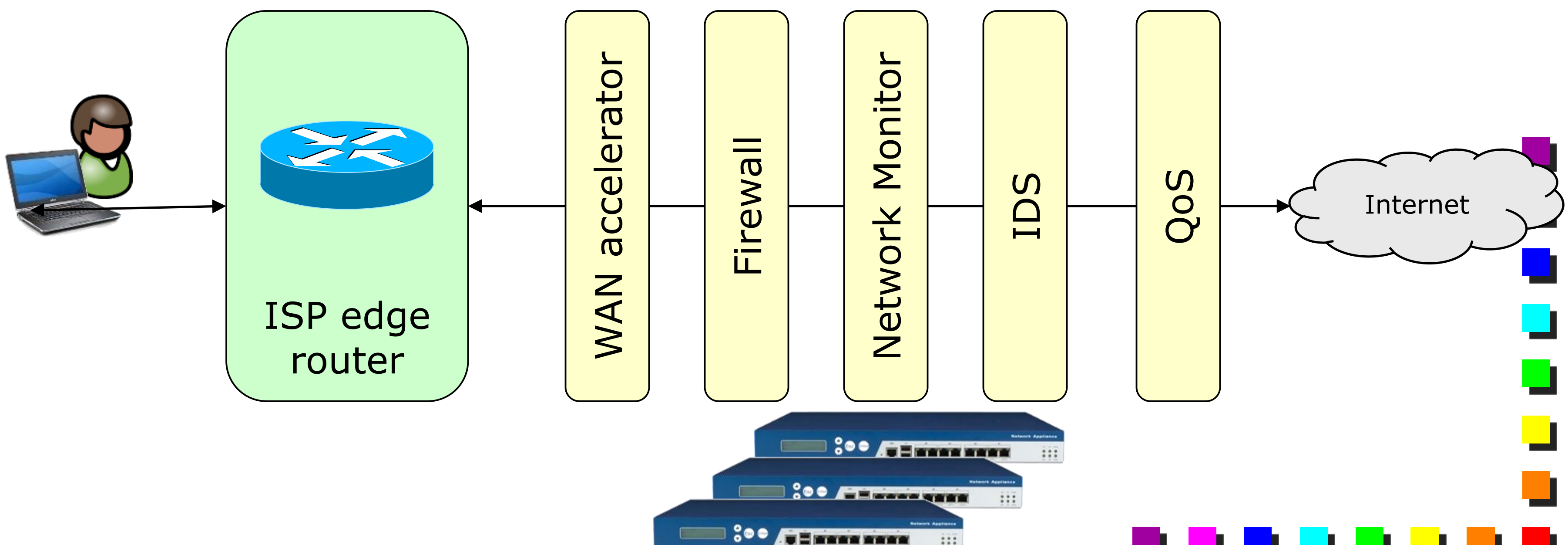
Network Functions Virtualization

Fulvio Risso

Politecnico di Torino

Service Function Chaining

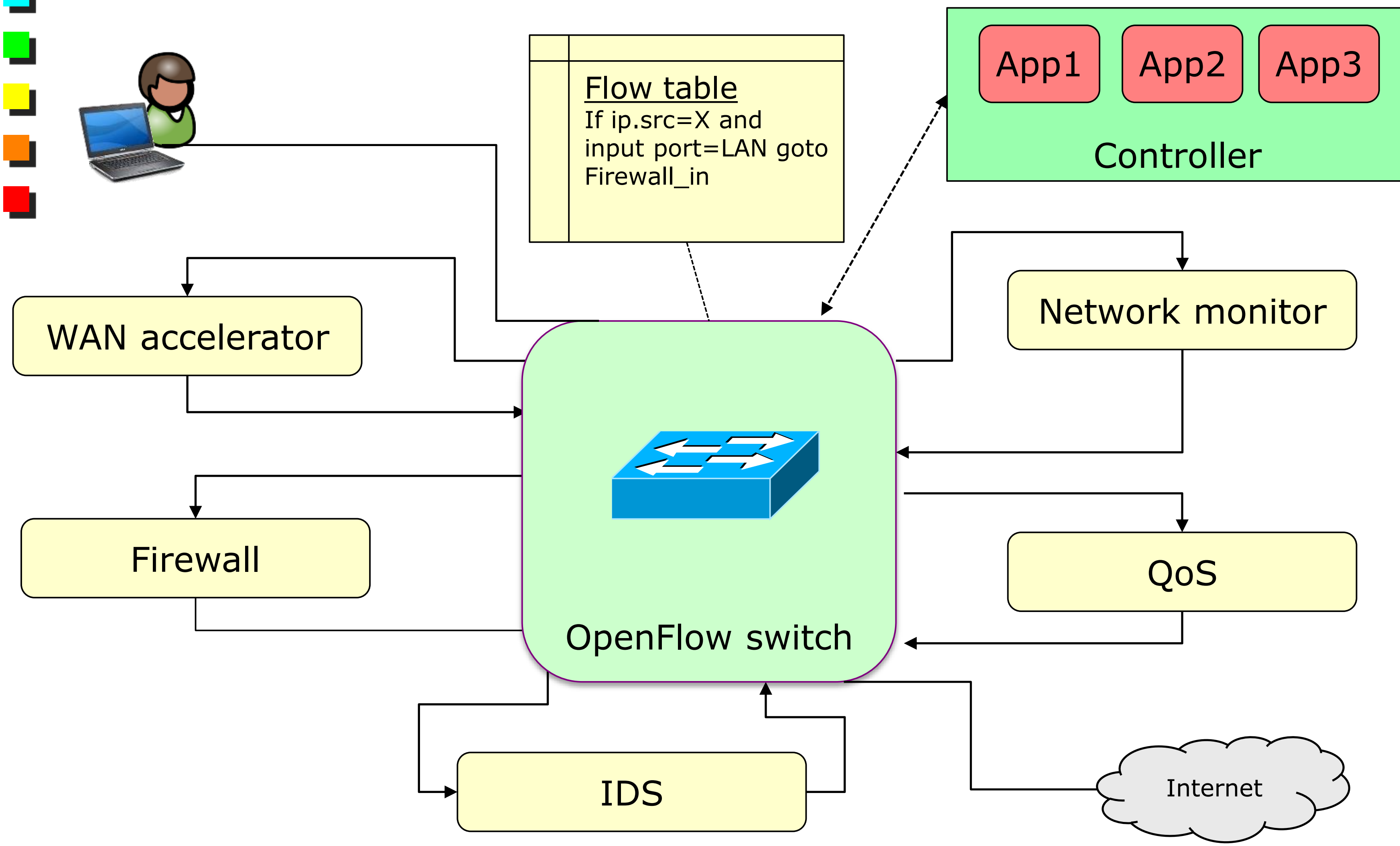
- Often, particularly at the edge of the network, we need to chain **different dedicated hardware appliances** to provide added-value services
- This is what is called a **chain** of **network functions**



Several (practical) problems with SFC

- Hardware resources not used at best
  - Some appliances may sustain an heavy load, while other may be almost unloaded and we are not able to share the available hardware resources (e.g., CPU, memory) between different services
- Service disruption when modifying the service chain
  - Each time we add/remove a middlebox, we have to disrupt the service
- Not easy to differentiate services among tenants
  - What about it a tenant buys a "secure access to the Internet", but other don't? How can we avoid that the traffic of the second tenant goes through the firewall as well?
    - This requires the firewall to support explicit configuration of the user privileges (i.e., per-application configuration)

Service Function Chaining with SDN (1)



Service Function Chaining with SDN (2)

- An OpenFlow switch can be installed to connect all boxes together
- OpenFlow rules can be used to steer the traffic from each user to the proper set of services
  - Rules can be either pre-provisioned, or provisioned on demand (e.g., user logs-in, and the controller instantiates the proper rules for this user, valid only for the duration of the user session)
- The controller can be installed locally to the machines
  - This looks like a nice setup for an edge POP of a telecom operator

SFC with SDN: characteristics

- Agility in provisioning new services
  - Install the box, then “routing” is done via software instead of connecting the box to the other with physical wires
- Maintenance and reliability
  - Cabling is done once
- Different customers can have different service chains
  - “routing” done via software, even possible to change its decisions based on other parameters (e.g., application layer content)

- Still difficult to partition a physical appliance among different tenants
  - Many small business customers, each asking for a firewall service

