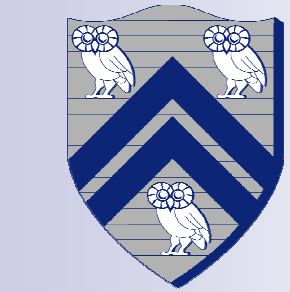
RiceNIC

A Reconfigurable Network Interface for Experimental Research and Education



Jeffrey Shafer
Scott Rixner

Introduction

- Networking is critical to modern computer systems
- Role of the network interface is changing
  - New communication mechanisms with host *(TCP offloading, RDMA…)*
  - New responsibilities *(Data caching, encryption, …)*
- Simulation is not sufficient to explore new architectures
- Research into network systems demands experimental prototypes
- RiceNIC was built as a fully-capable prototyping platform
  - In active use for experimental research at several institutions *(Rice, EPFL, UF, USF, HP Labs)*

Outline

- Experimental Research into Network Server Architectures
  - Simulation with whole-system simulators
    - Challenges when applied to network systems
  - Prototyping with software programmable NICs
    - Better approach, but limited by existing tools
- Introduce RiceNIC
  - Fully capable prototyping platform + practical to build
- Experimental Research with RiceNIC
- Conclusions

Simulation Challenges

- Network server architecture performance depends on interactions between…
  - System components (processor, memory, I/O interconnects)
  - User applications, OS and device drivers
  - Multiple computing systems separated by a network
    - Network (Internet) can be inherently chaotic
- All of these elements interact asynchronously

Simulator must model all elements with high fidelity

Adaptive Algorithms in Simulators

- High fidelity simulation critical when evaluating adaptive algorithms
  - Software control flow adjusts based on underlying system performance
- TCP is a widely used adaptive algorithm
- Small inaccuracies can progressively distort TCP performance as feedback loop develops
  - Example: Buffer drains slightly too slow, eventually fills and drops packets, TCP throttles bandwidth to compensate

TCP Simulation

- TCP (and other adaptive algorithms) react poorly to standard simulation techniques
  - Warm-up stage - functional simulator
  - Data collection stage - complete simulator
- Subtle pitfall
  - Accidentally measuring TCP slow-start performance during data collection
- Solution
  - Continuously execute simulator until TCP reaches steady-state, then measure performance
  - Minimum slow-start time is 150 million cycles (with no network delay)
  - Realistic slow-start time with network delay – order of magnitude higher

Simulation is Expensive

- One recent research project
  - Shortest tests are 120 seconds long (in real-time)
    - Minimum for accurate data collection at TCP steady-state
  - Hundreds of tests are run at different settings
- How long would simulation take?
  - 5 orders of magnitude slower (or more!) for cycle accurate simulation – 138 days per test
- How many compute machines are needed?

We built a prototype so we could run in real-time, not simulator-time

Prototyping Advantages

- Performance
  - Prototype can run in real-time
- Expense
  - Cheaper to build a few prototypes than buy a compute cluster for simulation
- Accuracy
  - A simulator models the entire computer (and has inherent assumptions and approximations throughout the system)
  - A prototype models the specific device being investigated (i.e. NIC) but uses a real system otherwise
    - Experimental error is constrained to the device being studied

Prototyping Challenges

- Existing prototyping platforms insufficient
- Can purchase software-programmable NICs for Ethernet and specialized interconnects (Myrinet, Infiniband)
- Hardware architecture (including control systems) is fixed
  - May constrain new software design
  - Software emulation of a new hardware architecture might be very compute-intensive
- Intellectual Property issues may limit range of available customizations

