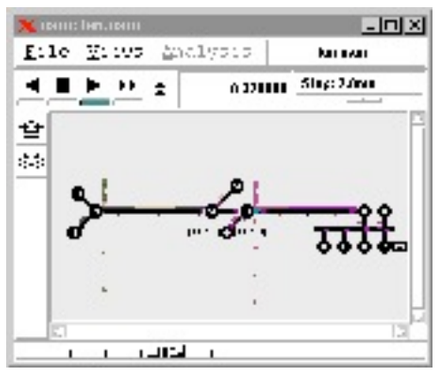
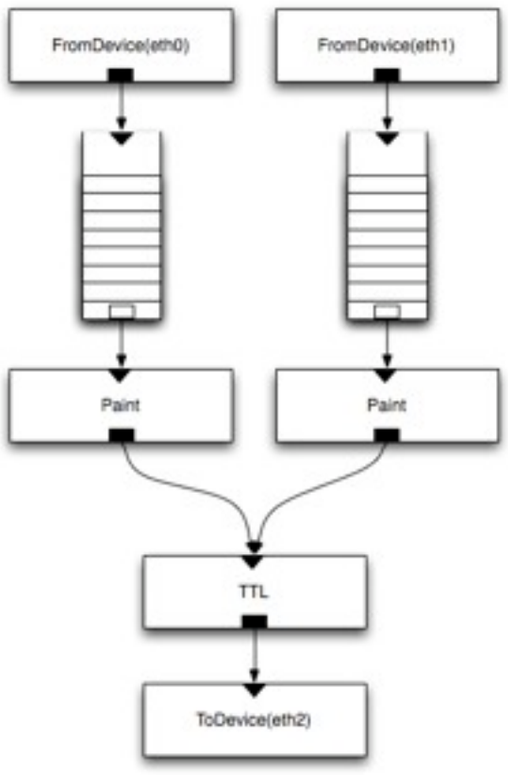
# Mininet: Squeezing a 1000 node OpenFlow Network onto a Laptop

