

An approach to education oriented TCP simulation

Marko Lacković, Robert Inkret, Miljenko Mikuc
Department of Telecommunications
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia
{marko.lackovic; robert.inkret; miljenko.mikuc}@fer.hr

Abstract: This article describes one approach to a TCP simulation. The main goal was to develop an educational TCP simulator, which would explain main TCP features, but with enough flexibility for further upgrade to more advanced TCP mechanisms. The simulator is a part of the TCP Education Suite, a complete teaching tool for the TCP and TCP related problems.

1. INTRODUCTION AND MOTIVATION

1.1 The role of a protocol study in a modern telecommunication curriculum

We are witnessing the rapid growth of the number of the Internet users. This is being followed by an enormous increase of computers connected to the network, which made the Internet the largest global network. The importance of the Internet in modern life also changed the education approach in the field of Telecommunications. Knowledge about basic Internet protocols (like HTTP, FTP, TCP, IP) have become crucial for every telecomm engineer. A protocol study became the integral part of the telecommunication curriculum.

1.2 Theoretical and practical approach – advantages and drawbacks

There are several approaches to the teaching of communication protocols. The most common way is to describe the most important mechanisms and characteristics of each protocol, without taking into consideration the whole protocol specification. Less important parts can be excluded as they don't contribute to the overall understanding. Students are being taught on a passive and simplified model described in a passive (verbal and pictorial) manner. This approach is usually characterized by the lack of comprehension. On the other hand almost every student has his own image of the Internet and its protocols acquired from his experience as an Internet user. This image has little or no theoretical background. At this stage, student often doesn't make connections between those two images. The possible solution to this problem could be practical supplements to theoretical lectures. These should include network traffic analysis using appropriate network (e.g. university LAN) and software. This approach is often avoided because the students don't have enough knowledge to understand even filtered traffic, what could make even greater confusion. The other solution is to simulate the network, using just the most important protocol parts, but still maintaining the functionality.

There is a considerable amount of available simulators that can simulate some parts, or all parts of the network architecture. Simulation kernels are also available. The most disturbing fact about them is their price or quality. They are either too expensive (commercial products) or inappropriate for education (usually parts of student projects). The most solutions are coded in free simulation kernels like Ptolemy [1] and NS [2]. No concrete solution will be pointed out as they are subject to more thorough testing with students.

1.3 Simulation/Programming Tools

Following the previous discussion, the choice of the simulation tool is not straightforward and can affect a lot the final simulator in terms of usability, flexibility and portability between different operating systems. According to authors' opinion the simulating system OMNeT++, which is free, offers a good background for a simple simulator and satisfies most demands.

Beside the simulator itself, the final educational suite should include a theoretical overview of the simulated protocol, as well as an examination procedure which would be used to test the knowledge of the chosen protocol. Macromedia Director tool [3] was chosen for the protocol tutorial and examination process implementation.

1.4 The transport protocol

The simulated protocol should represent all the elements regarding protocol specification and functionality. We thought that the transport protocol, and TCP as its representative, would be the best choice. The transport protocol is often considered the most important part of the whole network architecture for several reasons. It has the central part in a data exchange between remote hosts – it relieves the application from worrying about the communication and architectural details of the lower layers, and on the other hand converts data to a form that is convenient for transportation across the network. Besides architectural, there is a pedagogical purpose of simulating the transport protocol. The lower-layer protocols are often better understood because of their dependency on the physical components (network infrastructure). The concept of the transport protocol in most cases gives a student a vague picture that is difficult to comprehend. This is caused by a logical end-to-end service in a connectionless environment like Internet.

2. TCP FEATURES AND SIMULATION CHARACTERISTICS

2.1 TCP and standard architecture stacks

There are several network architectures (or network stacks) describing the whole network traffic using layers. The most spread is the OSI architecture which divides the network into 7 layers, and the TCP/IP architecture which comprises 5 layers. The simulated protocol TCP is a representative of a transport protocol or the fourth layer of the TCP/IP architecture. This layer also matches the fourth layer of the OSI architecture or the transport layer. There is no regulation that would prescribe what transport protocol should or must be used. Therefore we cannot identify the fourth OSI layer with the TCP. It's often said that the TCP isn't *de jure* but *de facto* standard.

