Virtualização de rede – desafios e tendências de evolução

IEEE ComSoc

Lisboa, 31.10.2013

Jorge Carapinha
jorgec@ptinovacao.pt

O que é Virtualização de Rede?

- ITU-T ("Framework of network virtualization for Future Networks", http://www.itu.int/oth/T3A05000073/en)
  - Network virtualization: A technology that enables the creation of logically isolated network partitions over shared physical network infrastructures so that multiple heterogeneous virtual networks can simultaneously coexist over the shared infrastructures. Also, network virtualization allows the aggregation of multiple resources and makes the aggregated resources appear as a single resource.
- IETF ("Framework for DC Network Virtualization", IETF draft-ietf-nvo3-framework-03.txt)
  - A Virtual Network is a logical abstraction of a physical network that provides L2 or L3 network services to a set of Tenant Systems. A VN is also known as a Closed User Group (CUG)
- Wikipedia
  - In computing, network virtualization is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a virtual network. Network virtualization involves platform virtualization, often combined with resource virtualization.
- Cisco http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns431/ns658/net_qanda0900aecd804a16ae.html:
  - (…)Network Virtualization is the efficient utilization of network resources through logical segmentation of a single physical network. An example of multiple logical networks over a common infrastructure could be different organizational units or departments on a single companywide network.

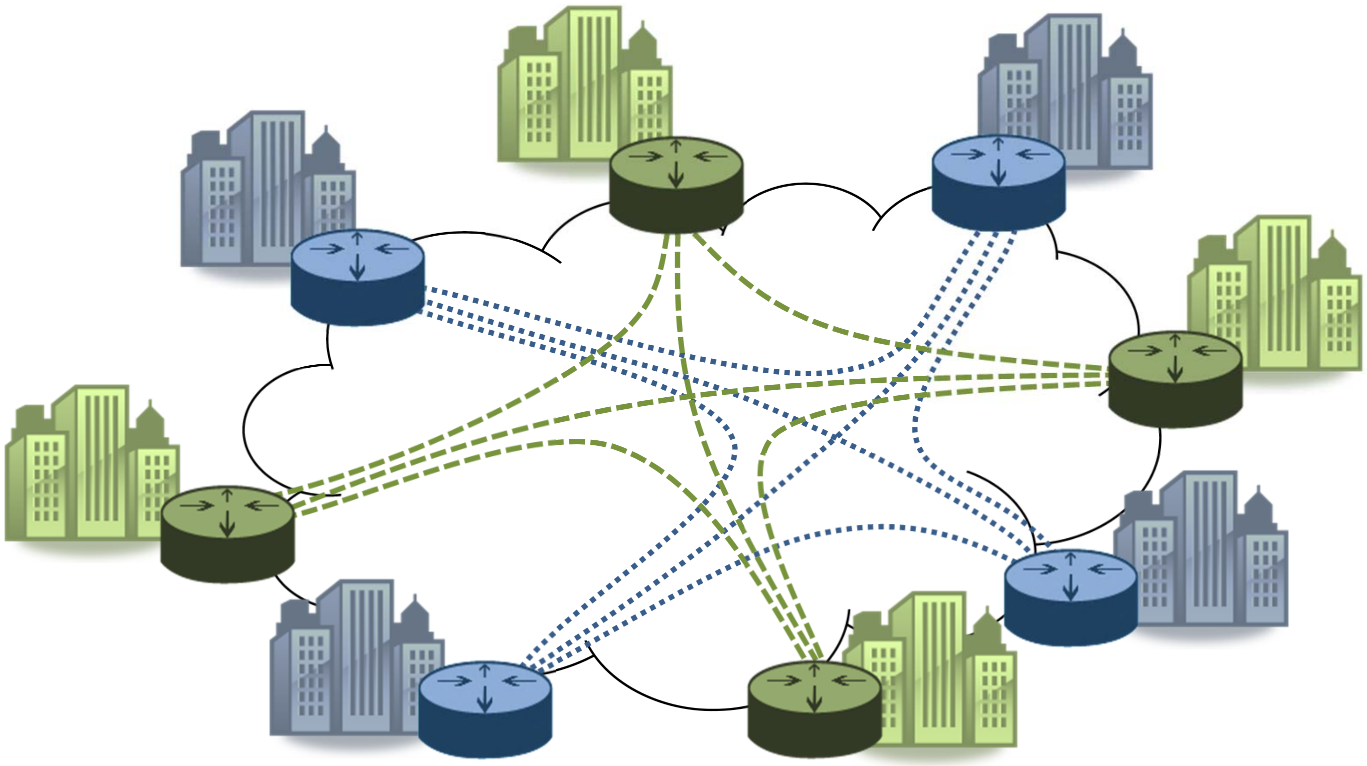
PT INOVAÇÃO

Requisitos de uma rede virtual

- **Serviços:** uma rede virtual deve poder oferecer o mesmo tipo de serviços normalmente disponíveis em redes físicas.
- **Fiabilidade:** uma rede virtual deve oferecer um grau de fiabilidade e robustez equivalente ao de uma rede baseada em recursos físicos dedicados.
- **Segurança:** uma rede virtual deve garantir níveis de segurança equivalentes aos de uma rede baseada em recursos físicos dedicados.
- **Desempenho:** uma rede virtual deve oferecer um nível de desempenho previsível e independente de factores externos à própria rede virtual (incluindo outras redes virtuais partilhando a mesma infraestrutura).
- **Interoperabilidade:** uma rede virtual deve poder interoperar com outras redes , físicas ou virtuais.
- **Operação / gestão:** uma rede virtual deve fornecer os instrumentos de operação, manutenção e gestão adequados.
- **Endereçamento:** uma rede virtual deve poder usar um plano de endereçamento fechado, flexível e independente do endereçamento de outras redes, incluindo outras redes virtuais, públicas ou privadas.

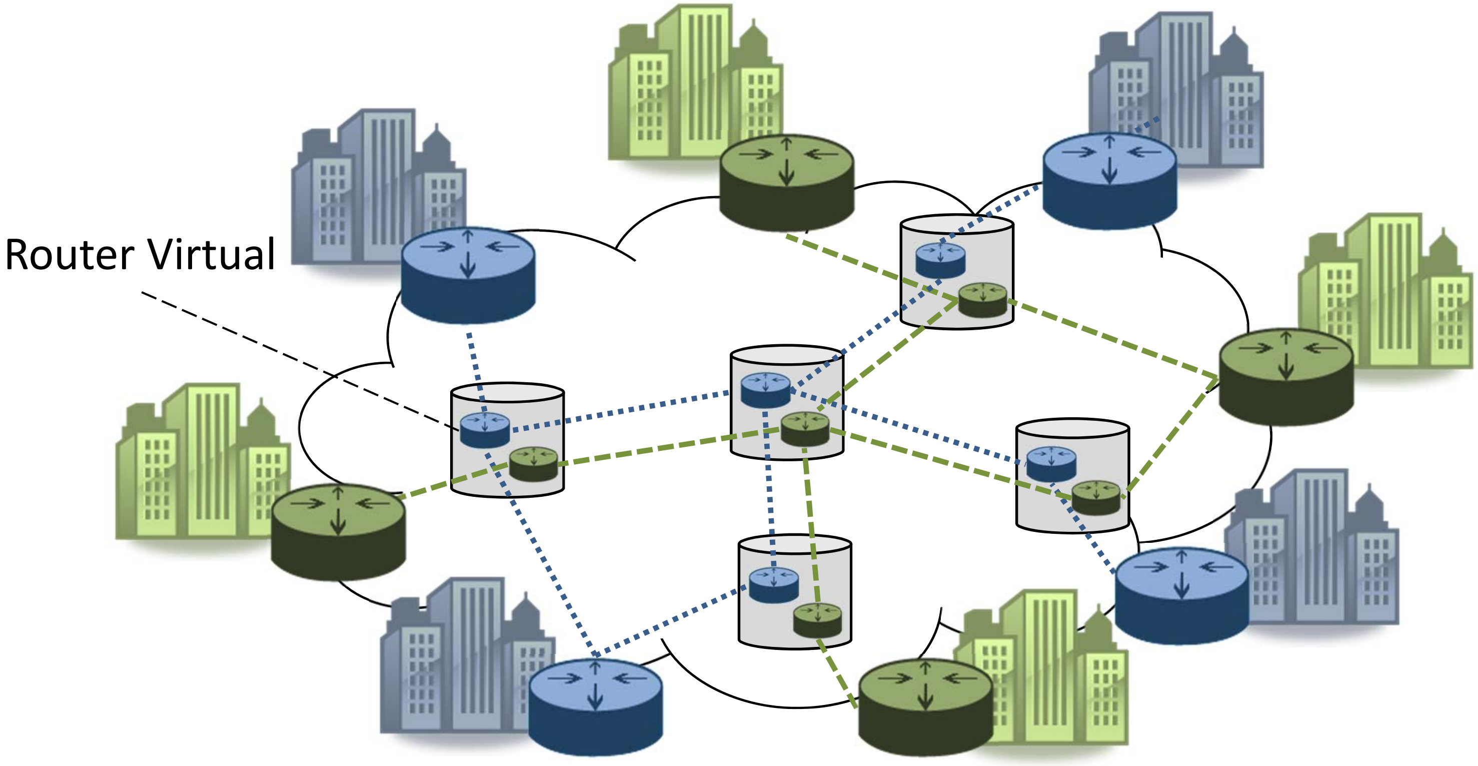
PT INOVAÇÃO

Virtualização no nível 2 - circuitos virtuais



Circuitos Virtuais

Virtualização no nível 3 – Routers Virtuais



Router virtual

A virtual router is a collection of threads, either static or dynamic, in a routing device, that provides routing and forwarding services much like physical routers.

A virtual router need not be a separate operating system process although it could be); it simply has to provide the illusion that a dedicated router is available to satisfy the needs of the network(s) to which it is connected.

(…)