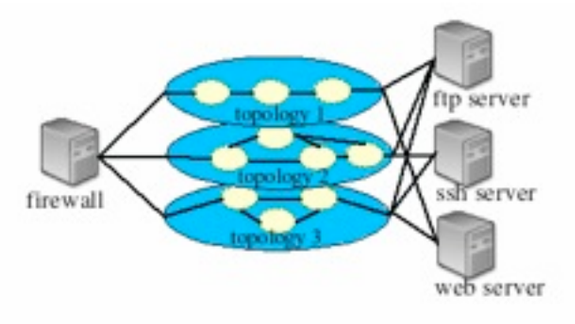

Bob Lantz, rlantz@cs.stanford.edu

November 19, 2009











Above figure: Clack in action

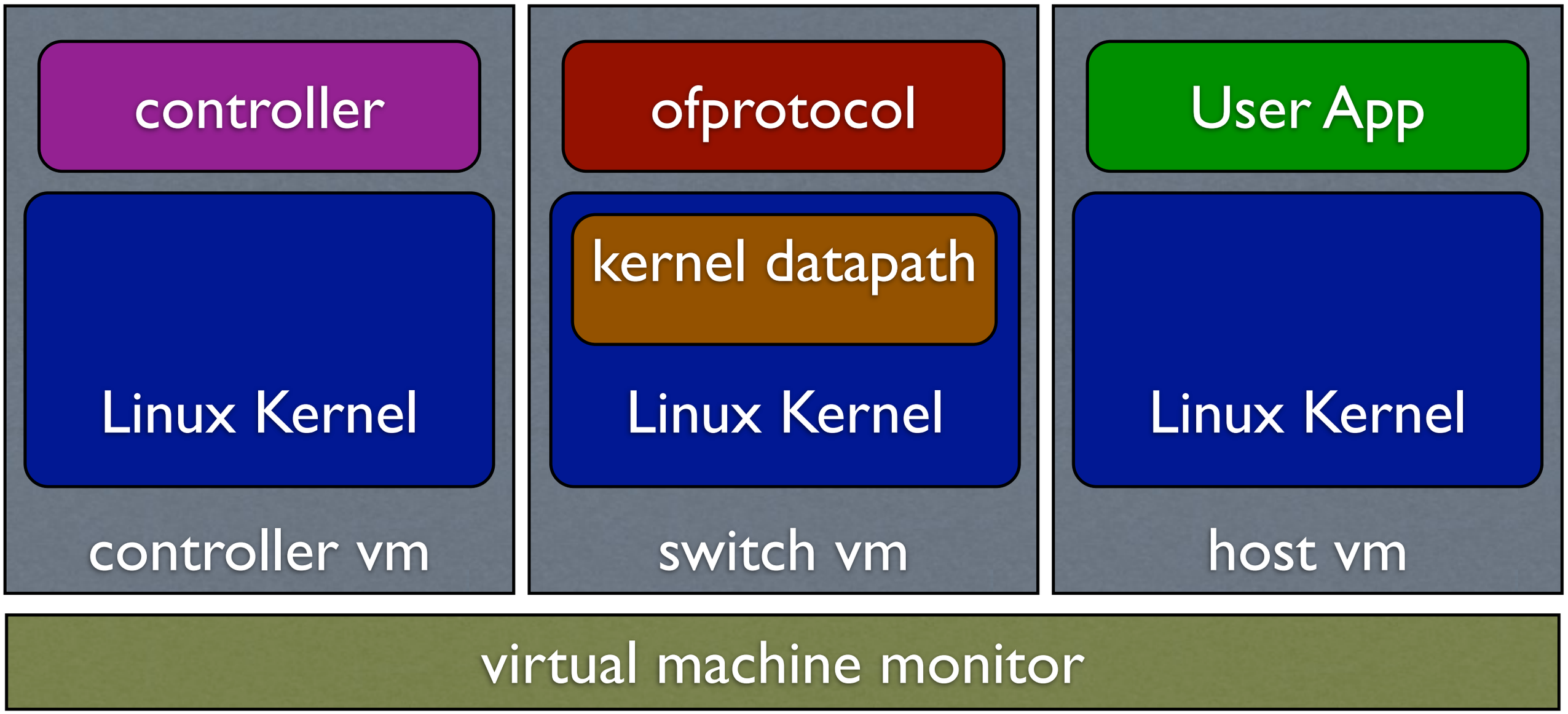


How To Do Network Research
- I'm trying to figure this out!
- Use OpenFlow, do cool stuff!
- But, you need a network to experiment/prototype on!
- Solution: virtual network on laptop
- my bias: large scale behavioral simulation

Virtual Networking Approaches
• Simulation: NS2
    - high detail/accuracy, low speed
• Custom SW switch: Click
    - more flexibility, less detail than ref. impl.
    - but need to fit multiple routers on single box