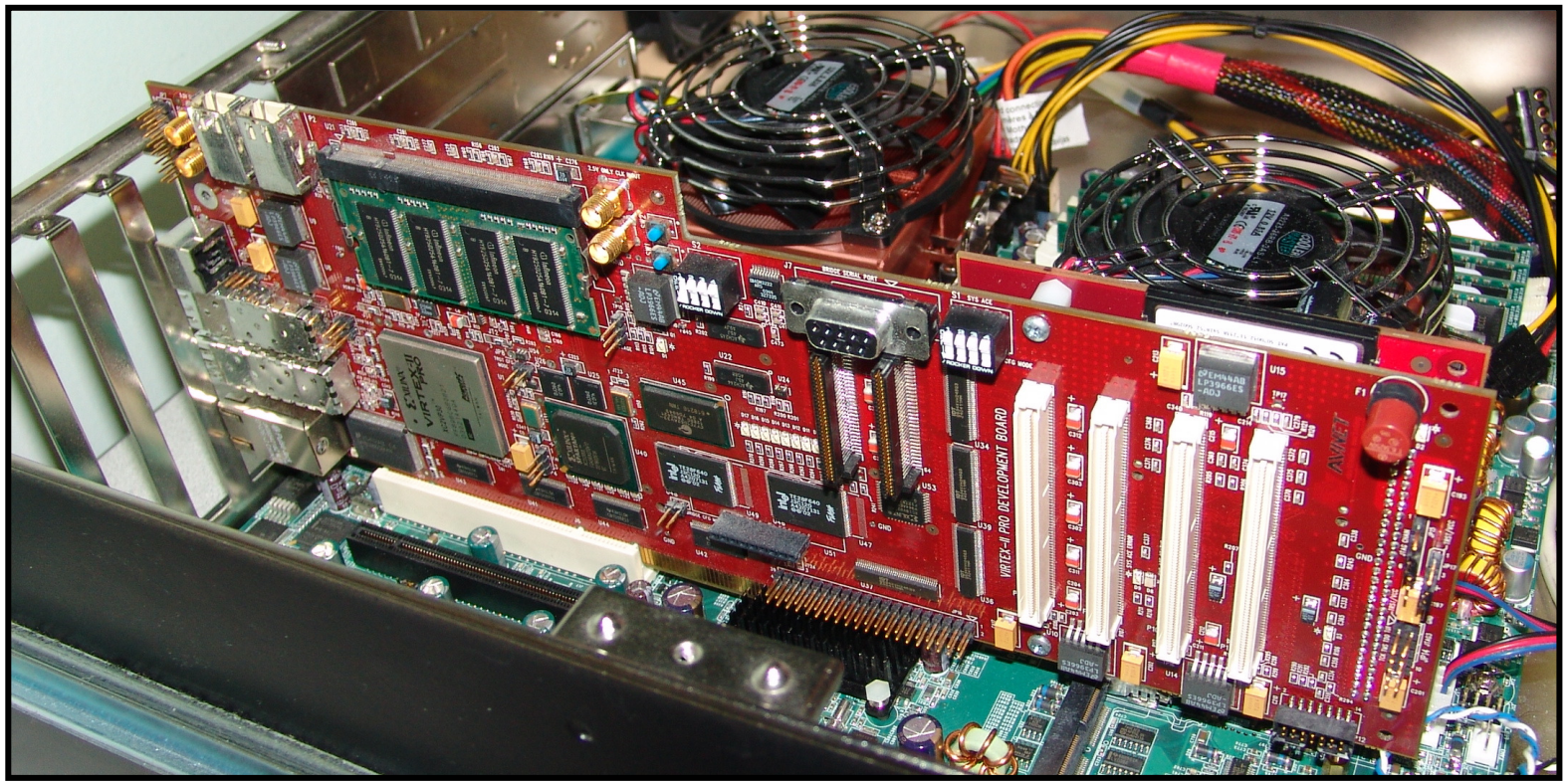
Past Experience with Prototyping

- Projects implemented on software-programmable NIC
  - TCP connection offloading
    - Move TCP stack to NIC for key connections
    - Implementation limited by processor and available memory (Throughput limited to 100Mbps on a Tigon2 gigabit NIC)
  - NIC data caching
    - Save frequent packet data on NIC
    - Cache size severely limited by Tigon2 memory capacity