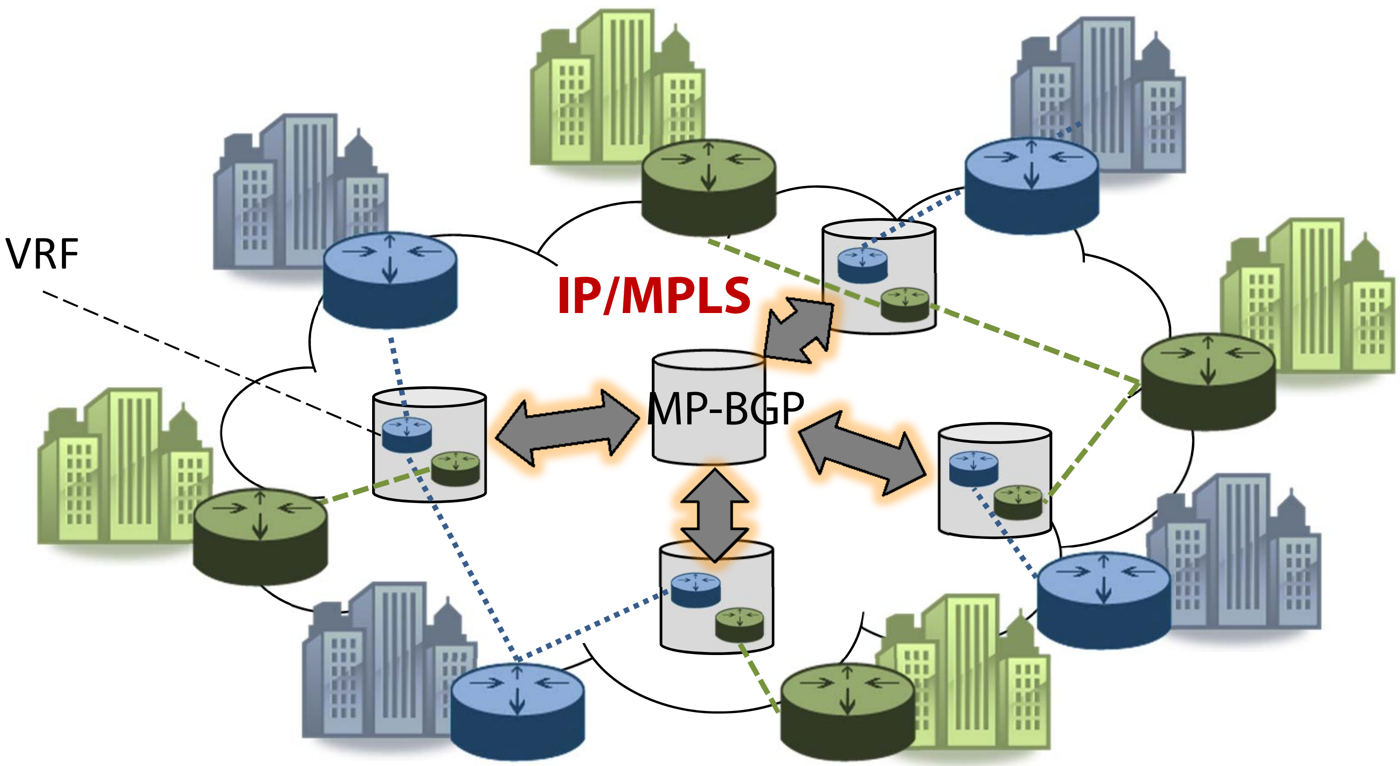
From the user (VPN customer) standpoint, it is imperative that the virtual router be as equivalent to a physical router as possible. In other words, with very minor and very few exceptions, the virtual router should appear for all purposes (configuration, management, monitoring and troubleshooting) like a dedicated physical router.

(…)

Every VPN has a logically independent routing domain. This enhances the SP's ability to offer a fully flexible virtual router service that can fully serve the SP's customer without requiring physical per-VPN routers. This means that the SP's "hardware" investments, namely routers and links between them, can be re-used by multiple customers.

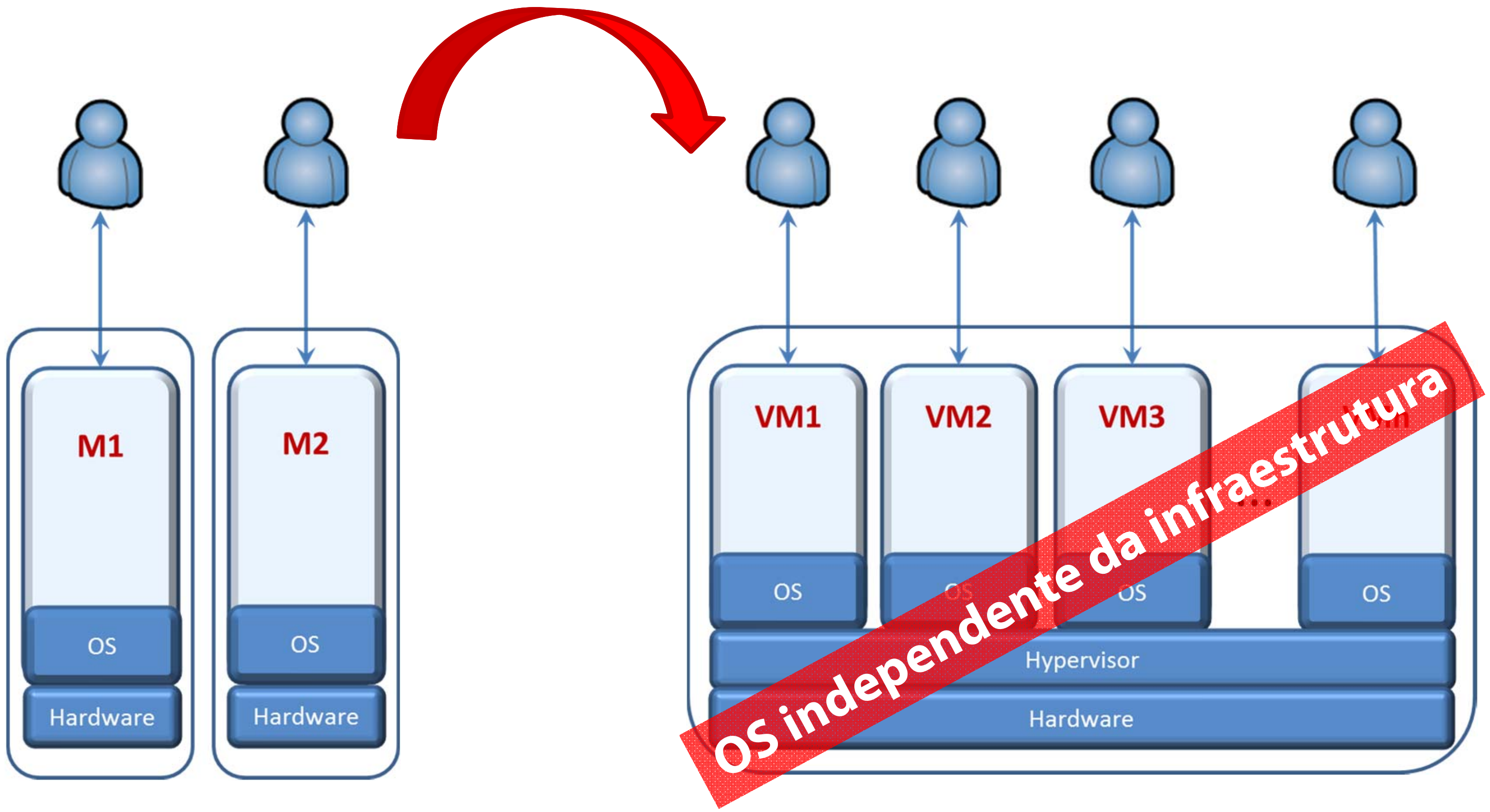
[K. Muthukrishnan, A. Malis,
RFC2917, "A Core MPLS IP VPN Architecture", Setembro 2000]

