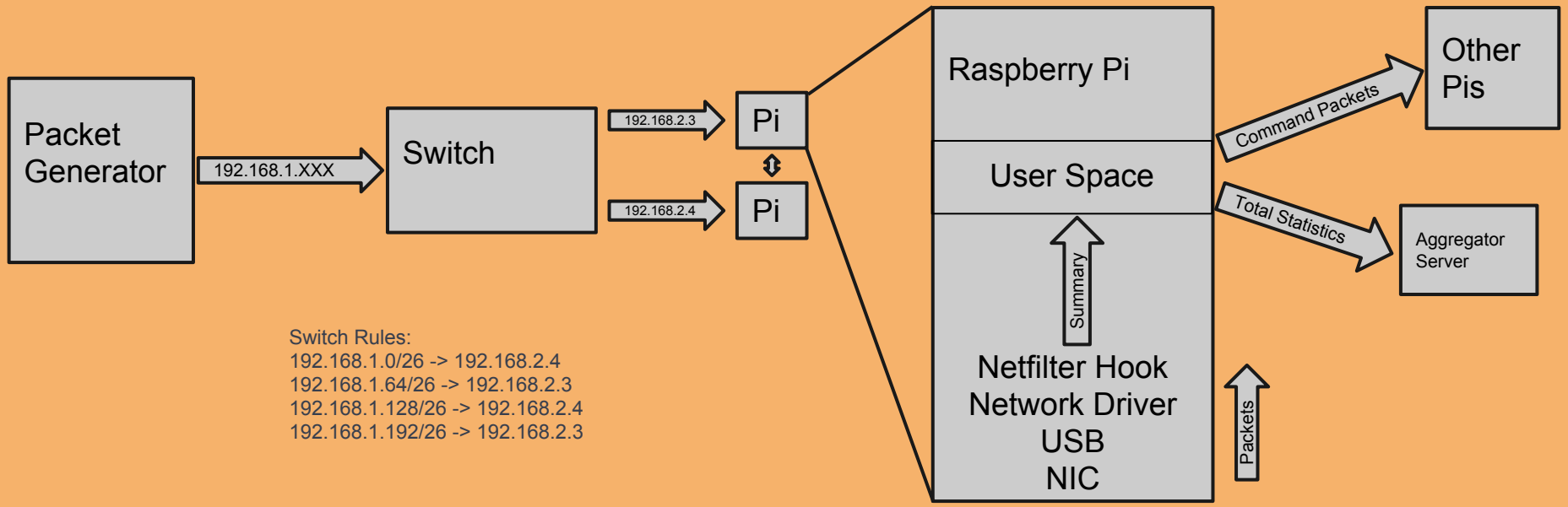
HoneyPi

A distributed Honeypot on Raspberry Pis

Motivation

- Building a distributed honeypot
- Evaluating Raspberry Pi performance and scalability
- Programming a switch to route packets
- Possible CS 3410 project

Architecture Diagram



Switch Rules:
192.168.1.0/26 -> 192.168.2.4
192.168.1.64/26 -> 192.168.2.3
192.168.1.128/26 -> 192.168.2.4
192.168.1.192/26 -> 192.168.2.3

Architecture Explanation

Packet Generator- generates command and data packets, and sends them to randomized IP addresses

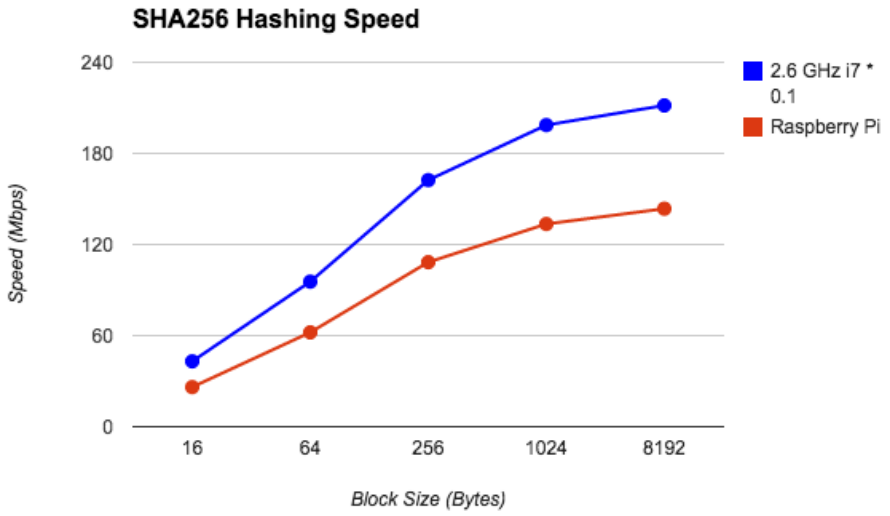
Switch- uses IP routing to partition the IP space among the Pis, dividing the packets between them