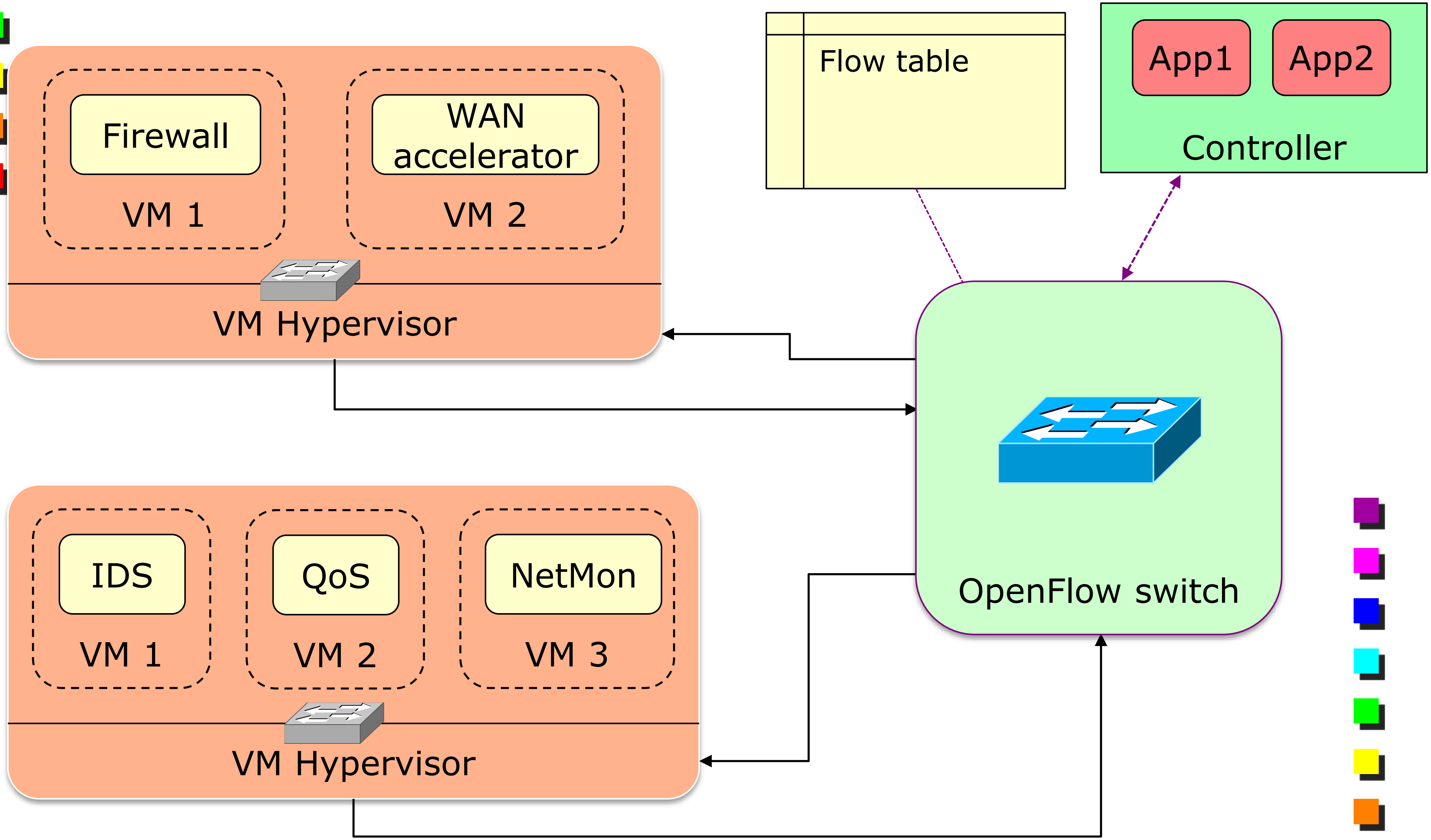
Network Functions Virtualization (1)

- Four main components:
  - Fast standard hardware (e.g. Intel servers)
    - Commercial-off-the-shelf (COTS) hardware
  - Software-based network functions
    - Network functions, previously running on a dedicated appliance, now become a software image, running on a standard server
  - Computing virtualization (e.g., Linux KVM)
    - All advantages of virtualization (quick provisioning, scalability, mobility, reduced CapEx, reduced OpEx, multitenancy, …)
  - Standard API (i.e., ETSI framework)
- Network Functions Virtualization is the capability to run any **network function** on a **standard hardware**, possibly with the help of **computing virtualization** to achieve an efficient use of resources

Network Functions Virtualization (2)

- Two possible deployment scenario for NFV services
- **Software based Devices**
  - Instead of having the service in a dedicated appliance, the service runs on standard hardware
    - E.g., Routers, Firewalls, Broadband Network Gateways (BNG) in a *white box* implementation
  - Often, virtualization is not used in this case (or used internally, without allowing the server to be integrated in the datacenter of the provider)
  - Commonly created through the use of DPDK-based functions
- **Function Modules**
  - Refers to both data plane and control plane
    - E.g., DHCP, NAT, Rate Limiting, etc.
  - Often they come as pure software packages

Service functions chaining with NFV



Advantages of NFV

- **1. Virtualization**: use resources without worrying about where it is physically located, how much it is, how it is organized, etc.
- **2. Orchestration**: manage thousands of devices
- **3. Programmable**: can change the behavior on the fly
- **4. Dynamic Scaling**: can adapt to different workloads
- **5. Automation**
- **6. Visibility**: Monitor resources, connectivity
- **7. Performance**: Optimize network device utilization
- **8. Multi-tenancy**
- **9. Service Integration**
- **10. Openness**: Full choice of service modules

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

Chaining vs general services (in NFV)

- Chaining usually refers to a service that is made up of a **stack** of modules
- Services are not always stackable
  - E.g., DHCP, DNS, web services need to operate in a LAN
  - How to model a LAN with a chain?
- Hence, NFV needs to be more flexible than just support chains

NFV and cloud

- NFV can be seen as a way to bring network services in the world of cloud technologies
  - Cloud: hosts web servers, database servers, big data applications, etc.
  - NFV: adds also network services to that picture
- Although apparently NFV can be realized mostly with existing technologies, in practice:
  - Cloud frameworks may not support well traffic steering, although they support well traditional LAN services
  - Network services are I/O intensive, while traditional cloud services are mostly CPU intensive
    - Some technologies need to be tuned (and/or modified) to support the high amount of network traffic that is generated by network services

NFV and SDN (1)

- NFV is about computing, SDN is about network paths
- NFV requires SDN for flexible traffic steering
  - Although, a point-to-point Ethernet is often enough for most of the purposes
- NFV and SDN are complementary
  - One does not depend upon the other
  - You can do SDN only, NFV only, or SDN and NFV
- A lot of discussions about SDN, not much debate about NFV

NFV and SDN (2)

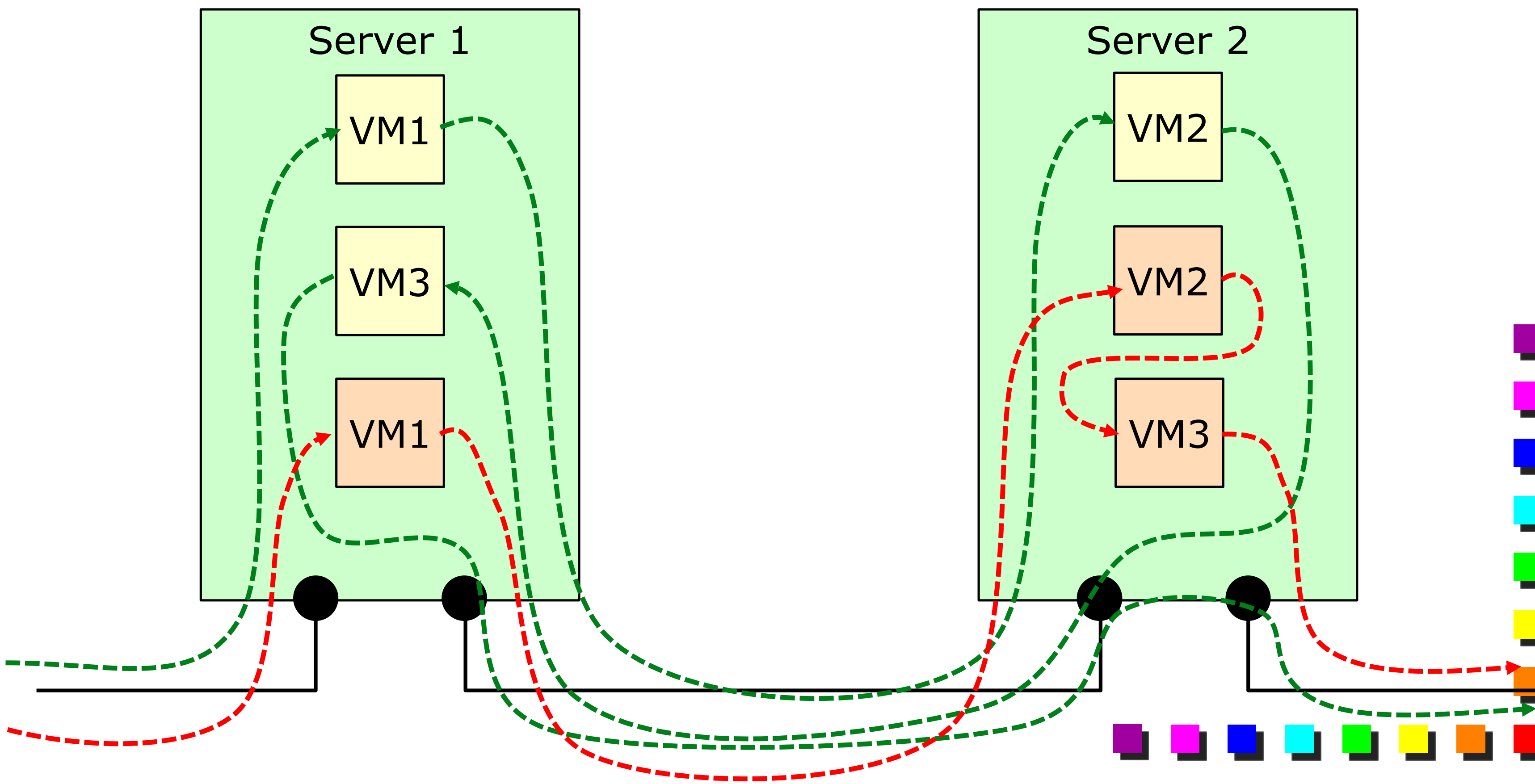
- Both have similar goals but approaches are very different
  - SDN needs new interfaces, control modules, applications must re-engineered
  - NFV requires moving network applications from dedicated hardware to virtual images on standard hardware

  - SDN heavily leverages accelerated hardware (the hardware switch)
  - NFV can hardly take advantages (right now) of accelerated hardware

  - Hence, SDN can be potentially much more efficient than NFV
  - NFV is currently much more flexible (in terms of possible supported applications) than SDN

VNF and network traffic (1)

- In theory, VMs can be deployed based on the resources that are available on the data center
- In practice, this may lead to very un-optimized paths

