10 Gbps TCP/IP streams from the FPGA for High Energy Physics

10 Gbps TCP/IP streams from the FPGA for High Energy Physics

2014 J. Phys.: Conf. Ser. 513 012042

(http://iopscience.iop.org/1742-6596/513/1/012042)

Journal of Physics: Conference Series **513** (2014) 012042 doi:10.1088/1742-6596/513/1/012042

10 Gbps TCP/IP streams from the FPGA for High Energy Physics

Gerry Bauer[6], Tomasz Bawej[2], Ulf Behrens[1], James Branson[4], Olivier Chaze[2], Sergio Cittolin[4], Jose Antonio Coarasa[2], Georgiana-Lavinia Darlea[6], Christian Deldicque[2], Marc Dobson[2], Aymeric Dupont[2], Samim Erhan[3], Dominique Gigi[2], Frank Glege[2], Guillelmo Gomez-Ceballos[6], Robert Gomez-Reino[2], Christian Hartl[2], Jeroen Hegeman[2], Andre Holzner[4], Lorenzo Masetti[2], Frans Meijers[2], Emilio Meschi[2], Remigius K Mommsen[5], Srecko Morovic[2,7], Carlos Nunez-Barranco-Fernandez[2], Vivian O'Dell[5], Luciano Orsini[2], Wojciech Ozga[2], Christoph Paus[6], Andrea Petrucci[2], Marco Pieri[4], Attila Racz[2], Olivier Raginel[6], Hannes Sakulin[2], Matteo Sani[4], Christoph Schwick[2], Andrei Cristian Spataru[2], Benjamin Stieger[2], Konstanty Sumorok[6], Jan Veverka[6], Christopher Colin Wakefield[2] and Petr Zejdl[2]

[1] DESY, Hamburg, Germany
[2] CERN, Geneva, Switzerland
[3] University of California, Los Angeles, Los Angeles, California, USA
[4] University of California, San Diego, San Diego, California, USA
[5] FNAL, Chicago, Illinois, USA
[6] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
[7] Now at Institute Rudjer Boskovic, Zagreb, Croatia

E-mail: petr.zejdl@cern.ch

Abstract. The DAQ system of the CMS experiment at CERN collects data from more than 600 custom detector Front-End Drivers (FEDs). During 2013 and 2014 the CMS DAQ system will undergo a major upgrade to address the obsolescence of current hardware and the requirements posed by the upgrade of the LHC accelerator and various detector components. For a loss-less data collection from the FEDs a new FPGA based card implementing the TCP/IP protocol suite over 10Gbps Ethernet has been developed. To limit the TCP hardware implementation complexity the DAQ group developed a simplified and unidirectional but RFC 793 compliant version of the TCP protocol. This allows to use a PC with the standard Linux TCP/IP stack as a receiver. We present the challenges and protocol modifications made to TCP in order to simplify its FPGA implementation. We also describe the interaction between the simplified TCP and Linux TCP/IP stack including the performance measurements.

1. Introduction

The central data acquisition system (DAQ) of the Compact Muon Solenoid (CMS) experiment at CERN collects data from more than 600 detector front-end drivers via custom links connected

