethgen QueryStats_FrameSent 0


# -------------- Stop Engine ----------------
IfEthGen ethgen StopEngine

IfEthGen ethgen QueryStats_TimeInterval 0
IfEthGen ethgen QueryStats_FrameSent 0


#
# ---------- Results -----------------
#
```

```
#
# date: 08 Feb 02
#
# comments:
#
# Weekend results
# CmpEthGen.cc          Version 1.42
# CmpTcl_EthGen.cc      Version 1.14
#
# FrameRate 100 fps

# Result:
# % IfEthGen ethgen SetRate_FramePerSec 100 0
# SetRate_FramePerSec: time interval to be set to 10000 usecs
# % IfEthGen ethgen QueryRate_FramePerSec 0
# QueryRate_FramePerSec: got frame rate as 100
# % IfEthGen ethgen PreRunCheck
# % IfEthGen ethgen StartEngine
# StartEngine: Thread running .....fps
# % ThrMain: lStartTime_sec.usec[0]  = 1013183019.848476

# %
# %
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013183026.768668 [0] = 692
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013183028.748903 [0] = 890
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013183034.589095 [0] = 1474
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013183036.519372 [0] = 1667
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013184502.933492 [0] = 148109
# %
# %
# %
# %
# %
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013184506.575326 [0] = 148473
# %
# %
# %
# %
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013184706.999929 [0] = 168416
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013184717.546011 [0] = 169470
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013418542.881715 [0] = 18914004
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013418765.711333 [0] = 18936187
# %
# %
# %
# %
# %
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013419562.48309 [0] = 19013520
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013419650.52687 [0] = 19022121
# % IfEthGen ethgen QueryStats_FrameSent 0
# QueryStats_FrameSent: 1013419653.686454 [0] = 19022484


# % IfEthGen ethgen StopEngine
```

```
# ThrMain: out of thread loop
# StopEngine: Frames sent out on Channel[0] = 19084446



source load.tcl
IfEthGen ethgen Config ../bsts/bsts SEQ_ETHER
IfEthGen ethgen SetRate_FramePerSec 10 0
IfEthGen ethgen QueryRate_FramePerSec 0
IfEthGen ethgen PreRunCheck
IfEthGen ethgen QueryState
IfEthGen ethgen StartEngine

IfEthGen ethgen QueryStats_FrameSent 0


IfEthGen ethgen StopEngine

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::CreateEthGen2.0
#
# Overview: create an instance of ethgen
#
# Input parameters:
#  Arg1: Library path to ethgen and loader
#
# Return Data: None
#
# Side effects: loads $aLibPath/libCmpTcl_Loader.so, $aLibPath/libCmpTcl_EthGen.so
#                and reads ./Registry
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc CreateEthGen2.0 {aLibPath} {

    if {[FindFile $aLibPath/libCmpTcl_Loader.so] == 1} {
   load $aLibPath/libCmpTcl_Loader.so
    } else {
   return {"could not find $aLibPath/libCmpTcl_Loader.so library"}
    }

    if {[FindFile $aLibPath/libCmpTcl_EthGen.so] == 1} {
   load $aLibPath/libCmpTcl_EthGen.so
    } else {
   return {"could not find $aLibPath/libCmpTcl_EthGen.so library"}
    }

    if {[FindFile ./Registry] == 1} {
   IfLoader loader ReadTypeRegFile Registry
   IfLoader loader InstCreate EthGen ethgen
    } else {
   return {"could not find ./Registry"}
    }

    IfEthGen ethgen QueryState

    return [IfEthGen ethgen QueryInterface]
}

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::DeleteEthGen2.0
#
# Overview: deletes the created instance of ethgen
#
```

```
# Input parameters:
#  Arg1: none
#
# Return Data: 0 for success and 1 for failure
#
# Side effects:
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc DeleteEthGen2.0 {} {

    IfLoader loader InstDelete EthGen ethgen
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::eth1
#
# Overview: handle to ethgen for active socket eth1
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth1 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth1"} {
   return "eth1 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
    eval $command
}

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::eth2
#
# Overview: handle to ethgen for active socket eth2
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth2 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth2"} {
   return "eth2 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
```