RFC 4364 (ex-2547) – BGP/MPLS IP VPNs



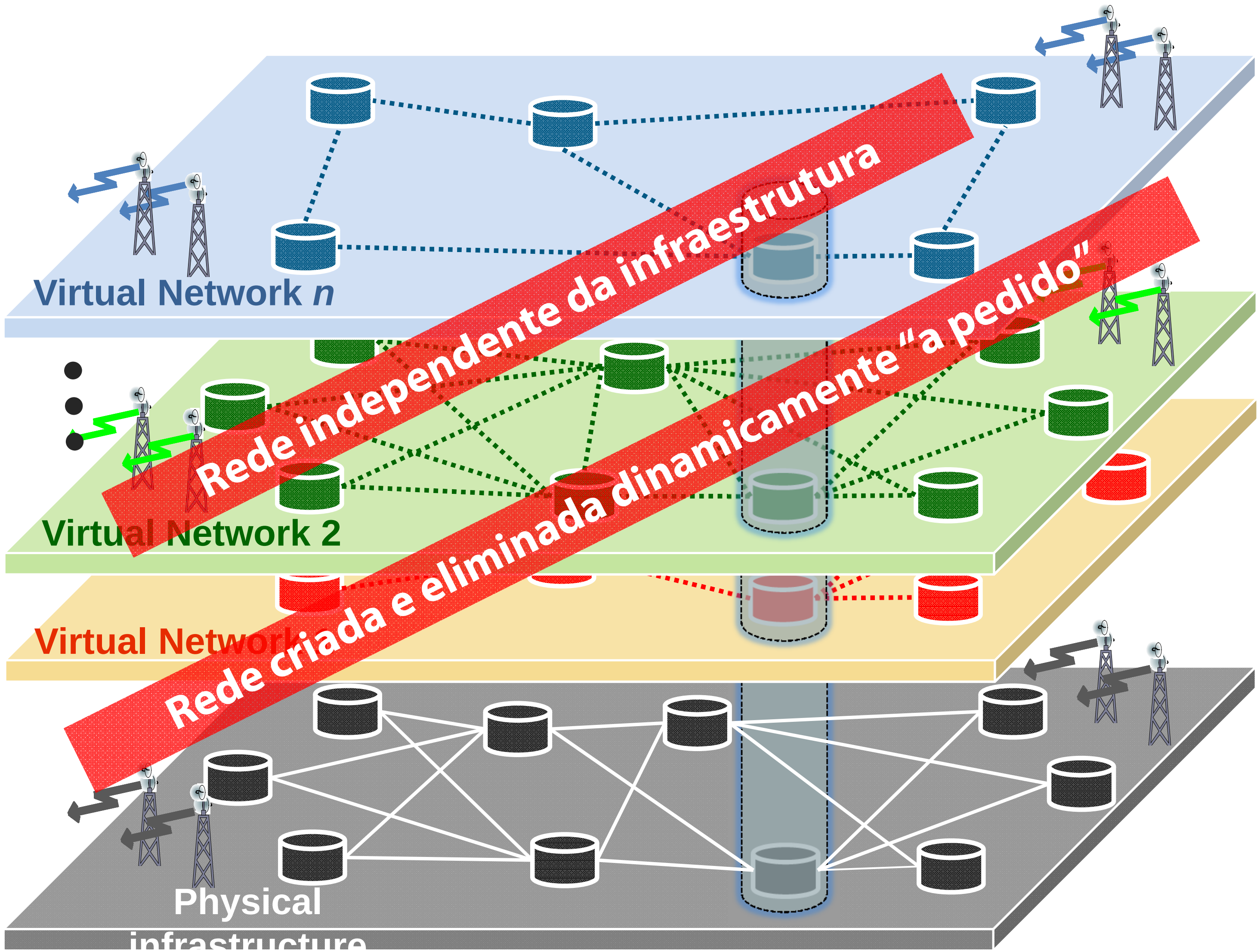
PT INOVAÇÃO

Virtualização da arquitectura x86



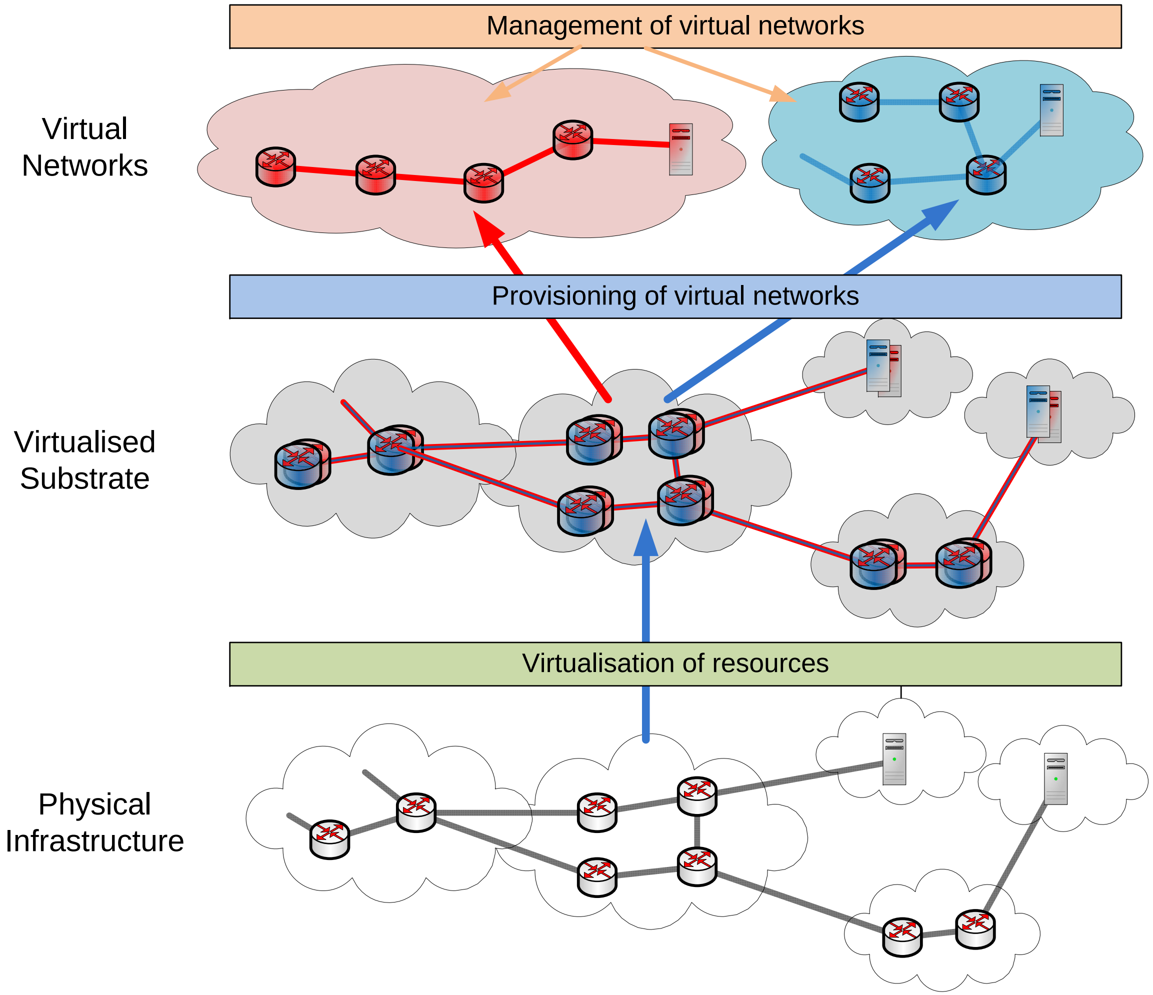
INOVAÇÃO

Virtualização de Rede



PT INOVAÇÃO

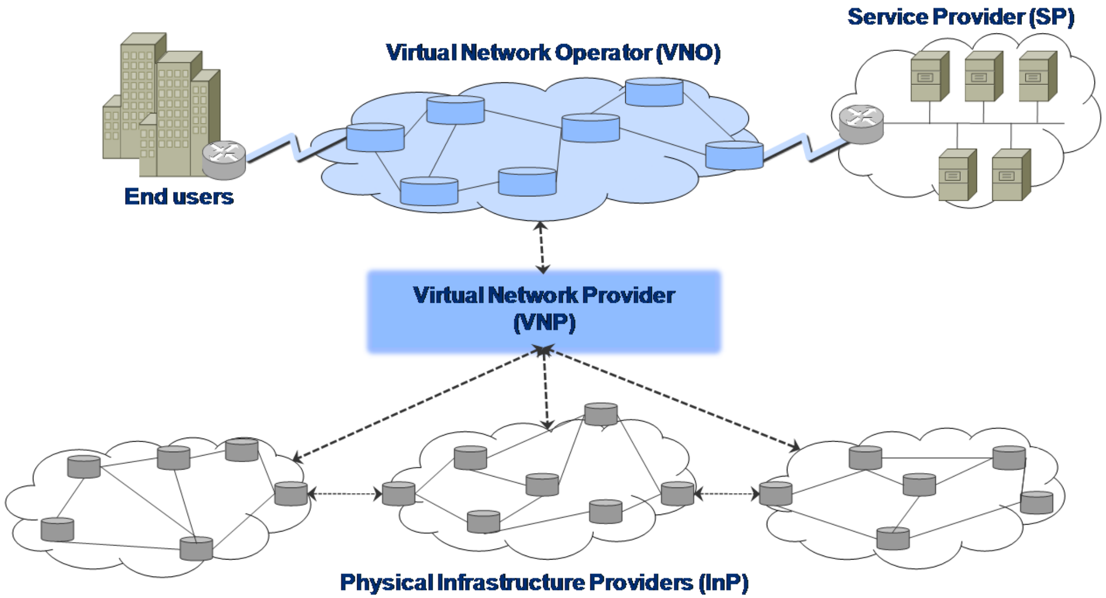
Separação rede virtual / infraestrutura



- Independent, isolated VNs, running different protocols, packet formats, management tools, etc.
- Collection of virtual resources, aggregated to build virtual networks
- Infrastructure made of virtualizable network resources

INOVAÇÃO

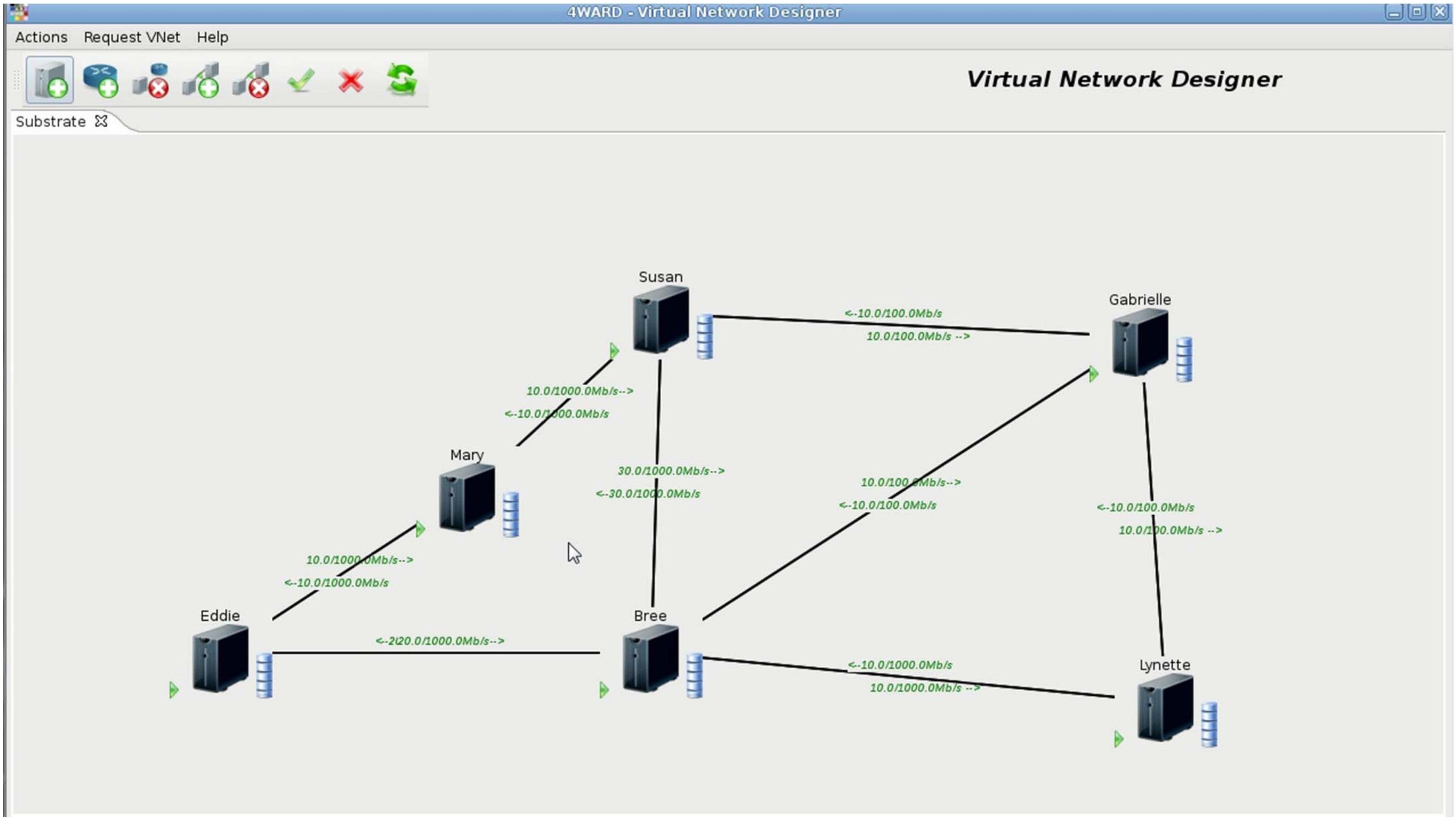
Novos modelos de negócio



- **InP** owns, controls and administers physical resources, which may be used, or offered to 3rd parties, to build custom-tailored VNs.
- **VNP** (optionally) assembles a VN, according to a given description, based on resources from one or more InPs. Likely to be the case when a VN spans multiple InP domains.
- **VNO** establishes, manages and operates VNs; handles end user attachment.
- **SP** provides services to end users; NV is supposed to be invisible from the SP perspective.
- **End user** is the user of the service offered by the SP (or directly by the VNO if a distinct SP does not exist as such).

INOVAÇÃO

Prova de Conceito - Virtual Network Designer



Cloud Computing

Características essenciais:

- On-demand Self-service
- Broad Network Access
- Resource Pooling
- Rapid Elasticity
- Measured Service

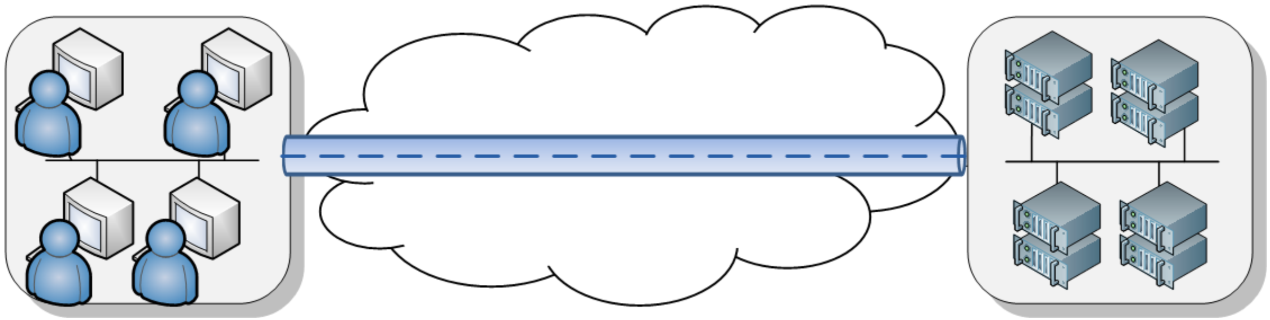
Modelos de *deployment*:

- Private Cloud
- Community Cloud
- Public Cloud
- Hybrid Cloud

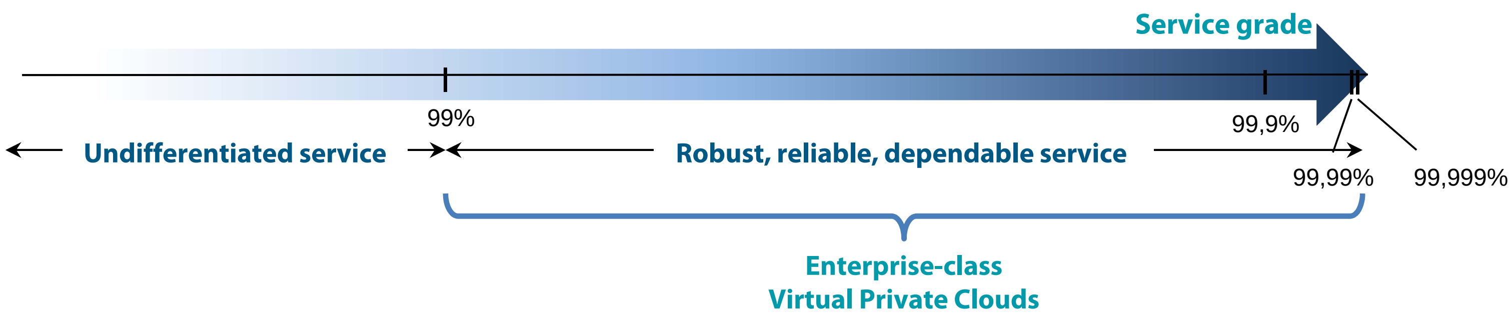
Modelos de serviço:

