

Versatile FPGA-based Hardware Platform for Gigabit Ethernet Applications

Matei Ciobotaru^{1,2} Mihai Ivanovici^{1,2}
Razvan Beuran^{1,2} Stefan Stancu³



¹ CERN, Geneva, Switzerland

² "Politehnica" University, Bucharest, Romania

³ University of California, Irvine, CA



Outline

- Motivation
- Hardware platform architecture
- Applications and results
 - Network Tester
 - Network Emulator

Motivation

- ❑ Experiments at the *Large Hadron Collider @ CERN*
 - Generate large amounts of data (Gigabytes/s)
 - Processing farms based on Gigabit Ethernet networks (1000's PCs and switches) (for the ATLAS detector)
 - Real-time Data Acquisition systems → reliable and predictable network operation

- ❑ The need for tools to evaluate the components of the Data Acquisition networks
 - A network tester for switch performance evaluation
 - A network emulator for application performance assessment
 - *All running at Gigabit speed!*

Solution

- ❑ Custom tool
 - Best fits our needs

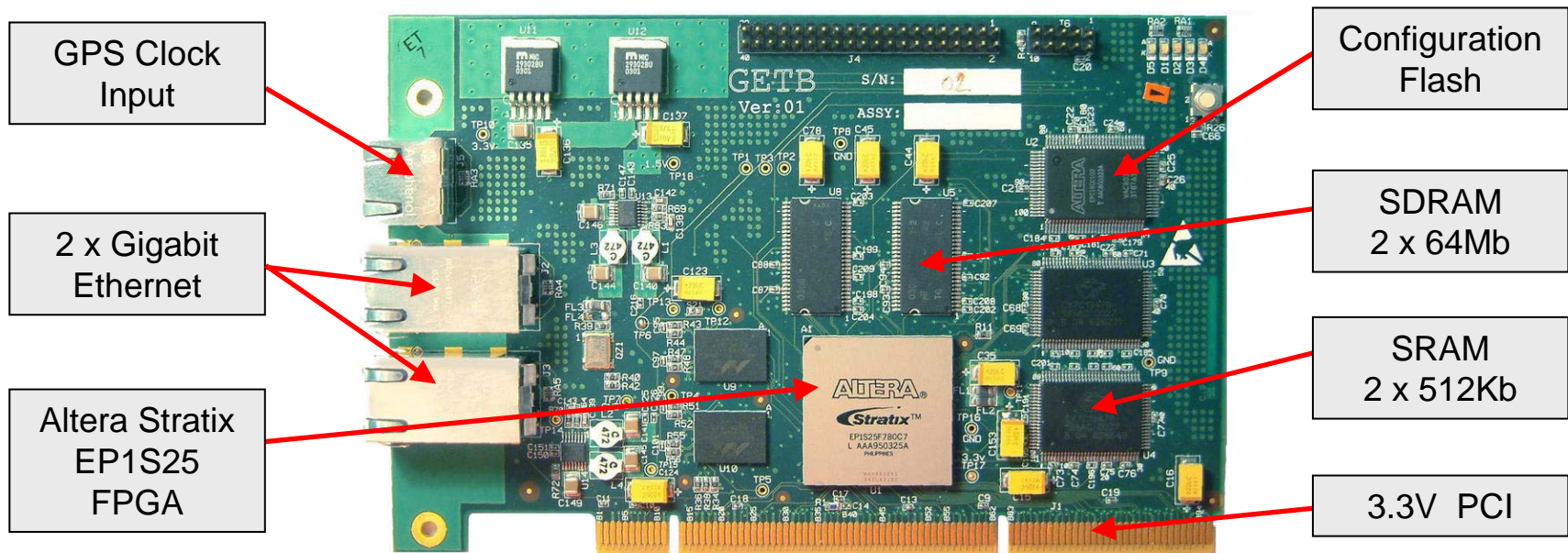
- ❑ Hardware implementation
 - High precision, fast operation

- ❑ FPGA-based platform
 - FPGA → programmable logic device
 - Multiple parallel processes

- ❑ PCI → Standard control interface
 - High density on commodity hardware (PC)

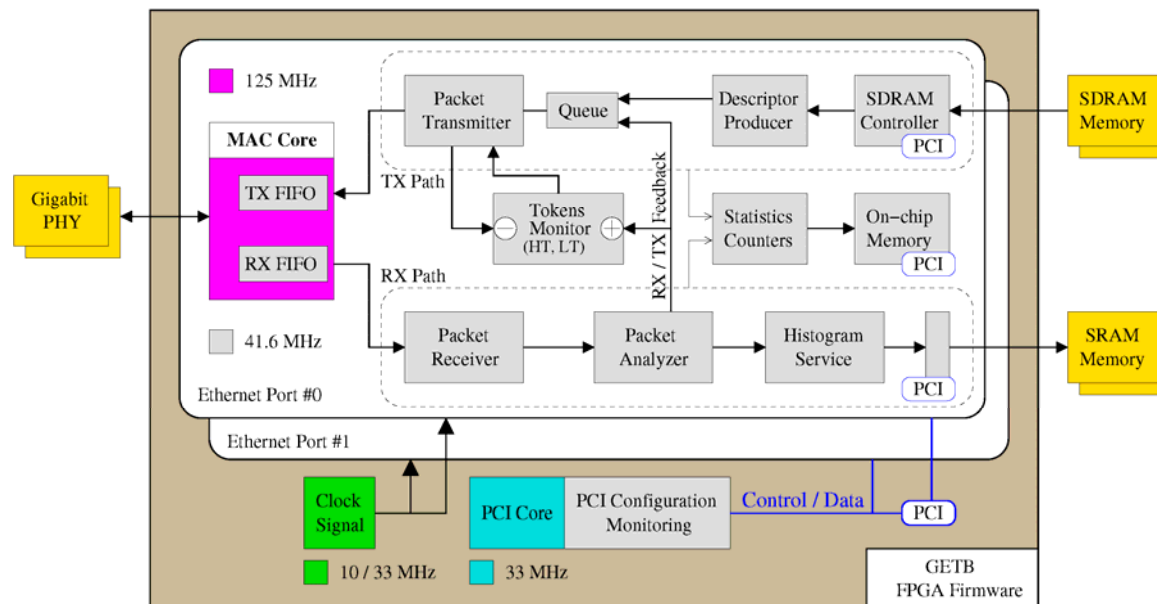
Hardware Platform – The GETB

- Custom-design PCI card
 - Altera Stratix FPGA
 - 2 Gigabit Ethernet ports
- GETB* = Gigabit Ethernet Test Board