```
    eval $command
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::eth3
#
# Overview: handle to ethgen for active socket eth3
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth3 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth3"} {
   return "eth3 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
    eval $command
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::Bsts_GetFrameData
#
# Overview: reads PDU data from bsts file
#
#           Does not support sequences - use  Bsts_GetPattern to get contents
#              of bsts sequences
# Input parameters:
#  Arg1: Bsts file name
#  Arg2: Pdu Name
#
# Return Data: returns a list of two elements [pdu length and data string]
#
# Side effects:
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc Bsts_GetFrameData {aFileName aFrameName } {
    set Channel [open $aFileName r]
    set lCount 1
    set lSuccess 0

    set lNoOfLinesInBstsFile 1584
    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lFrame [gets $Channel]
   if {$lFrame == $aFrameName} {
       set Layer [gets $Channel]
       set EthHdr [gets $Channel]
       set BufLen [gets $Channel]
       #puts $lFrame
       set lSuccess 1
       break
   }
```

```
   incr lCount
    }

    set HexData 1
    while { $HexData >= 0} {
        set HexData [gets $Channel]
   lappend lList $HexData
    }
    incr lCount

    close $Channel
    set TempList [join $lList ""]
    regsub -all " " $TempList "" ContHexString
    set lList [list $BufLen $ContHexString]

    if {$lSuccess != 1} {
   puts "could not find pattern - $aFrameName"
    }

    #puts $lList
    return $lList
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::Bsts_GetPattern
#
# Overview: get pattern from formatted file
#
# Input parameters:
#  Arg1: Bsts file name
#  Arg2: Sequence/Pdu Name
#
# Return Data: None
#
# Side effects:

#  Successful operation: returns list of frame repetation and frame name
#  Handles both sequences and PDUs for IfEthGen ethgen Config ../bsts/bsts ....
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc Bsts_GetPattern {aFileName aPatternname } {
    set Channel [open $aFileName r]
    set lCount 1
    set lSuccess 0
    set lNoOfLinesInBstsFile 1584
    set lFrameFlag true

    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lName [gets $Channel]

   if {$lName == "# Sequence list"} {
       set lFrameFlag false
   }
   if {$lName == $aPatternname} {

       if {$lFrameFlag == "false"} {
      set lPattern_Len [gets $Channel]
      for {set lCount 0} {$lCount < $lPattern_Len} {incr lCount} {
          set lElement [gets $Channel]
          lappend lList [split $lElement " "]
      }
       } else {
```

```
      # lName is now a frame
      set lMAC [gets $Channel]
      set lHdr [gets $Channel]
      set lFrameLen [gets $Channel]
      set lList [list 1 $lName]
       }

       set lSuccess 1
       break
   }
   incr lCount
    }
    close $Channel

    set lNewList [join $lList " "]

    if {$lSuccess != 1} {
   puts "could not find pattern - $aPatternname"
   return "null"
    }
    return $lNewList
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::FindFile
#
# Overview: checks existence of file in file structure
#
# Input parameters:
#  Arg1: Bsts file name
#
# Return Data: 1 if file found, 0 otherwise
#
# Side effects:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc FindFile {aFileName} {

