Software Defined Networks and Beyond

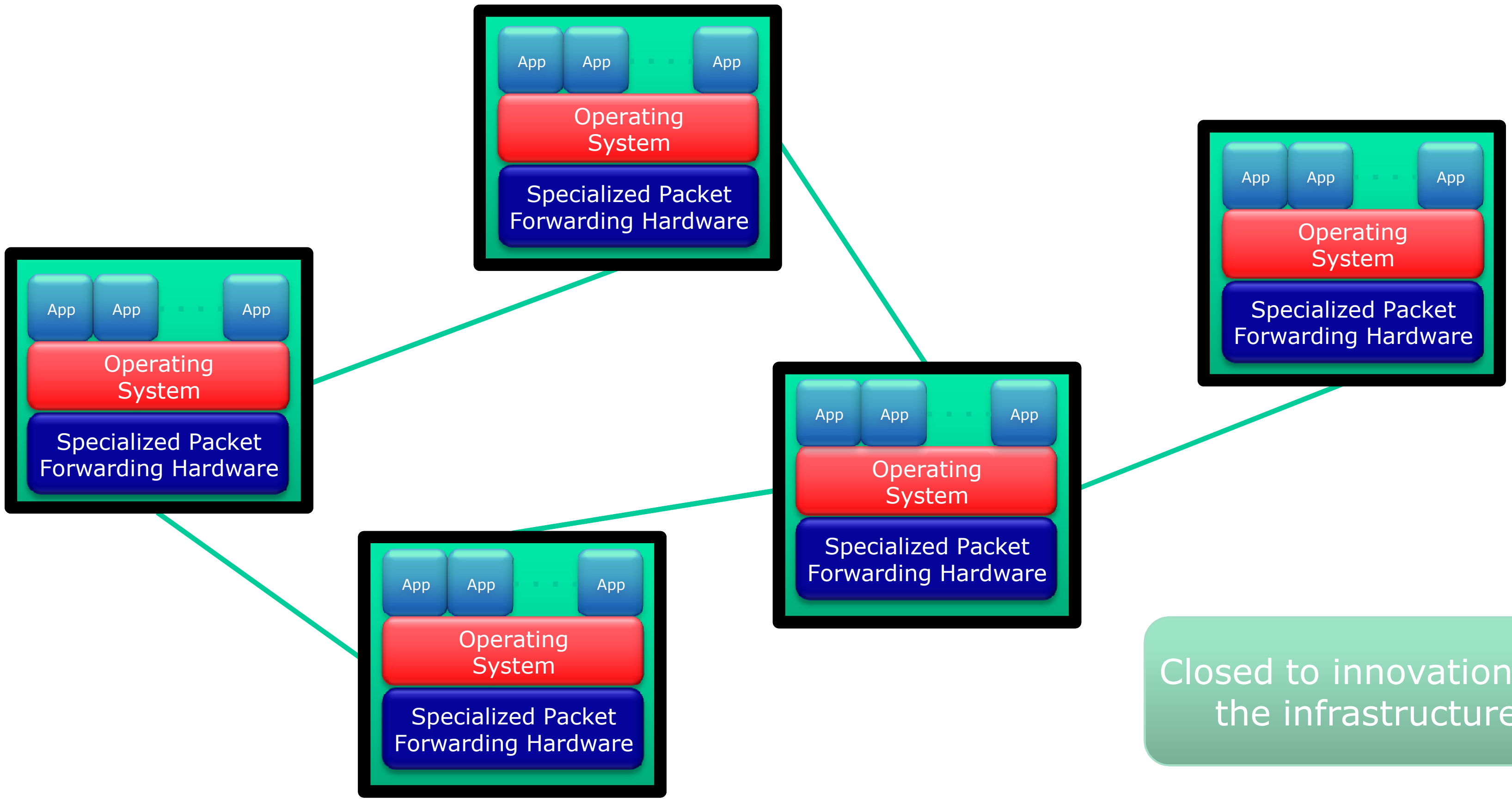
Fulvio Risso, Politecnico di Torino

http://fulvio.frisso.net

The Internet, after 30+ years

- Internet is still the one we defined 30 years ago
  - Same protocols, same philosophy
- Internet is a very efficient pipe that transport bits at high speed
- Can we add more value on those bits?
- Can we add more value to the network?

The Internet, after 30+ years



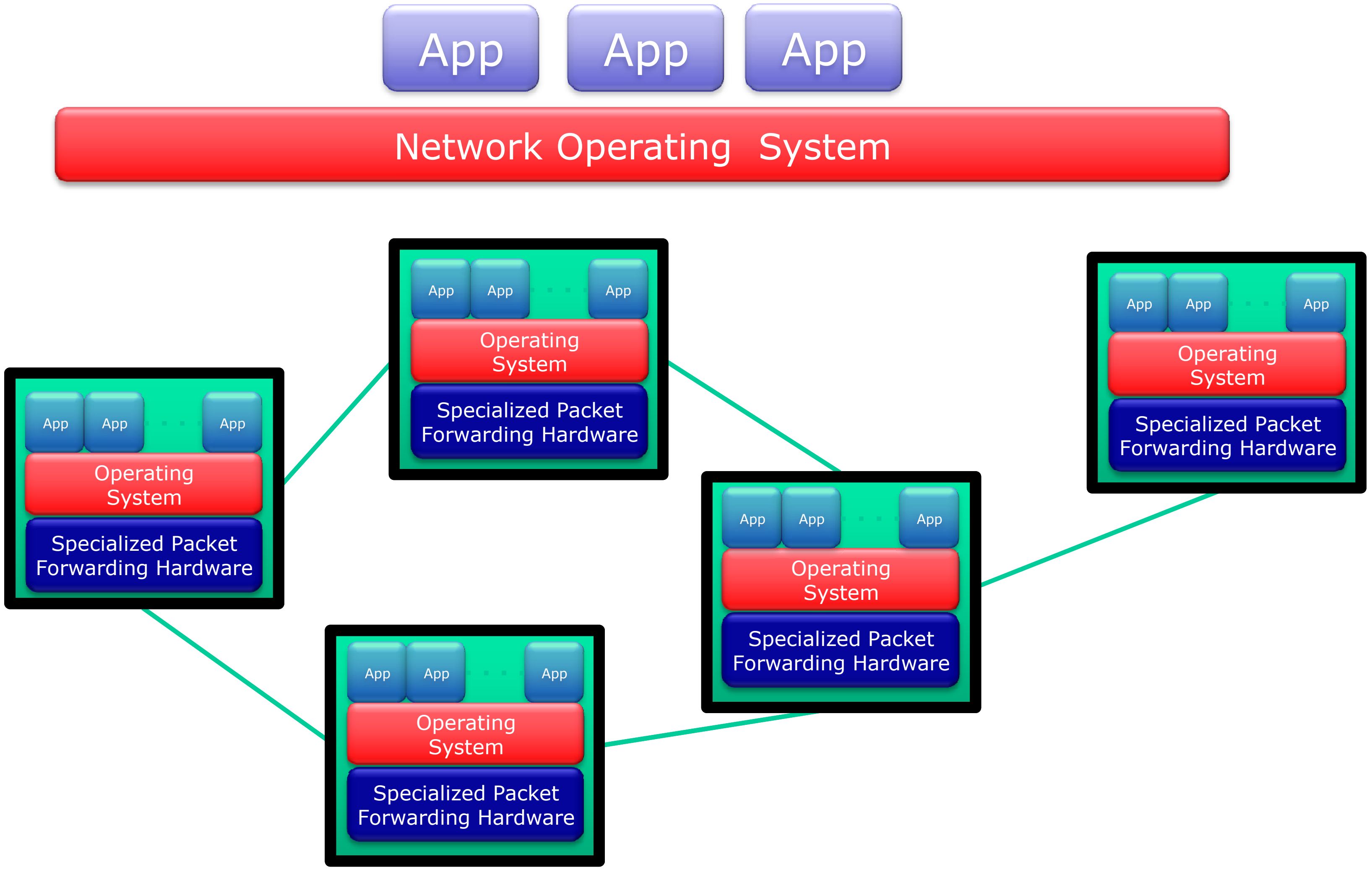
Closed to innovations in the infrastructure

http://www.rob-sherwood.net/GENI-Experimenters-Workshop.ppt

Software Defined Networks (1)

