

Experiences With the Intel IXP 1200

at UCLA

Peter Reiher

UCLA Computer Science

Department

September 18, 2003



Outline

- Educational
- Research



Our Educational Plans

- To teach a capstone design course in network protocol design
 - Concentrating on the network and transport layers
- Meant to synthesize senior undergraduates' knowledge in large design project
- Planned to use Intel IXPs as the underlying platform



More Details on Intended Course

- Based primarily around large projects
- For small teams of undergrad students
- In some advanced area of network protocols
 - Preferably with some research flavor
 - Not all students do the same project
- Goal: working implementation by end of course
 - Limited by UCLA's quarter system



Our Support From Intel

- Intel sent us 15 IXP 1200 machines
- Also supporting software and some documentation
- Offered us assistance in configuring and setting up the lab
 - Which we foolishly didn't take them up on



Preparations

- We supervised a bright undergrad who worked on a project
 - As a pre-pilot
- Two teaching assistants set up the IXP machines
 - Both to prepare for students and learn more themselves
- Dr. Mario Gerla and I planned pilot projects
- We recruited students from the undergrad networking class



Running the Class in the Winter Quarter

- One “team” of one student, one of three students
- The single student was a ringer
- The three person team did well



How the Class Worked

- We started with “homework” assignments
 - Designed to familiarize students with the IXP
 1. Survey of what others were doing with IXPs
 2. Install a working environment on an IXP
 3. Make slight alterations to simple tutorial program (a packet counter)



The Projects

- Single student in winter quarter implemented a crypto package
- Three student team built a firewall
- Single student in the spring quarter worked on honeynets
- Other than crypto package, results were more helpful to learning than research



The Firewall Project

- Team had moderately ambitious goals
 - Setting up a firewall that pipelined packets through multiple microengines
 - Each microengine would test at a different level of network stack
 - Wanted dynamic updates to firewall rules
- Weren't able to achieve nearly that much



Firewall Project, Con't

- What they actually achieved
 - Non-pipelined testing of a very limited number of conditions at different levels in stack
 - Only static setting of firewall rules

- And I mean **static**

```
.local expected_mask // IP address
immed[expected_mask,0x83B3]
alu[--,expected_mask,-,packet_mask]
br!=0[end#]
.endlocal //IP address
```



What Were the Problems?

- Time
 - The major factor preventing pipelining
- Lack of good documentation and models
 - Tough to figure out best way to get info from the Strongarm to the microengines correctly
 - Students wished one of the tutorials (or other form of example code) had given guidance

- Quality of tools

“The workbench shipped with the SDK CD is basically useless in the research environment.”



Honeynet Project

- Single student starting from scratch
- Basically, put an IXP in front of a honeynet
- Capture copies of all packets and dump them into the Strongarm
 - For later examination by standard tools
- Like other group, worked from an existing application
 - Layer 2 bridge software, in this case



Results of This Project

- Not very profound code or design
- What it did worked
- With one student working alone, hard to complain too much
- An illustrative complaint from the student:

“Half of your time programming an application will be spent creating the basic files that enable your code to link correctly and make sure that all input and output ports are bound correctly.”



What We'll Do This Year

- A refinement of the original plan
- Still a project based class
 - With student teams
- No attempt to run as a two-quarter sequence
- More formal lecturing up front
 - That helped students last year
 - But it wasn't very well organized



Project Process for This Year

- More instructor supervision in project choices
 - TAs felt students appreciated choice, rather than being assigned a project
- Use of “guest lecturers” to discuss possibilities
 - Mostly grad students working with IXP
- More formal project proposal, evaluation, and refinement
- More reasonable expectations
 - In a fraction of ten weeks, few students can work miracles



Goal for the Year

- To run it as a real course
 - A real model for future years
- Shooting for 10-15 students per quarter
 - We'd hoped for double that, but our previous experience tells us we aren't ready
- Plan to investigate use of equipment for graduate education, as well



Lessons Learned From Our Experience

- The IXP 1200 environment is hard to work with
 - As we'd been told
 - We'd been somewhat lulled by our ringer's success
- Having it all working before students show up is definitely best
- Intel IXP book not that helpful
 - Students relied more on PDF documents
 - Will try new Comer book this year



Lessons Concerning Projects

- Teams are best
 - The three student team achieved more than the non-ringer one student team
 - Both in degree of implementation and learning
 - Even considering extra manpower
- Best to tailor projects to reality
- The quarter system sucks for projects



An Alternate Experience

- The ringer- a really bright undergrad
- He learned to program the IXP 1200 himself
- And designed and programmed an implementation of our D-WARD anti-DDoS system
- Over a summer
- Careful redesign specific to IXP 1200 characteristics
 - Using both microengines and Strongarm
- The great ones can manage this platform
- We need to make it easier for the others



Our Educational Needs

- For educational purposes, more powerful equipment not critical
- Better programming environments desirable
- More working examples of functionality on IXP would be helpful



Our IXA Research

- Also performed on IXP 1200s
- On topics of network security
 - Detecting IP spoofing
 - Handling DDoS attacks
- Prototypes of both systems built on IXPs



iSAVE

- Protocol to detect IP spoofing at network routers
 - Not just at Internet entry point
- Based on determining proper incoming interface for particular source addresses
- Protocol passes information among routers to determine proper interfaces for addresses



iSAVE on a Router

- Router must check each incoming message
- Must have data structure to look up the message's source address
- Must recognize when it does not have information
 - And ask in an efficient manner
- Issues of timers and retries