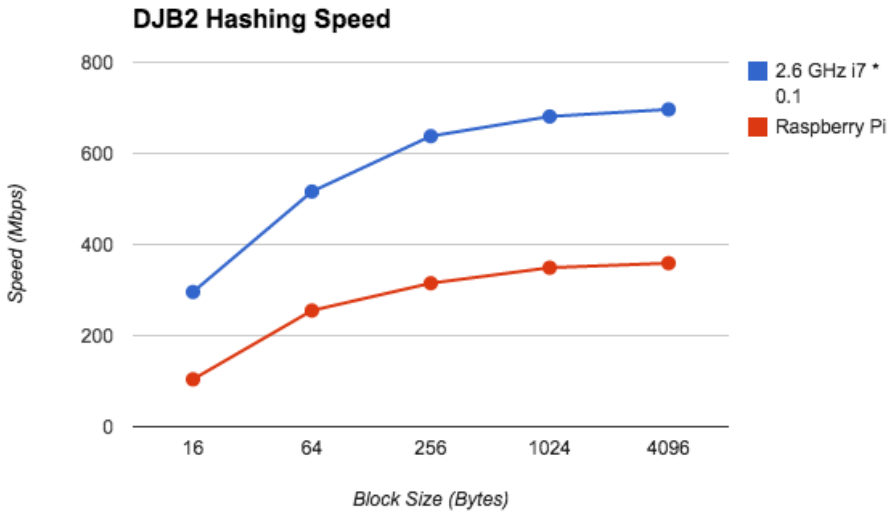
Honeypot Kernel Module- uses a netfilter hook to intercept packets and analyze them, sends captured data to user space program that aggregates statistics

Honeypot Read- reads packets from the kernel module, aggregates statistics in hashtables, and broadcasts received command packets to the other Pis

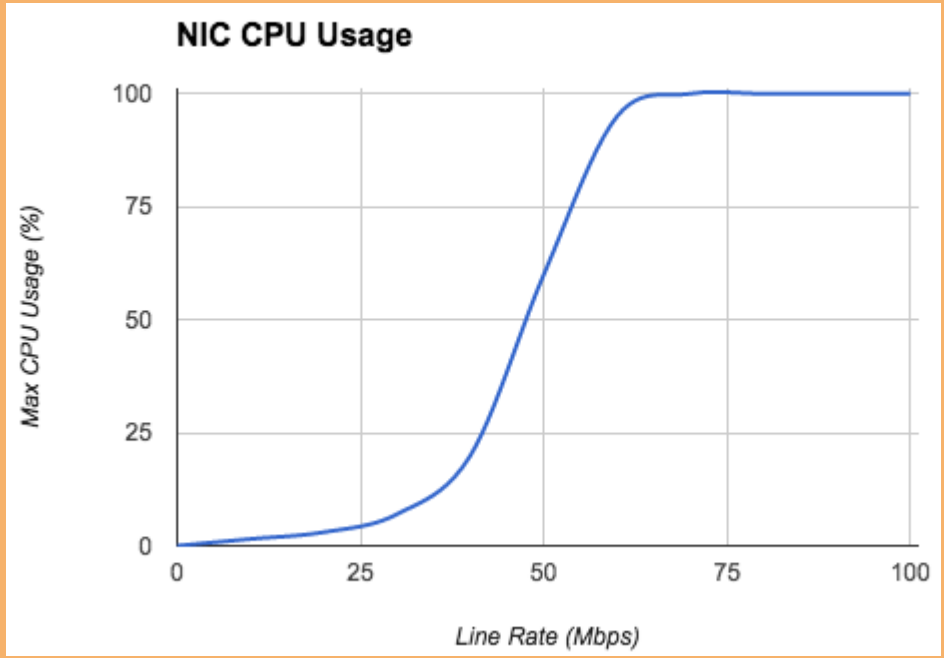
Statistics Aggregator- clients send their local statistics to the server, which combines them

