

# TCP-Splitter: A Reconfigurable Hardware Based TCP/IP Flow Monitor

David V. Schuehler  
dvs1@arl.wustl.edu

John W. Lockwood  
lockwood@arl.wustl.edu

Applied Research Laboratory (ARL)  
Department of Computer Science and Engineering  
Washington University in Saint Louis

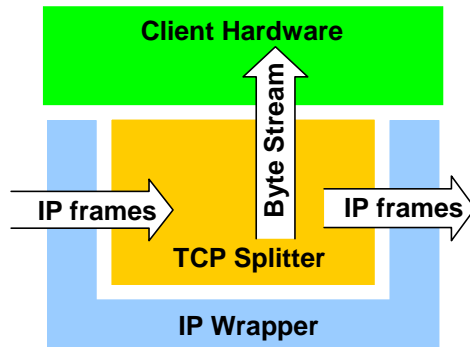
<http://www.arl.wustl.edu/arl/projects/fpx>

## Outline

- Motivation
- Hardware Platform
- Design
- Applications
- Results
- Questions

## Problem Statement

- Develop a lightweight network monitoring component that operates at multi-gigabit/second line rates.



## Why work with TCP?

- Over 85% on Internet traffic is TCP based
- Internet is growing
- TCP is a proven reliable transport for data delivery
- Provide high speed active networks the ability work with TCP flows

## Why not use a software based monitor?

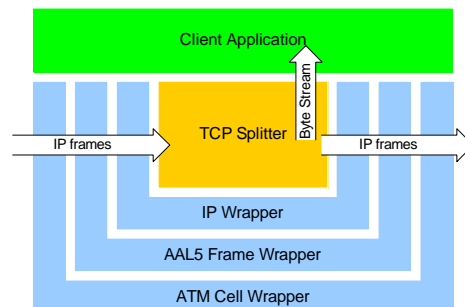
- Difficult to achieve desired performance

## Why not implement a full TCP stack ?

- Large memories required for reassembly
- Limited number of simultaneous connections
- Acts as a connection endpoint
- Not a lightweight solution

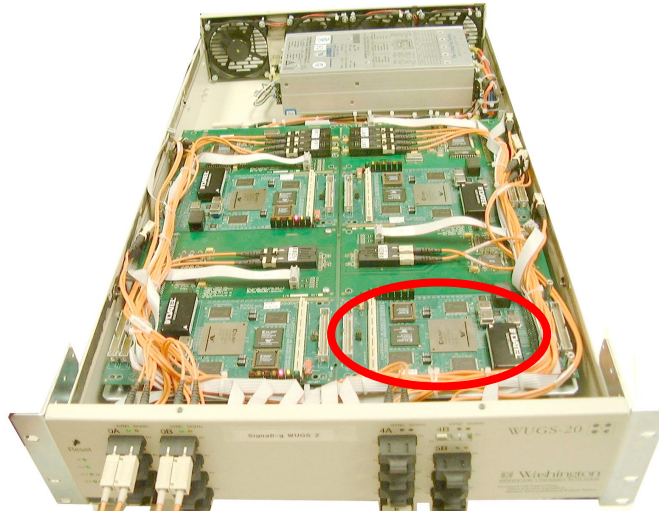
## Solution

- Develop TCP flow monitor: TCP-Splitter
- Leverage existing hardware infrastructure
- Expand upon Layered Protocol Wrappers research