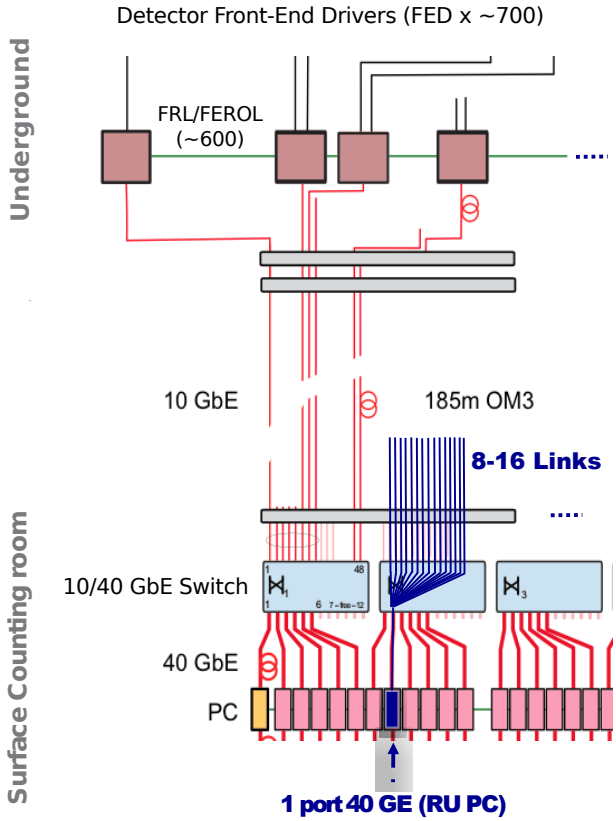

Figure 1. A part of the new DAQ front-end readout system. The data flows from the FRL/FEROL boards in the underground through optical 10GE links to the Readout Unit PCs (RU) at the surface via a layer of 10GE to 40GE switches which also merges several 10GE links.

to the interface boards called Front-end Readout Link (FRL). The FRL then transfer the data to a commercial Myrinet network based on 2.5 Gbps optical links.

During the first long-shutdown of the LHC in 2013 and 2014 the CMS DAQ system is being upgraded. As part of the upgrade process, new FRL hardware which replaces the interface to the Myrinet network has been designed. The new hardware called FEROL (Front-End Readout Optical Link) is based on an FPGA and provides a new 10 Gbps optical link connected to an Ethernet network. The data are transmitted over this link using the TCP/IP network protocol. TCP/IP will provide a reliable data connection between the front-end readout system in the underground and the rest of the eventbuilder network at the surface (Figure 1). More details on the new DAQ can be found in [1, 2]. The FEROL hardware is described in [3].

2. Requirements

The following overall requirements are identified for the connections between the new FRL hardware and DAQ eventbuilder network:

- Provide reliable (loss-less) connection between underground and surface equipment.
- Ability to provide back-pressure to the FEDs when upstream congestion is detected.
- Possibility of sending several TCP data streams over one 10GE link.
- Operation at 10 Gbps.

3. TCP/IP Network Protocol

TCP/IP is the suite of communications protocols used for data delivery over the Internet and other similar networks since the early 1980s. It is also referred to as the Internet protocol suite [4]. TCP/IP takes its name from the Transmission Control Protocol (TCP) defined in RFC 793 and the Internet Protocol (IP) defined in RFC 791. TCP and IP were the first two protocols defined in the suite. IP deals with packets (basic unit of data) and provides addressing and routing in an IP network. TCP establishes and manages a connection and provides reliable data transport over an IP network.

The following features are already part of TCP:

- *Reliable and in-order transmission* – Data sent over a network are protected by sequence number and checksum. The receiving host rearranges the received segments according to the sequence number. In case of packet loss, checksum error or link error the data are retransmitted.
- *Flow control* – End-to-end flow control respects the occupancy of the buffer in a receiving PC. In case the receiver's buffer is almost full, the TCP flow control will automatically decrease the rate of data sending to the receiver or temporary stop the transfer and allow the PC to process the received data.
- *Congestion control* – Avoids a congestion collapse (described in section 3.3) by limiting the rate of data sent to the network below a rate that would create a network congestion. As a result the congestion control allows transmitting multiple TCP streams on the same link and allows to merge several links, which are not fully utilized, together using a switch.

These features match the requirements specified in the previous section. TCP/IP is also a standard and well known protocol suite implemented in all mainstream operating systems. Debugging and monitoring tools are openly available (tcpdump, wireshark, iperf, ...) and don't need to be developed from scratch as it would be in case of using a different custom protocol. The network can be composed from off-the-shelf components where multiple vendors are available.