	IaaS	PaaS	SaaS
Aplicação			✓
Plataforma		✓	✓
Virtualização	✓	✓	✓
Hardware	✓	✓	✓

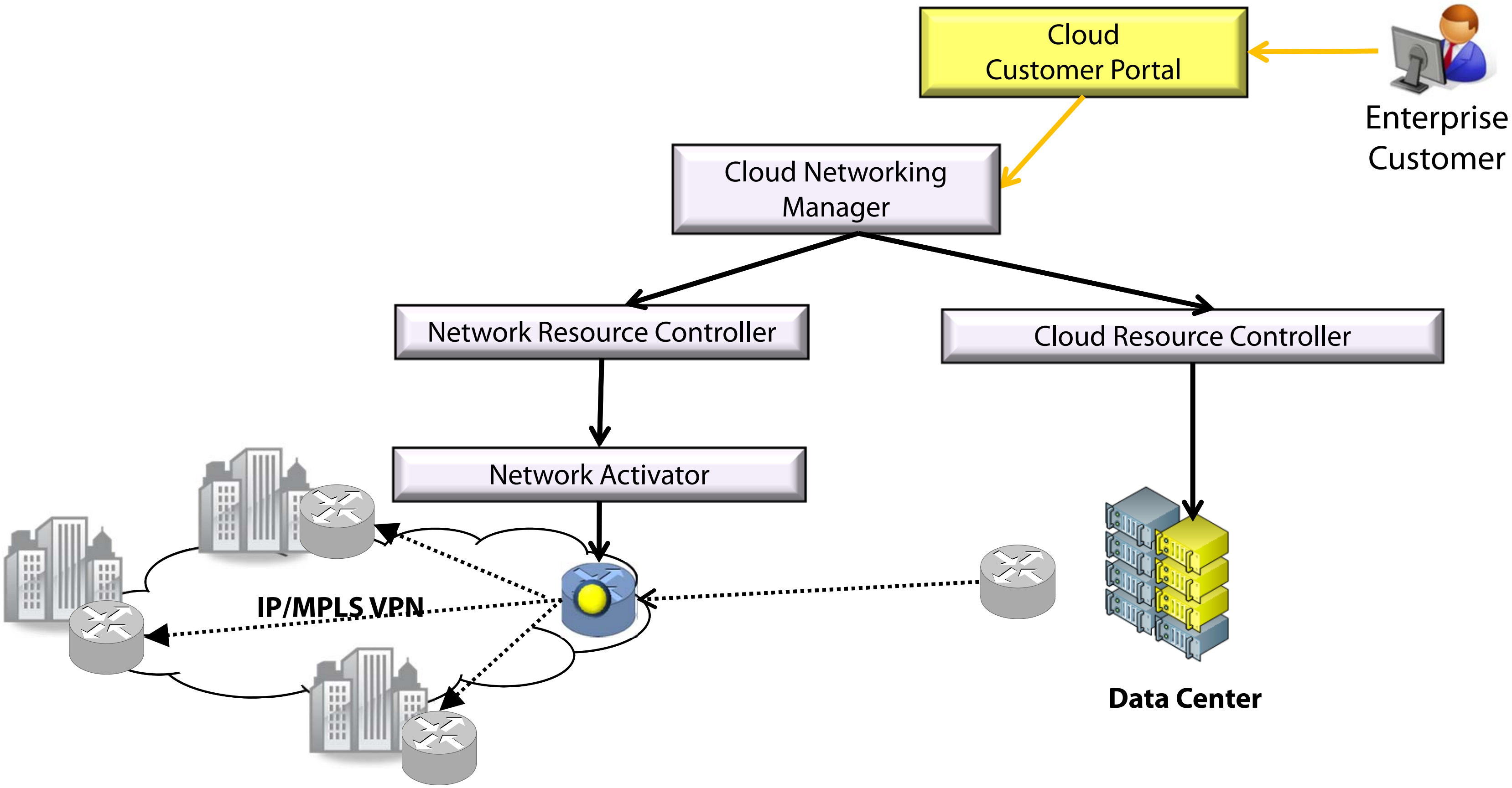
Rede: ingrediente fundamental da Cloud



- Dependability, predictable performance: "Always-on" mission-critical applications must be permanently available.
- Security and isolation: highly sensitive business data should be handled adequately.
- Flexible service provisioning, transparent and seamless integration in the enterprise network infrastructure.



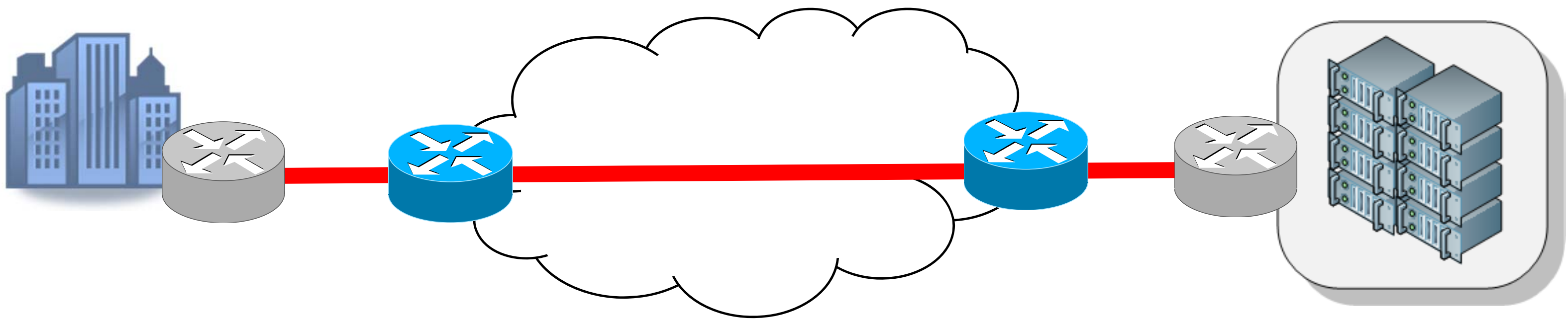
Desafio 1: Self-provisioning



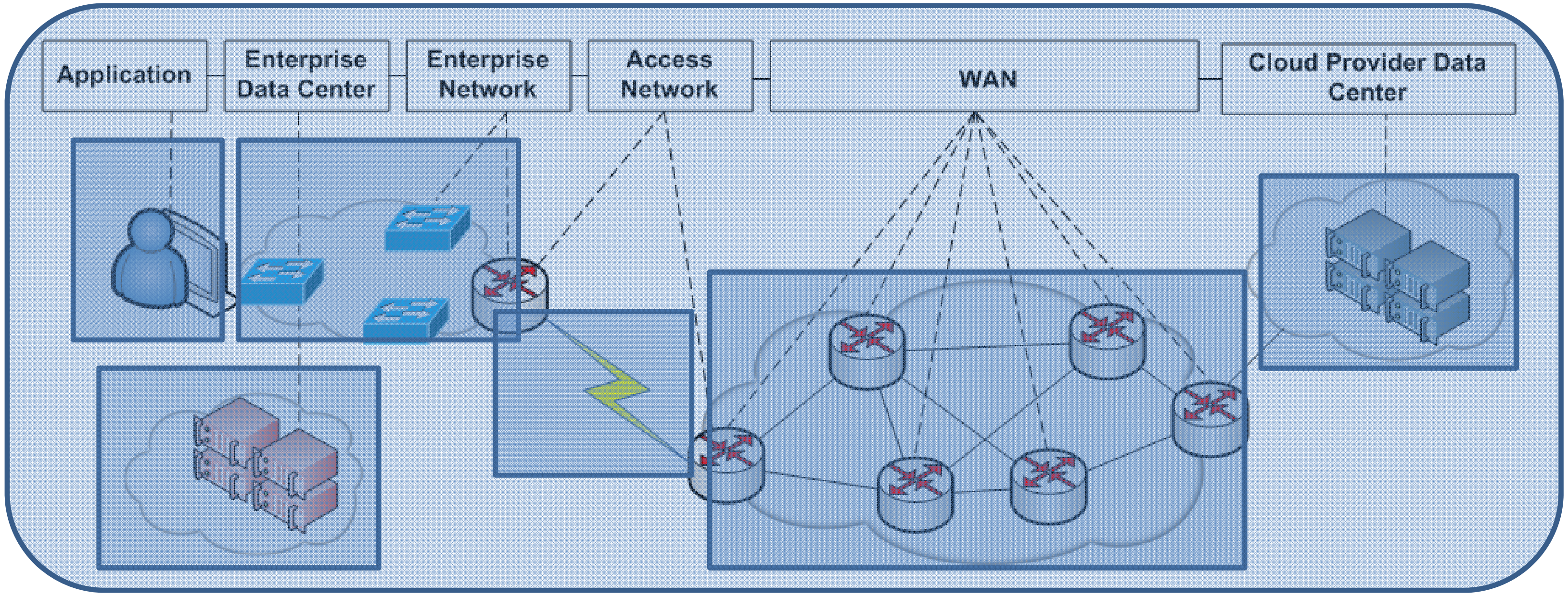
- By definition, clouds require on-demand allocation of resources where and when needed.
- Today's service provisioning cycle is too slow, not compatible with Clouds dynamics.