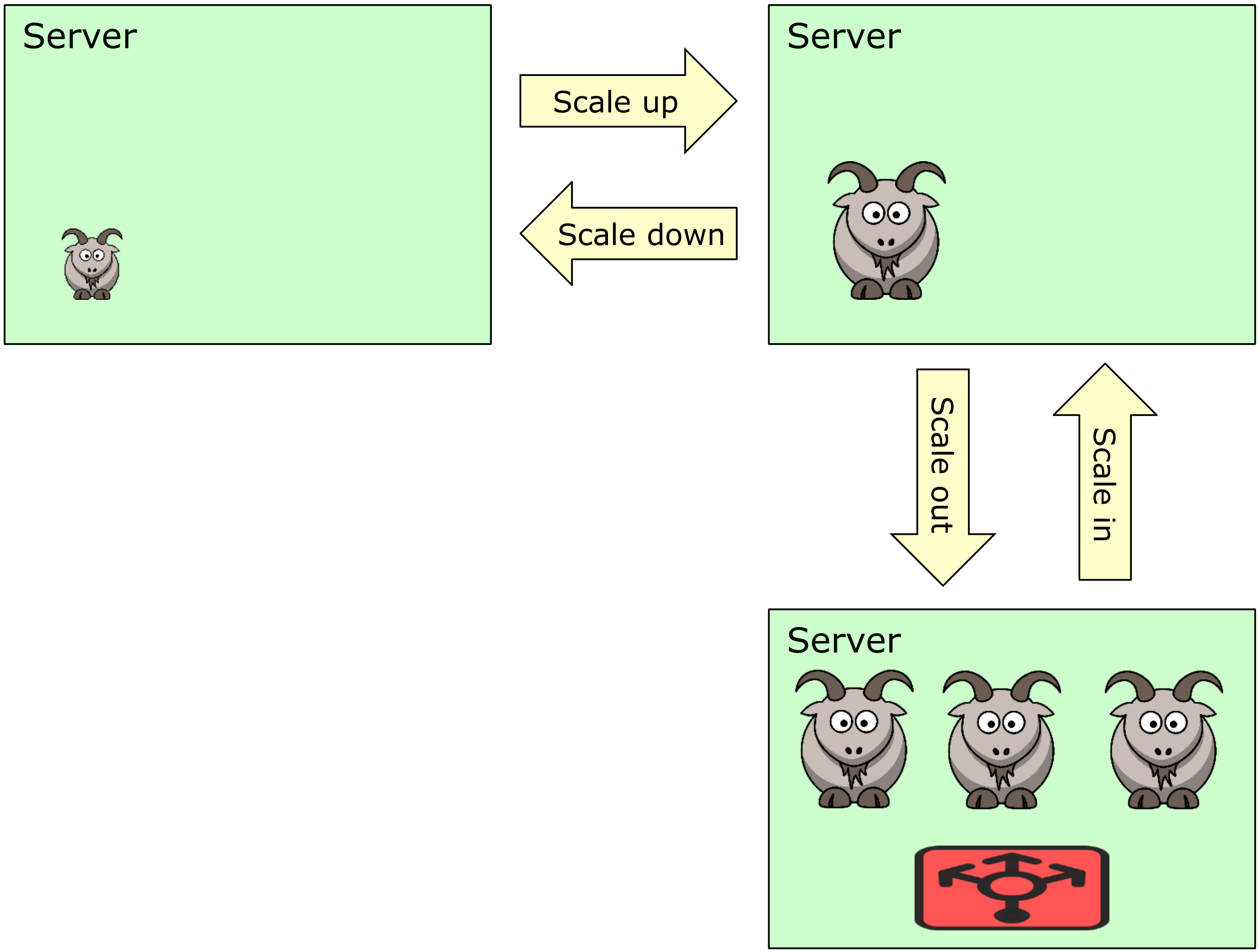


VNF and network traffic (2)

- NFV may have a huge impact on the traffic of your datacenter
  - NFV can generate a huge amount of traffic on the **network**
  - NFV can generate a huge amount of traffic **inside each server** as well
    - Packets may travel several times back and forth to the switch
    - We may need to optimize the computing technologies to reduce the load (e.g., SR-IOV, VirtIO, Shared memory)

- We need to predict the amount of traffic that is generated in the datacenter to avoid troubles

Scaling: definitions



NFV and scalability

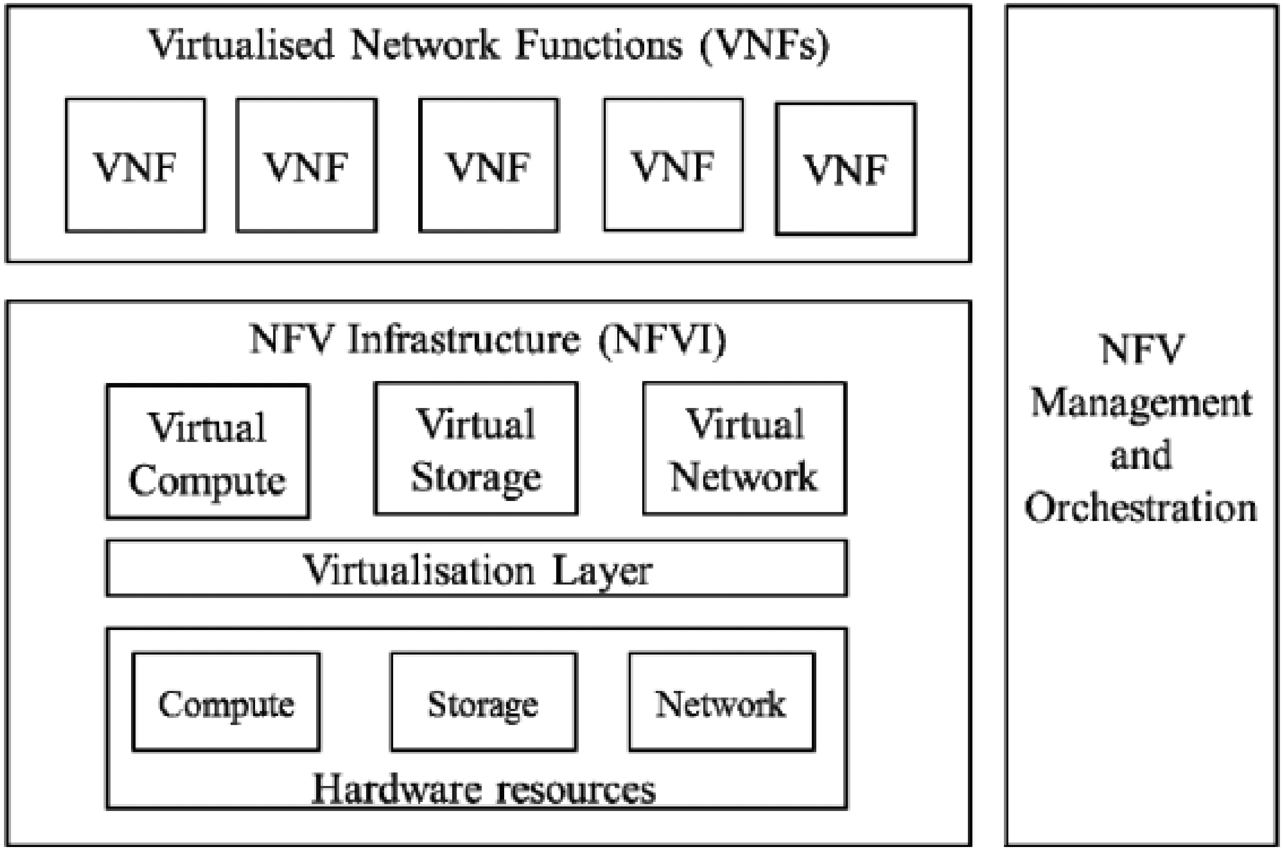
- VMs are good when we need to **consolidate** many (tiny) application instances on the same physical servers
- VMs are not very good when an application requires so many resources that even a **fully dedicated server is not enough** to deliver the service
- In the latter case, we have mainly two options:
  - Add a load balancer (e.g., using SDN) in front of the different instances and make sure that they can operate independently
    - Most applications work per-TCP-session, so if we split traffic this way, the application can operate properly
  - Modify the application in order to make it distributed
    - Application can have two set of variables: **local** ones (do not need to be in sync with the other instances) and **global** ones (each modification has to be propagated to all the instances)

The ETSI NFV model

ETSI NFV ISG

- ETSI NFV Industry Specification Group (ISG): define the requirements, architectural framework, interfaces for NFV
  - http://www.etsi.org/technologies-clusters/technologies/nfv
- Different Working Groups / Expert groups
  - Architecture for the virtualization infrastructure
  - Management and orchestration
  - Software architecture
  - Reliability and availability, resilience and fault tolerance
  - Public demonstrations and Proof of Concept
  - Performance
  - Security

High-level NFV framework



ETSI GS NFV 002 V1.2.1 (2014-12) Network Functions Virtualization (NFV); Architectural Framework

NFV terminology (1)

- **Network Function (NF): f**unctional building block with a well defined interfaces and well defined functional behavior
- **Virtualized Network Function (VNF)**: software implementation of NF that can be deployed in a virtualized infrastructure
- **VNF Set**: connectivity between VNFs is not specified, e.g., residential gateways
- **VNF Forwarding Graph**: service chain when network connectivity order is important, e.g., firewall, NAT, load balancer
- **NFV Infrastructure (NFVI)**: hardware and software required to deploy, manage and execute VNFs including computation, networking, and storage

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

NFV terminology (2)

- **NFVI Point of Presence (PoP)**: location of NFVI
- **NFVI-PoP Network**: internal network
- **Transport Network**: network connecting a PoP to other PoPs or external networks
- **VNF Manager**: VNF lifecycle management e.g., instantiation, update, scaling, query, monitoring, fault diagnosis, healing, termination
- **Virtualized Infrastructure Manager**: management of computing, storage, network, software resources
- **Network Service**: a composition of network functions and defined by its functional and behavioral specification
- **NFV Service**: a network services using NFs with at least one VNF