• Virtual network: VNS
    - closer to what we want for OF prototyping!

Goal: make something more like VNS that works with OpenFlow.

# Virtual networking with Virtual machines

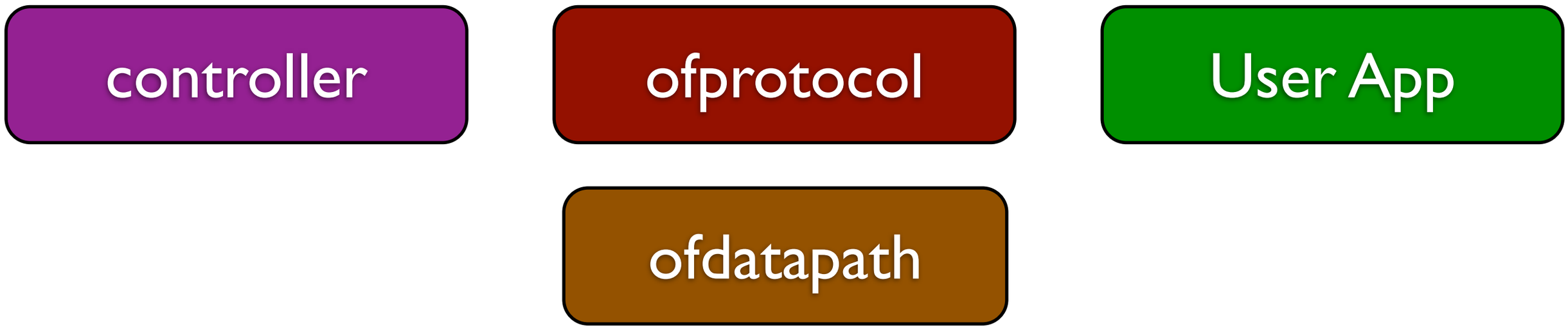



Virtual Networking with Virtual Machines
[diagram: VMs running openflow]
- Many advantages:
• Great work done on this already!
• Works great with OpenFlow
• Can use kernel-level implementation
• Fit a real network onto your laptop
• Migration, scalability to cluster
- Disadvantages:
• Need to run a VMM
• Complexity
- may be hard to set up, use, script
• Overhead
• Yiannis' poor laptop
• Overkill!!