Therefore, TCP/IP over Ethernet network was chosen as the new readout link. In order to achieve the maximum data throughput, TCP has been implemented in the FPGA as opposed to running in an embedded CPU inside the FPGA.

3.1. Simplified and unidirectional TCP

Implementing TCP in an FPGA is in principle a very difficult task. The software TCP implementation in the Linux kernel 3.4 takes about 15 000 lines of C code, not counting the IP layer, and is executed by a CPU. Running one 10 Gbps TCP stream can easily saturate one of the CPU cores[1].

To limit the TCP implementation complexity, we designed a simplified and unidirectional version of the protocol. Several simplifications were possible:

- Detector data flows in only one direction from a hardware sender to a receiving PC. It is a client–server architecture, where the client is the hardware sender and the server is the receiving PC running the full Linux TCP/IP stack. The client opens a connection (stream) to the server, sends the data and keeps the connection open until it is closed. In the simplified TCP version, the connection close is replaced by connection abort. If an error is detected, the connection is aborted immediately. In addition, the server cannot send any user data back to the client. Only the acknowledgment packets are sent back but these are part of the TCP protocol. Figure 2 shows the TCP state diagram. By removing the states

[1] Measured on DELL PowerEdge C6100 with Intel Xeon X5650 @ 2.67 GHz and Silicom(Intel) 10GE NIC.

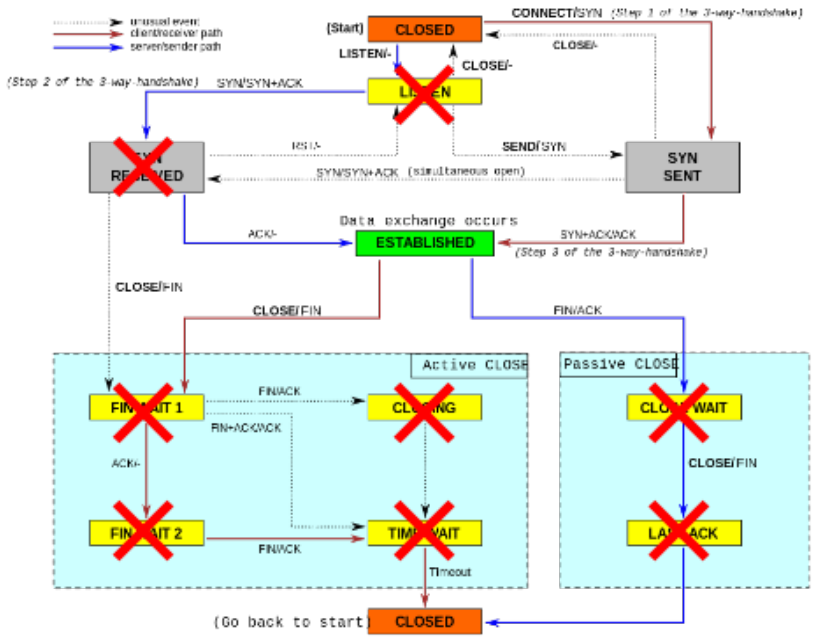

Figure 2. A simplified TCP state diagram [5]. The removed states are marked with red crosses. The states `LISTEN`, `SYN_RECEIVED`, `CLOSE_WAIT` and `LAST_ACK` are related to the server part of TCP and are not implemented. The states `FIN_WAIT_1`, `FIN_WAIT_2`, `CLOSING` and `TIME_WAIT` provide a reliable stream closing. It is replaced by a connection abort with a direct transition from the `ESTABLISHED` state to the `CLOSED` state.

related to the server and closing parts of TCP we reduced the number of required states in the diagram from 11 to 3.

- The DAQ network topology is fixed and not changed during data taking. The network is designed with sufficient bandwidth in to avoid packet congestion. This reduces the requirements for the TCP congestion control. The congestion control can assume that each TCP stream has always enough available bandwidth in the network and that the streams don't need to compete for it. Otherwise, the congestion control has to determine the amount of available bandwidth and carefully share it with other streams already running on the network. Therefore we simplified the complex TCP congestion control to an exponential back-off to decrease the throughput when temporary congestion is detected in order to avoid a congestion collapse. A fast-retransmit algorithm is also implemented to improve the throughput in case of single packet losses. In this case, the data are retransmitted immediately without waiting for a timeout.