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

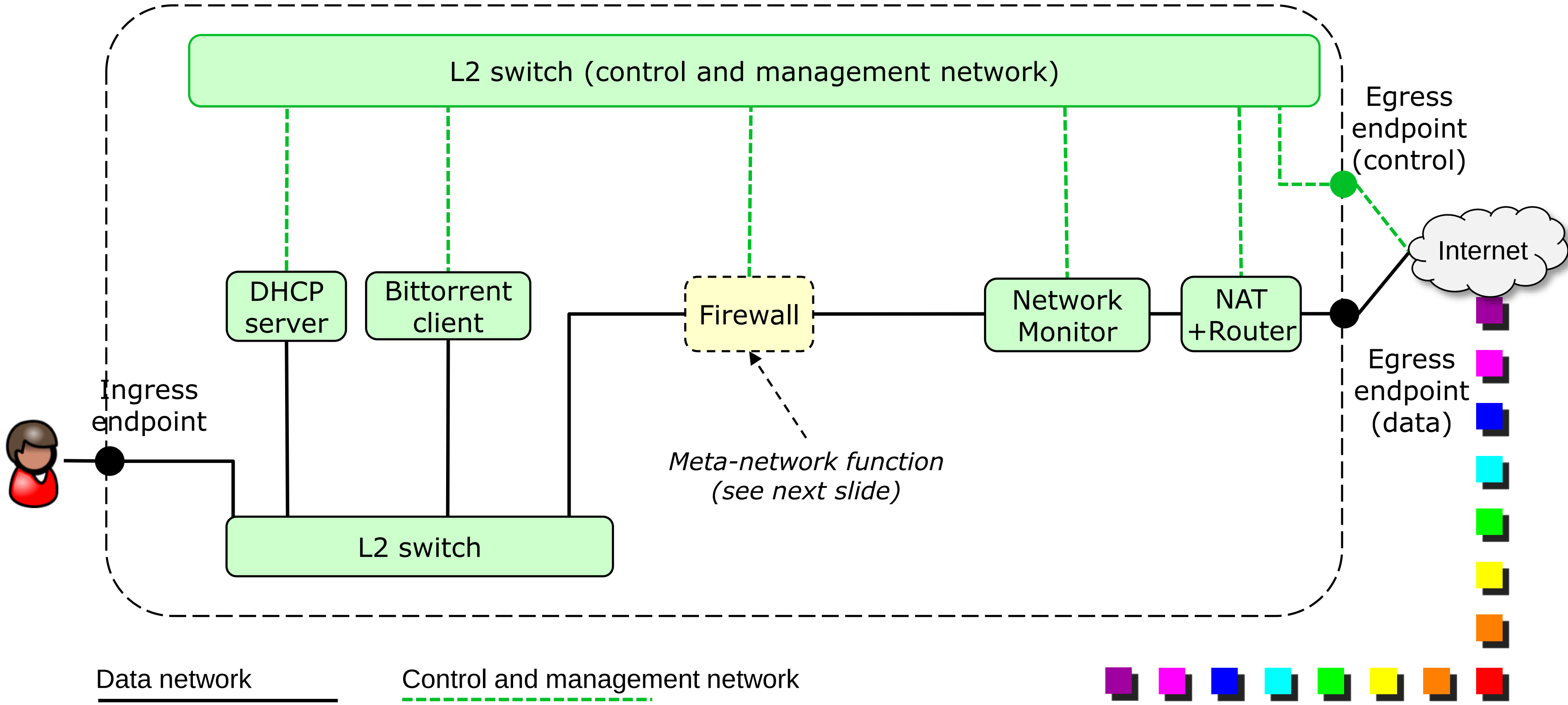
NFV terminology (3)

- **User Service**: services offered to end users / customers / subscribers
- **Deployment Behavior**: NFVI resources required by a VNF, e.g., number of VMs, memory, disk, images, bandwidth, latency
- **Operational Behavior**: VNF instance topology and lifecycle operations, e.g., start, stop, pause, migration, …
- **VNF Descriptor**: deployment behavior + operational behavior
- **NFV Orchestrator**: automates the deployment, operation, management, coordination of VNFs and NFVI
- **VNF Forwarding Graph**: connection topology of various NFs of which at least one is a VNF

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

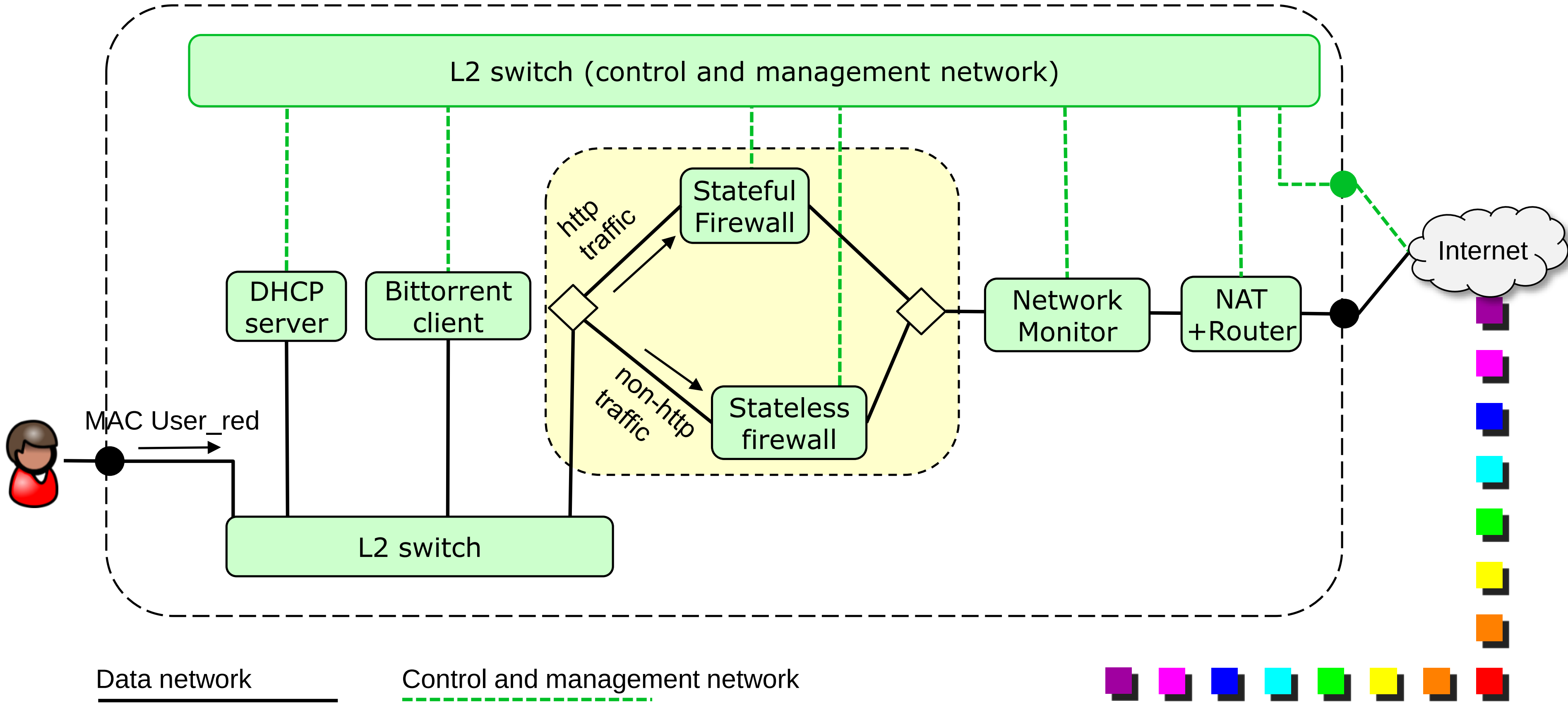
Forwarding graph

- High-level representation of the service in terms of functional blocks and their connections, similar to a service chain
- Example of a complex service