These prior experiences with experimental prototyping motivated the creation of RiceNIC

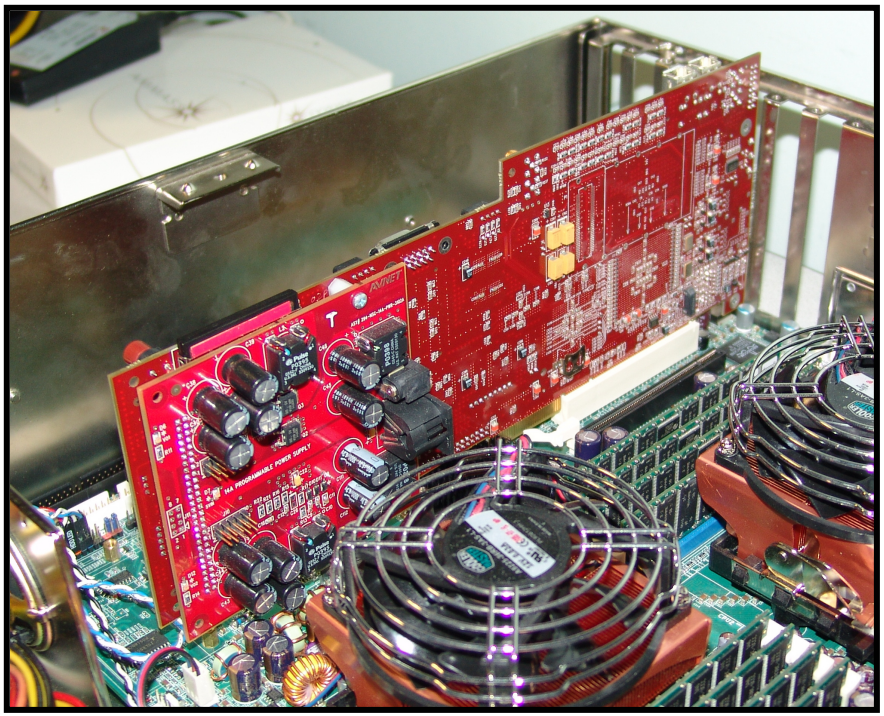
RiceNIC Overview

Gigabit Ethernet Network Interface Card

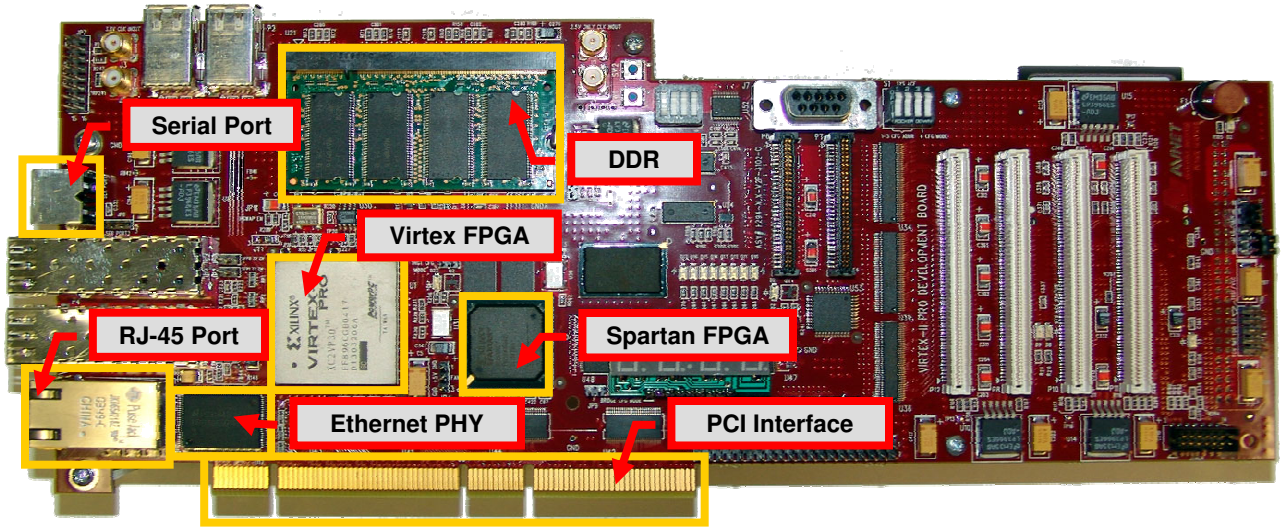


RiceNIC Overview

- Reconfigurable
  - FPGAs implement hardware architecture
- Programmable
  - Embedded processors provide high-level NIC control
- Performs at Ethernet line rate
- Reference design is freely available
- Targeted at research and education applications



RiceNIC Board



- 2 FPGAs
- 2 PowerPC processors (300 MHz)
- Serial Port
- On-NIC Memory (256MB DDR)
- Gigabit Ethernet PHY
- 64-bit / 66 MHz PCI bus

Debugging Tools

- Software Debugging
  - Serial Console (RS-232 port)
    - Command line interface to PowerPC processors
    - Runtime debugging / configuration changes / bootstrapping
  - Firmware Profiler
    - Timer based statistical profiler (similar to Oprofile)
    - Exports results via serial console
- Hardware Debugging
  - Logic analyzer (Xilinx Chipscope) on FPGAs

Debugging Tools are Essential for Experimentation

RiceNIC Design Timeline

- Development time
  - Hardware (FPGA) design - 1 year / 1 graduate student
  - Software (firmware / driver) design - 1 month / 1 graduate student
- Development stages
  - Learning Xilinx FPGA tools – 1 month
  - Design / Implementation – 9 months
  - Testing – 3 months
- Prototype can be re-used by other institutions just like a simulator
  - Current users include Rice, EPFL, UF, USF, and HP labs

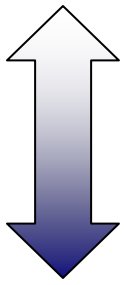
Constructing an experimental prototype is practical!

RiceNIC Applications

- RiceNIC supports a wide range of experimental research projects

Scope of Changes

Component Level



System Level

- Modify firmware
  (Network Address Translation on NIC)
- Modify FPGA
  (Low power networking with adaptive MAC)
- Modify firmware, FPGA, device driver, OS
  (Networking in virtual machines)
- Modify systems beyond the NIC
  (Reconfigurable networking lab)

Network Address Translation

- Scope focused on NIC firmware
- Added NAT module to RiceNIC
- RiceNIC still runs at full Ethernet speeds
- Implementation took 1 day
  - Debugging of NAT module greatly assisted by RiceNIC serial console which printed packet traces