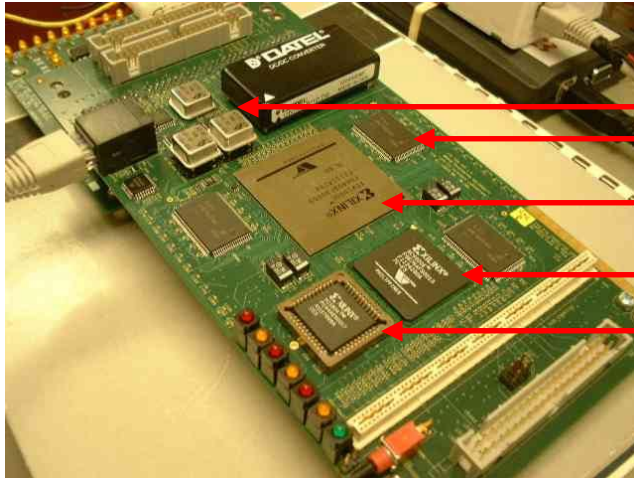


# HARDWARE PLATFORM

## Washington University Gigabit Switch



## FPX Module



- Oscillators
- Static Ram
- RAD (XCV1000E)
- NID (XCV600E)
- PROM

## DESIGN

## Goals

- High Speed Design
- Small FPGA Footprint
- Simple Client Interface
- Support Large Number of Flows
- Utilize existing protocol wrapper framework
- Execute within FPX environment, and systems like it

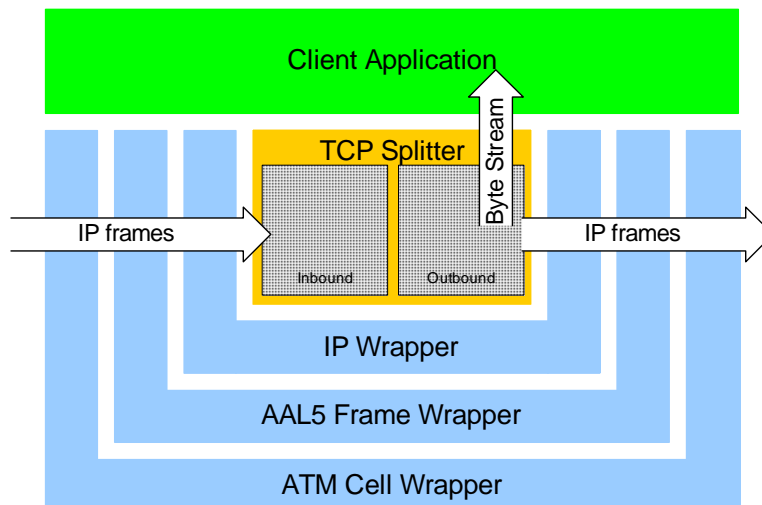
## Challenges

- Frames are dropped on the Internet
- Packets are reordering
- Flow state is needed for large number of flows
- Widescale deployment requires an efficient implementation
- Backbone networks must process data at multi-Gigabit/second rates
- Hardware library should be small

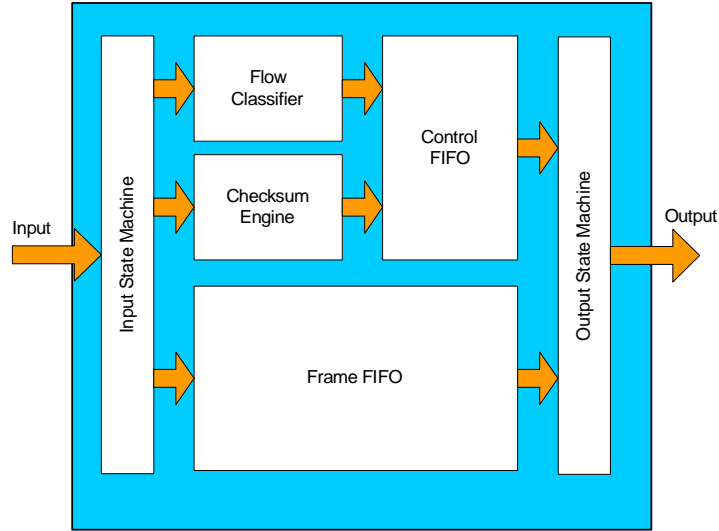
## Assumptions/Limitations

- Though traffic may take diverse paths through a network, all monitored traffic must flow through the node with TCP-Splitter
- Through flows are generally bidirectional, data is processed as a pair of unidirectional flows
- Though data may be sent out of order, data will be forced to be processed in-order