| 1991: PowerBook 100 | 2009: MacBook |
|---|---|
| 16 MHz CMOS 68000, 4.4K Dhrystones | 2.26 GHz Core 2 Duo, 7.8M Dhrystones |
| 2-8 MB RAM | 2-8 GB RAM, 3MB Cache |
| 20 MB SCSI HD, 1 MB/s | 250 GB SATA HD, 120 MB/s |
| 230 Kb/s Localtalk | 1000 Mb/s Ethernet |
| 5.1 lbs, $2300 ($3600) | 5.4 lbs, $1000 |

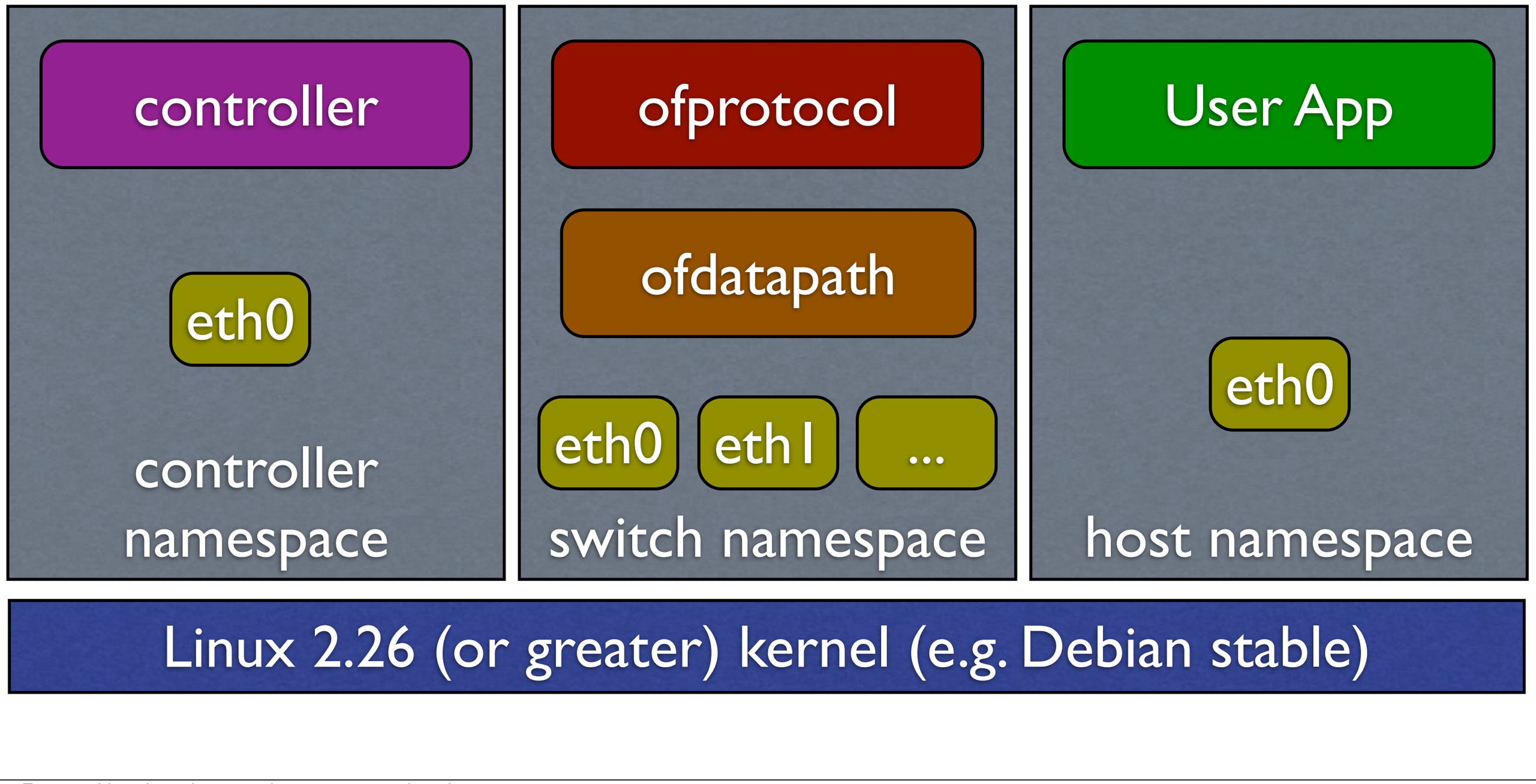
Apple laptops have in fact improved over time!
1900x faster (3800x if parallel! >>>x on GPU)
1000x memory
10,000x storage
... but only 100x disk read speed ;-(
4000x network
4.8 oz heavier ;-(
$300 cheaper (list price)
$1600 cheaper (2008 dollars)
(note I have a “pro” notebook, but hey)
Limitations: DRAM (1000x), disk performance (200x)
So: barring disk, we should be able to simulate 1000 1991-era laptops!
But: nobody does this, right?