- SDN proposes to create a new programmable interface to the *network*
- Software from third parties (e.g., network operators, residential customers, enterprise managers, datacenter operators) can be installed that can control network devices
- What does "control" mean?
  - Currently, "control" means mainly "control plane"
  - We can implement new "routing" protocols (e.g., customize paths for network engineering)
  - Create network slices "private" to different entities (e.g., virtual network operators, application service providers such as video streaming, CDN, etc)
- Not that different from the traditional control/datapath separation, although the "control" is programmable here

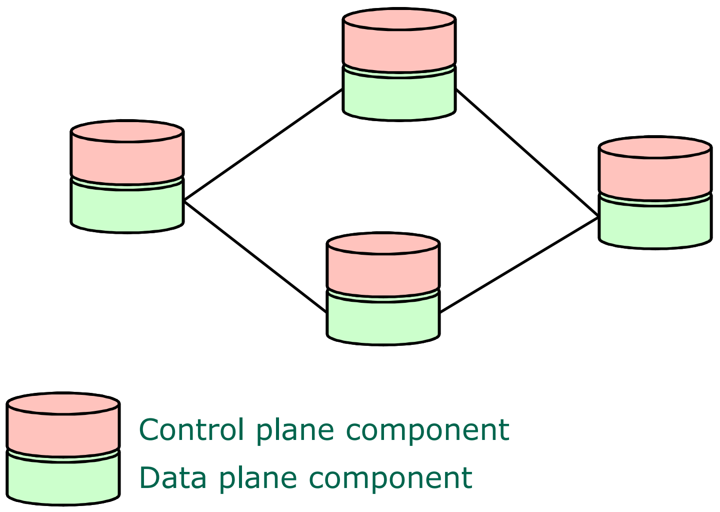
Software Defined Networks (2)


App
App
App
Network Operating System
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware

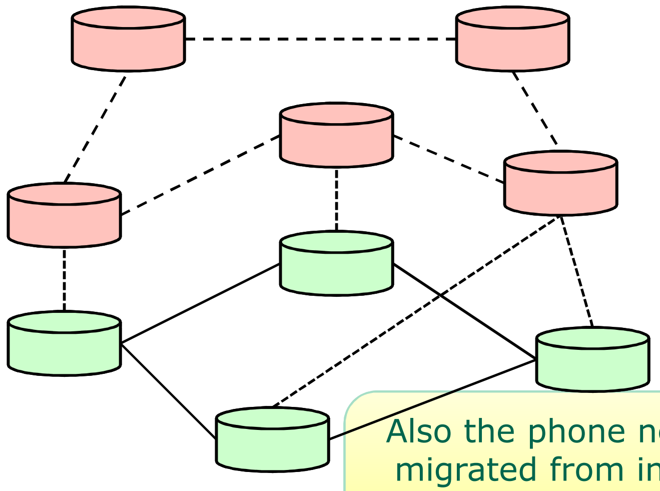
http://www.rob-sherwood.net/GENI-Experimenters-Workshop.ppt

Software Defined Networks in essence

Traditional
control plane architecture



Control plane architecture
with SDN

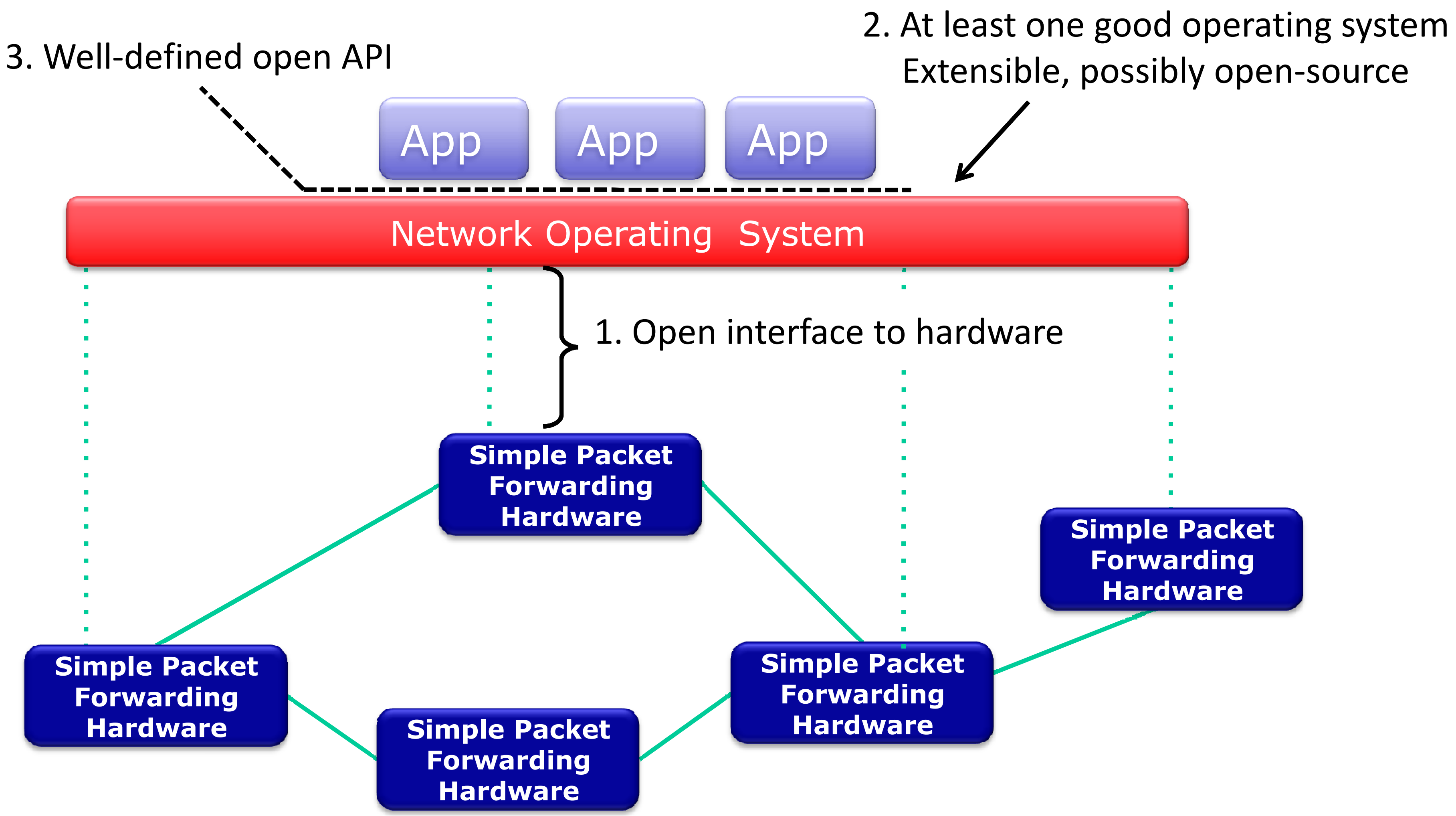


Also the phone network migrated from in-band signaling to SS#7 signaling some years ago

SDN is an approach to architecting the network control plane, where the behavior of the network is determined by software which is ***logically separate*** from the network devices.