## TCP-Splitter



## TCP Input Module Data Flow

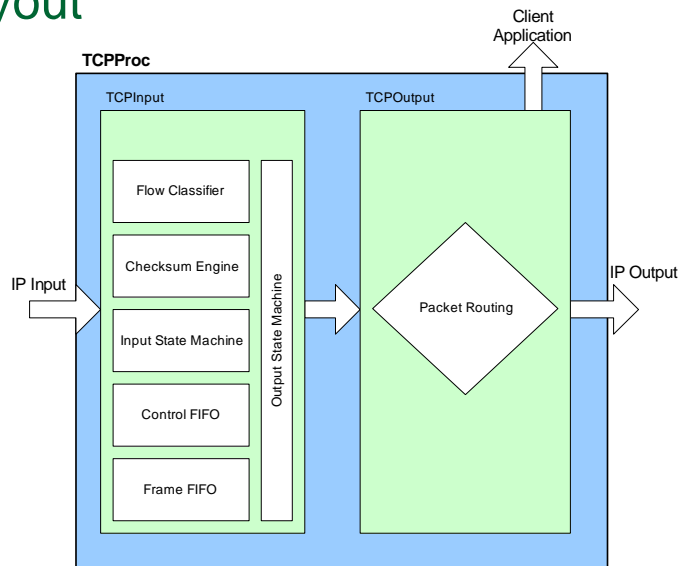


Hot Interconnects 2002



6

## Layout



Hot Interconnects 2002



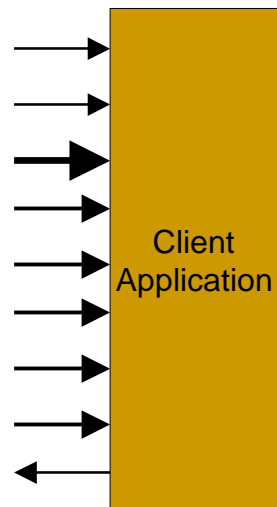
6

## Packet Routing

- Non-TCP packets → IP stack
- Invalid TCP checksum → Drop
- TCP SYN packets → IP stack
- (Seq # < Expected Seq #) → IP stack
- (Seq # > Expected Seq #) → Drop
- Else → Client App AND IP stack

## Simple Client Interface

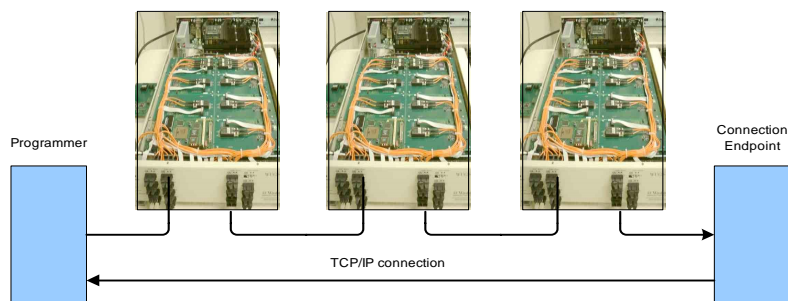
- 1 bit Clock
- 1 bit Reset
- 32 bit Data Word
- 2 bit Data Enable
- 3 bit Start/End of Data Signals
- 2 bit Valid Data Bytes
- N bit Flow Identifier
- 2 bit Start/End of Flow Signals
- 1 bit TCA