2.2 Basic TCP features

The transport protocol represents the lowest peer-to-peer connection, and its main responsibility is to provide a reliable service, and to relieve upper layers from worrying about the lower layer details. There are several issues every transport protocol should include in order to provide the expected service:

- Addressing,
- Multiplexing,
- Flow control, and
- Connection establishment/termination.

2.3 Simulation aspects of the TCP

The purpose of the simulator was to make an attractive teaching tool that would explain the main features of the TCP to undergraduate students with basic knowledge of the network architecture and the role of the transportation protocol in it. Postgraduate students from other engineering areas can use it to get a quick overview of the basic features. This approach doesn't require an in-depth analysis and the simulation of all aspects and mechanisms of the protocol. Only the most important issues should be included in simulation because of the clarity of the simulation alone. Additional tools can be used to analyze the network traffic, and give a better protocol insight than the complicated simulator.

The first problem was the consequence of the chosen approach. The TCP was to be modeled in the most appropriate way to accomplish the main goal – a creation of a simple, yet functional and easily comprehensible model of the transportation protocol (primarily TCP). We decided it would be better to use several modules to model the protocol. That approach would allow the user to analyze the change of data “inside” the transport layer and information needed and obtained by some communication mechanisms typical for the transport protocol. The user would be able to participate in the work of the whole

protocol by changing the information exchanged between modules.

The choice of the TCP mechanisms included in simulation had to make balance between clarity and functionality. Some features, like timers and buffers, had to be implemented in the model despite the fact that they aren't visible to the person analyzing input and output traffic of the TCP. Finally we decided to include following features: connection establishment and termination, addressing, multiplexing, data buffering, data segmentation, flow control (sliding window), error control (checksum).

2.4 OMNeT++

The TCP simulator was coded using the OMNeT++, a simulation [4]. OMNeT++ is a C++ based discrete event simulation suite for modeling communication networks and computer systems, and it obeys the GNU General Public License.

One of the most important demands on the simulation kernel was the request for the transparency of the mechanisms of the simulated protocol. This is almost impossible to achieve without using a GUI. This kernel uses the Tcl/Tk scripting language [5], which is a very good and efficient tool for creating applications in a graphical environment. The TCP simulator is portable to different platforms (it was written using Microsoft Windows NT environment and tested on Windows NT and Solaris (X -Window) operating systems) regarding the facts that Tcl/Tk is a multiplatform language and that the simulation kernel was written in C++.

2.5 Architecture

The simulated network consists of four computers (Figure 1a), each of them implementing a protocol stack. The main simulator window contains that network as seen from the TCP layer. All computers are interconnected making a mesh. Connections represent all possible TCP connections which are opened, used and closed during the simulation itself. End-to-end logical connections can be confusing to a beginner, but it should be taken into consideration that there are 3 more layers under this one, making the connection unreliable and changing the topology.

There are no other network devices (routers) that are normally used to connect computers on different networks (as in this case) because they don't implement the TCP, and don't participate in the TCP connection.

The sub module `stack` included in each computer implements the communication stack. It comprises two modules - `higher_layer` and `TCP` (Figure 1b). The `higher_layer` module serves as a traffic generator for the TCP and a destination for egress traffic from the TCP. This module should implement the application layer

(TCP/IP architecture) or the highest three layers of the OSI architecture. This is not the case in this simulator. Functionality and complexity of the higher layers were minimized because of the simulator's focus on the TCP functionality.

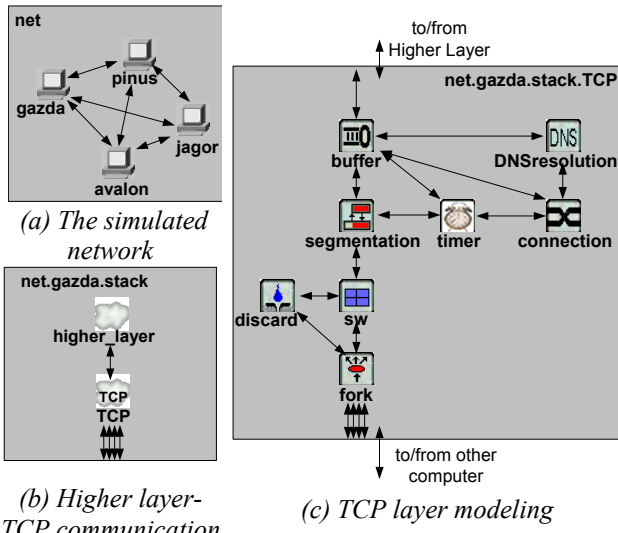


Figure 1 - The module hierarchy in the simulator

TCP uses a number of parameters for connection establishment/termination and data exchange (the sliding window mechanism).