FPGA Firmware

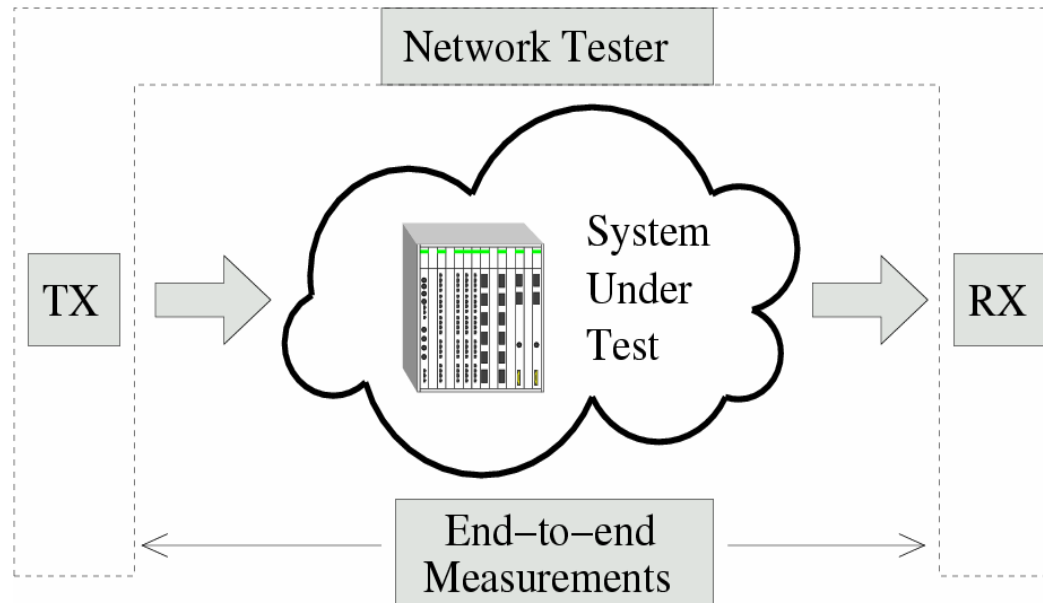
- Handel-C programming language
 - High-level C-like hardware description language – www.celoxica.com
 - Easy-to-write communicating parallel processes
 - Rapid development and translation into hardware
- Multiple clock domains (33–125MHz)
- Message passing architecture
 - Reusability & independent module design
 - Asynchronous concurrent processes
 - Low-level functionality shared by all projects
- Commercial IP cores (IP = Intellectual Property)
 - Gigabit Ethernet MAC , PCI Controller



Network Tester – Context

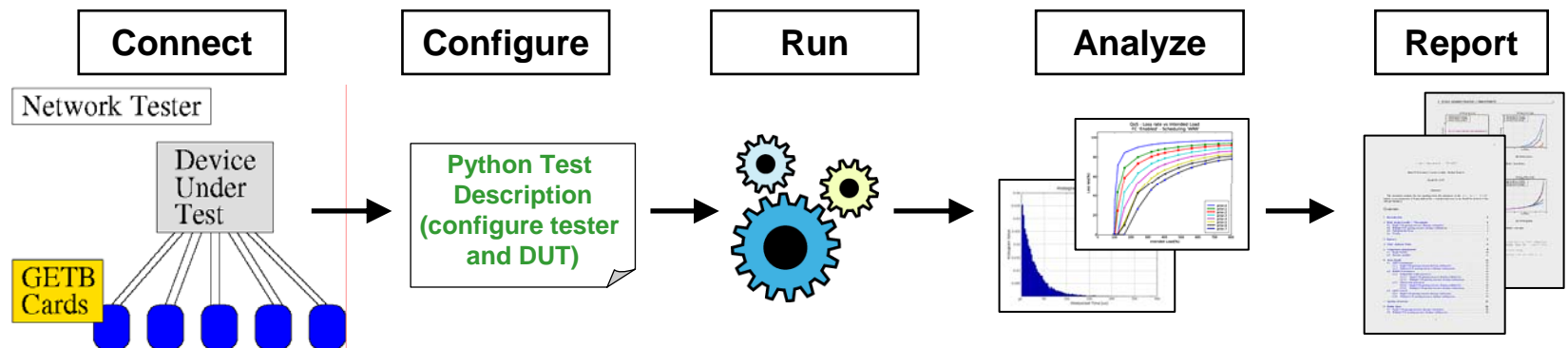
Study the performance and functionality of a given System Under Test (SUT)

- Inject artificial traffic into the SUT
 - Customizable traffic patterns
- Measure network parameters
 - Packet loss, one-way delay, throughput
- Existing commercial testers
 - Built for standard RFC benchmarks
 - Limited set of traffic patterns
 - Expensive



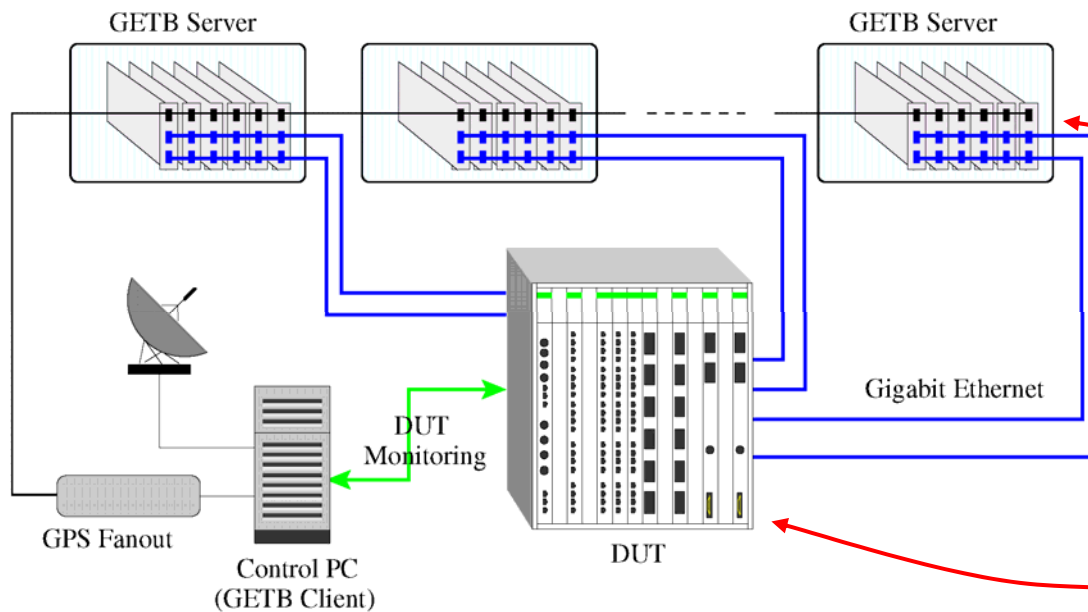
Network Tester – Features

- ❑ Traffic Generator and Measurement System
 - 1 Gigabit/s for all packet sizes (bi-directionally) – Raw Ethernet and IP
 - 1 GETB card → 2 Tester ports
- ❑ User-defined traffic patterns
 - Deterministic (packet descriptors)
 - Client-Server (request-response)
- ❑ Real-time measurements
 - Throughput, packet loss, latency and inter-packet time (IPG)
 - Histograms for latency, IPG, packet size
- ❑ Control system
 - Based on the Python scripting language
 - Distributed → 64 cards in 15 hosts (128 ports in total)