Desafio 2: Elasticidade de recursos

- Reconfiguration of computing and network resources must be synchronized.
  - Request of computing resources may imply reconfiguration of edge routers
  - Bandwidth capacity at WAN ingress/egress points should be dynamically reconfigurable



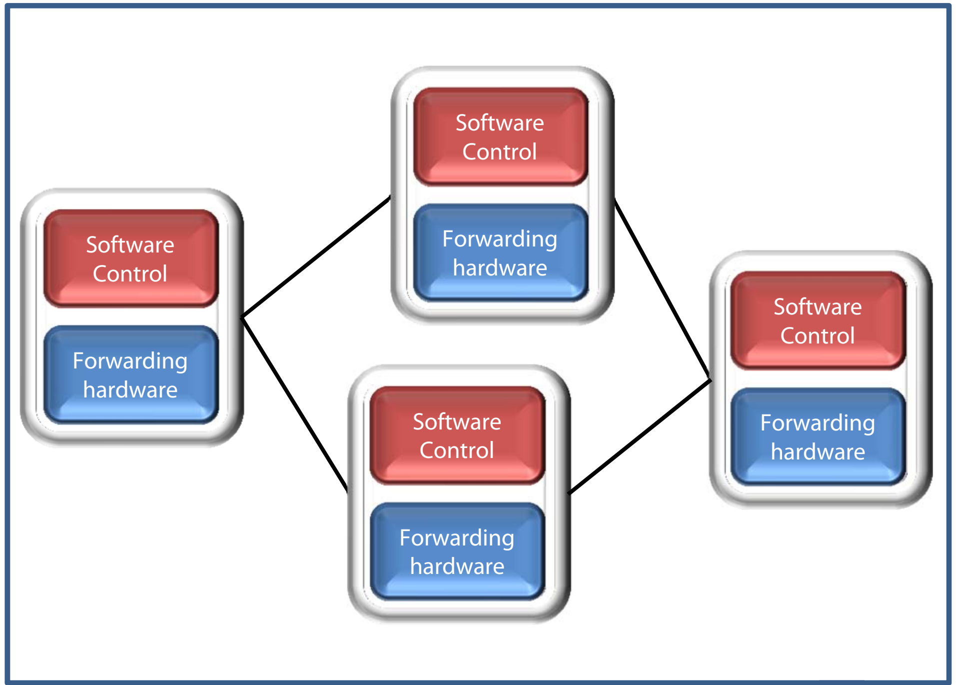
Desafio 3: Orquestração e controlo integrado dos recursos

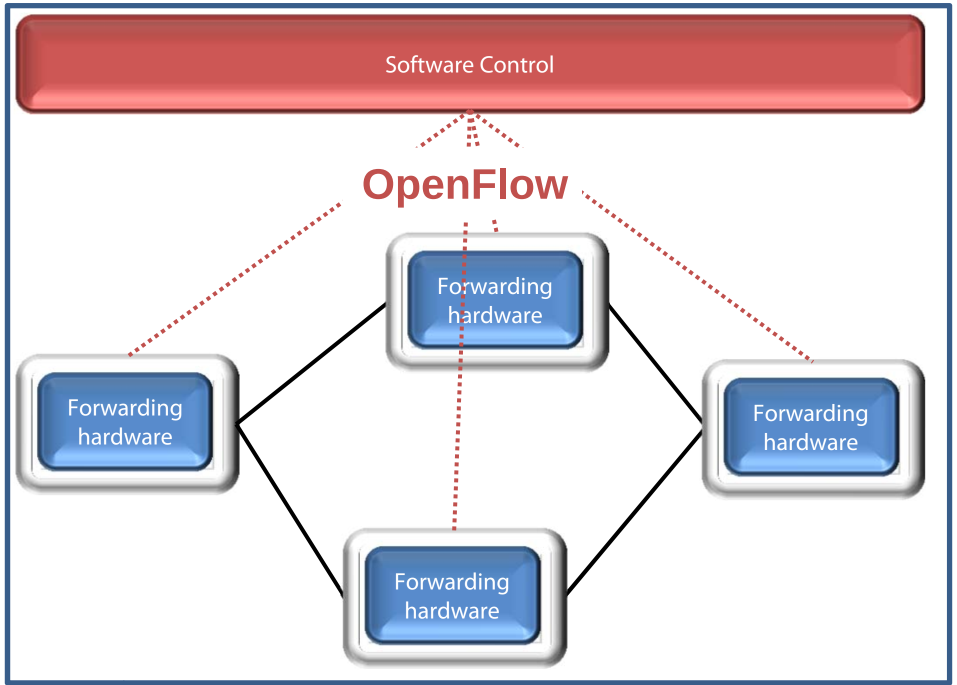


- Different network segments (DC, enterprise LAN, wired/wireless access, core network) can no longer be handled as silos; global resource orchestration is required.
- Control of applications, computational, storage and networking resources become closely intertwined
  - Fulfillment of end-to-end SLA requires an integrated view of all resources.

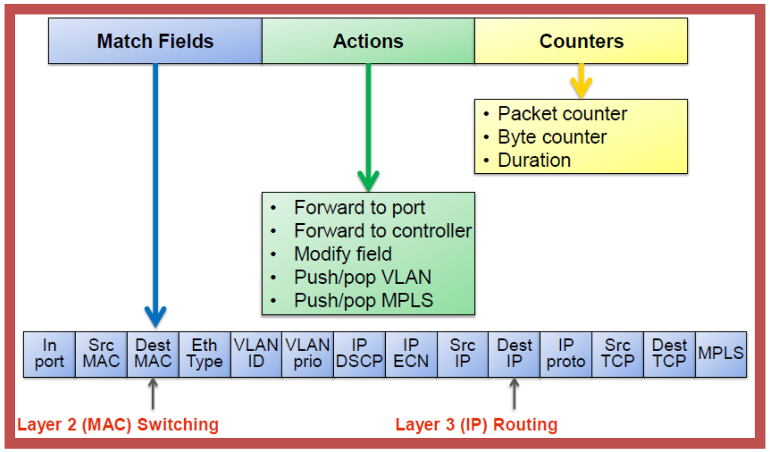
INOVAÇÃO

OpenFlow – Controlo e Transporte separam-se



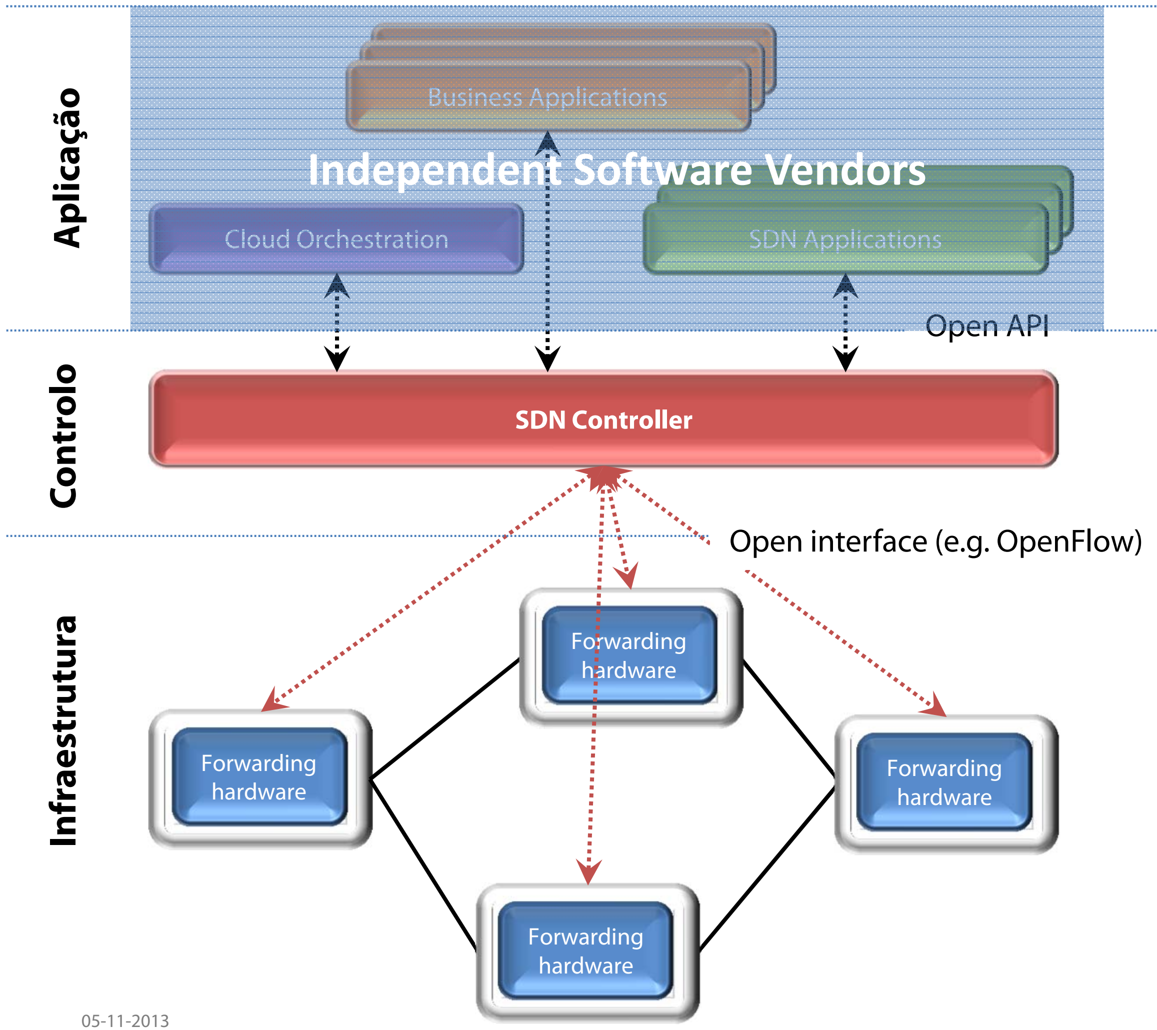


OpenFlow



INOVAÇÃO

Arquitectura SDN

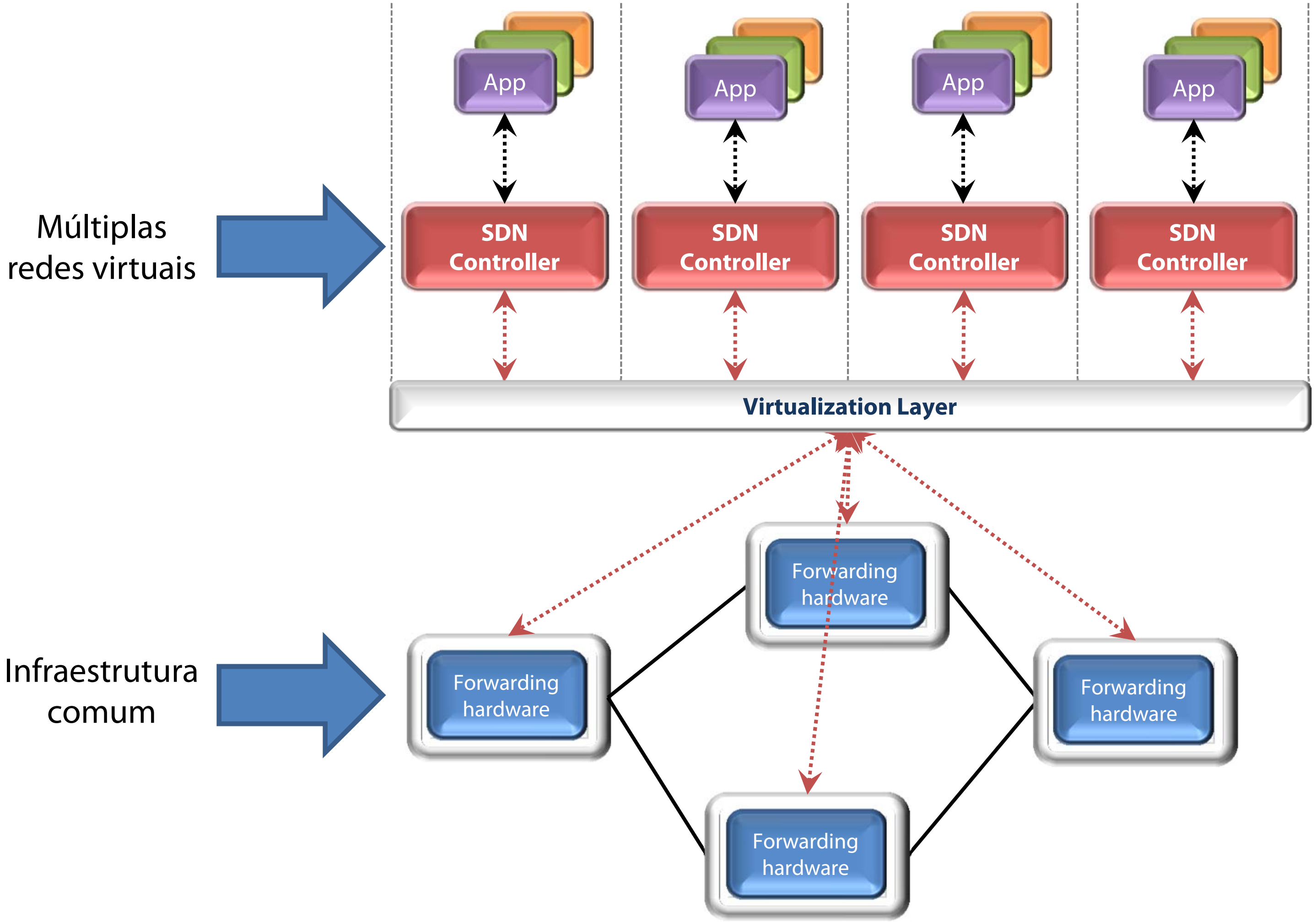


Princípios SDN
- Separação dos planos de controlo / transporte
- Abstração da infraestrutura de rede
- Programabilidade da rede por APIs abertas
- Visão global da rede