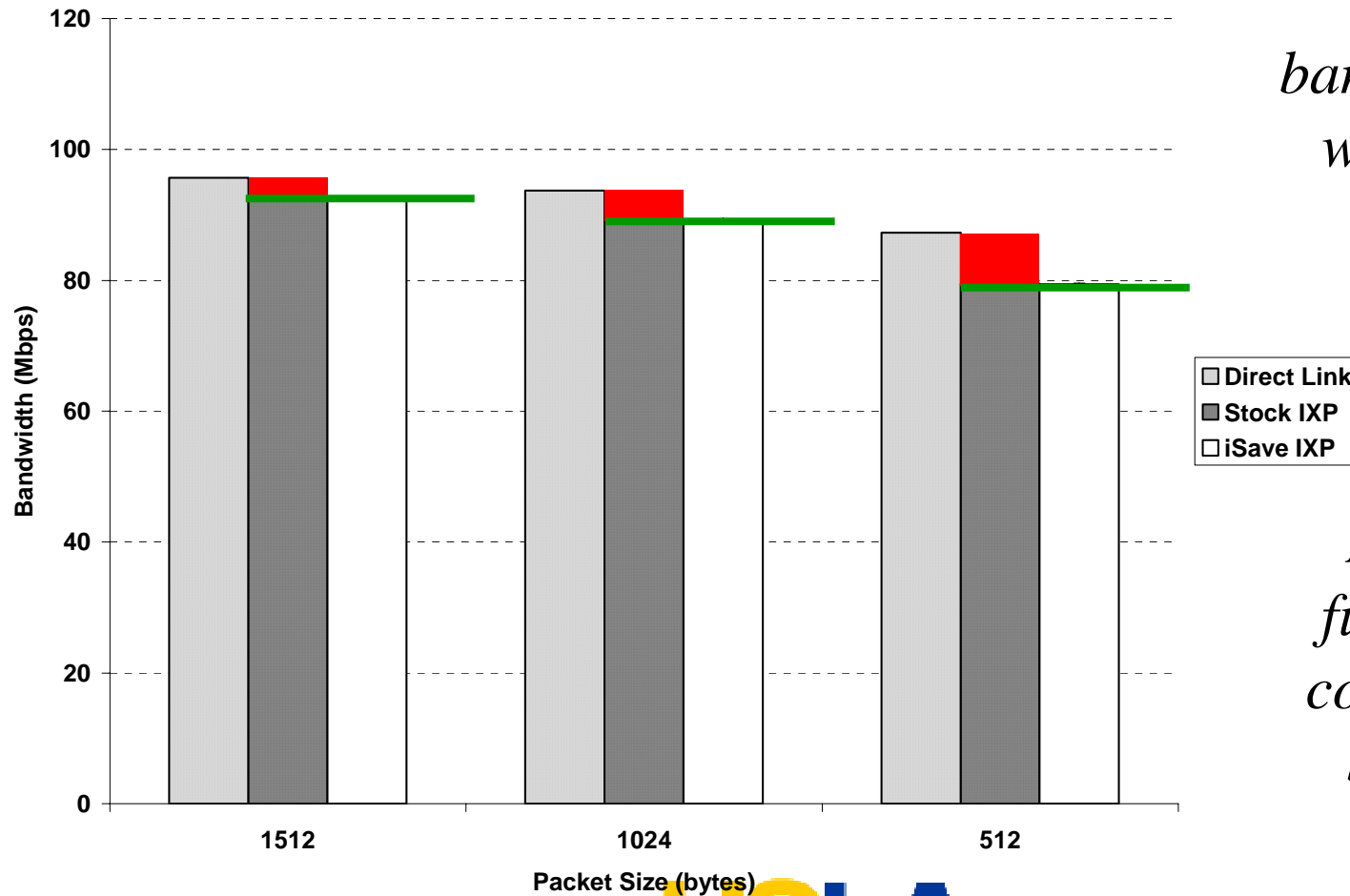


iSAVE on the IXP

- Implemented the filtering part of the protocol on the IXP
 - Not the part that sends updates
- Implemented on the IXP 1200
- Requires processing in both the fast path and the control path
 - On microengines and StrongARM, respectively
- Used standard IXP L3 forwarding code as a model



Performance of iSAVE on IXP



*Some
bandwidth lost
when using
IXP*

*But iSAVE
functionality
costs no more
bandwidth*



D-WARD

- Anti-DDoS system designed to work close to potential attackers
- Highly effective at stopping DDoS streams
 - While allowing legitimate traffic through
- Initially built in Linux router



iDWARD

- Implemented on IXP 1200
- Uses standard Intel Ingress and Egress ACEs
- Also uses new D-WARD ACE
 - To deal with D-WARD issues of packet handling
 - E.g., counting packets for flows and connections
 - Also applying rate limits and observing compliance to them
- The implementation is up and running



iDWARD IXP Architecture

- Uses two microengines
 - One handles all incoming packets
 - The other maintains the data structures used to track flows and connections
 - Separate microengines used because these operations have to happen simultaneously
- Little StrongARM code
 - Pretty much same operations required for all packets
 - Microengine was fast enough to handle them



iDWARD Performance

- Essentially achieves same throughput as raw bridging through the IXP 1200
- Better throughput than Linux router for small packet sizes
 - Avoids many OS overheads that are paid per packet
- Not quite as good as Linux router for large packet sizes
 - Basically, IXP has slower processor than Linux box
- Crossover point between two platforms between 128-byte and 256-byte packets



A Common Lesson

- You can do significant security-based processing on the IXP
- Without seriously compromising performance
- Even on the 1200
- You do need to design carefully



Future Research Plans

- Using IXP for other network security issues
- DefCOM
 - Anti-DDoS system with some core deployment
- Detection/prevention of worms
- Tracking interactive attackers who relay through multiple nodes
- Ubiquitous/mobile computing uses
- All require substantial per-packet processing at routers