The following standard TCP features are implemented:

- Jumbo frame support – up to 9000 bytes in one frame sent over an Ethernet network.
- Nagle's algorithm – merges data fragments to fully utilize the jumbo frame.
- Window scaling – the receiver can request more than 64KB of data to be sent.
- Silly window avoidance – stops sending data when the receiver is busy and requests a small amount of data.

The following TCP features are not implemented: timestamps, selective acknowledgments, out of band data.

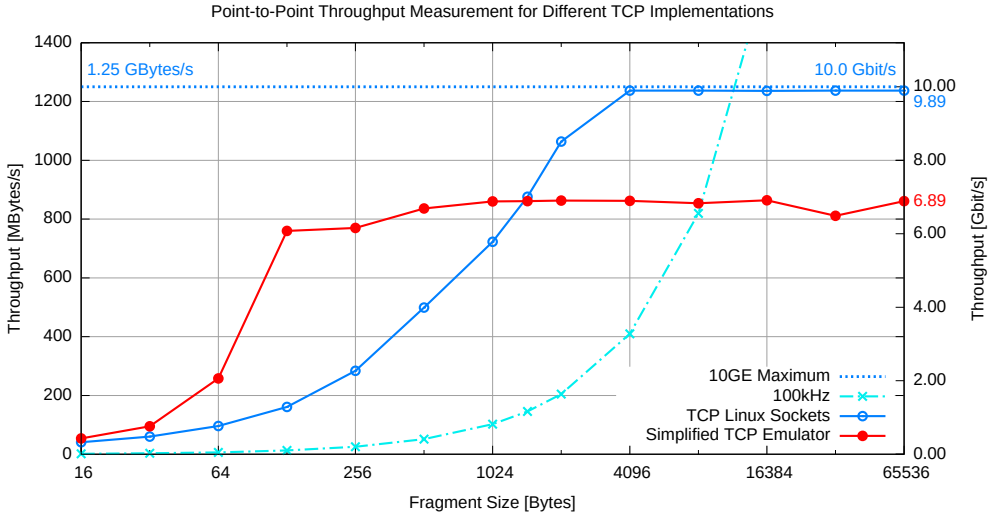

Figure 3. Throughput measurement between two PCs as a function of fragment size.

3.2. Software verification of the simplified TCP

For protocol verification and testing we developed an emulator implementing the simplified and unidirectional TCP/IP in software. It runs as an user space program and the network interface is accessed using a non-standard network socket provided by the PF_RING kernel driver and framework [6]. The PF_RING framework copies the received packets directly from the network device driver to the user space program and thus bypasses the standard kernel network processing. In particular, it bypasses the kernel TCP/IP stack which would normally interfere with the TCP/IP user space emulator.

Figure 3 shows the throughput of the simplified TCP emulator compared to Linux sockets. The maximal achieved throughput is about 7 Gbps for emulator and 9.89 Gbps for standard Linux TCP/IP sockets. The simplified TCP handles small fragment sizes more efficiently due to less movements in the PC's memory. The higher fragment sizes are handled less efficiently because PF_RING doesn't provide a kernel bypass for packet transmission. The standard Linux network processing is used. The user space process has to provide the full Ethernet frames and therefore is lacking any support for hardware acceleration like TCP and IP checksum calculations and segmentation offload.

