Hardware based packet filtering using field programmable gate arrays

April 15, 2010

Author: Timothy Whelan

Supervisor: Mr Barry Irwin
Rhodes University Computer Science Department

Background to the project

Currently in operation by the Security and Networks Research Group (SNRG) is an internet telescope that monitors packet flow over the internet and captures the packets of data that pass by the telescope. During the 5 years of operation over 500 gigabytes of data have been captured in packet form and the Hilbert Curve and Inetvis projects have been developed with the aim of visualising various aspects of the captured data [2]. Due to the vast number of packets that need to be processed when visualising the collected data, high performance means of filtering the captured packets to withdraw packets of interest are desired. Field programmable gate arrays (FPGAs) appear to be well suited to this purpose. FPGAs possess reconfigurable logic blocks [3] and hence are well suited to comparing properties of desired packets to fields within the captured packets and can be reconfigured to select packets based on alternative criteria.

Previous research

Yamaguchi et al. described in [6] how searches based on pattern matching can be made faster with the use of FPGAs with an almost proportional increase in the speed of the search as the FPGA used increases in size. Tsoi et al. [5] also describe how an FPGA can be implemented as a search engine for hash keys of the RC4 algortihm. The system designed by Tsoi et al. also shows how parallelism can be built into the FPGA system to increase the speed of searches. Of particular interest is a paper written by Ioannis Sourdis and Dionisios Pnevmatikatos [4] detailing how an FPGA system was implemented with the intention of matching strings used in pattern matching in network intrusion detection systems (NIDS). Sourdis and Pnevmatikatos show that their system exceeded a throughput of 11Gbps when comparing network packets passing through the FPGA with 50 different search strings used in the rule set of Snort, the open source software NIDS used in their research. This reseult is promising for my application of using an FPGA to filter packets based on properties of the packets as searching for strings in packets is deemed to be more computationally expensive than checking numerical fields in the packets [4]. Hence, for filtering based upon fioleds such as port numbers or IP addresses, the use of an FPGA can be expected to prove very efficient by current speed standards when compared to other, software based, packet filtering systems.

Project goals and research aims

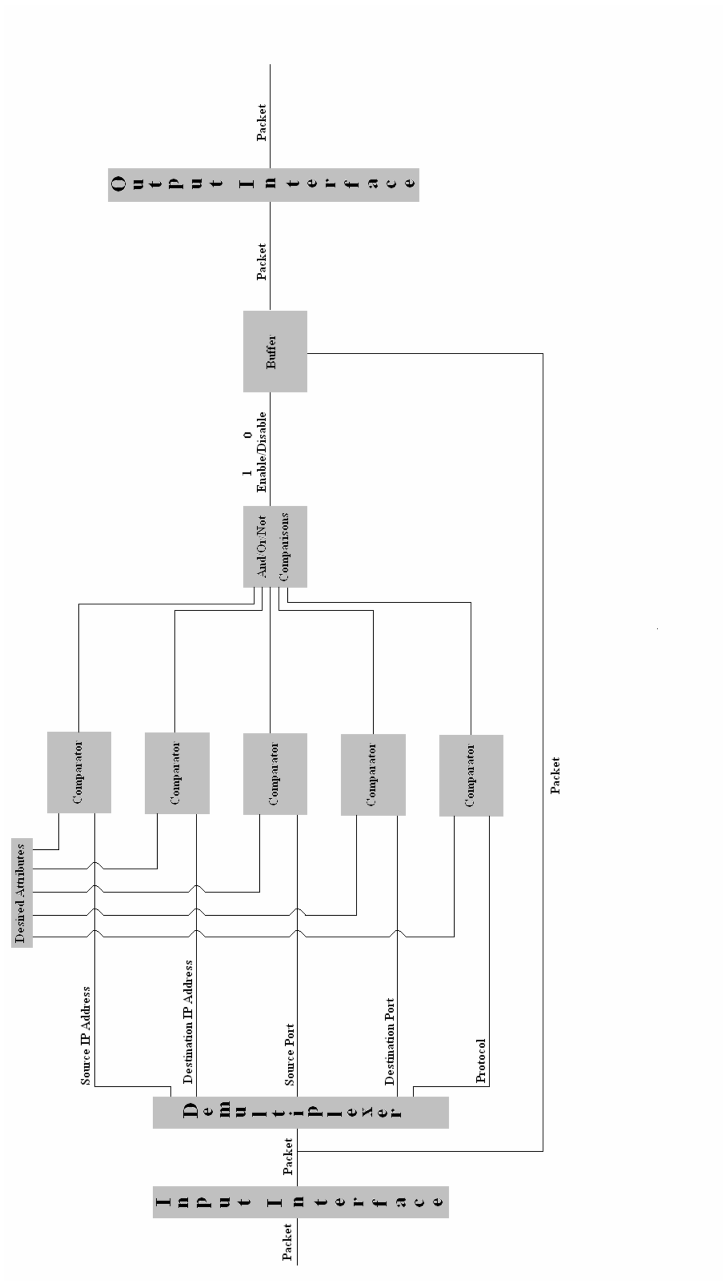
The goal of this project is that by the end of the project a working implementation of an FPGA packet filter will be produced that can filter packets based upon simple properties such as source/destination IP addresses or ports (including ranges of IP addresses or ports) or a protocol used by the packet thus enabling one to extract all SIP traffic discovered between a.b.c.0/8 and x.y.z.0/16 on ports above 1000 as an example. This will enable comparative analysis to be performed on timings for the FPGA implementation and existing systems such as the Inetvis and the Hilbert Curve projects. It is believed that, based upon previous research discussed above, an FPGA implemented packet filter may prove to be faster than current software based packet filtering systems. However, this project makes use of an FPGA constructed in 2002 or soon thereafter and so speed timings of the final implementation may only give estimates of what an FPGA system using a modern FPGA may be capable of.