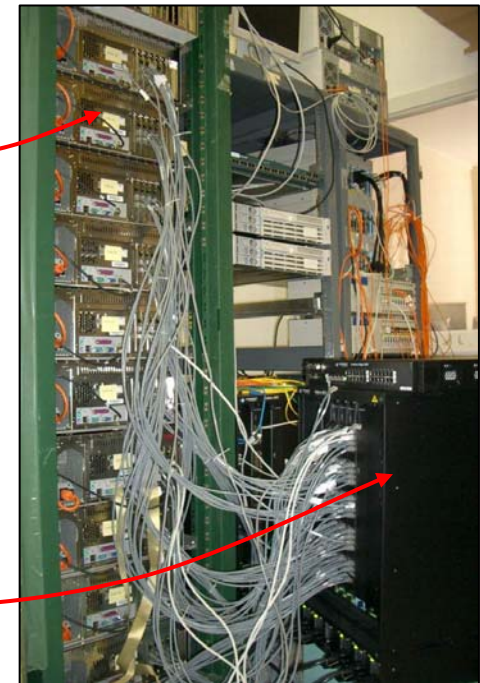


Network Tester – System Setup

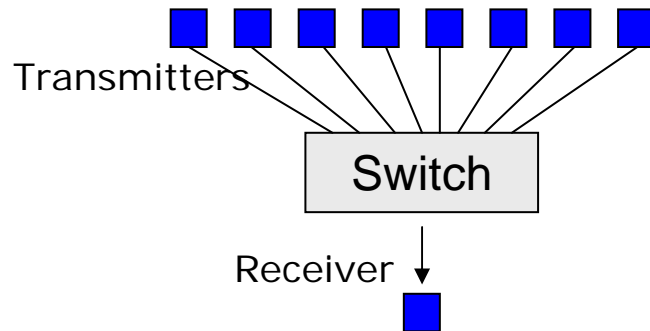
Block Diagram



Photo

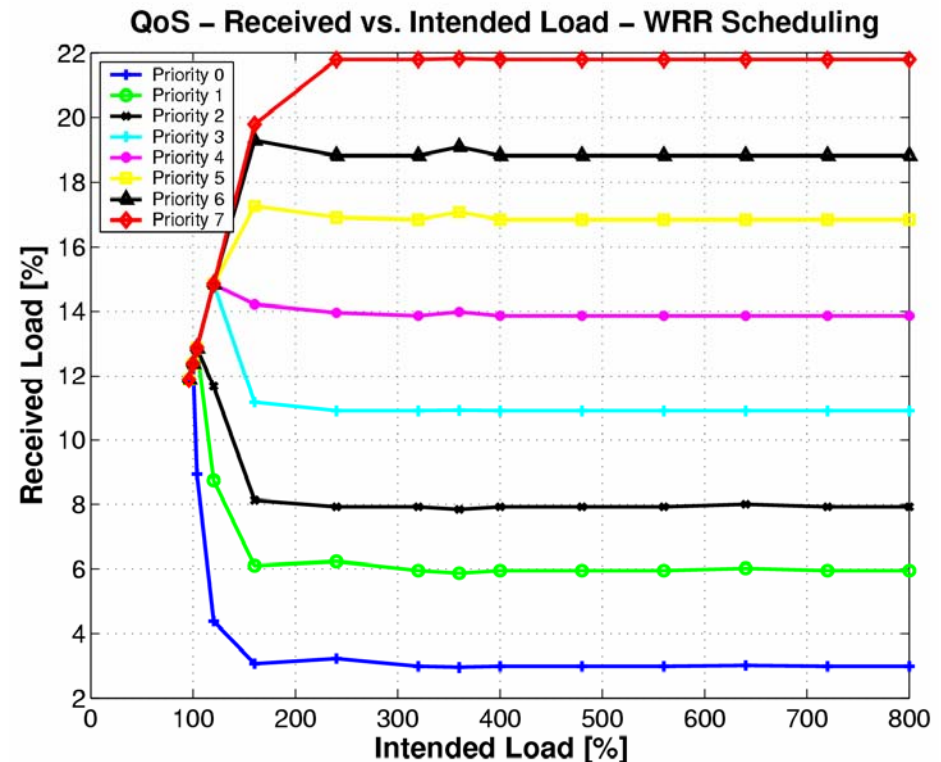


Network Tester – Results



- We tested QoS mechanisms, in this case Weighted Round Robin (WRR) scheduling
- Multiple transmitters (8) sent traffic to the same destination, each one with a different priority
- We measured the received rate for each transmitted stream

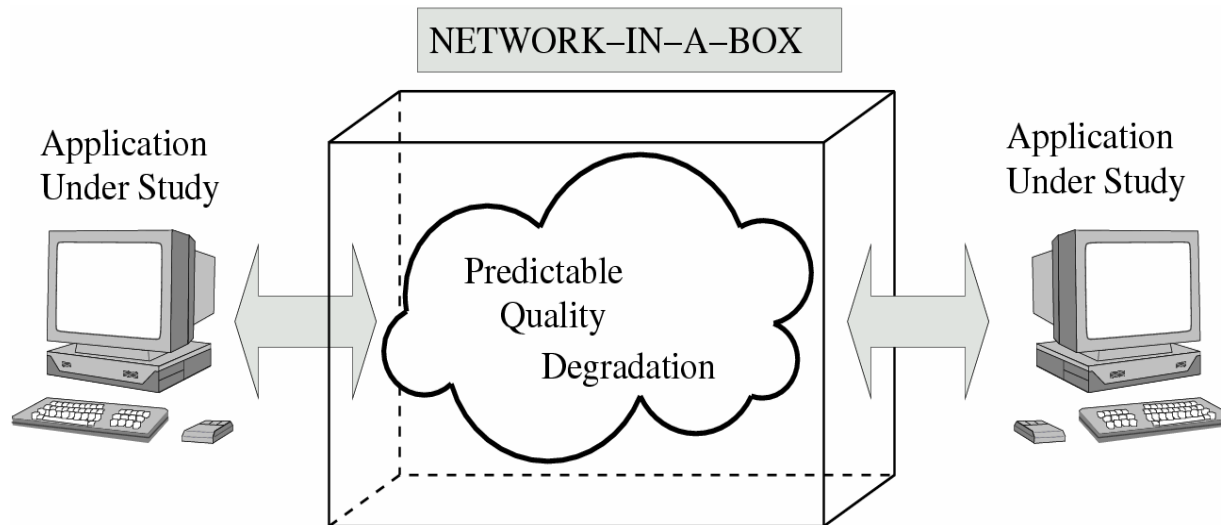
Priority	0	1	2	3	4	5	6	7
Weight [%]	3	6	8	11	14	17	19	22



Network Emulator – Context

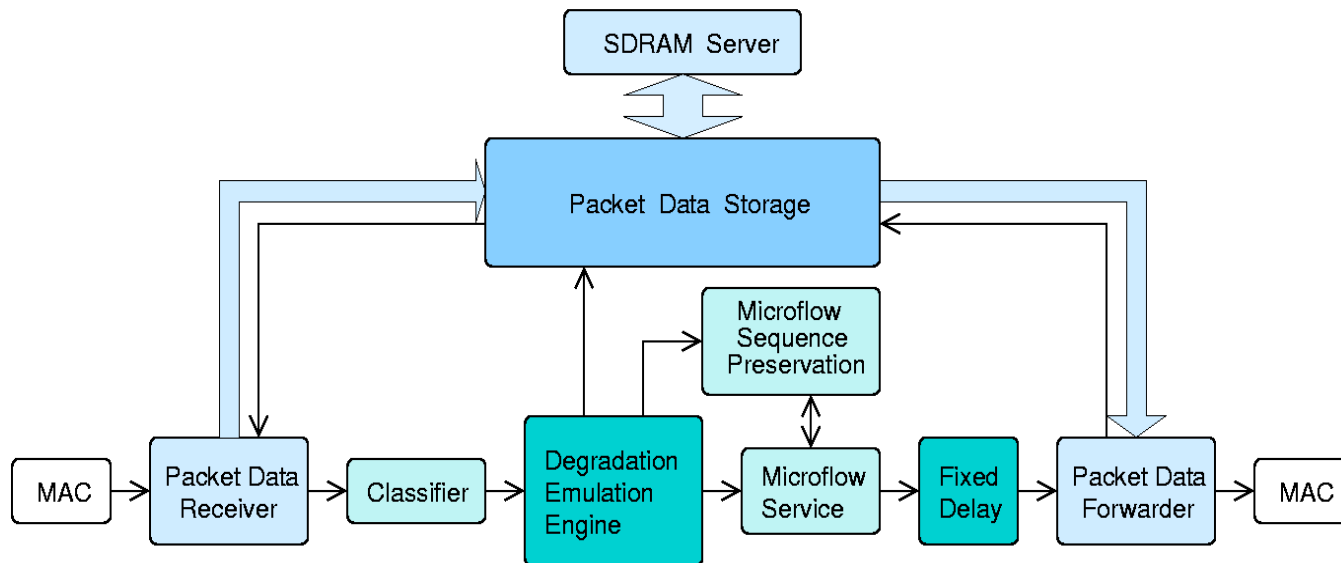
Study the behaviour of an application under different network conditions