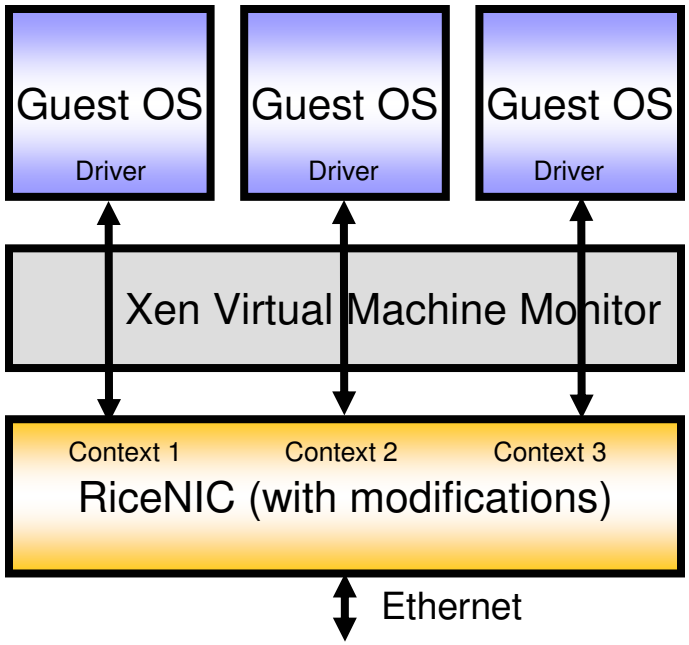
Low Power MAC

- Energy Efficient Internet Project †
  (at USF and UF)
- Scope focused on single component – MAC
- Modified FPGA hardware
- Replaced MAC core on RiceNIC FPGA with a custom low-power variant that supports adaptive link rates
  - This research could not be done on any software-programmable NIC!

† The Energy Efficient Internet Project: http://www.csee.usf.edu/~christen/energy/main.html

Networking in Virtual Machines †

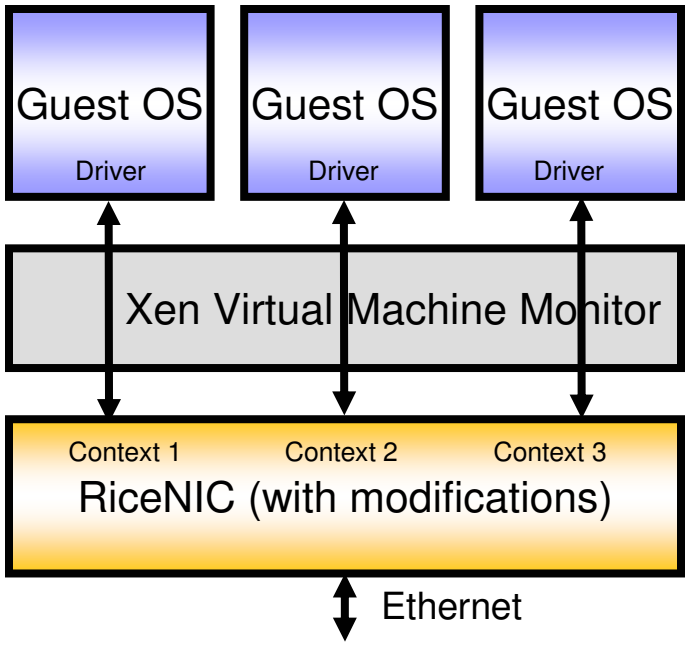
- Each guest OS can talk directly to single shared NIC
  - Separate interfaces for concurrent guests
- Experimental research project with many modifications
  - FPGA provides isolated contexts in memory + event notification
  - Firmware provides packet multiplexing / demultiplexing
  - Xen / OS modifications
  - Used 1 PowerPC processor and 12MB of RAM



† P. Willmann, J. Shafer, et.al, Concurrent Direct Network Access for Virtual Machine Monitors, *The International Symposium on High Performance Computer Architecture (HPCA'07),* Phoenix, AZ, (Feb 2007)