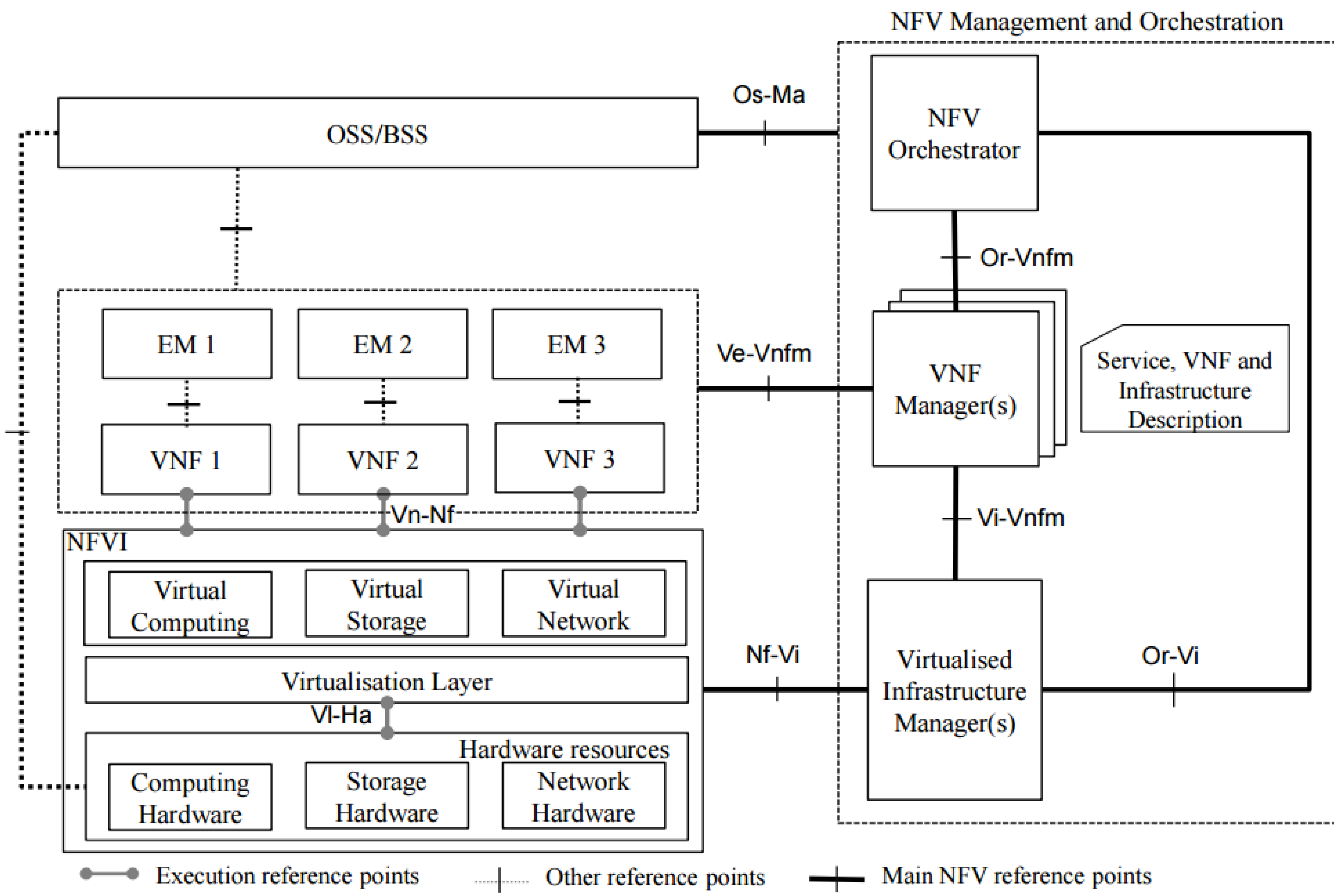


Forwarding graph: hierarchical decomposition

- Services can be hierarchically decomposed in smaller building blocks



NFV Reference Architectural Framework



ETSI GS NFV 002 V1.2.1 (2014-12) Network Functions Virtualization (NFV); Architectural Framework

NFV Reference Points

- Reference Point: points for inter-module specification
  - 1. Virtualization Layer-Hardware Resources (**VI-Ha**)
  - 2. VNF – NFVI (**Vn-Nf**)
  - 3. Orchestrator – VNF Manager (**Or-Vnfm**)
  - 4. Virtualized Infrastructure Manager – VNF Manager (**Vi-Vnfm**)
  - 5. Orchestrator – Virtualized Infrastructure Manager (**Or-Vi**)
  - 6. NFVI-Virtualized Infrastructure Manager (**Nf-Vi**)
  - 7. Operation Support System (OSS)/Business Support Systems (BSS) – NFV Management and Orchestration (**Os-Ma**)
  - 8. VNF/ Element Management System (EMS) – VNF Manager (**Ve-Vnfm**)
  - 9. Service, VNF and Infrastructure Description – NFV Management and Orchestration (**Se-Ma**): VNF Deployment template, VNF Forwarding Graph, service-related information, NFV infrastructure information

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

NFV Framework Requirements

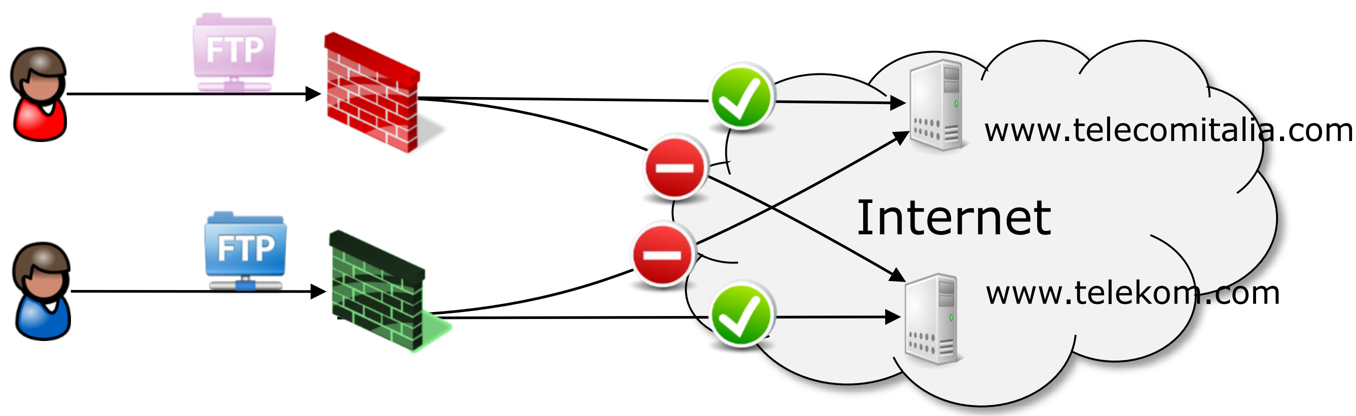
- **1. General**: partial or full virtualization, predictable performance
- **2. Portability**: decoupled from underlying infrastructure
- **3. Performance**: as described and facilities to monitor
- **4. Elasticity**: scalable to meet SLAs; movable to other servers
- **5. Resiliency**: be able to recreate after failure; specified packet loss rate, calls drops, time to recover, etc.
- **6. Security**: role-based authorization, authentication
- **7. Service Continuity**: seamless or non-seamless continuity after failures or migration
- **8. Service Assurance**: time stamp and forward copies of packets for fault detection
- **9. Energy Efficiency Requirements**: should be possible to put a subset of VNF in a power conserving sleep state
- **10. Transition**: coexistence with legacy and interoperability among multi-vendor implementations
- **11. Service Models**: operators may use NFV infrastructure operated by other operators

Partially adapted from http://www.cse.wustl.edu/~jain/cse570-13/m_17nfv.htm

An example of a complex service using NFV

Service overview

- Different users experiment a different and customized network service based on their credentials
  - User RED deploys a RED_NF-FG that includes a RED_SFTP server and a RED_FIREWALL
  - User BLUE deploys a BLUE_NF-FG that includes a BLUE_SFTP server and a BLUE_FIREWALL
- Network operator sets up some additional services in the network, active on all users

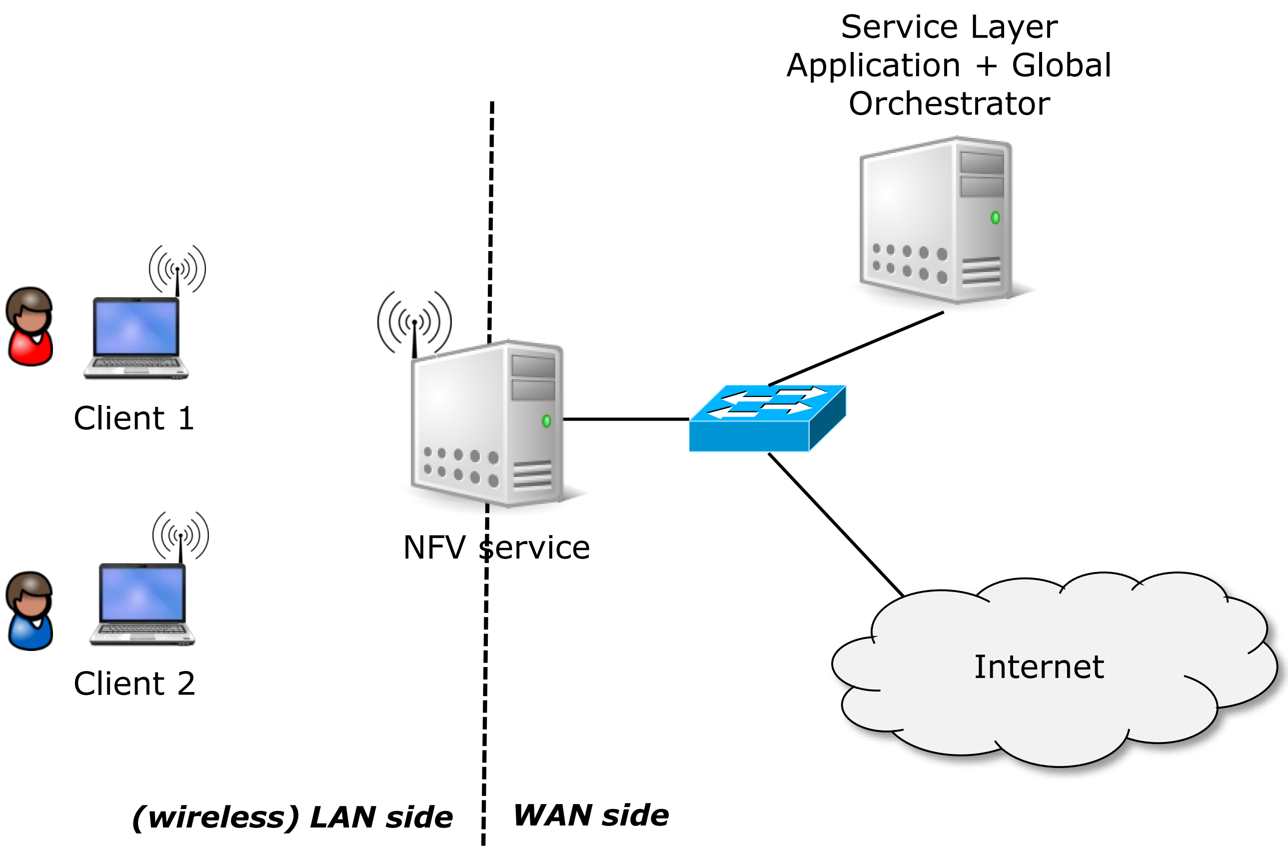


Demo details

- Each **end user** is associated to a service graph operating on his own traffic
  - I.e., packets coming from / sent to his **MAC** address
- The **network provider** forces the traffic of each user to cross an additional service graph under his control
  - Provides basic network services needed to connect to the Internet