This software could run in the network devices, a set/cluster of dedicated central server, application servers – or any combination of those.

The “Software-defined Network” (1)


3. Well-defined open API
2. At least one good operating system
Extensible, possibly open-source
App
App
App
Network Operating System
1. Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

http://www.rob-sherwood.net/GENI-Experimenters-Workshop.ppt

The “Software-defined Network” (2)

- Apparently, many people concentrated mostly on how to make the SDN, instead of what to do on it
- SDN is not just a new layer on top of existing routers, but it reorganizes the network architecture by moving some functions in other places
  - Controller (distributed/centralized) that keeps the intelligence
  - Simpler, cheaper, faster routers
- Here is where OpenFlow comes

The "Software-defined Network" (3)

- Offers a global view of the network to network apps
  - Right now, we have to configure each single device
    - VLAN, access lists, policies, QoS, …
    - The configuration may be incoherent on different devices
  - SDN wanted to give us the possibility to setup and application that operates across the whole network
    - E.g., check that a new user that connects to the network (from any port) is not infected; if so, give him his privileges/configuration
- Easy to translate this "global view" into a system that has a "global controller"
  - A unique controller is much easier to handle, but this is not the only option

What can we do on a SDN?

Optimize the network for our application

- E.g., private networks, CDN
- Fat-tree topologies in datacenters
- Bandwidth and resource optimization
- Service-specific packet treatment

Bring some network information into our application

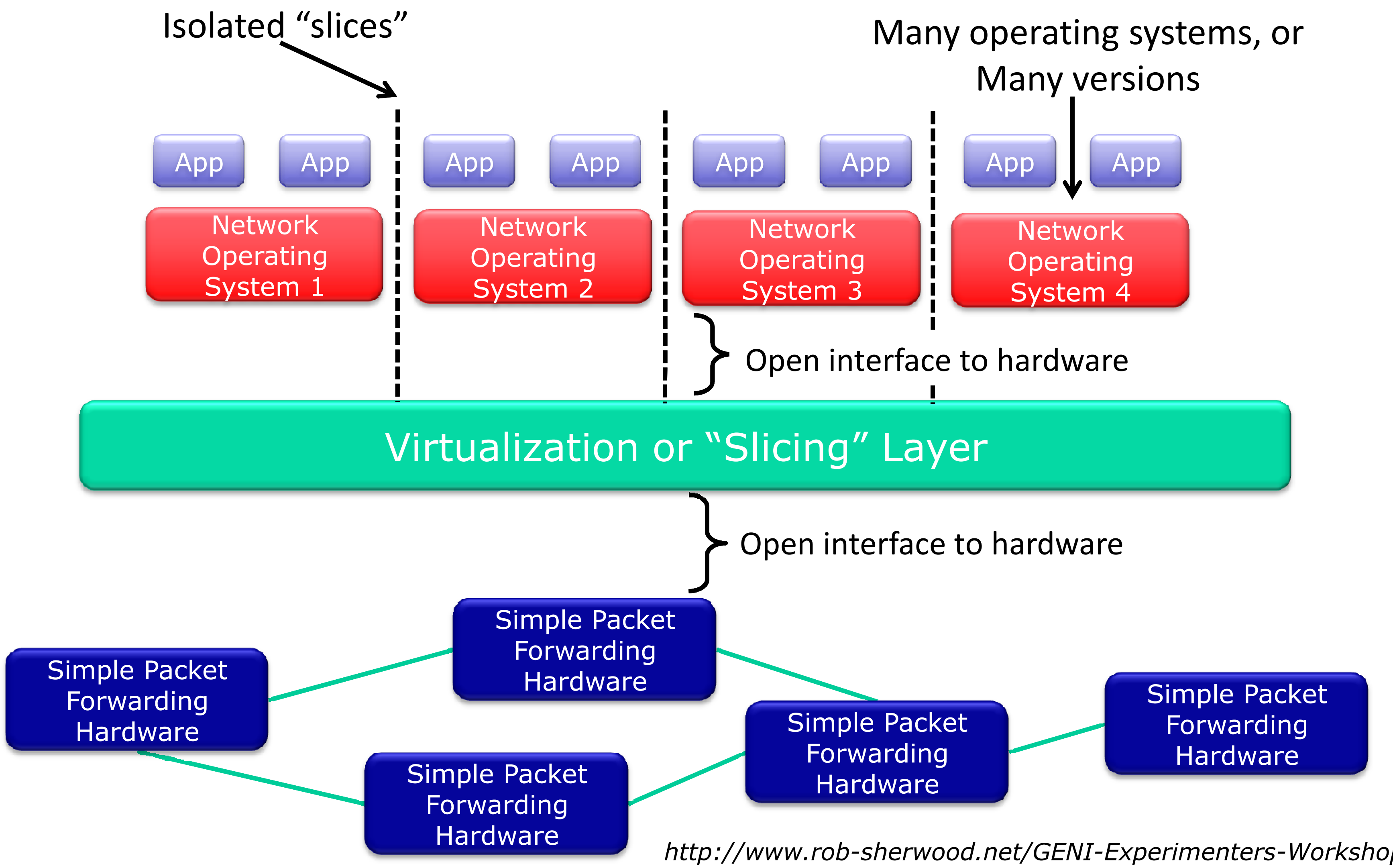
- Locality –aware P2P applications
- Actual location / topology
- Adjust behavior to real-time network usage

Applications we imagine are usually *network*-related applications

What do we do (now) on a SDN?

- SDN can be seen as a packet tunneling/switching technology that provides a pre-established forwarding path to specific service functions
- Technology enables selective traffic redirection based upon ephemeral classifiers
- Use cases
  - Video streaming
  - Content delivery networks
  - Virtual Private Networks - Social networks (with “physical” boundaries)
  - VM migration
  - Path optimizations (e.g., fat trees in datacenters, financials)

SDN and slices


Isolated "slices"
Many operating systems, or Many versions
App
App
App
App
App
App
App
App
Network Operating System 1
Network Operating System 2
Network Operating System 3
Network Operating System 4
Open interface to hardware
Virtualization or "Slicing" Layer
Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