Computers can establish a TCP connection to any other computer. A user interface was developed to make the process of connection description and parameterization easier (Figure 2). That parameterization is divided into higher layer properties, TCP properties and timers specification.

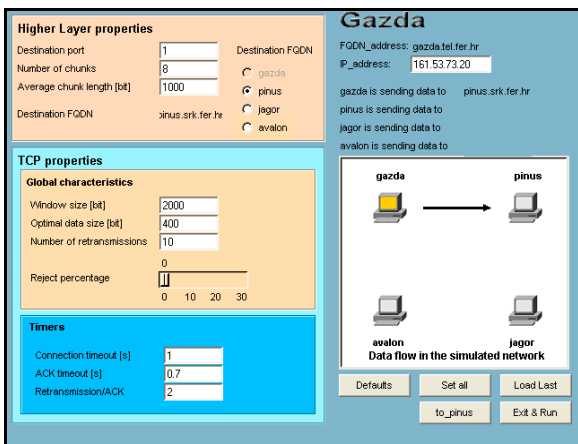


Figure 2 - User interface for simulator parameterization

A destination computer is determined for each source computer implying that each one will make one TCP connection during the simulation. Beside the destination FQDN (Fully Qualified Domain Name), the higher layer parameterization includes destination application port and

application traffic description (the number of chunks that are to be generated and the average chunk length).

TCP properties include initial window sizes (the sliding window mechanism) that are to be advertised by the senders, optimal data sizes that are to be transferred using specified connections and the number of retransmissions of the same segment before giving up one connection. Failures are simulated using the segment rejection percentage (percentage of segments that are rejected by the receiver).

The timer specification includes timeout for the connection timer. The connection will be terminated if it is idle during the whole interval. ACK timeout implements the cumulative retransmission strategy. The receiver won't acknowledge the received segments before the acknowledgement timer expires. Retransmission timeout expiration forces the sender to start retransmitting unacknowledged segments. That timer is the same for the whole queue of sent but unacknowledged segments.

The TCP itself was modeled using 8 modules (Figure 1c). A short description of all modules used to model the TCP layer follows.

Buffer

buffer stores the input data stream (represented in small data parts - chunks), and performs data aggregation/division. Data that application writes to the TCP layer don't necessarily correspond to the data stored in TCP segments. Transfer of the input data to the output is not always immediate. Too small data segments (20 octets of overhead + small data size) can increase the network load. Too large segments imply frequent IP fragmentation, and reduce performance.

After timer expiration, buffer sends out data segments of the most appropriate size (one of the simulation parameters).

DNS Resolution

DNS resolution converts a destination FQDN (Fully Qualified Domain Name) to an IP address. It contains the table of pairs FQDN – IP address for all computers in the simulated network.

This module is used just to emphasize the difference of addressing on the higher layer, the TCP and the IP layer, and is not part of the TCP in real implementations. It is accomplished usually by the application itself using the UDP for DNS queries. The real mechanism wouldn't be so transparent in the simulator, justifying this approximation.

Segmentation

Segmentation module adds header and pseudo header (communication primitive between TCP and IP) to the data of appropriate size that arrived from buffer. Data in the header depends on the connection data (socket/sliding window parameters/connection phase). Header data is shown in Figure 3.

```

ptr00A59F30 source_port (cPar) 1025 (L)
ptr00A59E40 destination_port (cPar) 1 (L)
ptr00A59D00 sequence_number (cPar) 1 (L)
ptr00A59D60 acknowledgement_number (cPar) 0 (L)
ptr00A59CF0 data_offset (cPar) 0 (L)
ptr00A59C80 URG (cPar) 0 (L)
ptr00A59BF0 ACK (cPar) 1 (L)
ptr00A59BA0 PSH (cPar) 0 (L)
ptr00A59B50 RST (cPar) 0 (L)
ptr00A59AF0 SYN (cPar) 1 (L)
ptr00A59A90 FIN (cPar) 0 (L)
ptr00A599E0 window_size (cPar) 2000 (L)
ptr00A59990 checksum (cPar) 201 (L)
ptr00A59920 urgent_pointer (cPar) 0 (L)
ptr00A59880 data (cPar) "This part of TCP segment contains data" (S)