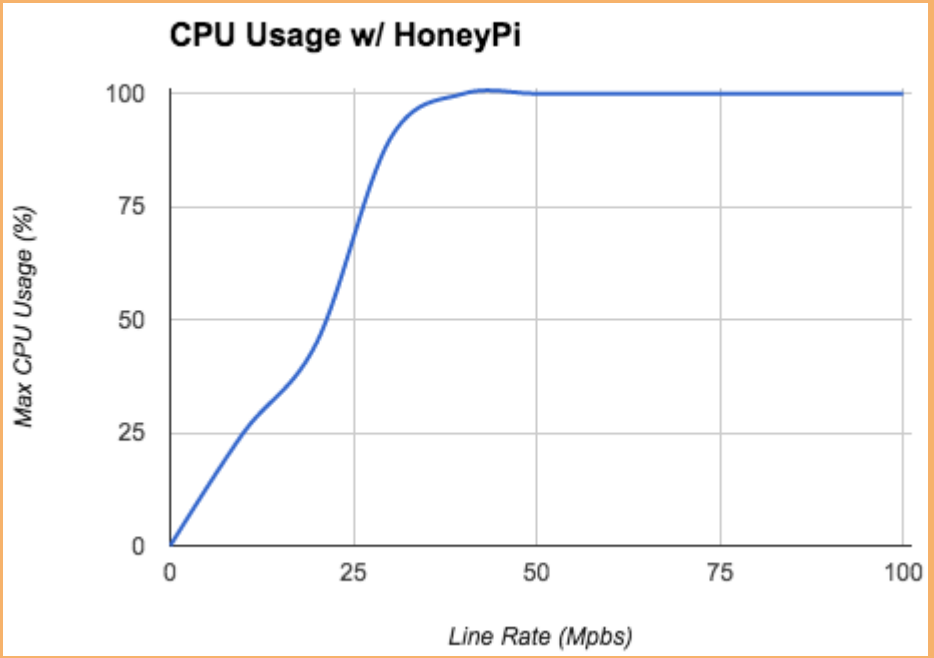
Raspberry Pi vs Laptop Hash Benchmarks



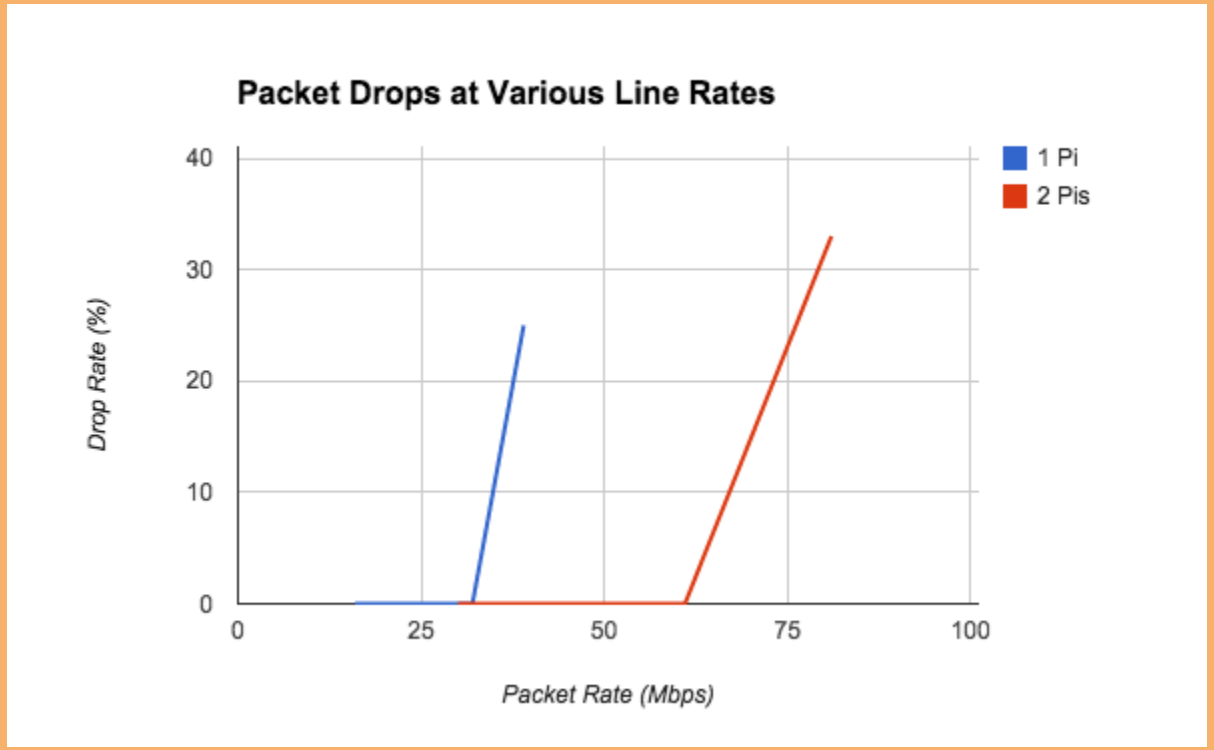


Raspberry Pi CPU Usage





Honeypot Results



Evaluation

- Kernel module could not run at line rate (100 mbps)- even without hashing the packet!
- Pi uses all of the CPU for the NIC (not even at line rate) without kernel module loaded
- You could hang a Pi just by sending it packets!
- SHA256 hashing was not working in kernel, so we used djb2 instead
- Packet generator scales very well on modest systems (over 500mpbs on a laptop), just need extra routing table entries on switch to add more Pis

Future Work

- Further optimization to the kernel module (packet capturing pre sk_buff, a-la NetMap)
- Creating a skeleton that could be used as a CS 3410 project
- Switching to the SHA256 hash function
- Porting to a more powerful board (ODroid)
- Scaling with more Pis

Conclusion

- Working distributed honeypot!
- Raspberry Pis are slow- both CPU and NIC
- But, the system can scale very easily and cheaply