- Network-in-a-box
 - Wide range of network conditions in a laboratory setup
 - Hybrid technique between computer simulation and tests in real networks
- Uses of a network emulator
 - Application assessment
 - Protocol development
- Short debugging cycles

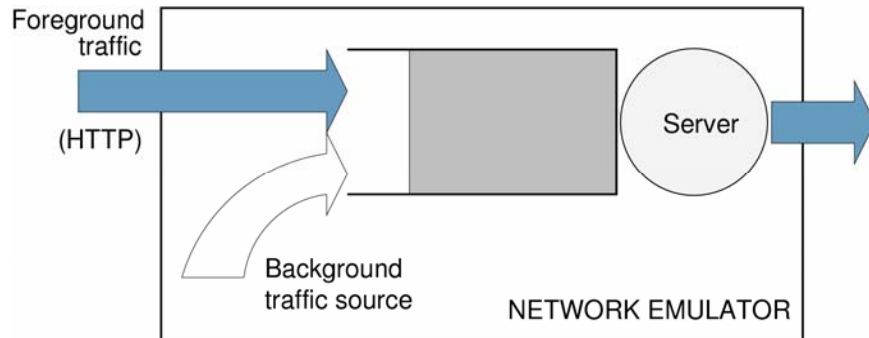


Network Emulator – Features

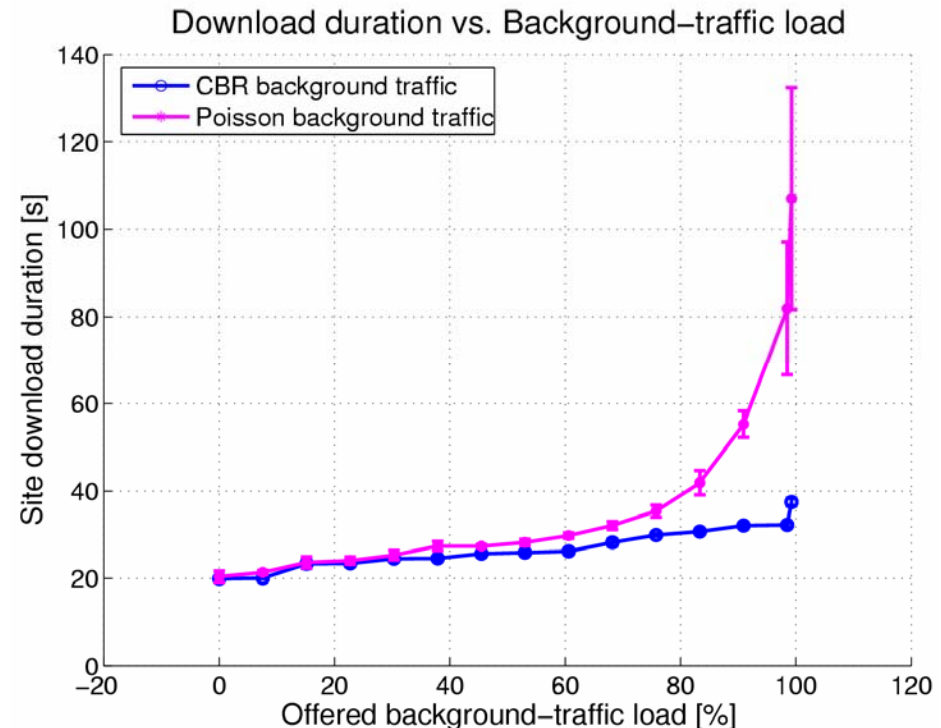
- User-defined network conditions
 - Correlated loss and variable delay
- Traffic differentiation
 - Flow classification (based on IP header)
 - Multiple queues
- Quality degradation through background traffic generation
 - Constant Bit Rate and Poisson distributions
- Rate limitation



Network Emulator – Results

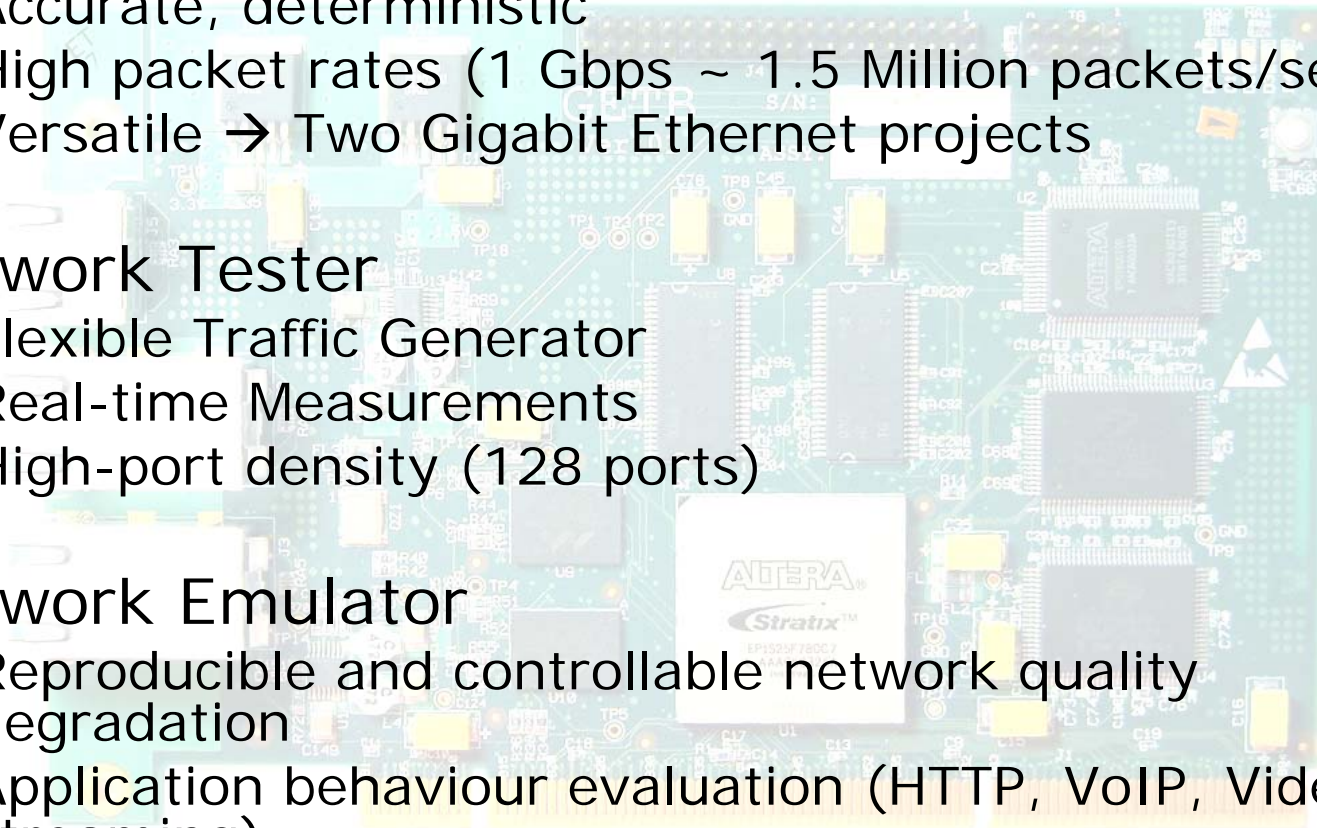


- Quality degradation through background traffic generation
- Tested application: Internet "browsing" (HTTP transfers)
- Site download duration vs. offered background traffic load
- Other applications: VoIP, Video streaming, etc.