Oportunidades:
- Visão global dos recursos
- Programabilidade da rede
- Fácil articulação com aplicações
- Ágil adaptação / reconfiguração
- Rápida inovação

Desafios:
- Escalabilidade
- Fiabilidade "Carrier-grade"
- Segurança

Virtualização de rede com SDN



INOVAÇÃO

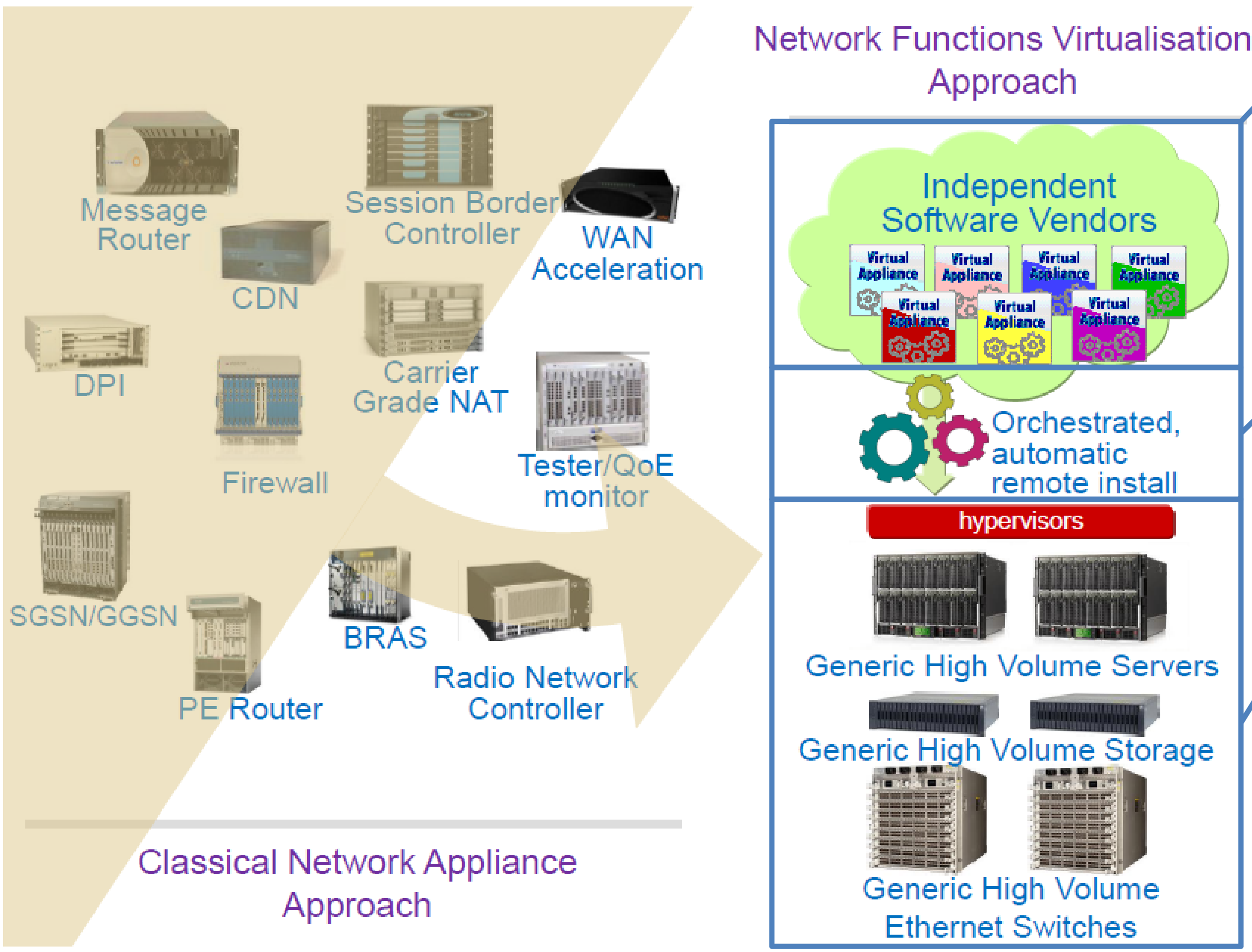
Código Python para implementar a funcionalidade de "learning switch"

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.openflow import PacketIn
from pox.topology.topology import Switch, Entity
from pox.lib.revent import EventMixin

import pickle

# Note that control applications are /stateless/; they are
simply a function:
# f(view) -> configuration
#
# The view is encapsulated in the NOM, and the configuration
results from manipulating
# NOM entities.
#
# To ensure statelesness (order-independence), the application
must never instantiate its own
# objects. Instead, it must "fold in" any needed state into the
NOM. The platform itself is in
# charge of managing the NOM.
#
# To "fold in" state, the application must declare a user-
defined NOM entity. The entities
# encapsulate:
# i. State (e.g., self.mac2port = {})
# ii. Behavior (i.e. event handlers, such as def
_handle_PacketIn() below)
#
# This is an example of a user-defined NOM entity.
class LearningSwitch (EventMixin, Entity):
  """
The learning switch "brain" associated with a single OpenFlow
switch.

When we see a packet, we'd like to output it on a port which
will eventually
lead to the destination. To accomplish this, we build a table
that maps
addresses to ports.

We populate the table by observing traffi
from some
source coming from some port, we know that
port.

When we want to forward traffic, we look up t  destination in
our table. If
we don't know the port, we simply send the message out all ports
except the
one it came in on. (In the presence of loops, this is bad!).

In short, our algorithm looks like this:

For each new flow:
1) Use source address and port to update address/port table
2) Is destination multicast?
Yes:
2a) Flood the packet
No:
2b) Port for destination address in our address/port table?
No:
2ba) Flood the packet
Yes:
2bb1) Install flow table entry in the switch so that this flow
goes out the appropriate port
2bb2) Send buffered packet out appropriate port
"""
  def __init__ (self, name, switch=None, macToPort={}):
    """
Initialize the NOM Wrapper for Switch Entities

switch - the NOM switch entity to wrap
"""
    # TODO: don't force user to inherit from Entity. We need this for
Entity.ID.
    # The long-term solution would be to create a second NOM layer for
user-defined
    # entities.
    Entity.__init__(self)
    self.name = name
    self.switch = switch
    self.log = core.getLogger(name)

    # We define our own state
    self.macToPort = macToPort

    if isinstance(switch, Entity):
      # We also define our behavior by reg
(_handle_PacketIn)
      self.listenTo(switch)

  def _handle_PacketIn
    """ Event han                              n the learning switch
algorithm """
    sel                           er! packet_in_event: %s" %

                us the packet """
          ODO: there should really be a static method in pox.openflow that
   structs this
      # this packet for us.
      msg = of.ofp_packet_out()
      msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
      msg.buffer_id = packet_in_event.ofp.buffer_id
      msg.in_port = packet_in_event.port
      self.switch.send(msg)

    packet = packet_in_event.parse()
    self.macToPort[packet.src] = packet_in_event.port # 1
    if packet.dst.isMulticast():
      flood() # 2a
    else:
      if packet.dst not in self.macToPort:
        self.log.debug("port for %s unknown -- flooding" % (packet.dst,))
        flood() # 2ba
      else:
        # 2bb
 port = self.macToPort[packet.dst]
        self.log.debug("installing flow for %s.%i -> %s.%i" %
                 (packet.src, packet_in_event.port, packet.dst, port))
        # TODO: there should really be a static method in pox.openflow that
constructs this
        # this packet for us.
        msg = of.ofp_flow_mod()
        msg.match = of.ofp_match.from_packet(packet)
        msg.idle_timeout = 10
        msg.hard_timeout = 30
        msg.actions.append(of.ofp_action_output(port = port))
        msg.buffer_id = packet_in_event.ofp.buffer_id
        self.switch.send(msg)

  def serialize(self):
    # TO   this is a hack... need a better way of differntiating IDs
(r        ) from raw objects (local case)
               ance(self.switch, Entity):
                id = self.switch.id