http://www.rob-sherwood.net/GENI-Experimenters-Workshop.ppt

The good and the bad of SDN

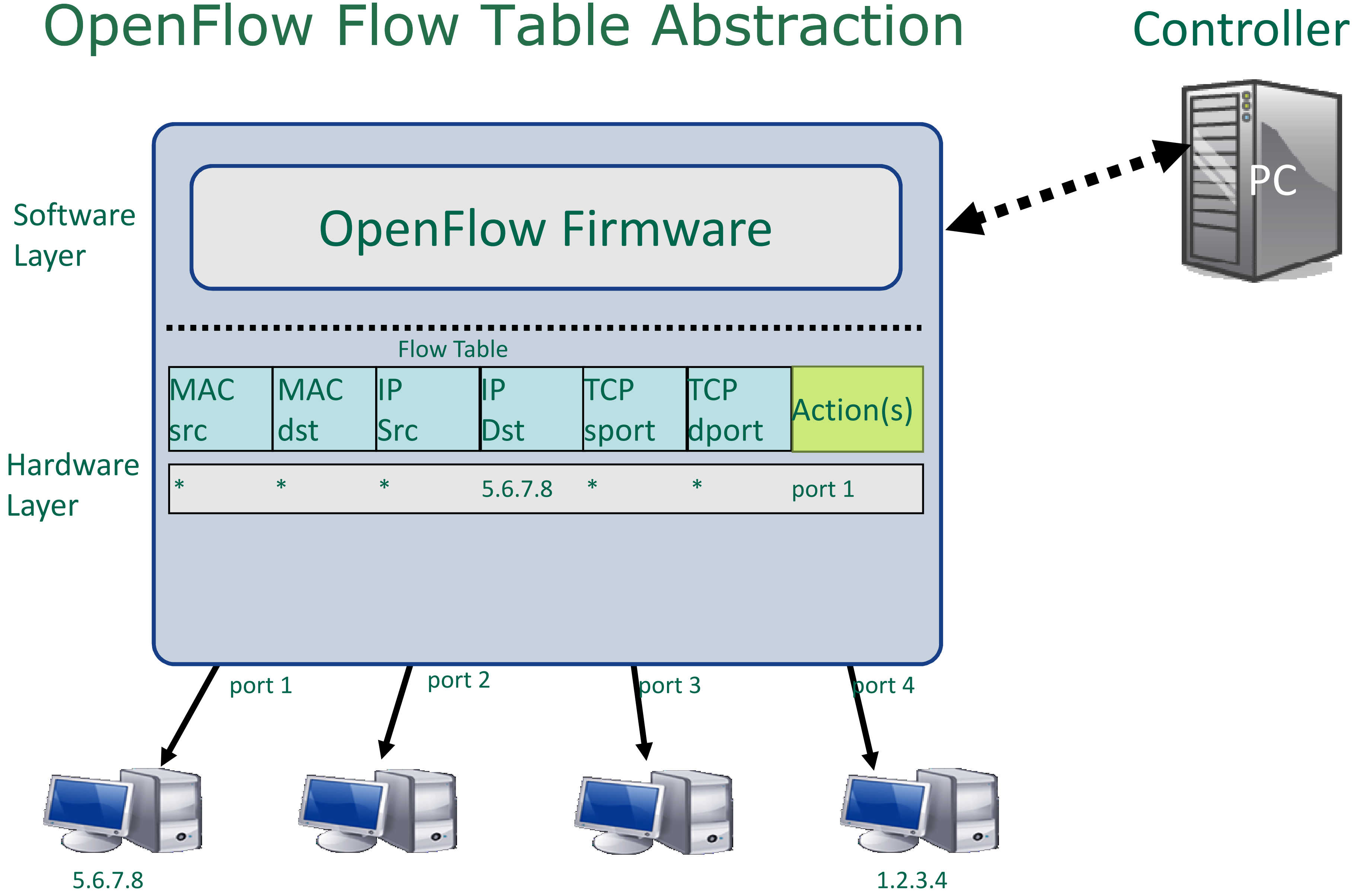
Good	Bad
• Network virtualization (slicing) • E.g., virtual operators, VPNs, social networks at the network level • Network operating system • Network API • Network as an unique entity	• Simple packet forwarding hardware • Major network vendors will try to do their best to kill you • Not scalable (hardware speedup needed) • Focus on the *network* • Ehm... Is there anything else to consider? (more details later)

A quick look at OpenFlow

Introduction

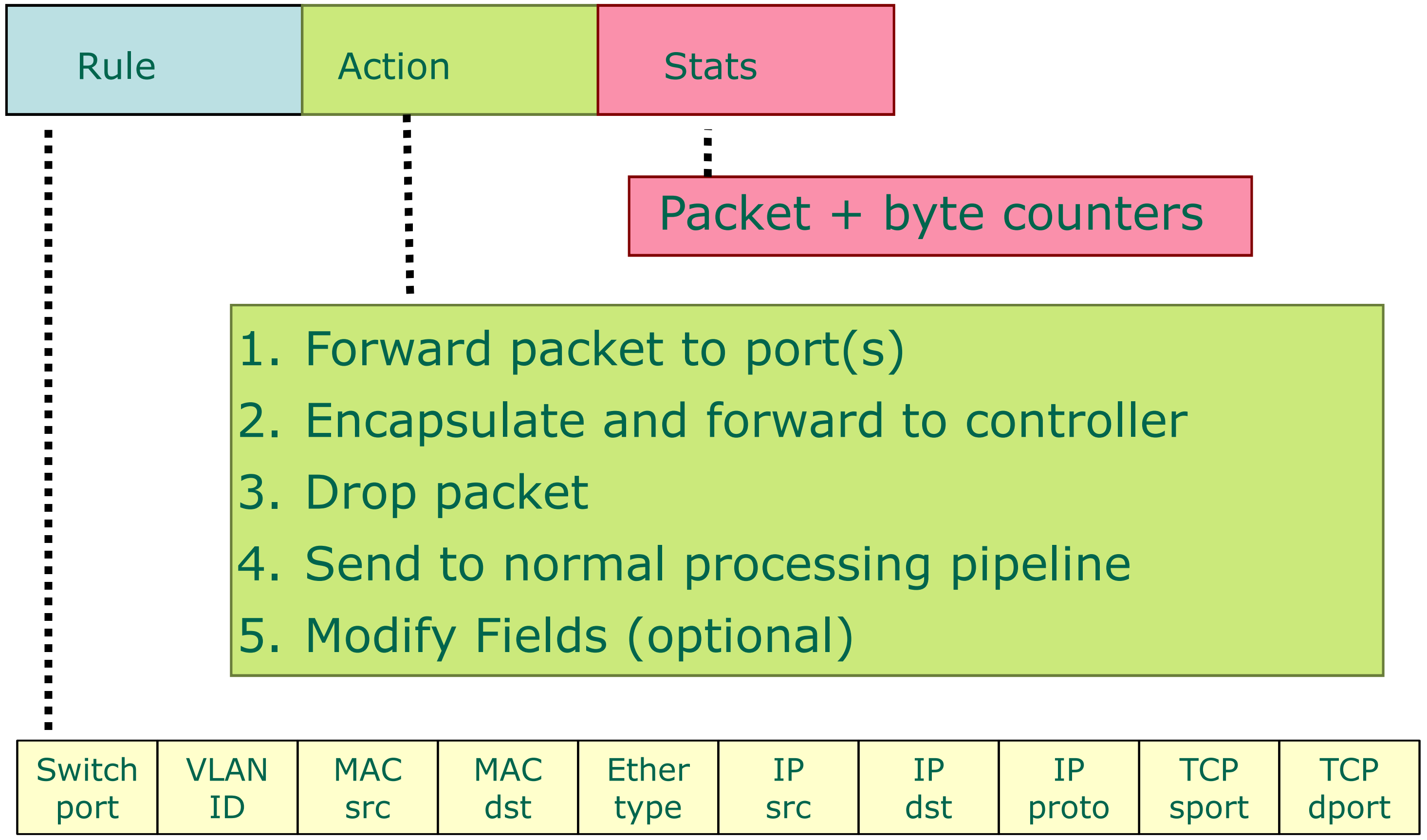
- OpenFlow idea launched around 2008
  - Ancestors (e.g., Ethane) published even earlier
- OpenFlow and SDN
  - For sure, it represents a way to implement SDN
  - Often confused with SDN
    - The proposed OF architecture was associated to the only way to implement SDN
    - Flow-based, centralized controller, reactive control