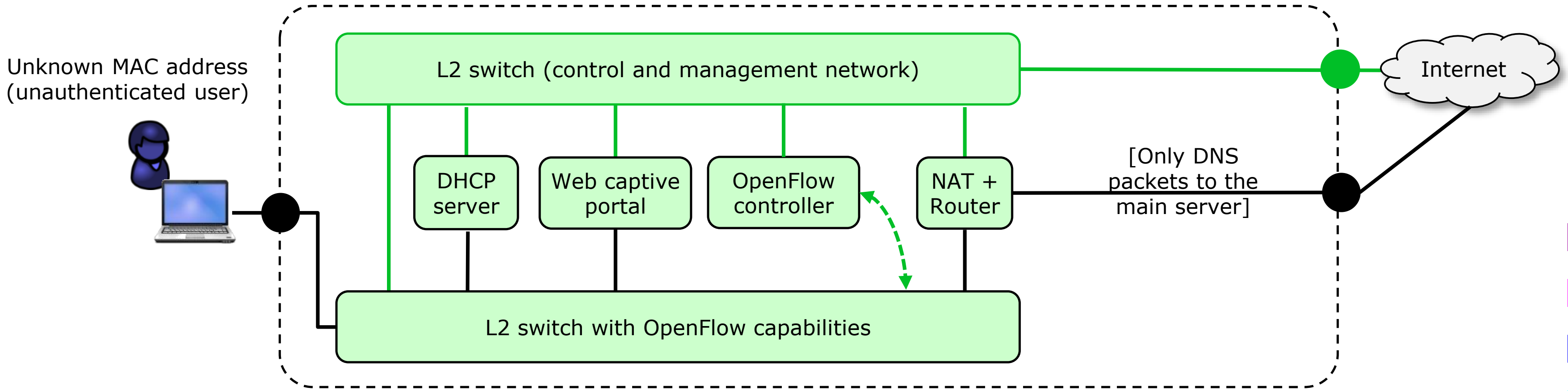
- When the user connects to the network:
  - is authenticated using a particular graph (**Auth_NF-FG**)
  - his own service graph **User_NF-FG** is instantiated on the Universal Node
  - His traffic is forced to cross the **User_NF-FG** first, then the **ISP_NF-FG** before reaching the Internet

Possible physical setup



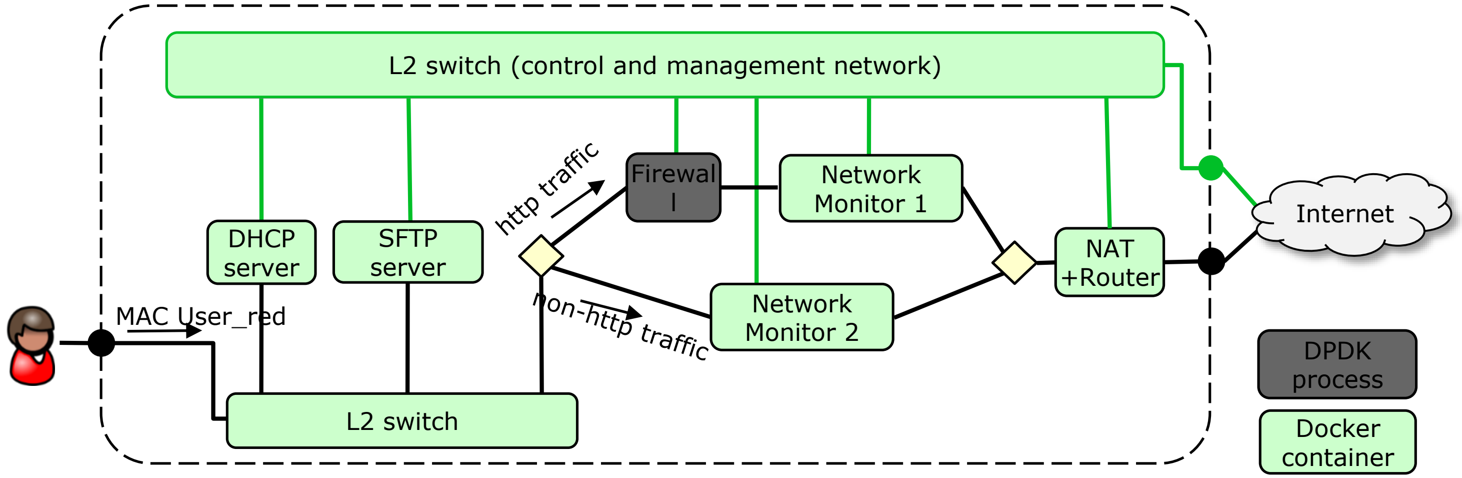
Authentication NF-FG (Auth_NF-FG)

- Handles the traffic generated by unknown devices (unknown MAC addresses)
- Provides a way to authenticate users



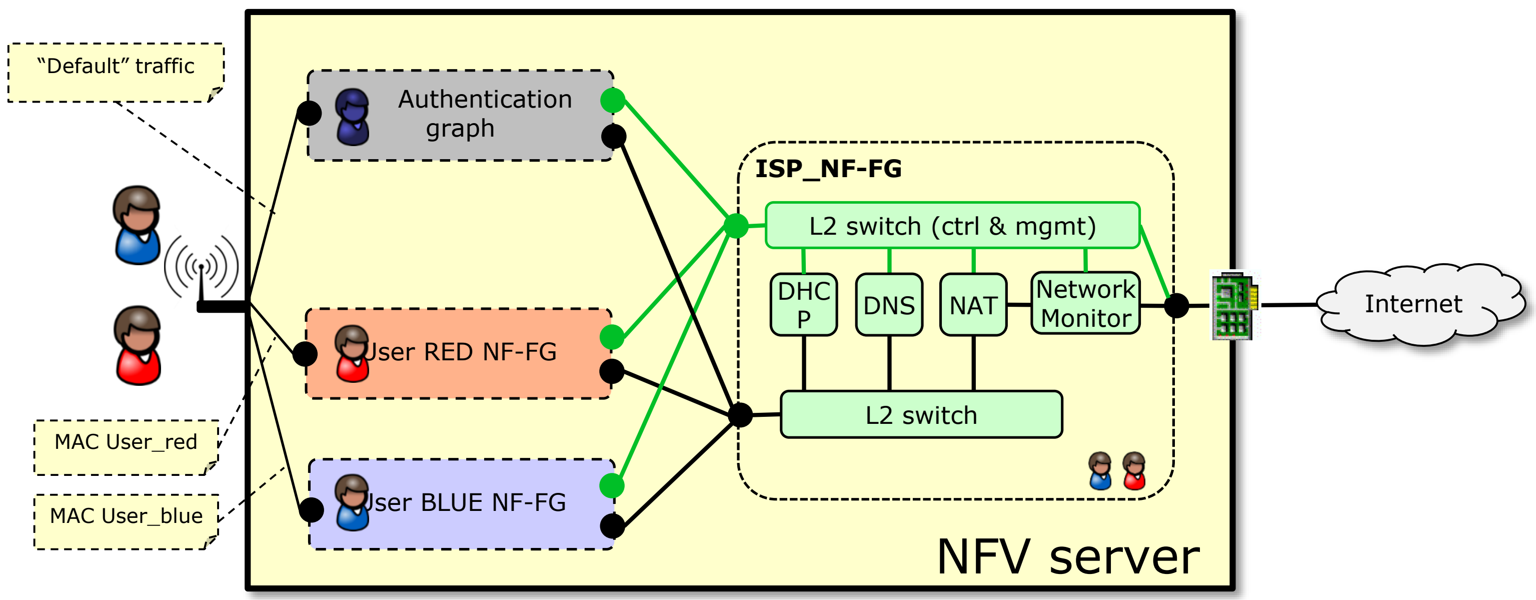
User_NF-FG

- Each user is associated to a specific NF-FG
  - Includes both transparent (e.g., firewall) and non-transparent (e.g., SFTP server) functions