```

Figure 3 - TCP header data

Timer

Timer module implements five timers: timer for buffer (avoiding long delays between data input and output), 2MSL (Maximum Segment Lifetime) timer used during connection termination, connection establishment timer, acknowledgement timer and retransmission timer

Connection

Connection module performs connection establishment (the *three way handshake* procedure). It assigns a client port for a new connection.

Sw

Sw module implements the sliding window protocol (flow control). It contains input and output queues for each established connection. It also implements the reliability features: acknowledgement and retransmission and performs the error control (checksum), terminates connection (four segments exchange) and deletes socket information after the connection is fully closed.

Discard

Discard module deletes TCP segments delivered to the wrong host (detected at fork), or that are damaged during transportation (wrong checksum, detected at sw).

Fork

Fork module directs input traffic to appropriate module analyzing the segment's flags and detecting the phase of the connection (connection establishment – to connection module, data exchange and connection termination – to sw module).

2.6 Applications

The purpose of this simulator is to make some issues regarding TCP more comprehensible. Although it was planned to be used just as supplemental to the theoretical lectures, not as the only way of teaching TCP, the whole suite itself can serve as a complete teaching tool. One of the main assumptions is that the user who knows the TCP basics tests the “live” TCP network using simulator and sees if he got all the facts right. The main advantages of the simulator are the clear and transparent insights to the most important TCP features combined with interactivity.

There are two ways the simulator can be used. One way is to use the simulator with predefined or custom

parameters, and just observe the work of the TCP by watching the sequence of exchanged messages inside the TCP or between TCP entities (Figure 4). User can examine the contents of each exchanged message (Figure 3). This can be appropriate for the first stage when the student refreshes his TCP knowledge on the ideal network with no data losses. The next stage could be a simulation with defined traffic reject percentage (this simulates the damages and loses during transportation). Some mechanisms (such as rejection and retransmission) should be visible now. The predefined parameters allow the instructor to create some interesting situations (different ratios of acknowledgement and retransmission ratio or the ideal data length in the TCP segment and initial window size). The other way of using the simulator is to “manually” simulate errors or change the course of action in the network by changing some of the parameters in the message header. This can be used to simulate effects during connection establishment and termination that are not visible in the ideal network.

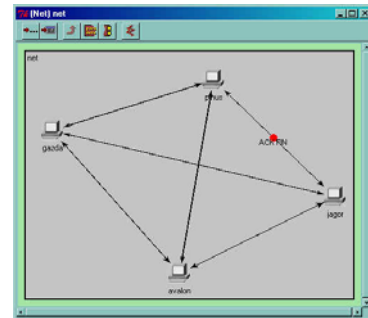


Figure 4 - Message exchange monitoring in the main simulator window

There are several topics that should be covered by simulation in order to fully demonstrate a communication between two remote TCP layers. The whole TCP concept of data exchange can be divided into 3 main parts. The first part is the connection establishment (initiated by the first data chunk from the higher layer), which includes the exchange of three segments. This part includes the analysis of flags and their role in the communication. The data exchange itself allows better understanding of the sequence numbers and acknowledgement. Students can observe changes in the input and output queues of the module that implements the sliding window mechanism. The last data chunk from the higher layer initiates the active close of the connection. There are four segments to be exchanged before the complete connection close. Different client and server behavior during the connection termination are visible by observing actions in the TCP layers of remote computers.

Beside the exchanged messages' characteristics (flags, message names), the current TCP state can be determined from the phase of the modules inside the TCP layer that follows states in the TCP state and transition diagram. The

TCP can be specified using 11 states and numerous transitions that depend on the role of a computer in the network (client or server) [6].

3. TCP EDUCATION SUITE

The whole suite is intended to be used as a complete TCP education tool (Figure 7a), which includes explanations of basic and advanced TCP features and simulation/demonstration. It includes four parts:

- TCP Simulator
- TCP Tutorial (Figure 5),
- TCP Connection Painter (Figure 6), and
- Ethereal.