OpenFlow Flow Table Abstraction



MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action(s)
*	*	*	5.6.7.8	*	*	port 1

OpenFlow Basics: Flow Table Entries

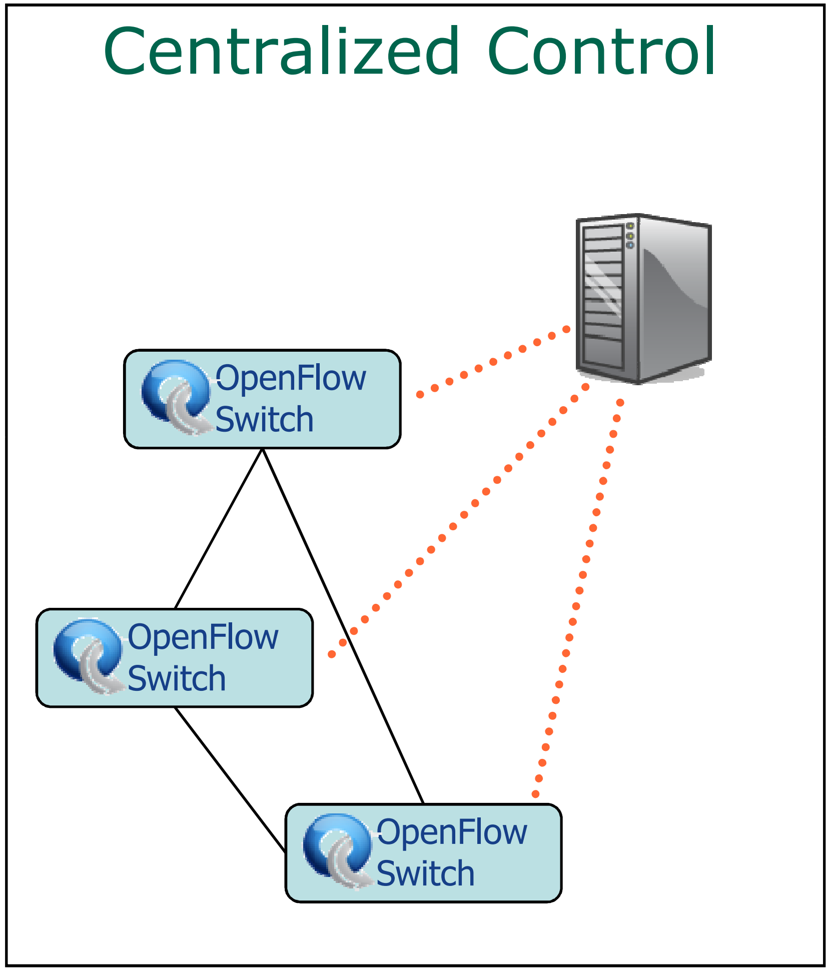


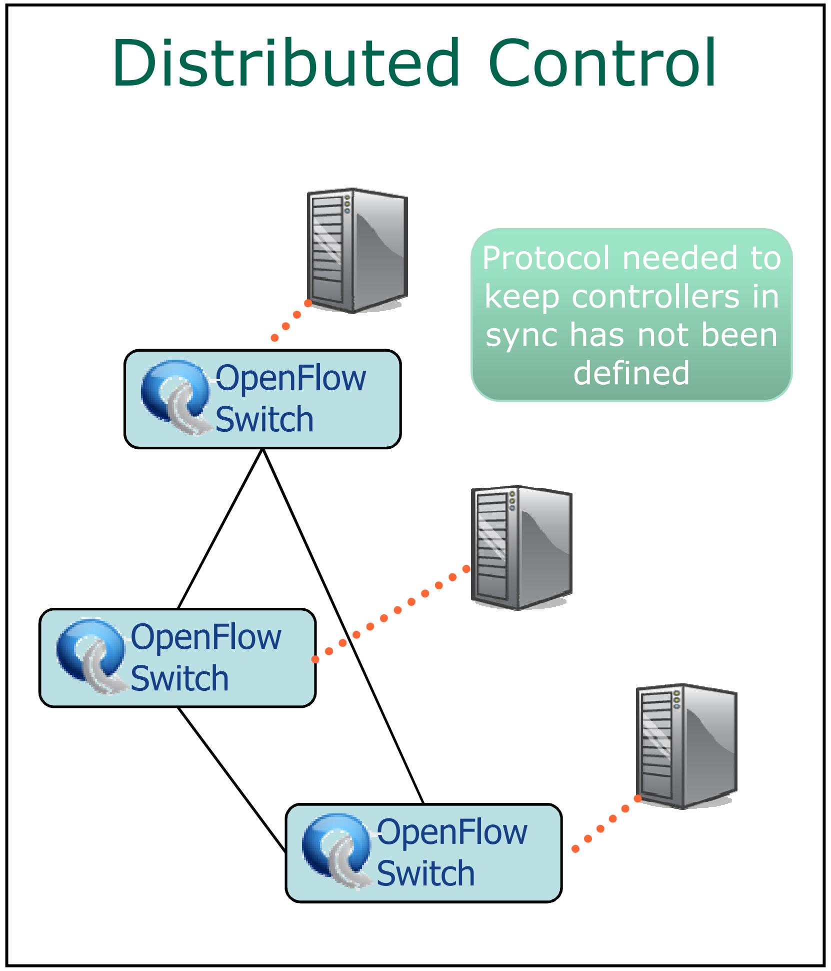
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields (optional)

Switch port	VLAN ID	MAC src	MAC dst	Ether type	IP src	IP dst	IP proto	TCP sport	TCP dport

+ mask for selecting which fields to match

OF: Centralized vs Distributed Control




Distributed Control
OpenFlow Switch
OpenFlow Switch
OpenFlow Switch
Protocol needed to keep controllers in sync has not been defined

OF: Flow Routing vs. Aggregation

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

Both models are possible with OpenFlow

OF: Reactive vs. Proactive

Reactive

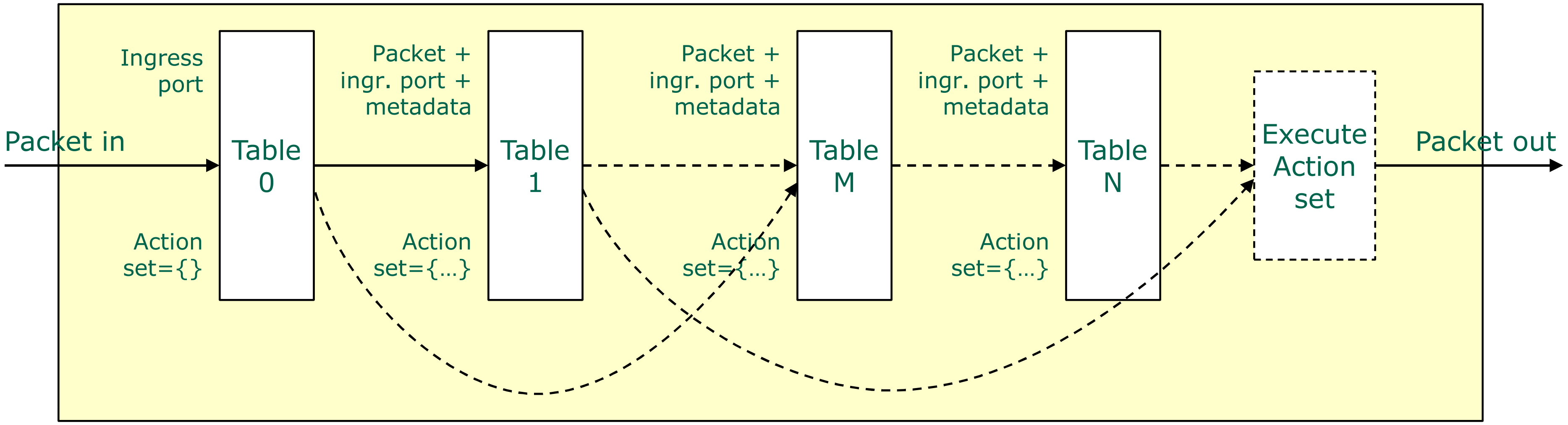
- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