System Requirements

Designs are to be based upon the Xilinx XCS10XL FPGA using the Xilinx Integrated Software Environment (ISE) on a Windows XP platform. A minimum of 128MB of RAM will be needed by the Xilinx ISE. The Xilinx ISE further specifies the following minimum system requirements [1]:

- 500MHz CPU speed
- Colour VGA monitor with a minimum resolution of 640 x 480
- a parallel port must be available on the system to enable designs to be uploaded to the FPGA

Initial Project Design



Packets enter the FPGA through an input interface. Ideally this would be an Ethernet interface but initial prototyping might be done on a USB interface so that input rates can be reliably controlled. A copy of the packet is then placed in a buffer while the source/destination IP/port address and the protocol of the packet are placed on the inputs of comparators by a demultiplexing unit. The matching criteria such as the IP address range of desired packets is placed on the second input of the comparators. The header attributes are compared with the desired attributes and the result (1 if match, 0 if not matched) is passed on to another unit which combines the results to determine if the packet must be accepted or discarded by passing an enable or disable control bit to the buffer containing a copy of the packet. If the buffer receives an enable bit the packet is forwarded to the output interface, typically similar to the input interface. If a disable bit is sent to the buffer the packet is discarded and the process is repeated for the next packet. Note: additional comparators can be used to check header attributes against disjoint sets of desired attribute e.g. desired destination port numbers are <100 AND >1000. Currently captured packets are stored in the libpcap format.

Risks associated with FPGA platforms

FPGA's make use of volatile RAM for storage of operating instructions which require power to be supplied to the device to load instructions onto the FPGA and to remain available until the FPGA's task is complete. Should any loss of power occur the FPGA will lose all data stored on the FPGA including the instructions that specify the operation of the FPGA. This problem can be mitigated though by making use of the otpion to load instructions from a serial PROM chip that can be attached to the FGPA and configured to automatically download instructions to the FPGA upon power up.
Developments and progress in hardware technologies can be easily incorperated into the system by recompiling the software specifying the system design for any new FPGA systems that might be developed enabling the system to constantly operate on the latest designs with minimal disruption to the system platform. However, should the format of packets change after system implementation the system would need to be redesigned to accomodate the changes in packet formatting. With the continual rapid development of network technologies, particularly the impending implementation of IPv6, this is a definite risk to the design of any system that will filter packets in a fashion similar that intended by this system.
Other usual risks associated with hardware are also present; should such a system be implemented fully protection from moistrue, extreme temperatures and power fluctuations will need to be provided.

Intended Timeline

Date	**Objective**
2nd Mar	**Present Project Proposal**
3rd - 20th Mar	**Develop FPGA capable of sorting data input**
21st Mar - 19May	**Focus on literature review**
21st May - 17th July	**Extend initial design to include packet formats and simplify I/O**
18th - 19th July	**Prepare presentation of project work to date**
20th July - 10th Aug	**Write report up to and including Introduction**
11th - 26th Aug	**Perform speed comparison tests with FPGA system and current software implementations of packet filters**
27th Aug - 12 Sept	**Work on short paper**
13th - 28th	**Sept Final optimisations to FPGA design and perform more timings**
29th Sept - 19th Oct	**Continue project report**
20th - 24th Oct	**Prepare final presentation**
25th - 30th Oct	**Complete project report**
1st Nov	**Hand in project report**

References

[1] *ISE 1.5i Release and Installation Guide*, 2002.

[2] Security and networks research group current projects, 2010.

[3] Comer, D. *Digital logic and state machine design*, 3rd ed. Oxford University Press, 1995. ISBN 0-91-510723-3.

[4] Sourdis, I., and Pnevmatikatos, D. *Fast, Large-Scale String Match for a 10Gbps FPGA-Based Network Intrusion Detection System.* Springer Berlin / Heidelberg, 2003. http://www.springerlink.com/content/etq4r1xaeqvtl0db/.

[5] Tsoi, K. H., Lee, K. H., and Leong, P. H. W. A massively parallel rc4 key search engine. In *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (2002). http://www.computer.org/portal/web/csdl/doi/10.1109/FPGA.2002.1106657.

[6] Yamaguchi, Y., Maruyama, T., and Konagaya, A. High speed homology search with fpgas. In *Pacific Symposium on Biocomputing* (2002). http://helix-web.stanford.edu/psb02/yamaguchi.pdf.