Summary

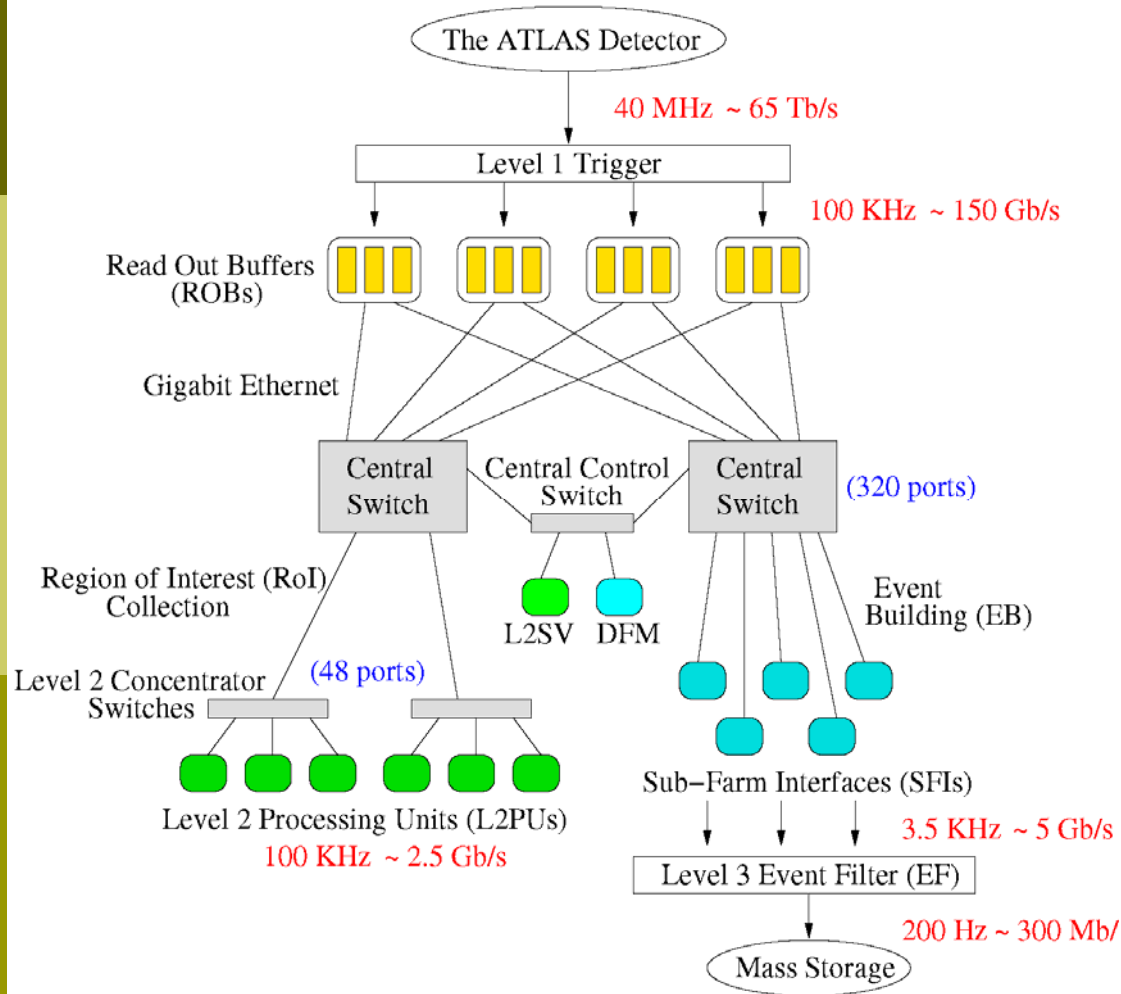
- ❑ Custom-design FPGA-based platform
 - Accurate, deterministic
 - High packet rates (1 Gbps ~ 1.5 Million packets/second)
 - Versatile → Two Gigabit Ethernet projects
- ❑ Network Tester
 - Flexible Traffic Generator
 - Real-time Measurements
 - High-port density (128 ports)
- ❑ Network Emulator
 - Reproducible and controllable network quality degradation
 - Application behaviour evaluation (HTTP, VoIP, Video streaming)



End of presentation

▣ Backup slides

ATLAS TDAQ Network



■ ATLAS TDAQ Network

- Gigabit Ethernet
- Layer 2 only
 - Only switches
 - ~ 1000 end-nodes
- Sustained rates
 - 5 Gb/s on average

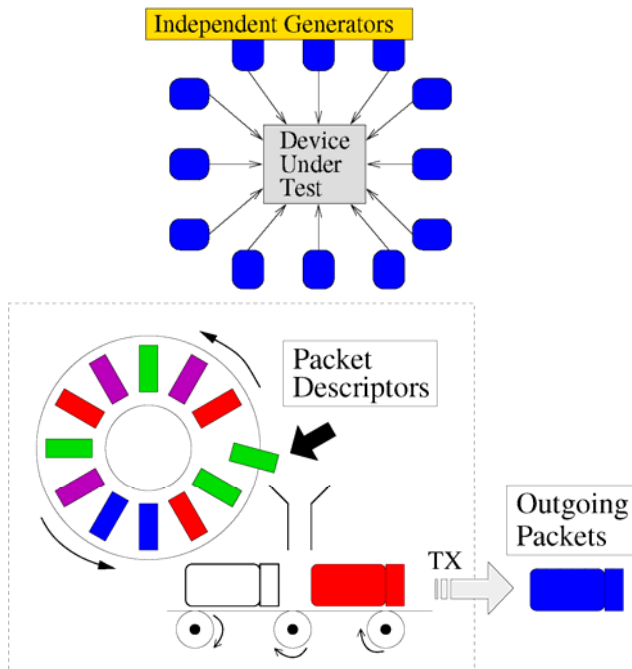
■ Requirements

- Minimal packet loss
- Minimal latency
- High Performance Switches
- Try before you buy

Network Tester – Packet Generation

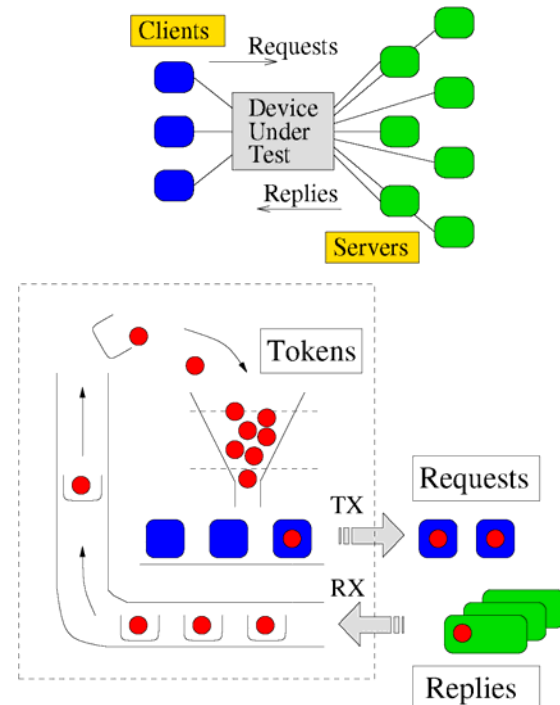
□ Independent Generators

- Packet descriptors
 - Modify MAC addresses, packet size, inter-packet time, IP headers, etc.
 - Wide range of traffic patterns
- Ethernet and IP packets, VLANs