Both models are possible with OpenFlow

OpenFlow 1.2 (Dec 2011)



- Multiple tables, not just a simple pipeline
- *Each* flow in table M can be redirected to another table N (M<N) (i.e., no loops in the pipeline)
- Some actions (e.g., send to controller, or drop) can be applied immediately after each table; other are added to the action set
- At the end of the pipeline, the action set is executed in a predefined order (not in the order the action set was created)
- Is it a good idea? More complicated than OF 1.0, while it does not solve the problem of creating a suitable abstraction of the network device

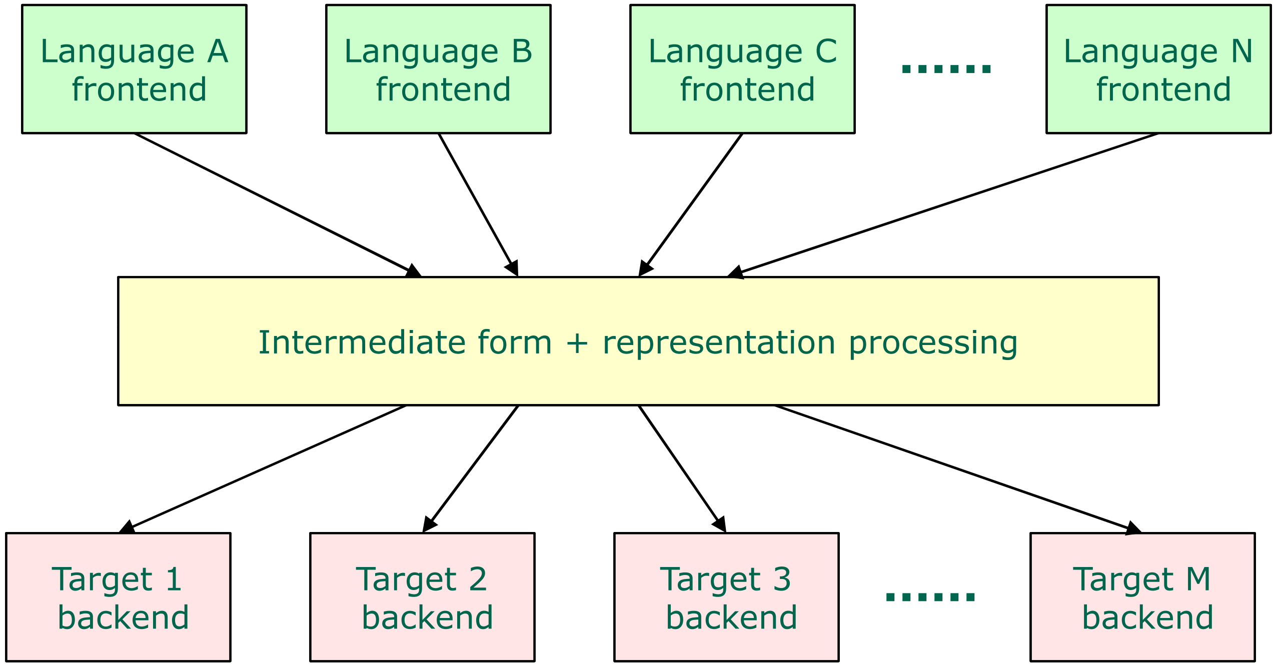
Openflow limitations

- Probably good for separation of control and data planes
  - For instance, this is something already existed for years (the separation between supervisor/linecards is not that far)

- Not good enough for data plane abstraction
  - Too simple switch model
  - Only tables
  - What about specialized hardware (encryption, content matching, etc?)

Openflow of the future (2.0?)

- Extends the “MAT model” (Match – Action- Table) of current OpenFlow



Some notes:

- FORCES did not make it
- Openflow (1.2) does not seem appropriate
- What else? A NetVM approach?

Similar to the LLVM / NetVM model

OpenFlow standardization

- Delegated to the Open Networking Foundation
  - You have to contribute with 30K$/year to be truly open
- Members
  - Many names, but the ball seems to be in the hands of the content providers
    - Google, Facebook, Yahoo, Microsoft, NTT, Verizon, Deutsche Telecom
    - Network manufacturers cannot sit in the board
  - Is the fee a way to discourage small companies/university to be active in that forum?
- Did you notice how many papers on SDN/OpenFlow come from the likes of Google?
  - Quagga, MPLS on OF, …

The good and the bad of OpenFlow

Good	Bad
• A possible way to implement SDN • So far, probably the only available choice • We may have something else in the future, but this is what we have now • Does FORCES exist? • May be suitable for some environments • Datacenters, enterprise • Introduces some more data-plane primitives • Although still not appropriate	• Simple packet forwarding hardware • OpenFlow 2.0 still a dream • Does not seem appropriate to large networks (ISP) • Not too much open • Still rather focused on the *network* • Although something not *topology-oriented* is starting to appear

SDN and OpenFlow

- SDN is a broader concept, and in particular
  - SDN != OpenFlow
    - Looks like SDN is *what* do to, OpenFlow is *how* to do
  - OpenFlow is one of many APIs that may be used by SDN
  - SDN is about *network programmability*
    - Contrast device programmability
- One of the hard problems with SDN is finding the appropriate abstractions and APIs
- OpenFlow particularly well suited to classification (edge application) and programming of ephemeral forwarding state
  - Still need APIs for QoS, Tunnels, …

Adapted from David Mayer, Ethernet Technology Summit, San Jose, CA, Feb 2012

Startups and beyond

- Nicira
- Big Switch
- Pica8
- Embrane (?)
- …
- *Verba volant*… gossip only on the air…

Some additional considerations

Is OF/SDN a threat for network vendors?

- If the network device will be stupid, yes
  - Probably, this is not really the way to go
  - So, probably it will not be a threat
- Will network vendors loose their grip anyway?
  - Not sure about this; probably, things are going to change
  - What about a NetAppStore completely controlled by each single network giant?
    - Apple is not making that much money from its AppStore (1% of the profits?)
    - Apple is selling a huge amount of iPhone/iPad, though
  - Some numbers
    - HP, Dell: 7-15% margins
    - Cisco: 70% margins

Cisco and the ISR line

- Router + x86 processing board tied in the same chassis
  - Interconnected through an high speed network (GbE)
  - Not too much integration
  - Possibility to register events in the data path that trigger an action on the x86 board
  - It reminds the old days of the Cat5K with switching and routing in the same chassis
- Applications
  - Not very "network related"
  - Turn lights on/off, handle alarms, fax server, WAN optimizer, etc.
- Was this successful?
- Is this what we need?
  - Is this so different from a router and a server stacked one on top of the other?