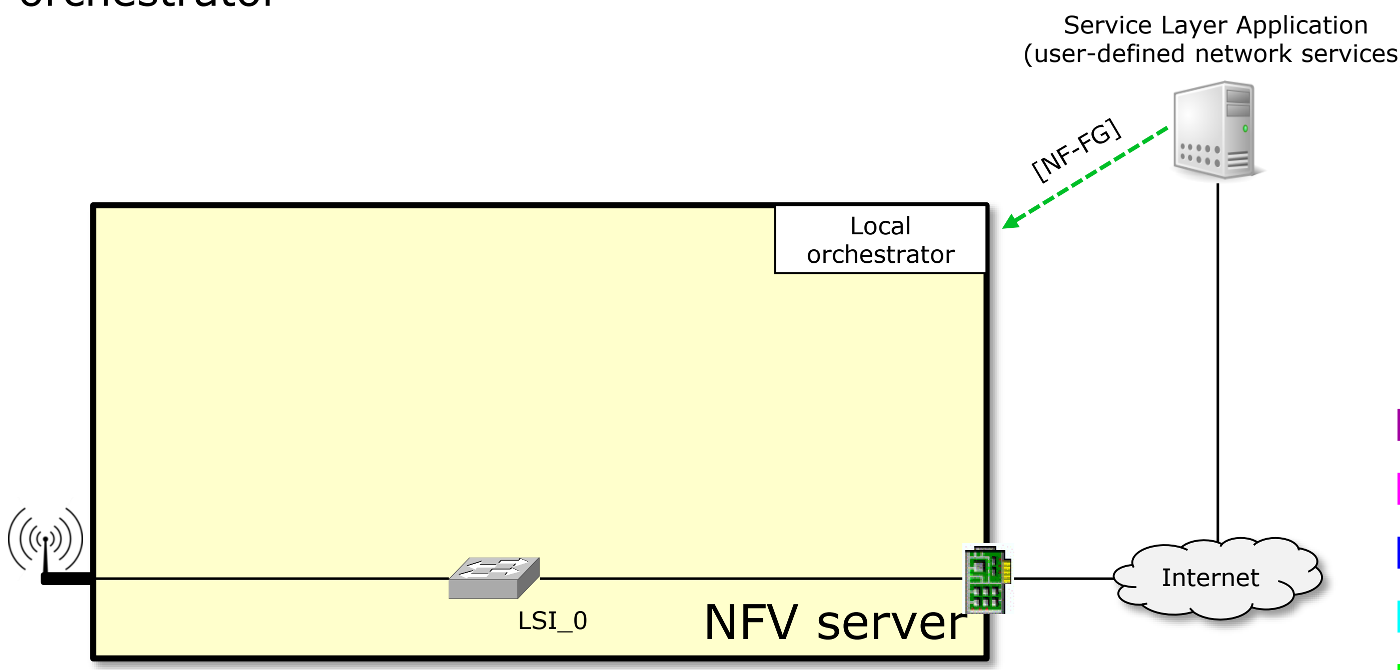
ISP_NF-FG

- Example of a possible set of Network Functions under the control of the ISP



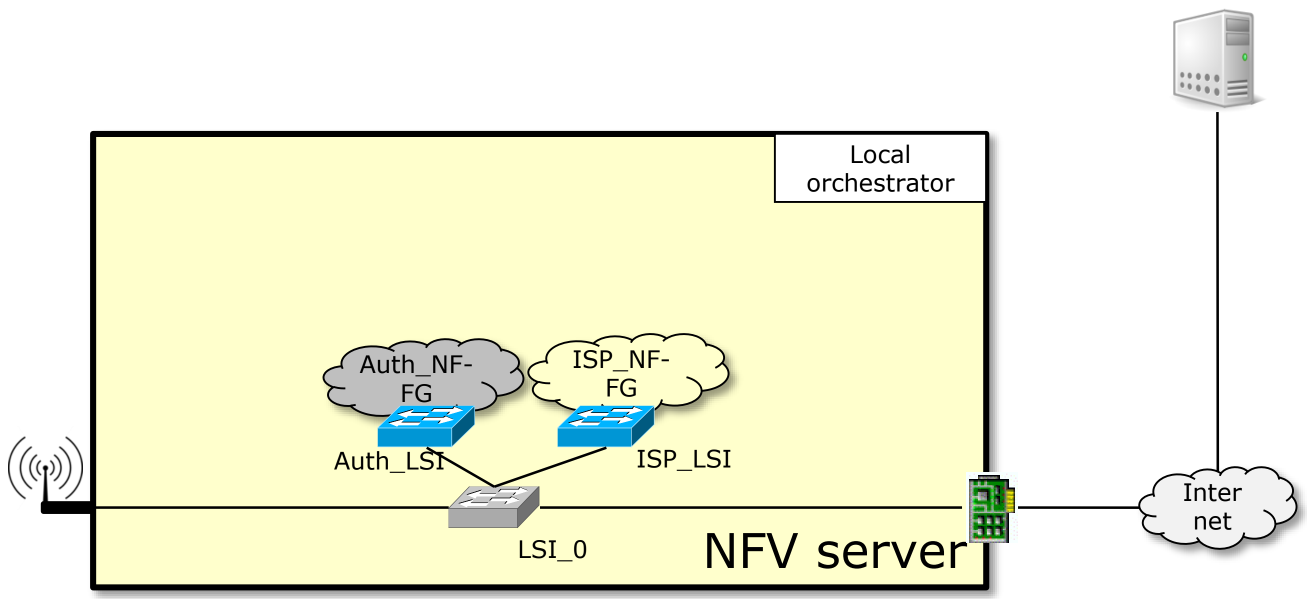
Example step 0: system startup

- Basic software running
  - Softswitch running (LSI0)
  - Local orchestrator



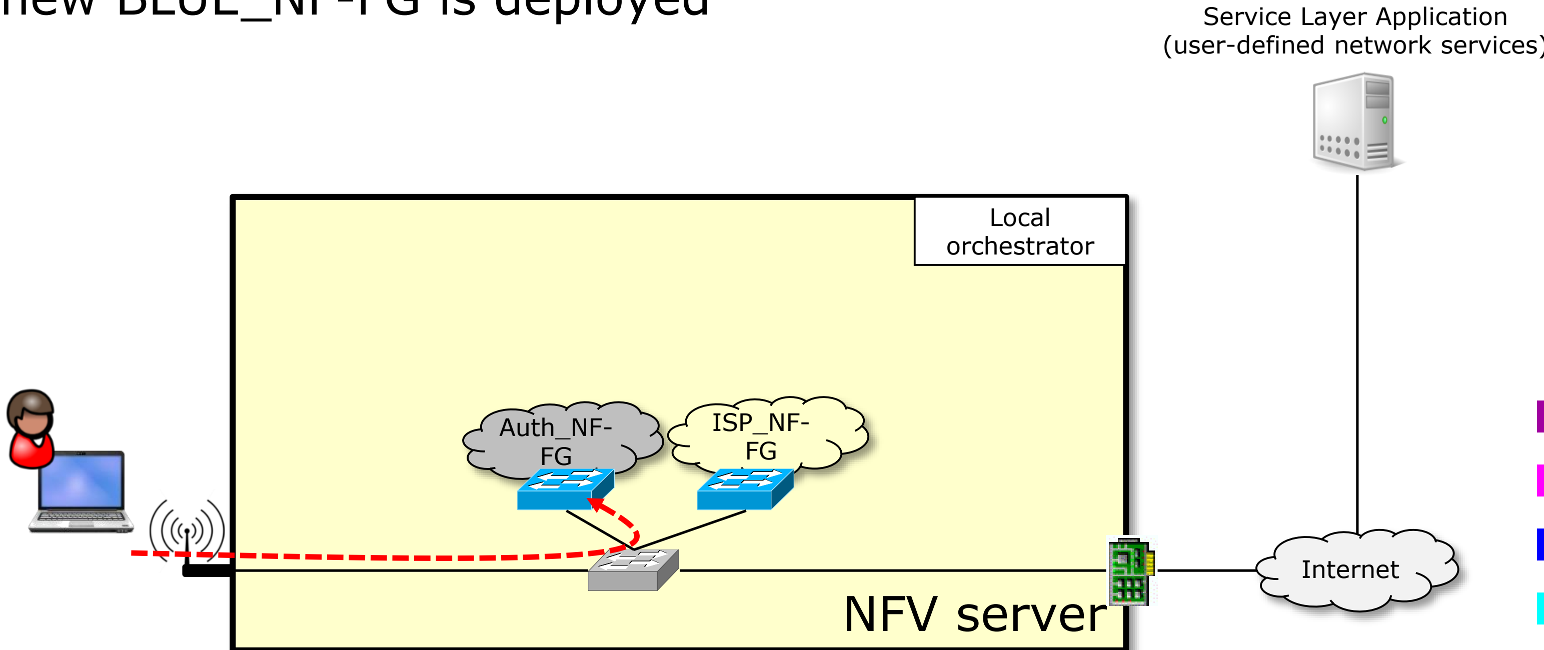
Example step 1: graph startup

- Auth_NF-FG and ISP_NF_FG automatically deployed
- Three Logical Switching Instances (LSI) active
- All incoming (from the WiFi) traffic sent to AuthLSI