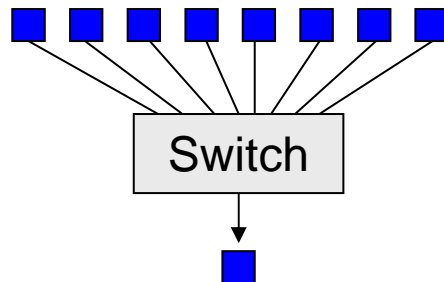


□ Client Server

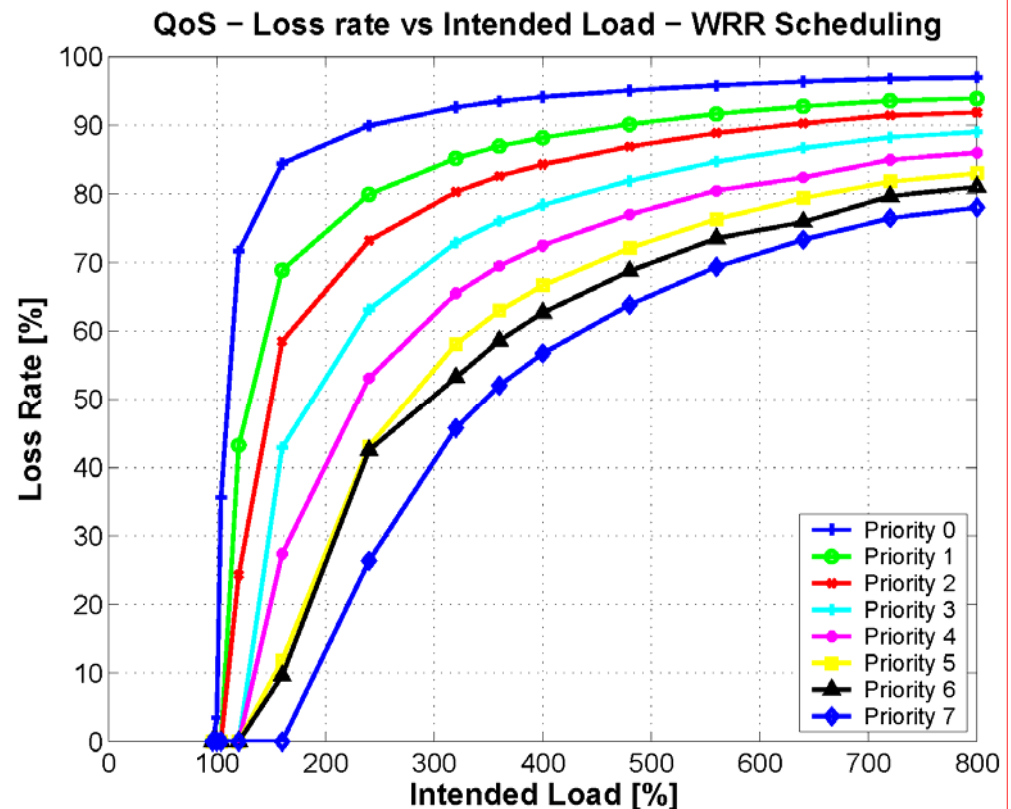
- ATLAS-like request-reply traffic
- Clients send data request packets to servers; Servers send back replies
- Clients use a token-based system to control network load



Quality of Service (Variant)

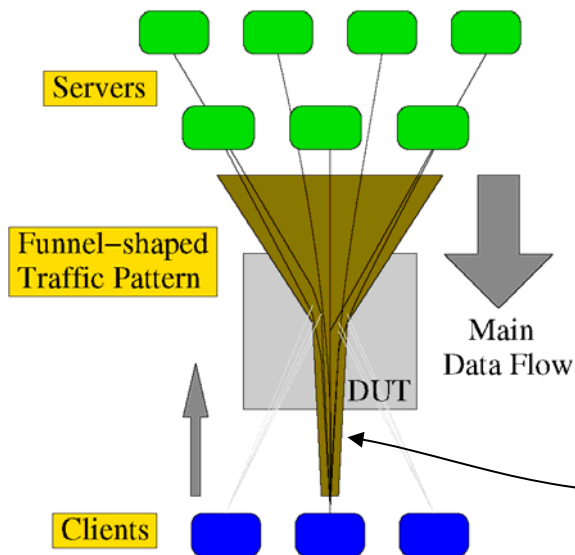


- 8 transmitters, each sending traffic with a different priority
- The switch uses Weighted Round Robin Scheduling
- Result obtained using data collected by one GETB tester port

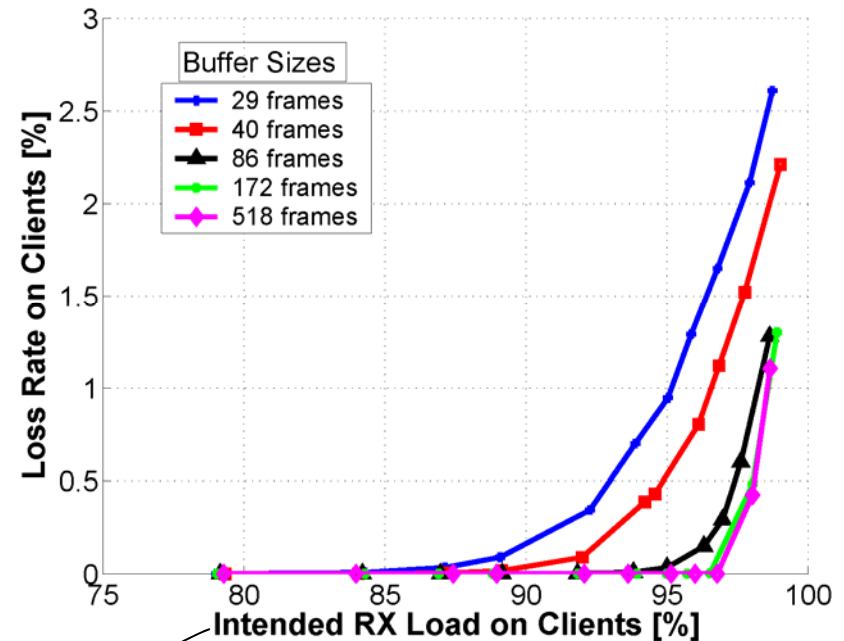


Network Tester – ATLAS Traffic

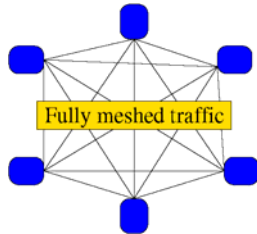
- Performance for ATLAS Traffic (funnel-shaped traffic pattern)
 - Depends on the number of buffers
- Determine the maximum load the clients can receive without loss