Network apps

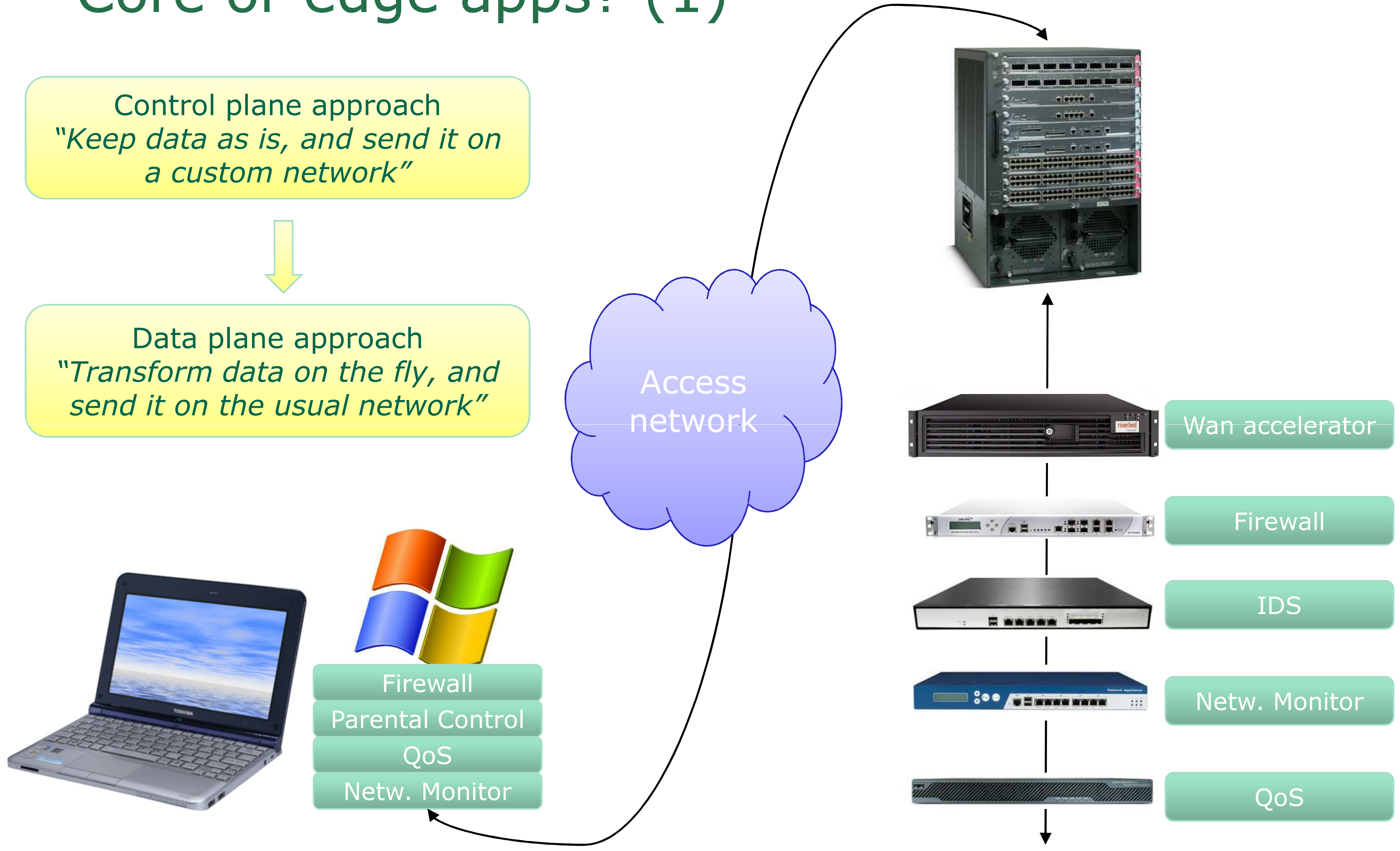
- We envision there is a class of applications that need to interact very closely with the router data path
- Those applications must reside on the data path of the network device
  - Often they have to handle part of the traffic
- Possibly, they should make use of the hardware speed-up available there
- Not a good idea to install a router with a server on top of it, and run those apps on the server

Parental Control
QoS
Wan accelerator
Firewall
IDS
Netw. Monitor
QoS

Core or edge apps? (1)

Control plane approach
"Keep data as is, and send it on a custom network"

Data plane approach
"Transform data on the fly, and send it on the usual network"



Core or edge apps? (2)

- SDN introduced the "control/data path" separation, but most of the efforts is in the *control*
  - Controlling the network, controlling paths, traffic engineering, making network-aware apps
  - OpenFlow, in fact, looks more flexible, but still most of the proposals focus on the *control* side
- What about the data path?
  - Let's "put more intelligence in processing traffic"

Edge applications:

We can change how the edge network node processes the traffic, not how the network transport it

Core or edge apps? (3)

- One approach does not exclude the other, but
  - We feel that smart and customizable edge nodes can enable more services
  - It solves practical issues
    - Has anybody seen the mess of multiple appliances in cascade in edge networks?
  - There may be more money
  - We are interested in the second approach ☺

Core or edge apps? (4)

- The core remains unchanged
  - Probably not always a good idea to make smarter the data path processing in core routers
    - Core routers must be fast, fast, fast, fast
    - It may make sense in some specific environment
  - Any attempt to create a clever core network failed (so far)
  - Hard to upgrade the core network
    - Cost, difficulties to get advantages when only a portion of the core has been upgraded
- The edge needs to be upgraded
  - More feasible migration path, as the gradual replacement of edge network nodes is simpler and it guarantees immediate advantages to the users on the new nodes