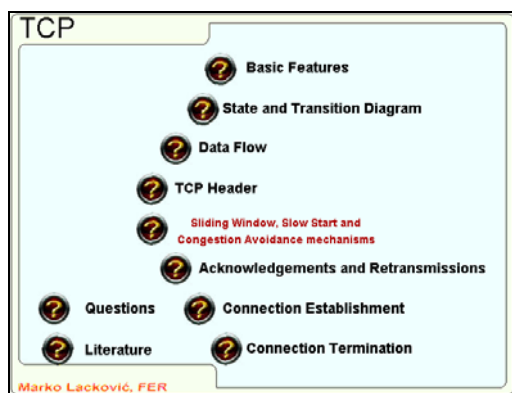


Figure 5 – TCP Tutorial

TCP Tutorial is an interactive theoretical supplement to the simulator. It covers all common TCP features (non implementation specific), including those not included in the simulator (like Karn or Nagle algorithm [6]). Those advanced features would make the simulator more difficult to use and would blur basic features. It has been developed using the Macromedia Director tool, and can be used on Windows platforms as a standalone executable file, or in a HTML form using Macromedia Internet browser plug-in, allowing its use on the Unix/Linux based platforms.

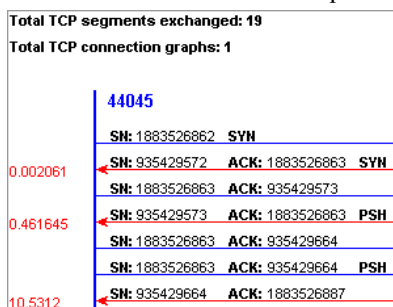


Figure 6 – Connection Painter tool

However, more advanced points are covered using Ethereal, a free protocol analyzer based on network traffic snapshot analysis [7]. It is portable between Windows and Unix/Linux platforms thus not limiting the portability of

the whole suite. Ethereal allows analysis of TCP headers (along with other protocol headers), and trace of all segments belonging to one TCP connection.

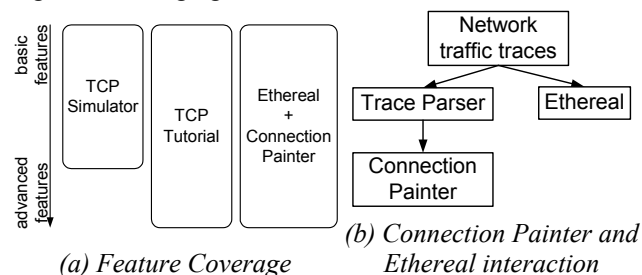


Figure 7 - TCP Education Suite

The Connection Painter tool was developed using Tcl/Tk as a supplement to Ethereal. It is capable of reading traffic trace files used by Ethereal using a parser which filters the TCP traffic and converts it to a suitable form for Connection Painter (Figure 7b). Painter is capable of drawing connection diagrams allowing visual supplement to Ethereal's textual view (Figure 6).

4. CONCLUSIONS AND PERSPECTIVES

It's very difficult to evaluate the quality of an education simulator before it was thoroughly tested in the classroom. The Suite was used as a part of the "Communication Protocols" curriculum (prof. Lovrek). More than a hundred 4th year students from the Telecommunication department were involved. The main goals were achieved. The simulator was simple enough to be used without preparation, yet able to explain basic and some advanced TCP features.

There are two ways of improving the simulator – adding new protocols (like UDP which would allow user to analyze both transport protocols and differences in their applications) and implementing more advanced TCP features (like the slow start mechanism or the Nagle algorithm).

REFERENCES

- [1] Ptolemy home site <http://ptolemy.eecs.berkeley.edu>
- [2] Network simulator home site <http://www.isi.edu/nsnam/ns/>
- [3] <http://www.macromedia.com/software/director/>
- [4] <http://www.hit.bme.hu/phd/vargaa/omnetpp.htm>
- [5] Brent Welch, "Practical Programming in Tcl and Tk", Prentice Hall, Upper Saddle River, New Jersey, 1999
- [6] W. R. Stevens, "TCP/IP Illustrated, Volume1 (The Protocols)", Addison-Wesley Publishing Company, Reading, Massachusetts
- [7] Ethereal home site <http://www.ethereal.com>