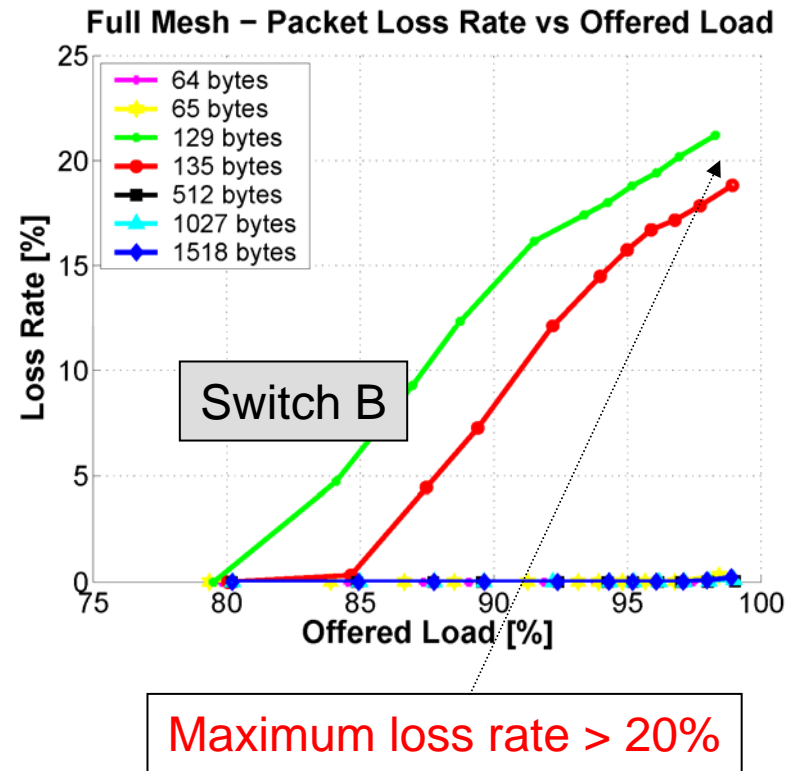
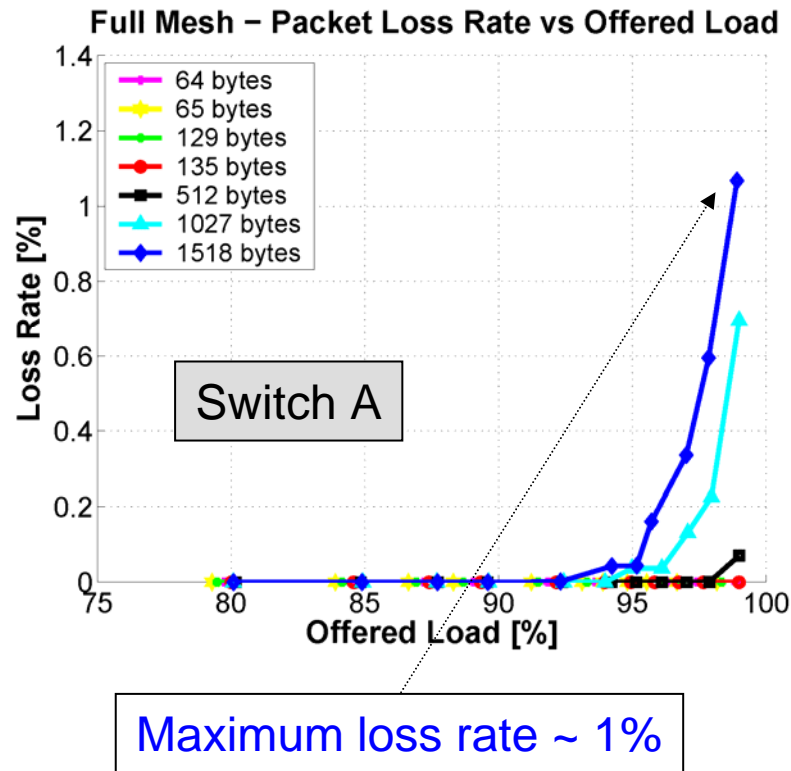
ATLAS Traffic – Loss Rate vs Intended RX Load on Clients



Network Tester – Full-Mesh Traffic

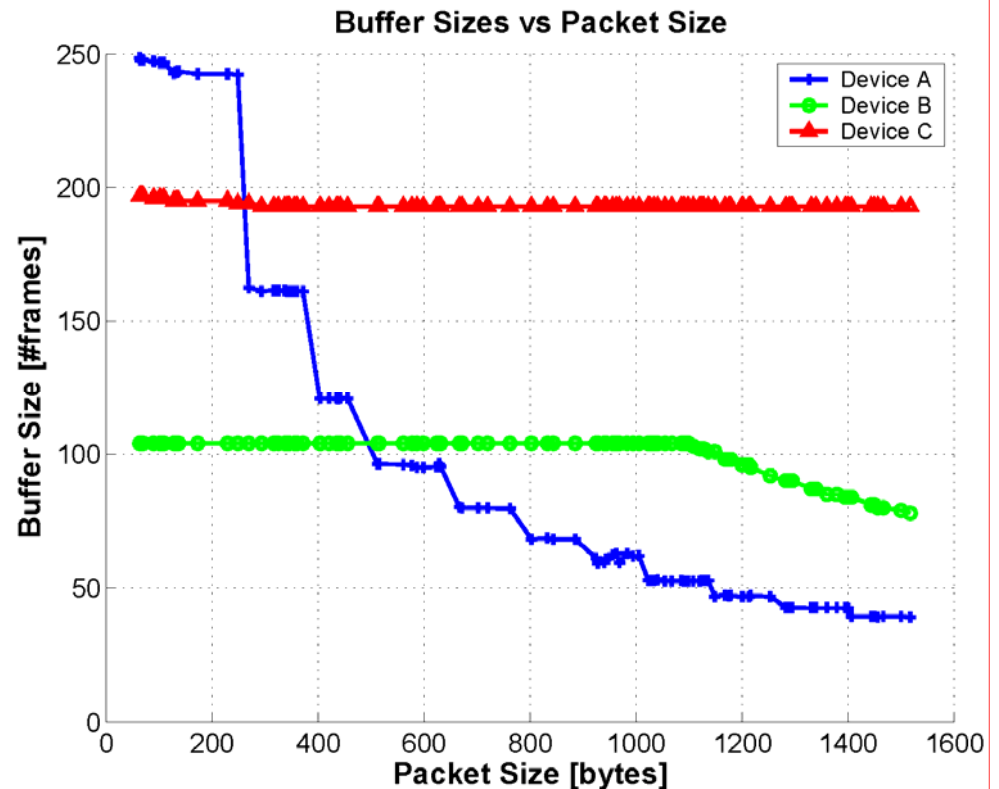


■ Fully-meshed traffic performance



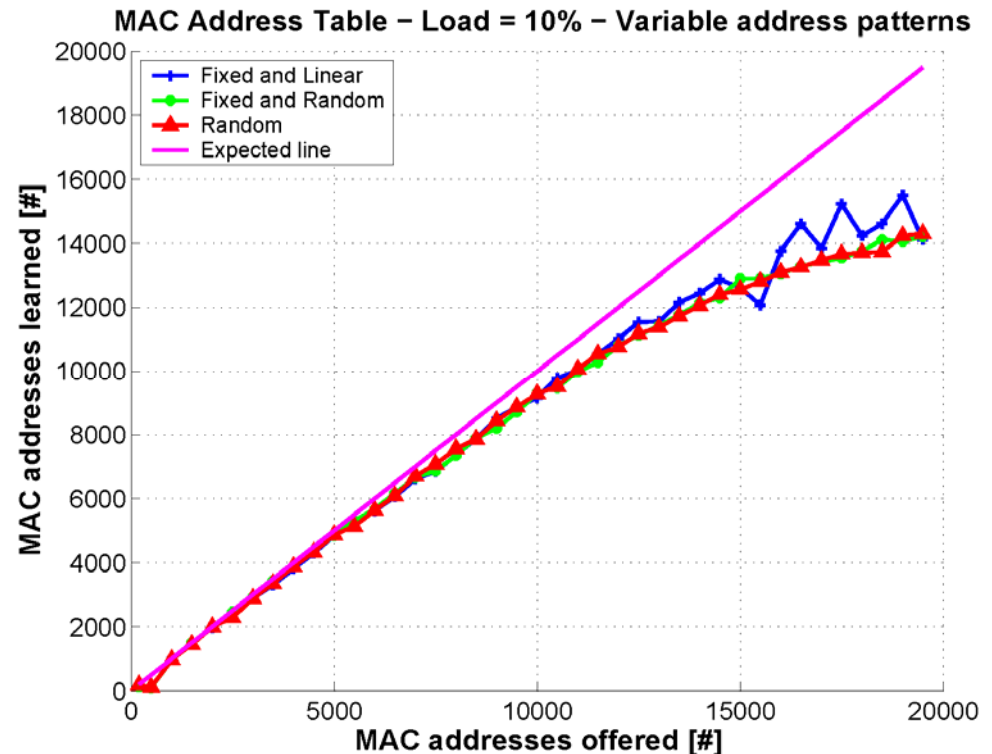
Buffer Sizes

- Custom method for measuring buffer sizes
- 3 devices → 3 different memory management techniques
- Size of buffers important for the ATLAS traffic pattern