# Processes



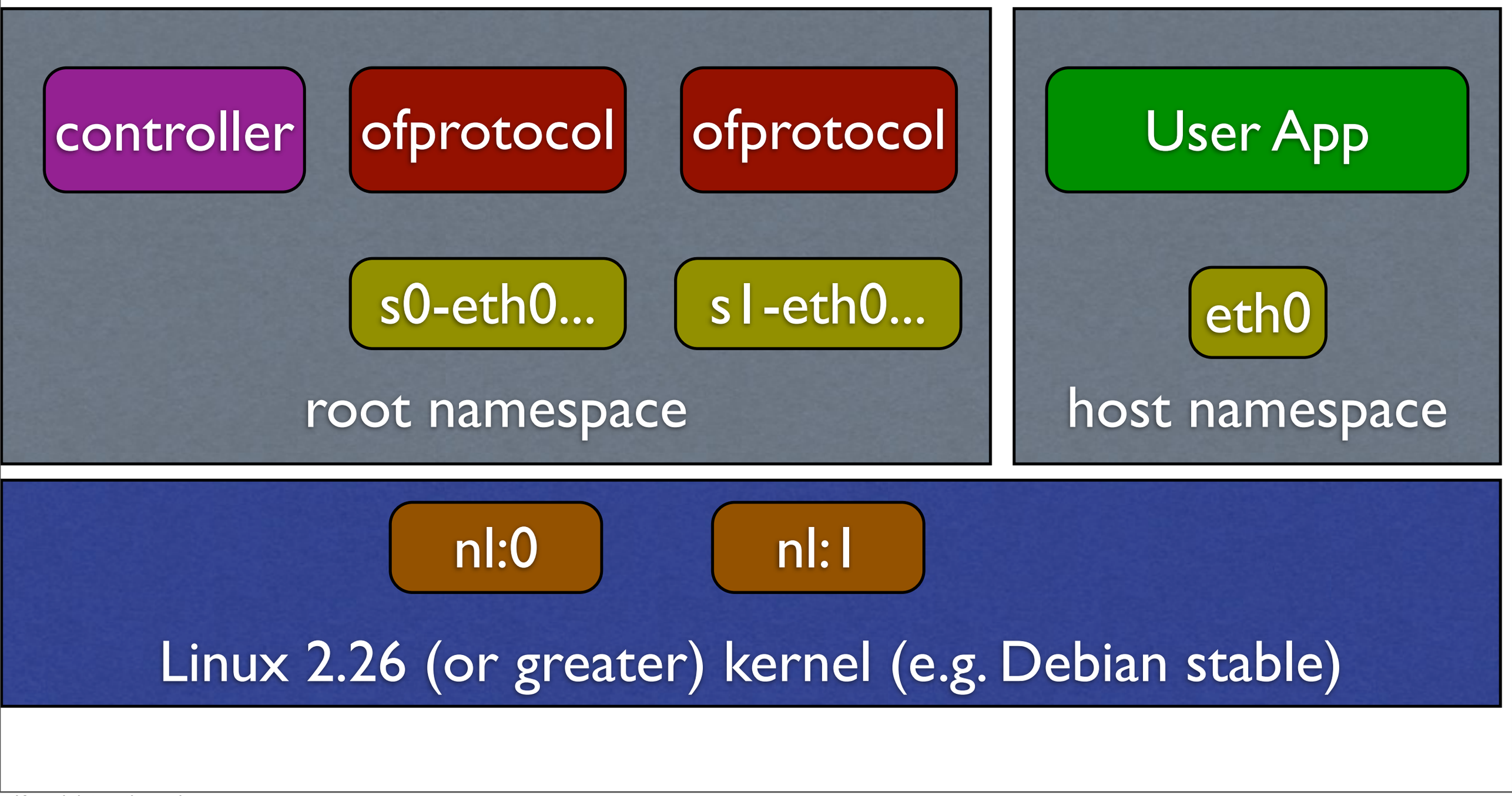

Alternative to VMs: Process-based virtualization
- Nearly every OS already has virtualization built in
    - it's called "processes"
- Extend processes with
    - own file system ('chroot')
    - own process space ('/proc')
    - own network devices ('network namespaces')
    - resource limits
- Examples: Solaris zones, FreeBSD jails, Linux containers
- Lower overhead, easier to set up!
- For networks, we only need the network part

# Mininet architecture



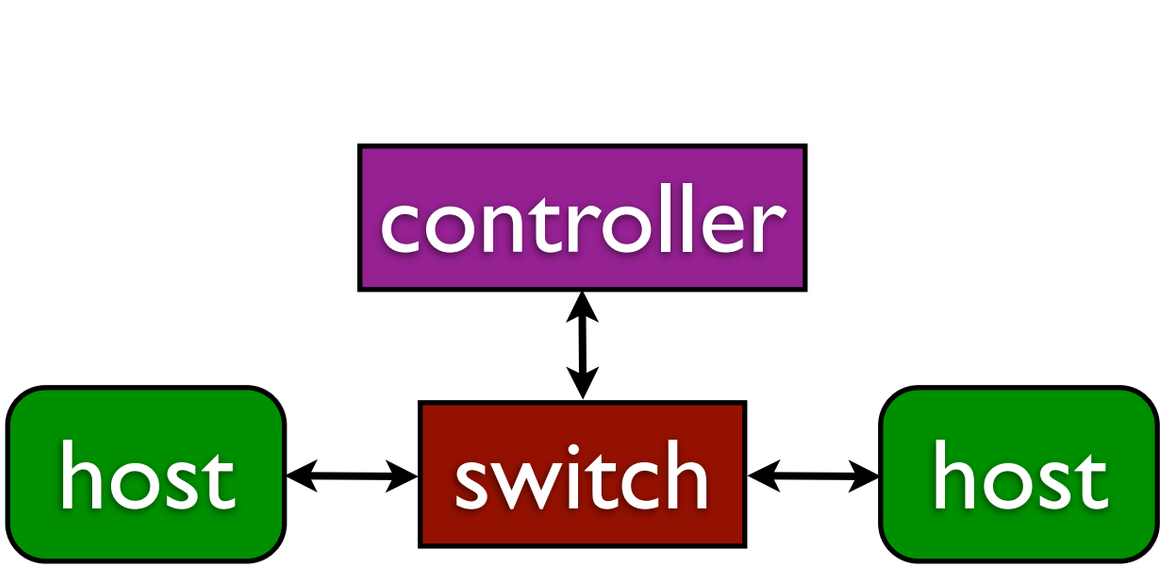

- For now: Linux kernel + network namespaces = happiness
- Very simple architecture:
    - supervisor process (mininet.py)
    - subprocesses (nodes)
        - /bin/bash in network namespace
        - local network devices (veth ends)
        - pipes to communicate to supervisor
    - switch nodes also have:
        - management interface
        - OpenFlow switch (udatapath or openvswitch)
    - controller nodes also have:
        - management interface
        - controller of your choice (nox, controller(8), etc.)