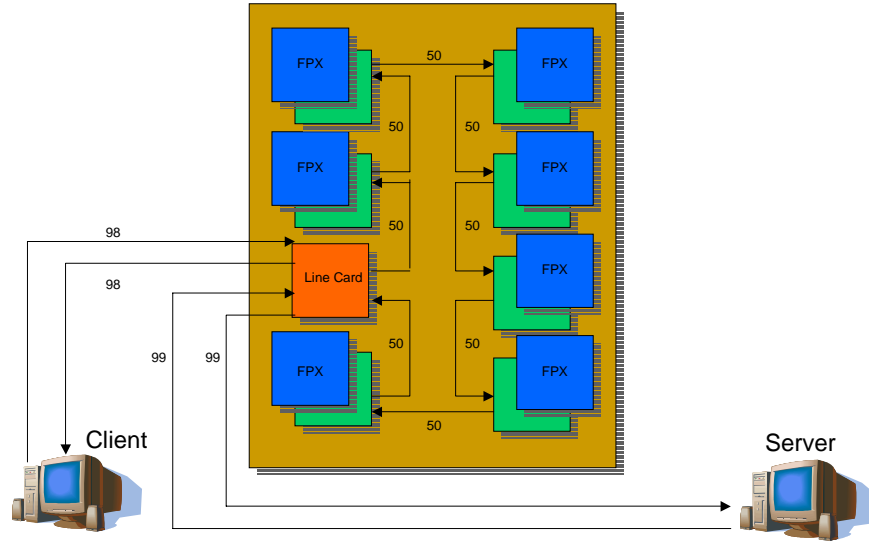
# APPLICATIONS

## Multi-Device Programmer

- Listens to TCP/IP conversation
- Extracts programming information
- Sends programming information to device
- Simultaneously programs multiple devices



## Stacked programmer



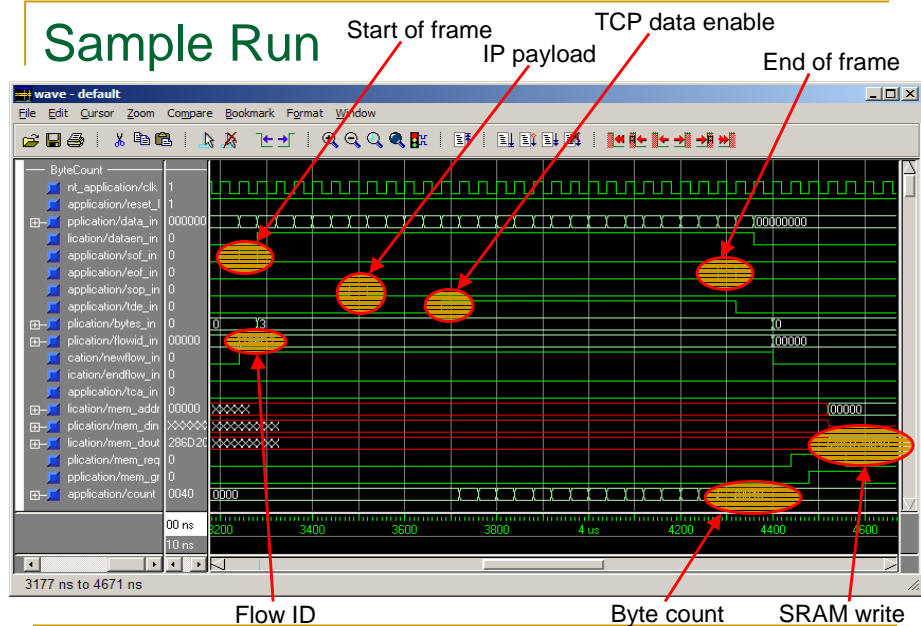
## RESULTS

## Synthesis Results for Xilinx XCV1000E-7

	TCPsplitter	Full Wrappers (Cell + Frame + IP + TCP + Client)
Space/LUTs	617 (2%)	4954 (20%)
Register bits	503 (2%)	4933 (20%)
Processing delay	7 clock cycles *	44-68 clock cycles *

\* Plus length of packet in 32 bit words

## Sample Run



## Conclusion

- ❑ TCP-Splitter developed, simulated, and tested in reconfigurable hardware
- ❑ Monitors 256k TCP/IP simultaneous flows in real-time
  - Synthesizes at 101 MHz
  - Processes data at 3.2 Gigabits/secondHardware design scalable to OC-192 and OC-768
- ❑ Requires only limited resources:
  - TCP-Splitter uses 2% of Xilinx XCV1000E
  - Complete IP stack uses 20% of XCV1000E
- ❑ Eliminates the need for large reassembly buffers

## Acknowledgments

- Harvey Ku
  - ❑ Multi-Device Programmer
  
- Florian Braun and James Moscola
  - ❑ IP Protocol Wrappers