3.3. Congestion Control

The TCP sender attempts to transmit the data at the maximum rate driven by the rate of the incoming data from the detector, but can be limited by the congestion control settings. Due to fluctuations in throughput caused by the receiving PCs (Linux is not a real-time OS), the TCP buffers in the sender can accommodate temporarily a higher amount of data compared to sustained transmission. These data are sent out later at a higher rate. This peak can create temporary congestion in the switch, and depending on the switch buffer size, induce packet loss. The sender compensates the lost packets by packet retransmission and thus increases the outgoing rate even more. If the amount of the retransmissions is not controlled, the system will stay in the congested state even after the initial cause of the congestion disappeared. This state is known as a congestion collapse. It is a stable state with low data throughput, the rest of the bandwidth is taken by the retransmissions.

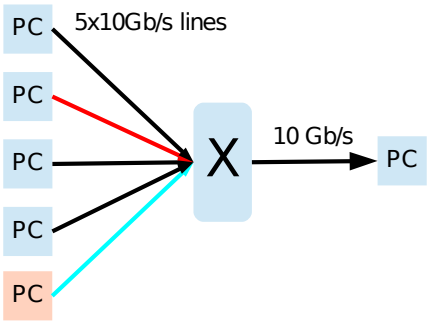

Figure 4. Testbed configuration for congestion control verification. The five sender PCs are running the simplified TCP emulator, the receiver PC is running standard Linux TCP/IP stack. The sender PCs are configured to send streams carrying fragments of 2048 bytes at 125 kHz, therefore the data rate is 2.048 Gbps.

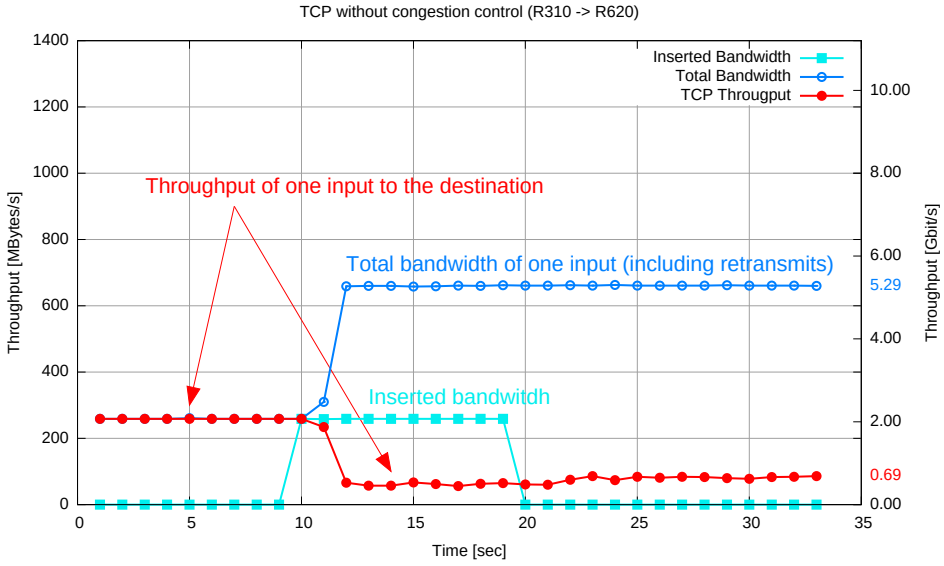

Figure 5. The data throughput and total bandwidth used by one link connected to the switch during network congestion without congestion control in place. When the link which originally created congestion stops sending, the system does not recover (congestion collapse).

Figure 4 shows the testbed we used for the verification of the simplified congestion control algorithm. The sender PCs are configured to send one TCP stream each at 2.048 Gbps.

The first test is without congestion control in place. Figure 5 shows the throughput of one link from the sender PC. At the beginning of the test, only four senders are used. After 10 seconds the 5^{th} sender starts to transmit data. The total attempted throughput of all five links is now 10.24 Gbps which is a little bit more than the bandwidth of the outgoing link. The switch starts dropping some packets out. TCP starts to retransmit the lost packets. This increases the data rate up to 5 Gbps for one link, causing even more congestion in the switch. The total useful throughput drops to $5 * 0.69$ Gbps $= 3.45$ Gbps. After another 10 seconds (at Time $=$ 19 s) the 5^{th} sender is stopped. However, the switch stays in the congested state (congestion collapse) due to the uncontrolled TCP packet retransmissions of the remaining 4 senders. The useful data throughput of one link stays at 0.69 Gbps.