            tch_id = self.switch

    serializable = LearningSwitch(self.name, switch_id)
    serializable.log = None
    return pickle.dumps(serializable, protocol = 0)
```

A flexibilidade tem um custo!

Fonte: https://github.com/strategist333/hedera/blob/master/pox/pox/nom_l2_switch_controller/learning_switch.py

NFV - Virtualização das *Funções* de Rede



Aplicacional

- Incentivo à inovação
- Menor *Time-to-Market*
- Aumento de receitas
- Ciclo de inovação e maturação dos serviços mais rápido

Operacional

- Maior eficiência operacional
- Maior escalabilidade
- Maior flexibilidade e agilidade
- Resiliência

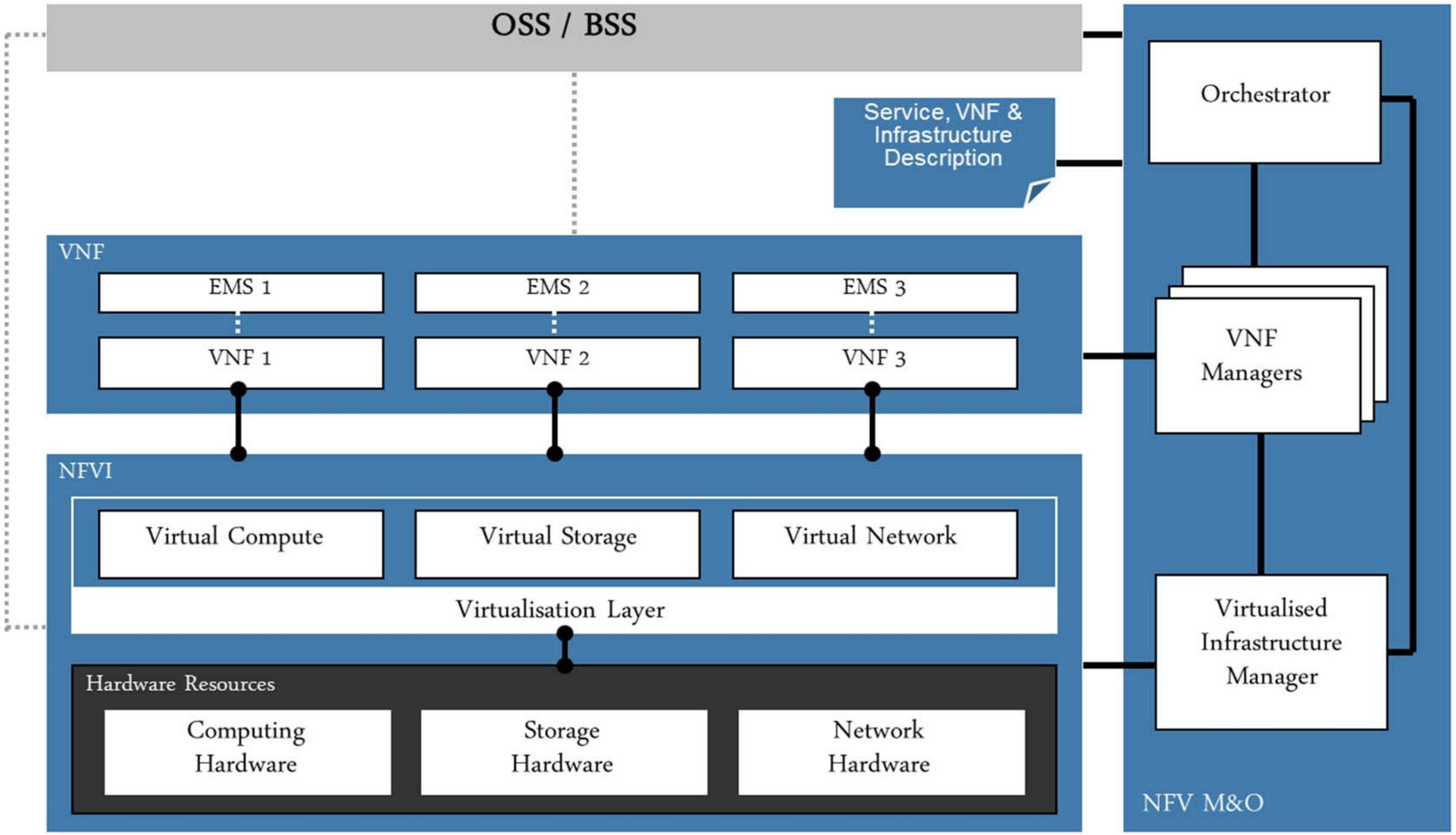
Infraestrutura

- Redução de CapEX e OpEx
- Redução do consumo energético; aproveitamento eficiente de recursos da rede
- Redução de espaço físico necessário para alojar *appliances* físicas
- Interoperabilidade num ambiente *multi-vendor*

NFV "in a nutshell": *Consolidação dos vários tipos de equipamento de rede, e virtualização das respetivas funções, em servidores de baixo custo, switches, dispositivos de armazenamento*

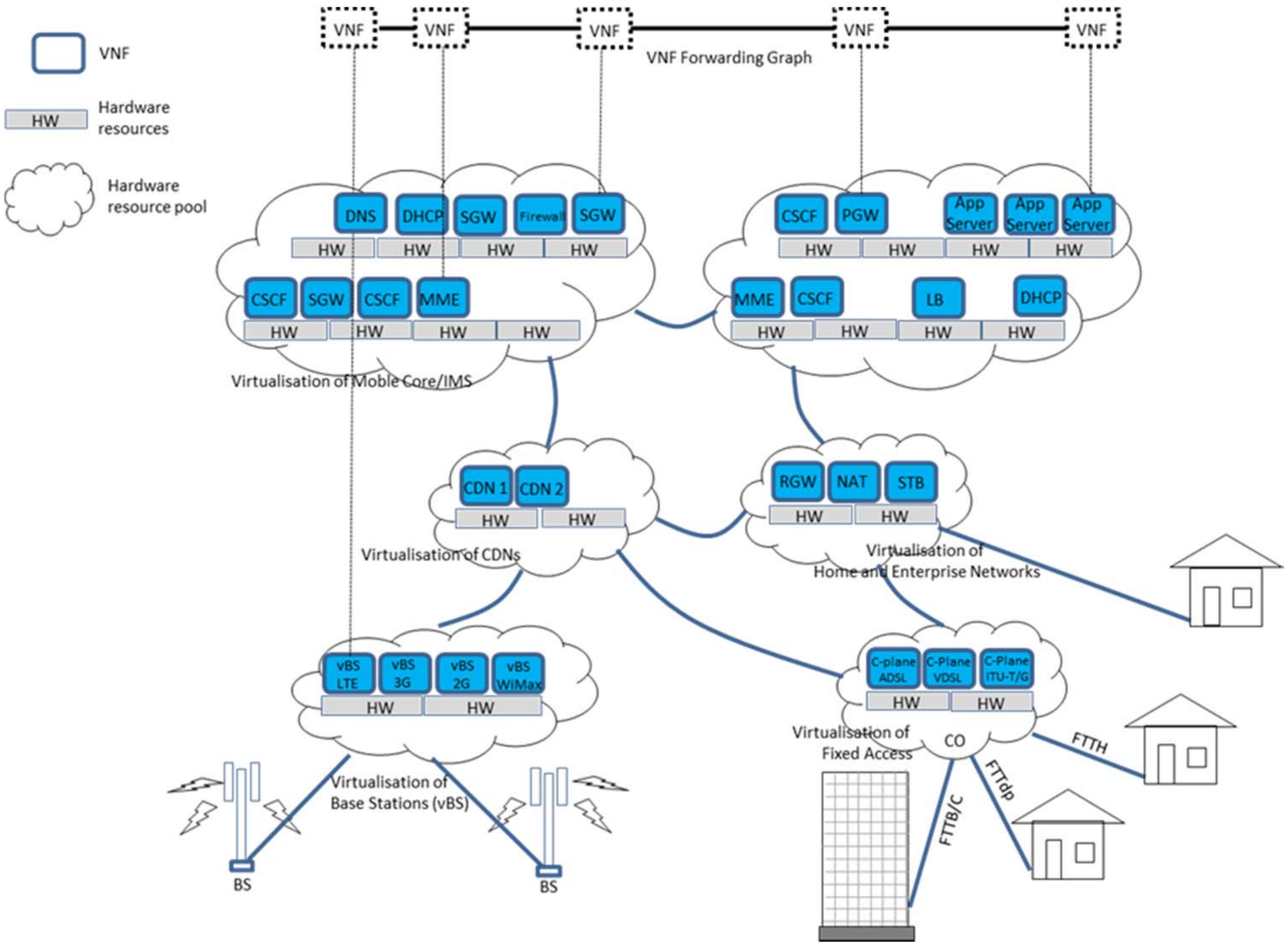
PT INOVAÇÃO

ETSI NFV – Arquitectura de Referência



Fonte: ETSI GS NFV 002 v1.1.1 “Network Functions Virtualisation (NFV); Architectural Framework”

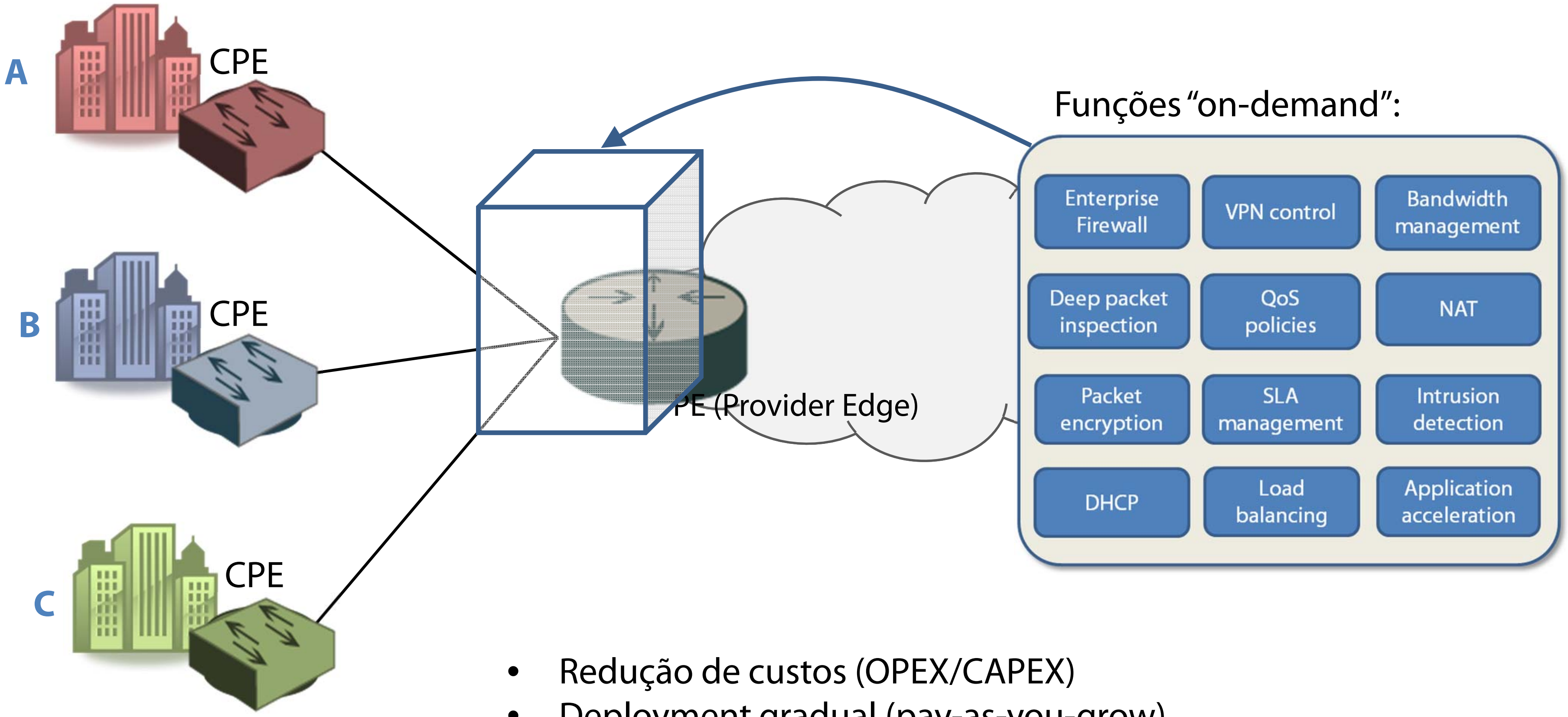
ETSI NFV – Use Cases



1. Network Functions Virtualisation Infrastructure as a Service
2. Virtual Network Platform as a Service (VNPaaS)
3. Virtual Network Function as a Service (VNFaaS)
4. Virtualisation of Mobile Core Network and IMS
5. Virtualisation of Mobile base station
6. Virtualisation of the Home Environment
7. Service Chains (VNF Forwarding Graphs)
8. Virtualisation of CDNs (vCDN)
9. Fixed Access Network Functions Virtualisation