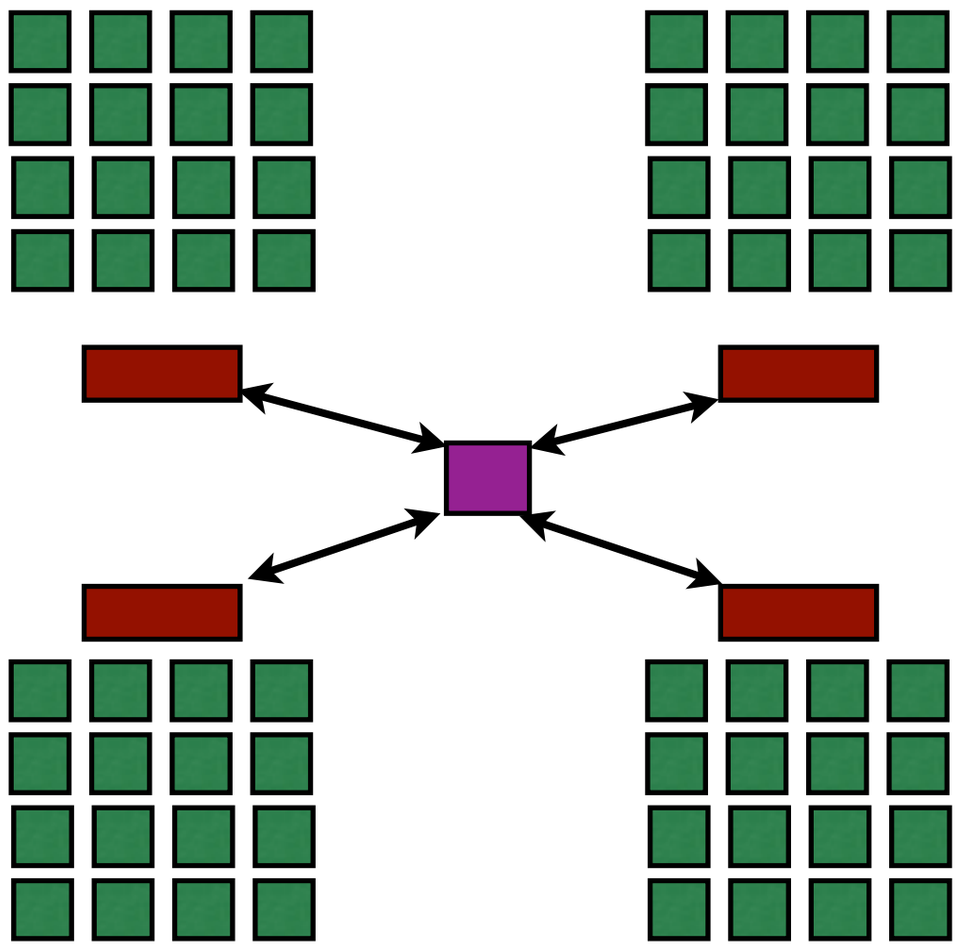
# Mininet architecture




Kernel datapath version

# Demo(s)



Demo: 512 node OpenFlow Network!