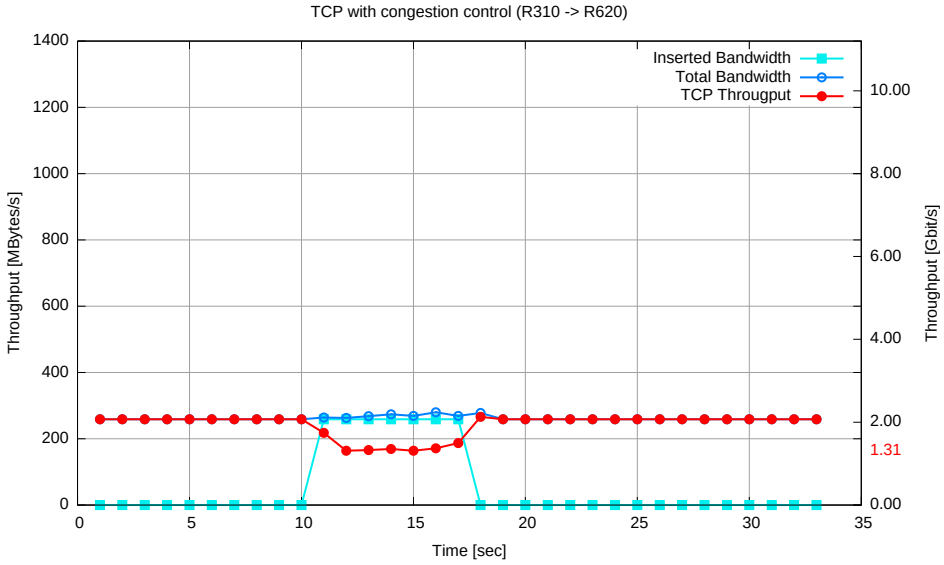

Figure 6. The data throughput and total bandwidth taken by one link connected to the switch during network congestion with the congestion control in place.

Figure 6 shows the same test with the simplified congestion control in place. The total bandwidth of one stream is increased during the congestion, but it is not causing any additional congestion in the switch. The useful throughput decreases during the network congestion, but when the source of the congestion disappears, the system recovers. Therefore, the congestion control – even simplified – is still able to recover from a temporary network congestion.

4. Summary

We presented a simplified and unidirectional version of TCP protocol suitable for a direct implementation in a programmable hardware (FPGA). The simplifications are compatible with RFC 793 specification, and therefore a PC with the standard Linux TCP/IP stack can be used as a receiver. The protocol including the simplified congestion control was verified with the software emulation.

As a part of the new CMS data acquisition system, the simplified TCP is foreseen to run in 600 devices based on an FPGA and providing reliable 10 Gbps data connections between the front-end readout system in the underground and the eventbuilder network at the surface.

Acknowledgments

This work was supported in part by the DOE and NSF (USA) and the Marie Curie Program.

References

[1] Holzner A et al. The new CMS DAQ system for LHC operation after 2014 (DAQ2) (these proceedings)
[2] Mommsen R K et al. File based high level trigger (these proceedings)
[3] Zejdl P et al. 10 Gbps TCP/IP streams from the FPGA for the CMS DAQ Eventbuilder Network, submitted to JINST.
[4] Wikipedia, the free encyclopedia. The Internet protocol suite. Available: `http://en.wikipedia.org/wiki/Internet_protocol_suite`
[5] Wikipedia, the free encyclopedia. The TCP State Diagram. Available: `http://en.wikipedia.org/wiki/Transmission_Control_Protocol`
[6] Deri L et al. PF_RING: High-speed packet capture, filtering and analysis. Available: `http://www.ntop.org/products/pf_ring/`