TCP Offloading Engine

Project Proposal - CSEE 4840, Spring 2014

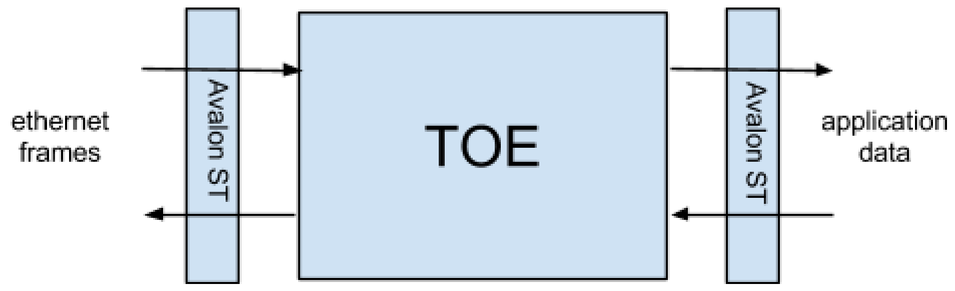
Clementine Barbet (cb3022) - Christine Chen (cpc2143) - Qi Li (ql2163)

I. Motivations

TCP/IP serves predominantly as a protocol suite for exchanging data over the Internet. It runs across Ethernet networks which provides up to 10G data rate capability. At such high speeds, the processing of the TCP/IP stack represents a significant overhead for the CPU, and the Linux kernel itself can limit the throughput.
A solution to these problems, called TCP offloading, consists of building dedicated hardware that implements the TCP/IP stack, instead of handling this stack with software.

II. High-Level Project Description

The aim of this project is to build a 10G TOE (TCP Offloading Engine) soft IP Core that uses the Altera standard interfaces (Avalon ST) for sending and receiving Ethernet packets. Hence, this TOE IP Core would allow users to efficiently handle TCP/IP over 10G Ethernet on various Altera platforms, such as SoCKIT or AoE Solarflare.



III. First Milestone: TCP/IP protocol

TCP/IP is a connection-oriented protocol that aims at allowing reliable data transfers. As a result, it is quite a complex protocol.

The first step of the project is ensuring that our team acquires a thorough understanding of

TCP/IP through state diagrams and through the implementation of a minimal TCP/IP using regular Linux methods for reading / writing raw Ethernet frames (partly extracted from the Linux kernel).

This will also allow the group to determine more precisely the subset of TCP that will be implemented in the TOE.

IV. Second Milestone: Testing Framework

The major difficulty in implementing the TOE is to test its compliance with the TCP/IP protocol, especially because there is an enormous amount of possible transactions. In order to ease development, we need to build a testing framework. Producing this framework constitutes our second milestone.

Three ideas are being considered:

- First, pcap files would be used as a golden reference. This method could be useful at the beginning but could prove to be too limited in scope for implementations down the line. For instance, pcap files include various labels of data such as sockets and time stamps, so to include these in the testing would require hard-coding of the variables. This, in itself, could limit verification to a pre-determined set of conditions.
- The second solution could be to simulate our IP core in hardware and have C code for choosing the outputs of the IP core and sending them to the network via raw socket calls; conversely, whenever a packet is received in software, it would be outputted to the IP core.
- The last solution could be to combine Modelsim with C language by, instead, using files for communication. It amounts to writing Verilog test tools that accept data from Avalon ST and writes to a file. A separate C program file will be used to read / write the files, passing data to/from the network.

V. Third Milestone : Implementation

For the last milestone, we plan on having a functional TOE, implemented using a combination of both SystemVerilog and our testing framework, running on the SoCKIT board.