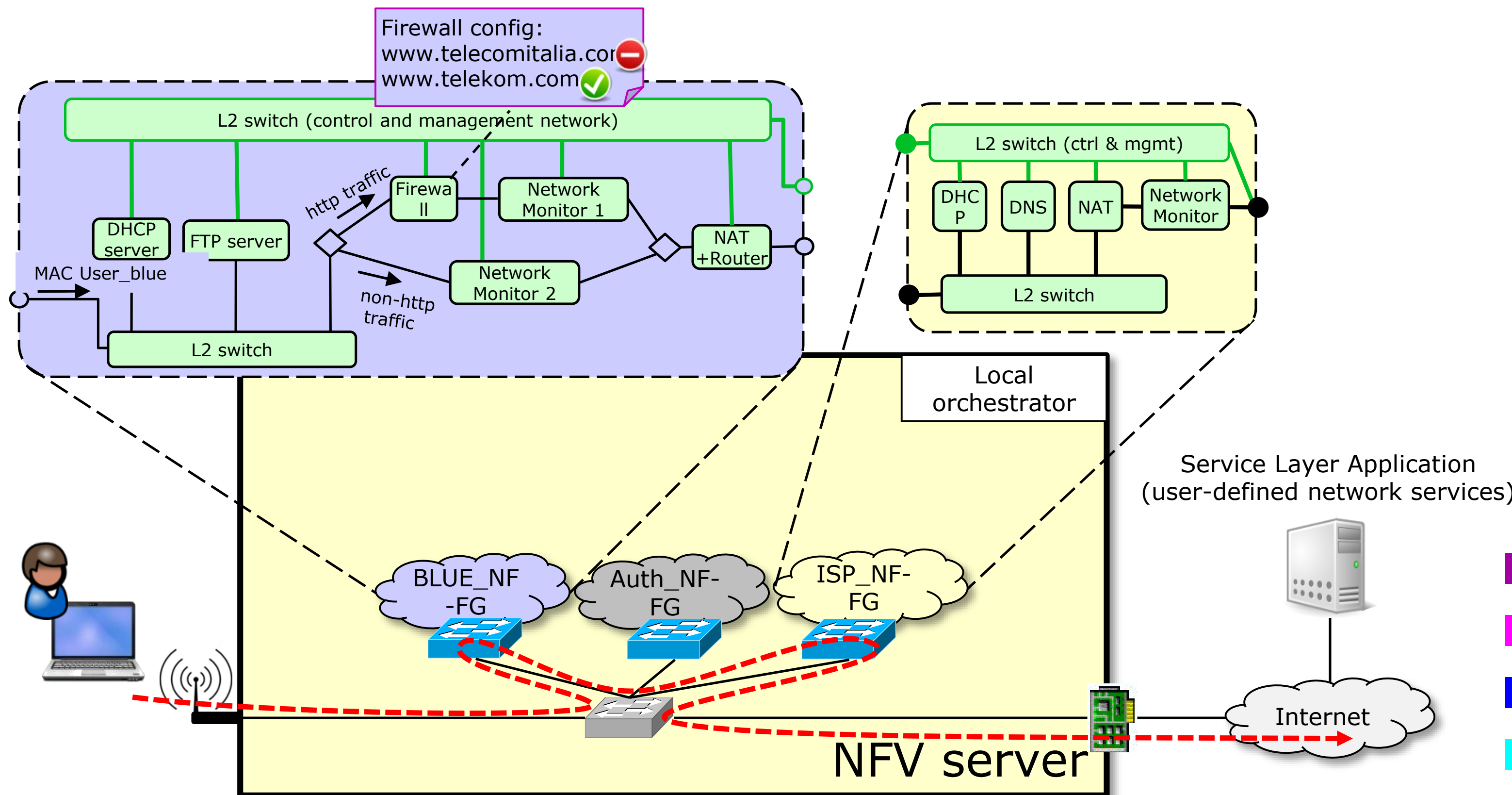


Example step 2: user (BLUE) connects

- User web traffic redirected to a captive portal
- User (BLUE) authenticates
- A new BLUE_NF-FG is deployed

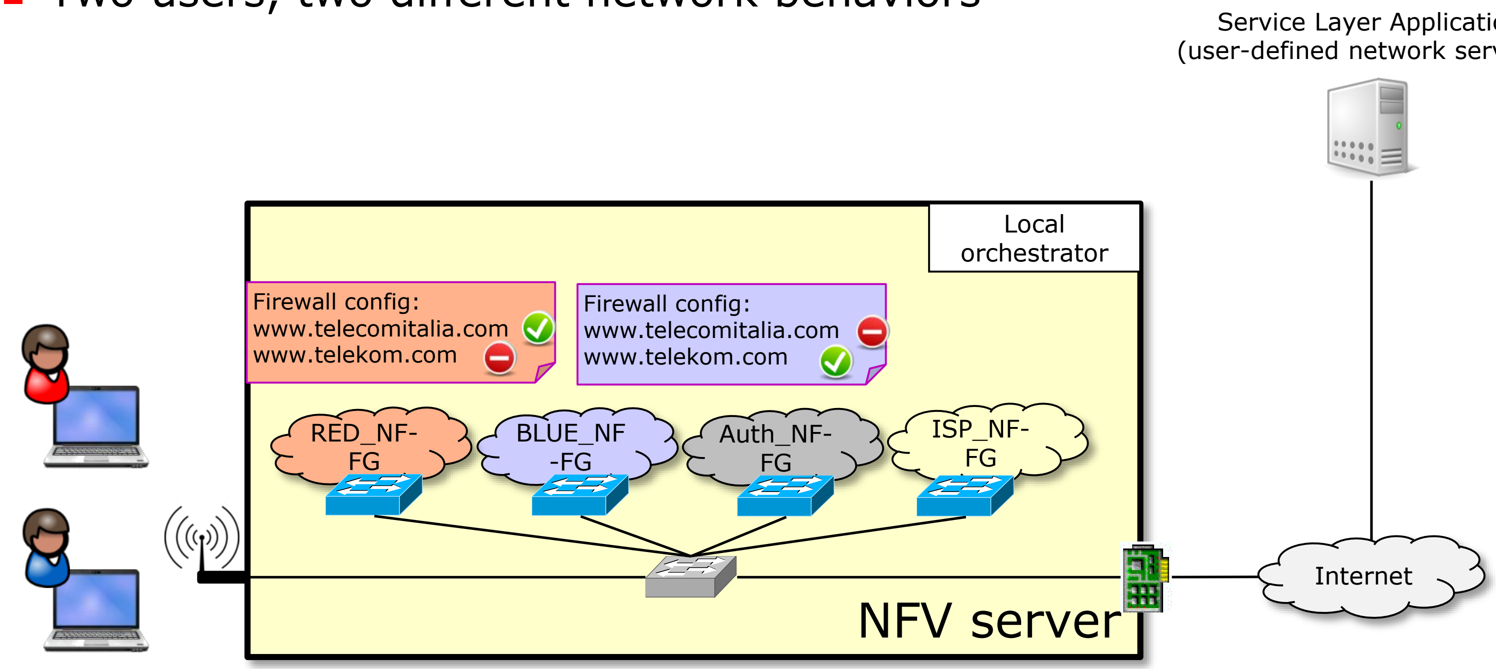


Example step 2b: user BLUE connected



Example step 3: users BLUE and RED

- Logging in User RED from another laptop
  - Two users, two different network behaviors



Conclusions

- Shown the capability of an NFV server to deploy arbitrary network services, starting with a minimal set of components (softswitch, local orchestrator, NF-FG)
  - On-demand deployment of (user-defined) service graphs and run-time modifications of the attaching points
  - Support for a large number of network functions
  - Support for complex and cascading service graphs
  - Support for network functions with different architectures
    - Docker containers, DPDK native processes
  - Support for transparent (e.g., firewall, network monitor) and non-transparent (e.g., SFTP server, DHCP/DNS servers) network functions

Additional content

Future integrated routers for NFV (1)

General purpose processing linecards

Network processing linecards

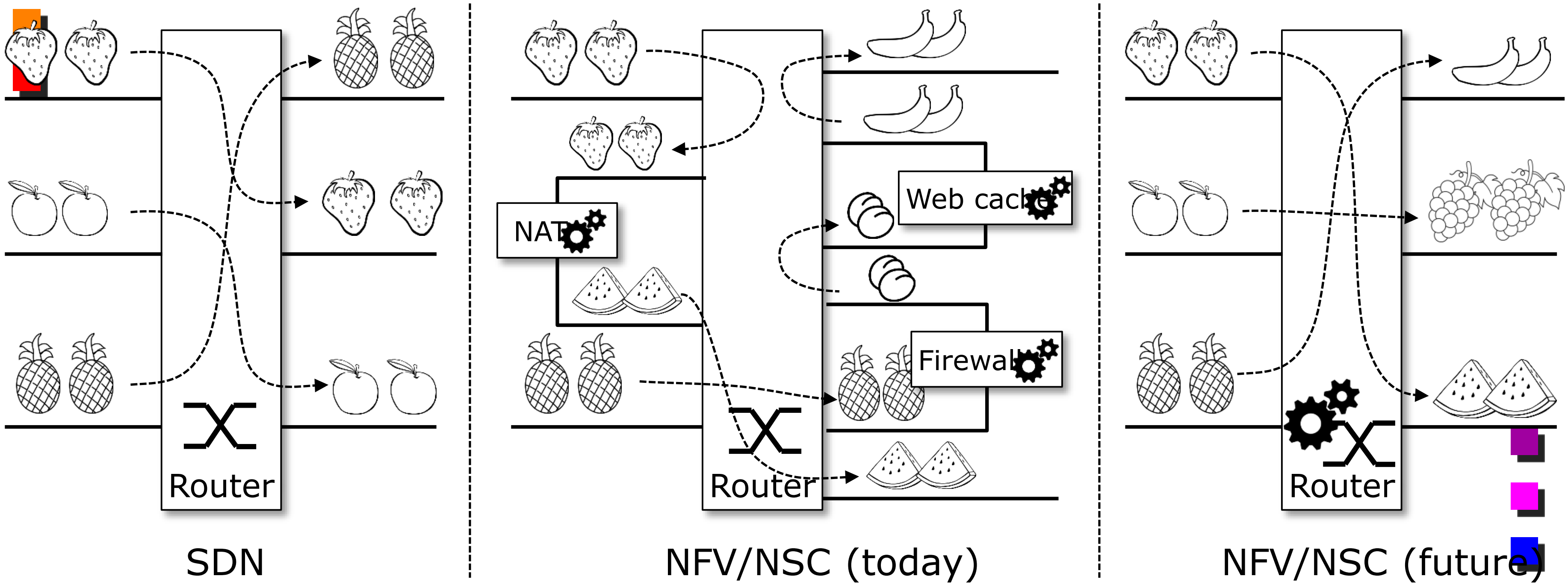
Future integrated routers for NFV (2)

- Routers with general purpose processing + network processing
  - More efficient (possibility to exploit hardware accelerators)
  - More scalable
  - Reduced network traffic to/from the router
- Routers may become "hybrid" platforms supporting different kind of applications
  - From network applications to traditional VMs
- Different hardware accelerators may be available
  - Each application can be mapped on the best processing component
  - Mapping can be done even at finer granularities (at the module level)

Some integrated routers available right now

- 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.
- Not very successful
- Not what we need
  - It was not so different from a router and a server, stacked one on top of the other

SDN vs. NFV/SFC



Slim or fat routers?

- SDN predicts routers are slim

- Slim routers suggest that the network will become a dumb pipe

- Slim routers are good for “low value, low margins” vendors

- Data plane performance requirements (and the relative simplicity of data plane tasks) suggest that (at least edge) routers need to be fat

- Fat routers may suggest that the network will still have advanced processing capabilities
  - Hence, intelligence
  - Hence, value

- Fat routers are good for current network vendors