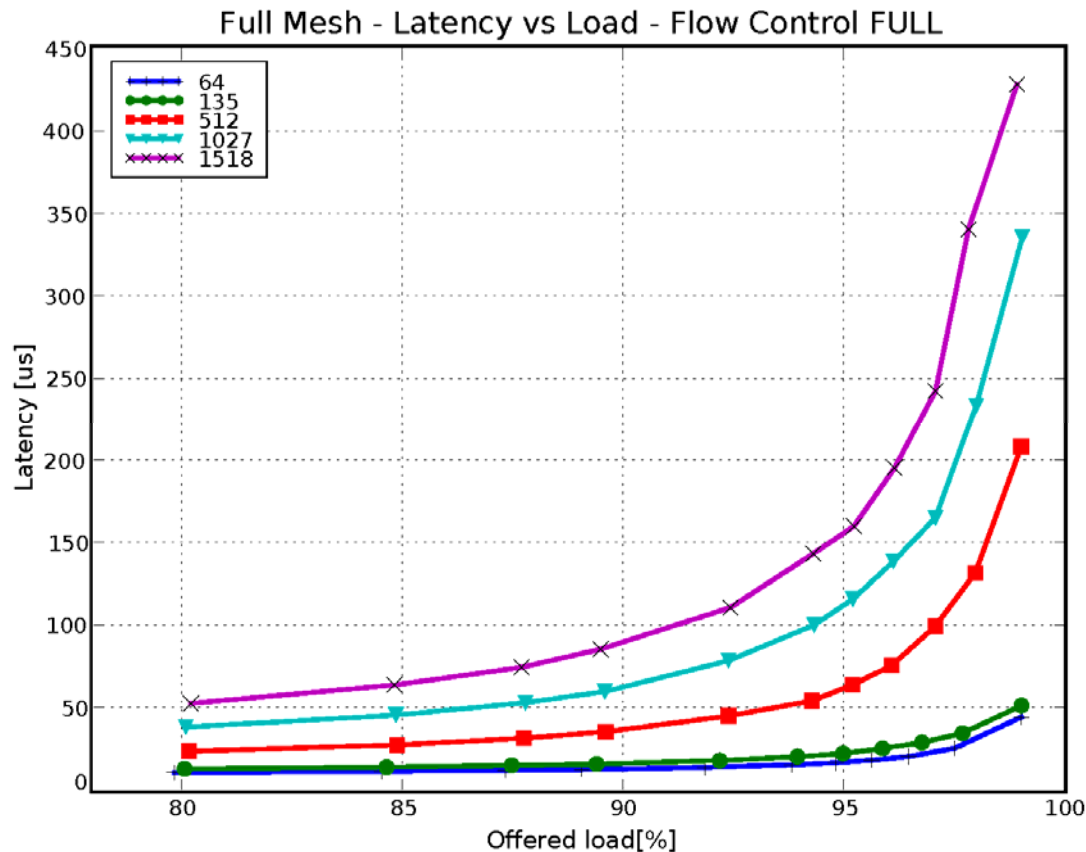
MAC Address Table

- Switch advertises 16000 entries in the MAC table
- Measurement reveals problems for more than 5000 addresses



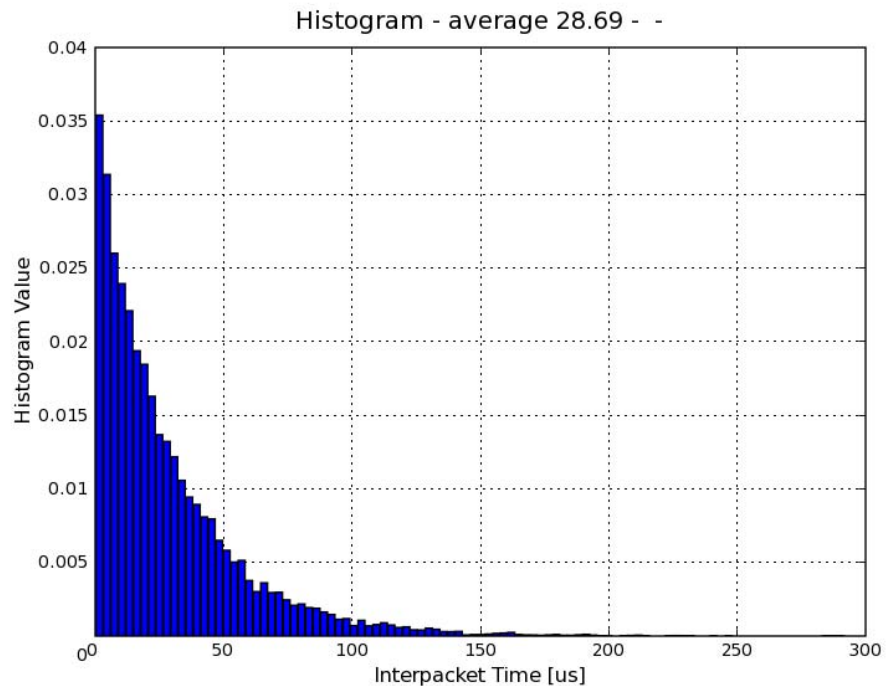
Latency Measurement

□ Latency for a fully-meshed traffic pattern



Sample Histogram

- Example of histogram of Inter-packet Time for a Negative Exponential distribution
- Features
 - Packets can be classified according to source or VLAN ID
 - Latency, IPG and packet size can be histogrammed
 - User defined resolution and histogram window (start offset, length)



Handel-C – www.celoxica.com

- ❑ Hardware description language
 - Like VHDL, but with syntax similar to C
 - The result of the compilation is the description of an electrical circuit
- ❑ Contains built-in parallel constructs
- ❑ Synchronization primitives: channels, semaphores
- ❑ Special features
 - Arbitrary widths on variables
 - Enhanced bit manipulation operators
- ❑ Simple timing model
 - Each assignment is one clock cycle
- ❑ Support for hardware constructs
 - Multiple clock domains, on-chip memories, external interfaces

Sequential Block

```
// 3 Clock Cycles
{
    a=1;
    b=2;
    c=3;
}
```

Parallel Block

```
// 1 Clock Cycle
par{
    a=1;
    b=2;
    c=3;
}
```

Python Language – www.python.org

□ Features

- Object oriented
- Easy to learn, read, use
- Extremely portable
- Extensible (new modules)

□ What is it used for?

- Rapid prototyping
- Scientific applications
- Extension language
- Web programming

```
class Stack:
    "A well-known data structure"
    def __init__(self):    #
        constructor
        self.items = []
    def push(self, x):
        self.items.append(x)
    def pop(self):
        x = self.items[-1]
        del self.items[-1]
        return x
    def empty(self):
        return len(self.items) == 0
```