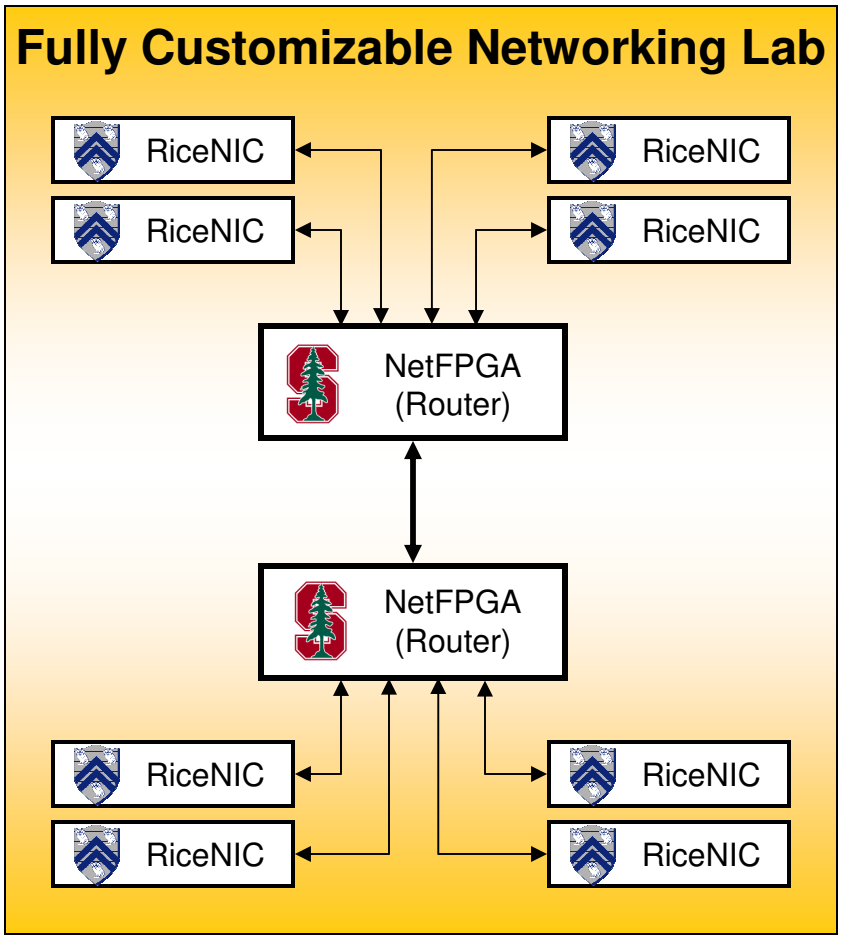
Virtual Machines Networking

- Prototyping (with RiceNIC) critical to project success
- Could not use software programmable NIC
  - Needed to change hardware architecture
  - Software emulation too slow
- RiceNIC prototype ran at full speed
- Used RiceNIC to experimentally determine key architectural features
  - Minimum on-NIC packet buffer size per virtual machine
- Could not obtain equivalent results via simulation in a timely fashion



Reconfigurable Network Ecosystem

- Scope of experimental prototyping extends beyond single system
- Create a reconfigurable and programmable network ecosystem
  - All endpoints and routing fabric are modifiable
- Use RiceNIC as interface with host computers
- Use NetFPGA *(from Stanford)* as routers / switches
- Experimentally explore new networking architectures



RiceNIC in Education

- Simple class project *(undergraduate)*
  - NAT on NIC implementation (1 day)
- Intermediate project *(undergraduate)*
  - Perform deep packet inspection in FPGA hardware
- Advanced project *(graduate)*
  - Use fully customizable networking lab (with NICs, routers and switches) to explore new network protocols and architectures
- Debugging tools used for research (serial console, profiler, logic analyzer) are equally valuable in the classroom for use by non-experts

Obtaining RiceNIC

- RiceNIC is free for research and education applications
  - Platform includes the FPGA configuration (bitstream), VHDL source code, embedded PowerPC firmware, and Linux device driver
- Purchase Avnet Virtex-II Pro development board
- Additional requirements to modify FPGA designs
  - Xilinx development software (ISE, EDK, Chipscope)
  - New Xilinx licenses for low-level hardware cores
  - Can still modify firmware without any FPGA changes or additional licenses

Conclusions

- Research into network systems demands experimental prototypes
  - Complex, asynchronous interactions among system components
  - Simulation insufficient for new architectures
  - Prototyping tools such as RiceNIC are critical for the development and understanding of future computer systems
- Constructing an experimental prototype is practical
- RiceNIC is awesome!
  - Provides hardware and software flexibility for a highly-capable experimental platform
  - Currently employed in research projects at several institutions

Questions?

- RiceNIC website:

http://www.cs.rice.edu/CS/Architecture/ricenic/

LACSI

AMD

XILINX®

hp invent