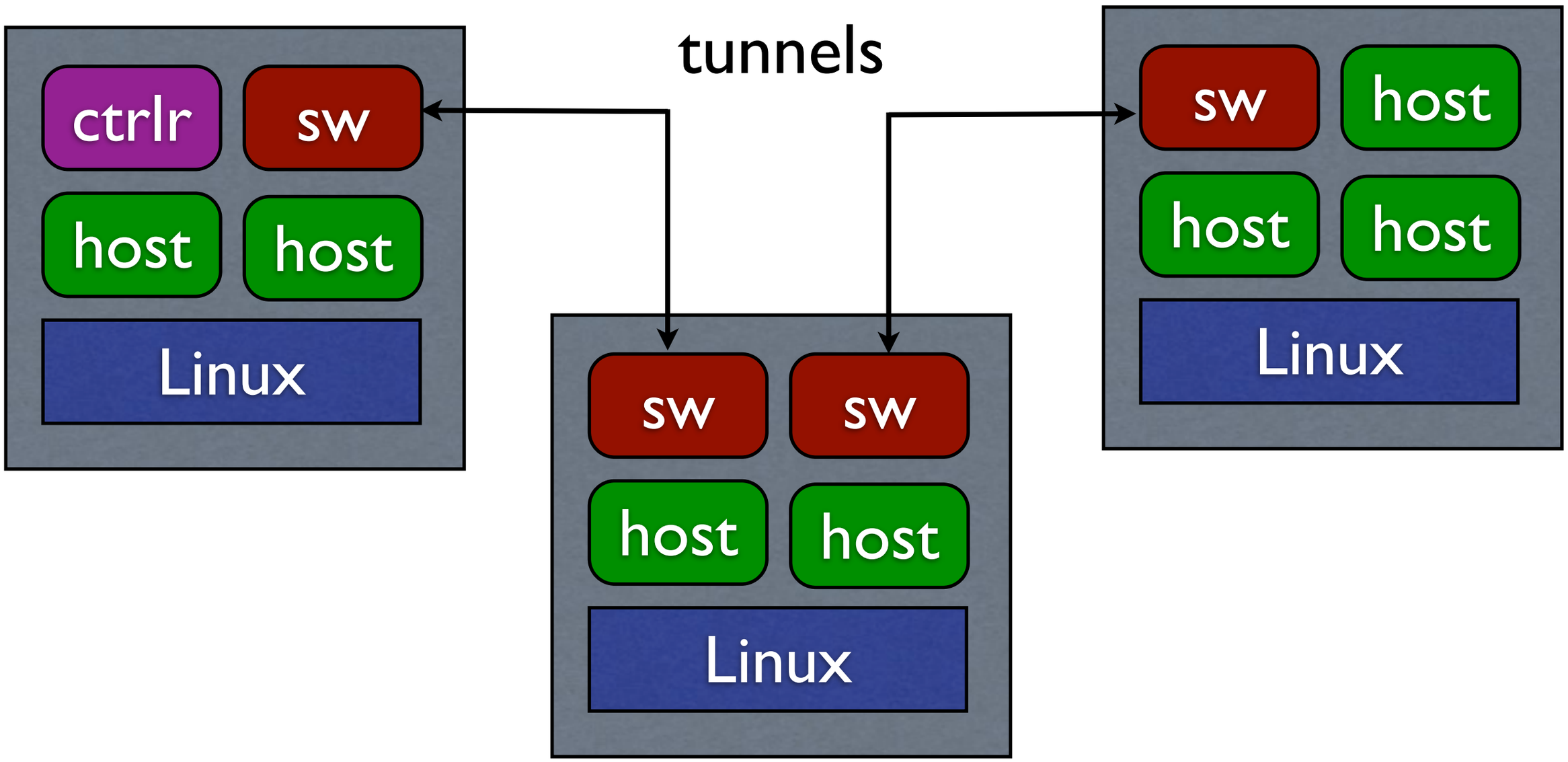
- iperf results for 1024-node network
- interactive (Tk) graph of throughput for various network sizes

Results:
- squeezes large network onto your laptop
    - caveat: 2 MHz processors, 2MB RAM, xxx Mbps
- simple: mininet.py is 200 lines of python

# Status/preliminary results

- Performance: 2-host iperf through udatapath w/reference controller: 512 Mbps
- Scalability: 512-node network: ping test works; 1000-node network: starts up in ~5 minutes
- Simplicity: core is ~200 lines of Python

# Mininet: cluster edition




Mininet: Cluster Edition (future work)

- would like more scalability and performance
- more like VNS!
- solution: run mininet on a cluster
    - replace bash with sshd
        - now all nodes need management interface
    - add tunnels to connect devices on different machines
- goal: simulate all of Stanford (25000 nodes) on our NetFPGA cluster!

# Mininet Conclusions

- Process-based virtualization is great!
  - Easy: basically a few lines of python
  - Scalable: net ns, fork/exec vs. boot, shared fs, OS, pages with low overhead -> 1000 nodes seems doable
  - Fun/Powerful: Composable, parametrized topologies, fast startup
- Hope to merge into OpenFlow testing/ tutorial/prototyping infrastructure

Key ideas:

- Cheap virtualization: processes vs. vms
  - network namespace: the only virtual feature we need
  - fork/exec vs. boot
  - shared fs saves disk space
  - linux page sharing vs. vmware page sharing

- Simplicity
  - 500 lines of python
  - composable, parametrized network topologies

- Easy way to get a miniature OpenFlow network
  - testing infrastructure
  - prototyping
  - tutorials
  - research