    return [file isfile $aFileName]
}
```

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::CreateEthGen2.0
#
# Overview: create an instance of ethgen
#
# Input parameters:
#  Arg1: Library path to ethgen and loader
#
# Return Data: None
#
# Side effects: loads $aLibPath/libCmpTcl_Loader.so, $aLibPath/libCmpTcl_EthGen.so
#                and reads ./Registry
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc CreateEthGen2.0 {aLibPath} {

    if {[FindFile $aLibPath/libCmpTcl_Loader.so] == 1} {
   load $aLibPath/libCmpTcl_Loader.so
    } else {
   return {"could not find $aLibPath/libCmpTcl_Loader.so library"}
    }

    if {[FindFile $aLibPath/libCmpTcl_EthGen.so] == 1} {
   load $aLibPath/libCmpTcl_EthGen.so
    } else {
   return {"could not find $aLibPath/libCmpTcl_EthGen.so library"}
    }

    if {[FindFile ./Registry] == 1} {
   IfLoader loader ReadTypeRegFile Registry
   IfLoader loader InstCreate EthGen ethgen
    } else {
   return {"could not find ./Registry"}
    }

    IfEthGen ethgen QueryState

    return [IfEthGen ethgen QueryInterface]
}

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::DeleteEthGen2.0
#
# Overview: deletes the created instance of ethgen
#
# Input parameters:
#  Arg1: none
#
# Return Data: 0 for success and 1 for failure
#
# Side effects:
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc DeleteEthGen2.0 {} {

    IfLoader loader InstDelete EthGen ethgen
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
# Name: ::eth1
#
# Overview: handle to ethgen for active socket eth1
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth1 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth1"} {
   return "eth1 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
    eval $command
}

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::eth2
#
# Overview: handle to ethgen for active socket eth2
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth2 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth2"} {
   return "eth2 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
    eval $command
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::eth3
#
# Overview: handle to ethgen for active socket eth3
#
# Input parameters:
#  Arg1: none
#  Optional Arg3:
#
# Return Data: None
#
# Side effects: makes up full command from args supplied
#
#  Failure:
```

```
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc eth3 {args } {

    if {[IfEthGen ethgen QueryInterface] != "eth3"} {
   return "eth3 not open"
    }
    set command [concat {IfEthGen ethgen} $args]
    eval $command
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::Bsts_GetFrameData
#
# Overview: reads PDU data from bsts file
#
#           Does not support sequences - use  Bsts_GetPattern to get contents
#              of bsts sequences
# Input parameters:
#  Arg1: Bsts file name
#  Arg2: Pdu Name
#
# Return Data: returns a list of two elements [pdu length and data string]
#
# Side effects:
#
#  Failure:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc Bsts_GetFrameData {aFileName aFrameName } {
    set Channel [open $aFileName r]
    set lCount 1
    set lSuccess 0

    set lNoOfLinesInBstsFile 1584
    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lFrame [gets $Channel]
   if {$lFrame == $aFrameName} {
       set Layer [gets $Channel]
       set EthHdr [gets $Channel]
       set BufLen [gets $Channel]
       #puts $lFrame
       set lSuccess 1
       break
   }

   incr lCount
    }

    set HexData 1
    while { $HexData >= 0} {
        set HexData [gets $Channel]
   lappend lList $HexData
    }
    incr lCount

    close $Channel
    set TempList [join $lList ""]
    regsub -all " " $TempList "" ContHexString
    set lList [list $BufLen $ContHexString]

    if {$lSuccess != 1} {
   puts "could not find pattern - $aFrameName"
```

```
    }

    #puts $lList
    return $lList
}


# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::Bsts_GetPattern
#
# Overview: get pattern from formatted file
#
# Input parameters:
#  Arg1: Bsts file name
#  Arg2: Sequence/Pdu Name
#
# Return Data: None
#
# Side effects:

#  Successful operation: returns list of frame repetation and frame name
#  Handles both sequences and PDUs for IfEthGen ethgen Config ../bsts/bsts ....
#  Failure:
#
#
# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc Bsts_GetPattern {aFileName aPatternname } {
    set Channel [open $aFileName r]
    set lCount 1
    set lSuccess 0
    set lNoOfLinesInBstsFile 1584
    set lFrameFlag true

    while {$lCount <= $lNoOfLinesInBstsFile} {
   set lName [gets $Channel]

   if {$lName == "# Sequence list"} {
       set lFrameFlag false
   }
   if {$lName == $aPatternname} {

       if {$lFrameFlag == "false"} {
      set lPattern_Len [gets $Channel]
      for {set lCount 0} {$lCount < $lPattern_Len} {incr lCount} {
          set lElement [gets $Channel]
          lappend lList [split $lElement " "]
      }
       } else {
      # lName is now a frame
      set lMAC [gets $Channel]
      set lHdr [gets $Channel]
      set lFrameLen [gets $Channel]
      set lList [list 1 $lName]
       }

       set lSuccess 1
       break
   }
   incr lCount
    }
    close $Channel

    set lNewList [join $lList " "]

    if {$lSuccess != 1} {
   puts "could not find pattern - $aPatternname"
```

```
   return "null"
    }
    return $lNewList
}


# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Name: ::FindFile
#
# Overview: checks existence of file in file structure
#
# Input parameters:
#  Arg1: Bsts file name
#
# Return Data: 1 if file found, 0 otherwise
#
# Side effects:
#
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
proc FindFile {aFileName} {

    return [file isfile $aFileName]
}
```