Control plane approach

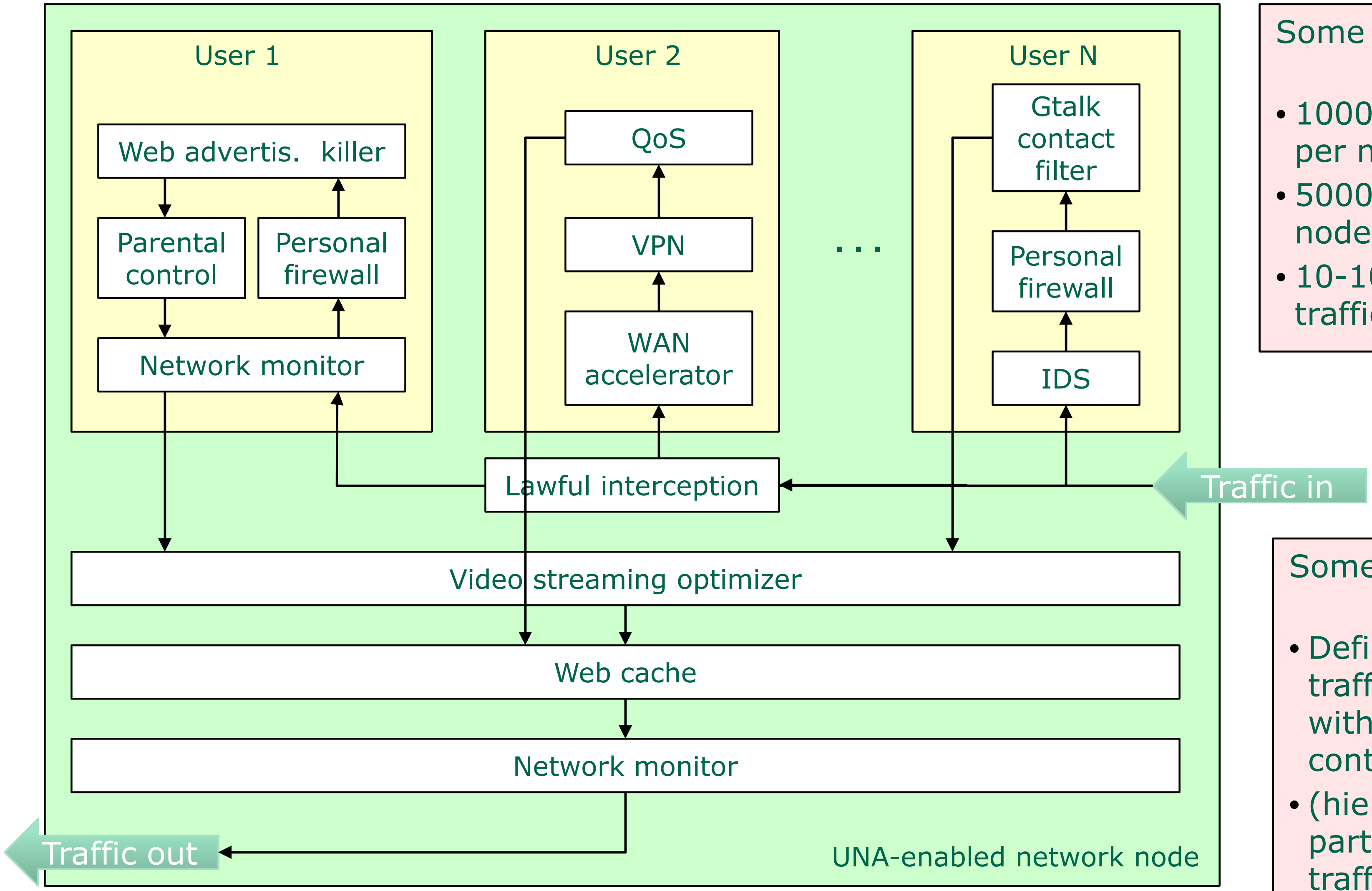
"Keep data as is, and send it on a custom network"

↓

Data plane approach

"Transform data on the fly, and send it on the usual network"

User-defined Network Applications



Some numbers

- 1000+ users per node
- 5000+ apps per node
- 10-100 Gbps traffic

Some challenges

- Define the traffic flow within each container (user)
- (hierarchical) partition of the traffic

Client-based or network-based UNAs?

- Two possibilities for most network applications
  - Bump in the stack in each user client
  - Appliance in the network

- Network based, for the same reason we’re going to the cloud
  - Same network behavior independently from the client we use (smartphone, laptop, public kiosk)
    - My kids do no longer have the “personal computer”… everything that connects to the Internet is fine for them
  - Saving resources on clients
  - Possibility to optimize the application if runs in the network domain
    - E.g., content delivery, web cache

Challenges

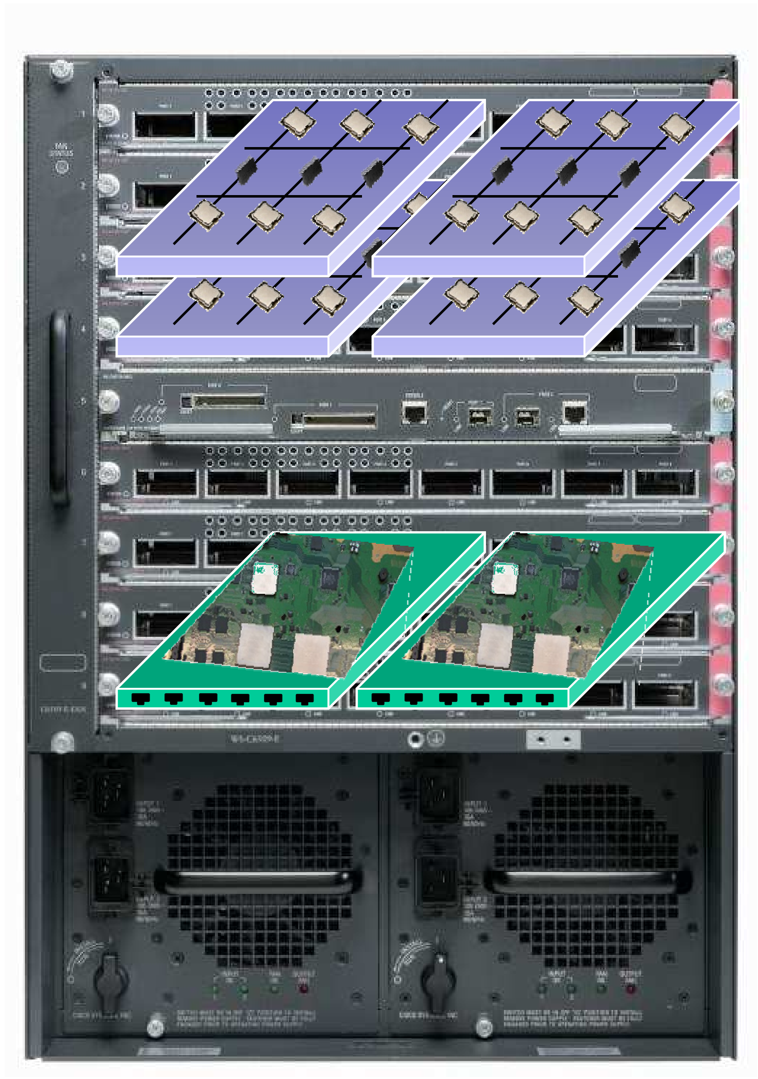
- Network (hierarchical) virtualization
  - Multiple network slices (e.g., containers for users, etc)
  - Are these slices one inside the other or do we have to foresee more complex topologies?
  - FlowVisor can provide us the foundation for this feature
- Application container
  - Scalability (many containers)
  - Isolation (CPU/resource protection)
  - Efficiency (hardware speedup?)
  - Portability (live migration)
  - Easy to code (in the end, users have to create apps)
  - Light VMs? Java?

Conclusions

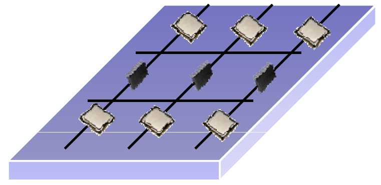
- OpenFlow (1.2) is not production ready
- SDN is probably going to happen
- Huge impact on traditional network vendors
- Huge impact on traditional ISP as well
- A lot of money around
  - Financial, datacenter, enterprise
  - Probably not limited to those environments in the future

Thanks for your attention!

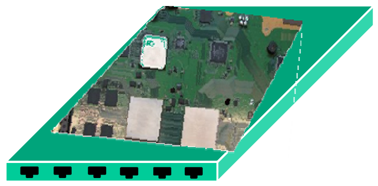
Network computing (1)



The new generation network node



Processing linecards



Interface linecards

Rumors of some new startup in this area

Network computing (2)

- High speed, partially programmable data path
- Linecards with several "general-purpose" CPUs
  - Local memory + global (shared) memory
  - Maybe some coprocessors for function-specific speedups
- Very high speed interconnect between CPUs
  - Don't care where my data is
  - Seamless job migration from one core to another
- Intrinsically scalable software
  - Re-define network algorithms in a "map-and-reduce" fashion?

Performance

Flexibility

Scalability

Upgradeability

Of course, the hardware needs to evolve, but the software is the real challenge!