Fonte: ETSI GS NFV 001 v1.1.1 “Network Functions Virtualisation (NFV); Use Cases”

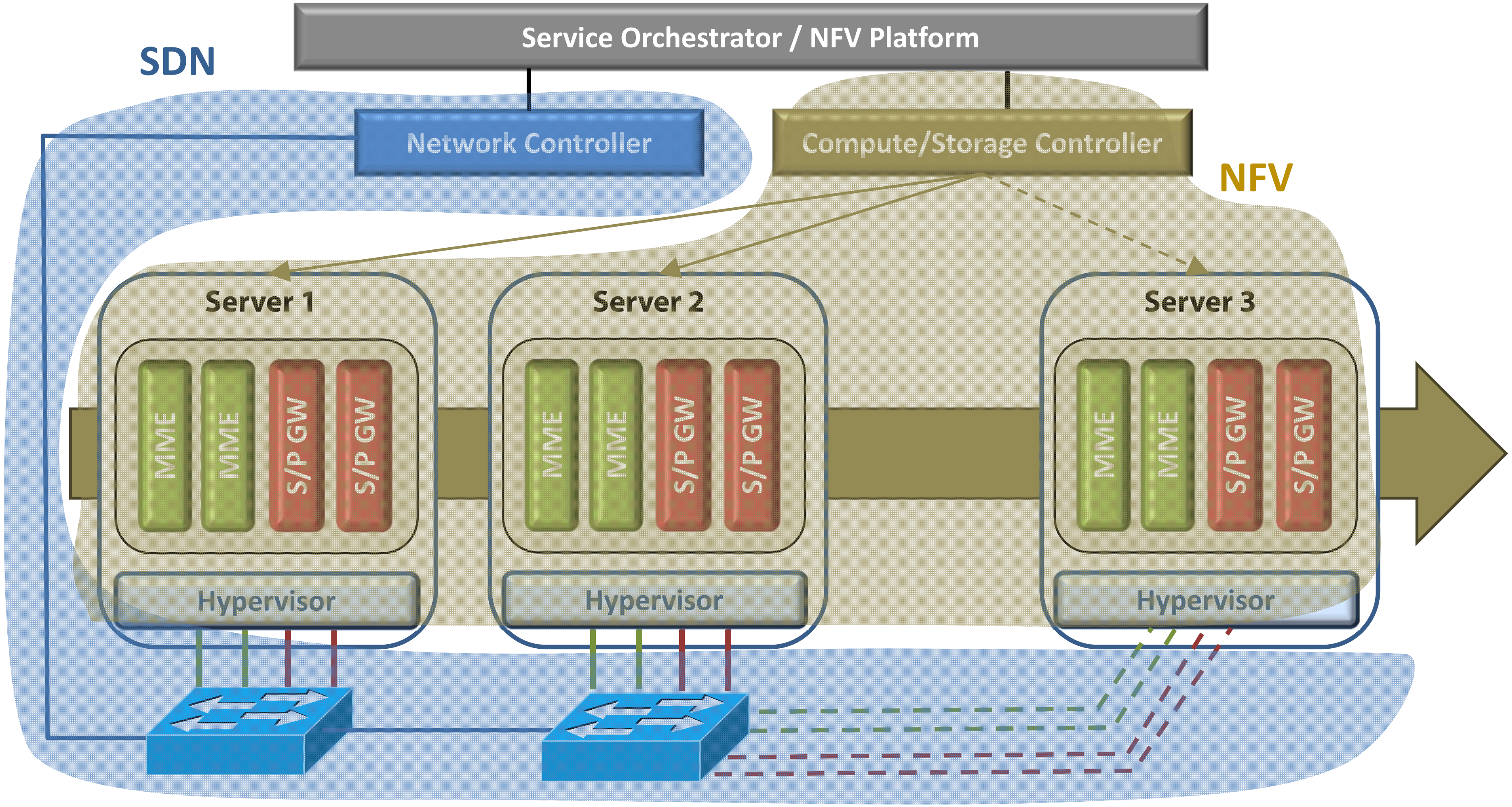
Virtualização do CPE



- Redução de custos (OPEX/CAPEX)
- Deployment gradual (pay-as-you-grow)
- Controlo partilhado do CPE, delegação de funções seleccionadas do CPE para o cliente
- Redefinição dos papeis PE/CPE

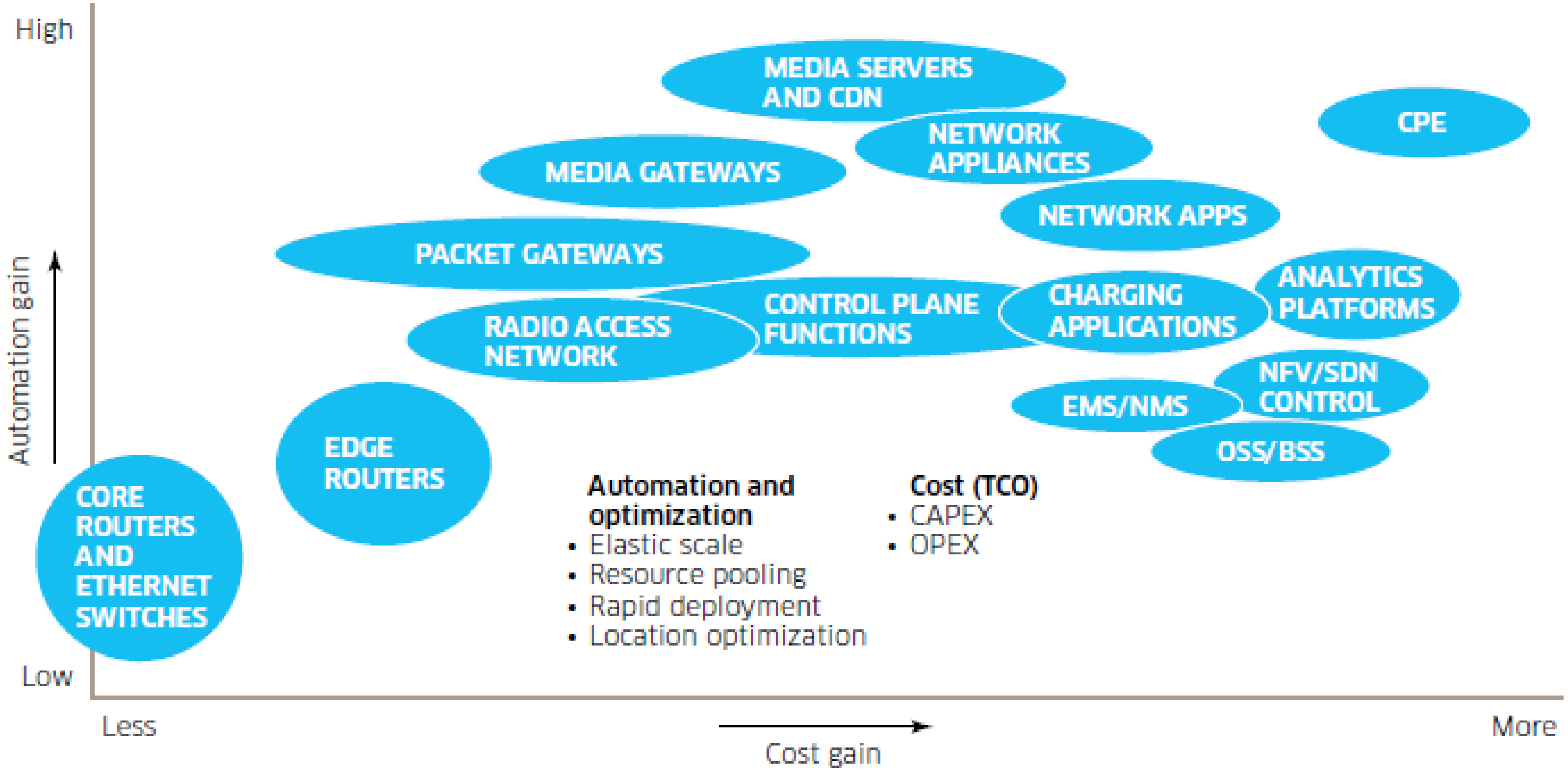
INOVAÇÃO

Virtualização do Mobile Core (EPC)



INOVAÇÃO

NFV - Funções candidatas



Fonte: Alcatel-Lucent Strategic White Paper, “Network Functions Virtualization – Challenges and Solutions”

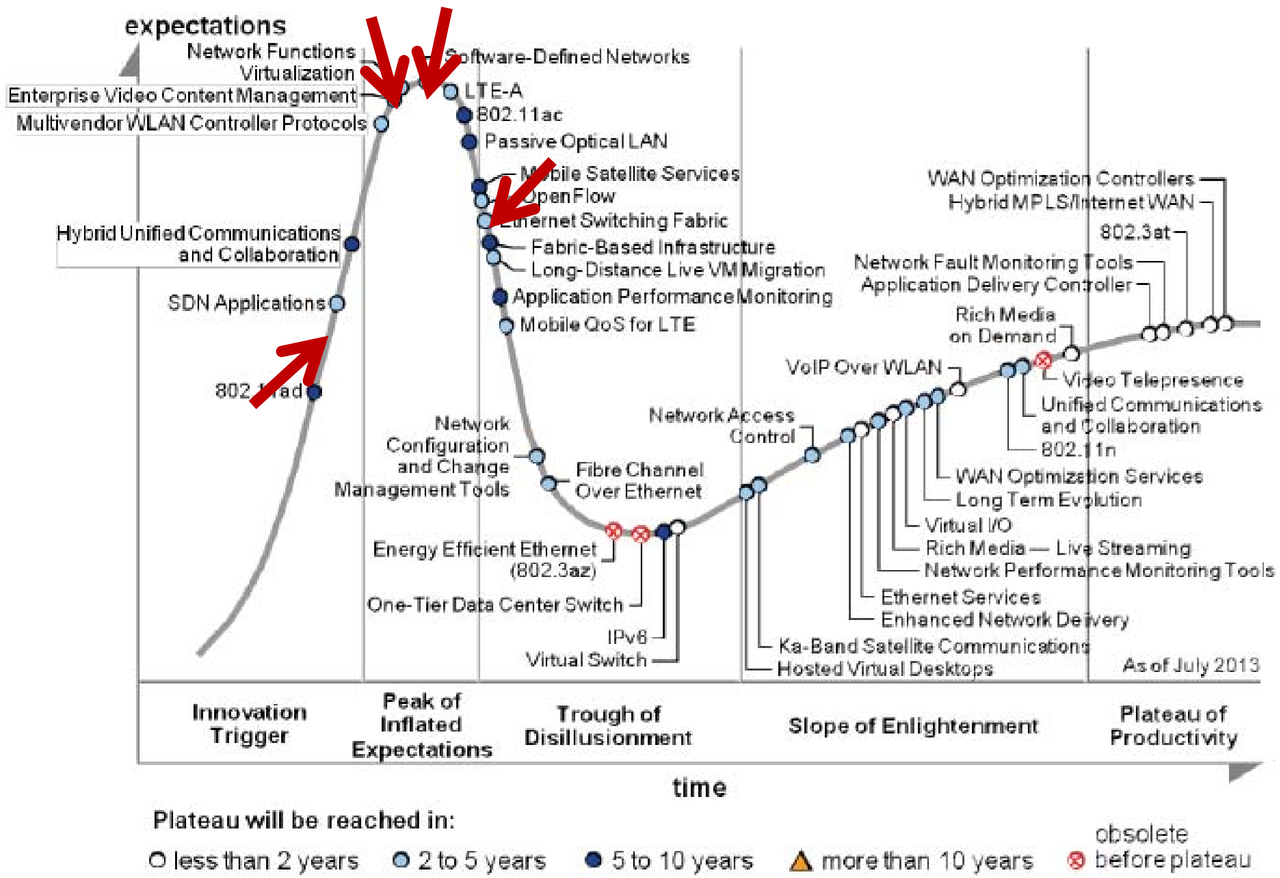
Futuro – Algumas questões a responder

- Como definir uma estratégia de migração para SDN/NFV e como garantir a interoperabilidade com tecnologias tradicionais?
- Como tirar partido dos benefícios da virtualização de funções de rede sem pôr em causa o desempenho e a robustez da rede?
- Como lidar com gestão de falhas (e.g. detecção, localização, reparação) em ambientes virtualizados?
- Quais as limitações de escalabilidade impostas pela centralização do controlo da rede em SDN?
- Tenderá a virtualização a diminuir ou aumentar os custos operacionais da rede?
- Que impacto terá na indústria a redução dos equipamentos de rede às funções de transporte?
- Que aplicações poderão tirar maior partido da virtualização e da programabilidade da rede?
- Que novos actores e modelos de negócio poderão surgir da virtualização de rede e da separação entre serviços/aplicações e infra-estrutura?

INOVAÇÃO

Gartner Hype Cycle 2013 – Networking & Communications

Figure 1. Hype Cycle for Networking and Communications, 2013



Source: Gartner (July 2013)

Fonte: Gartner, Hype Cycle for Networking and Communications, 2013

Obrigado!