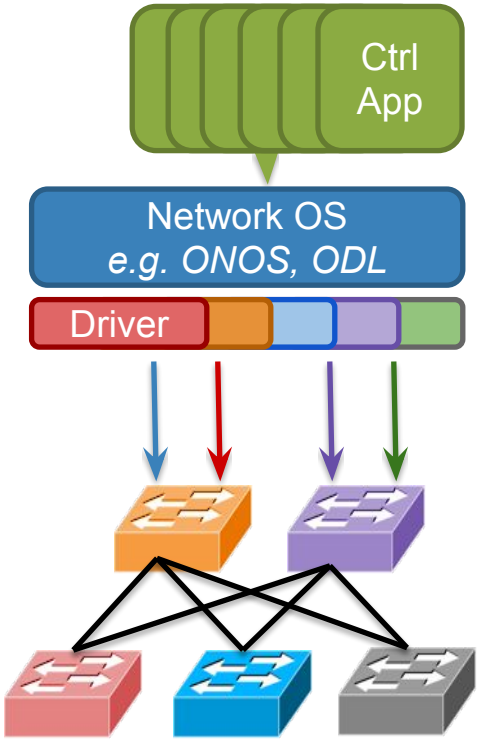
Next-Generation SDN

A high level introduction and motivation to the ONF's UPAN Reference Design

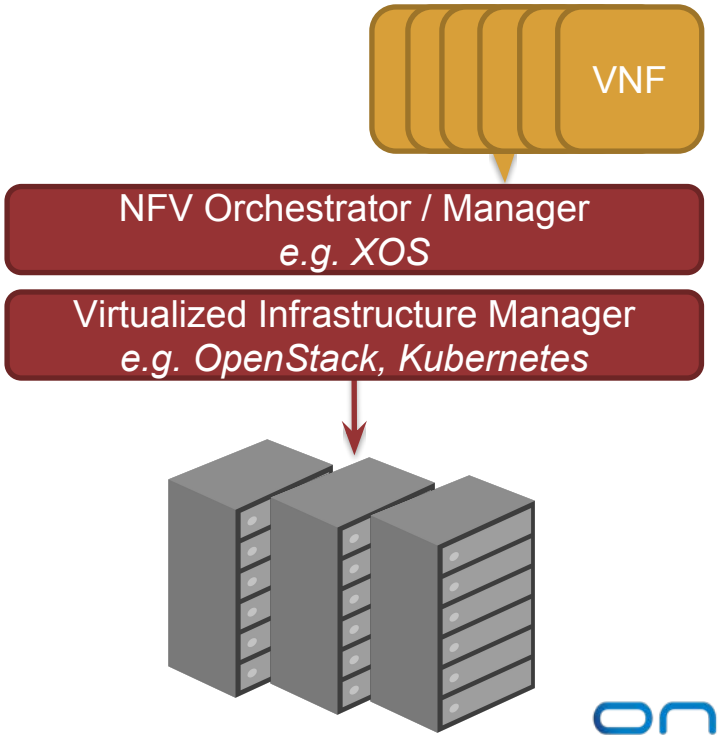
ONF Connect
December 5, 2018

A Brief Retrospective

SDN v1: Adapt to existing, heterogeneous hardware



NFV v1: Virtualize specialized networking boxes



Next-Generation SDN Tenets

Prescribe configuration, pipeline definition, forwarding state, and network intent

- Defines unambiguous contract between control and data plane

Disaggregate the control plane and VNFs

- Enables function placement to be optimized on resource type and location

Unify the “network” and “compute” infrastructure

- Simplifies deployment; improves resource utilization

Build toolchain for end-to-end debugging, verification, and upgrades

- Improves reliability and availability; enables rapid iteration

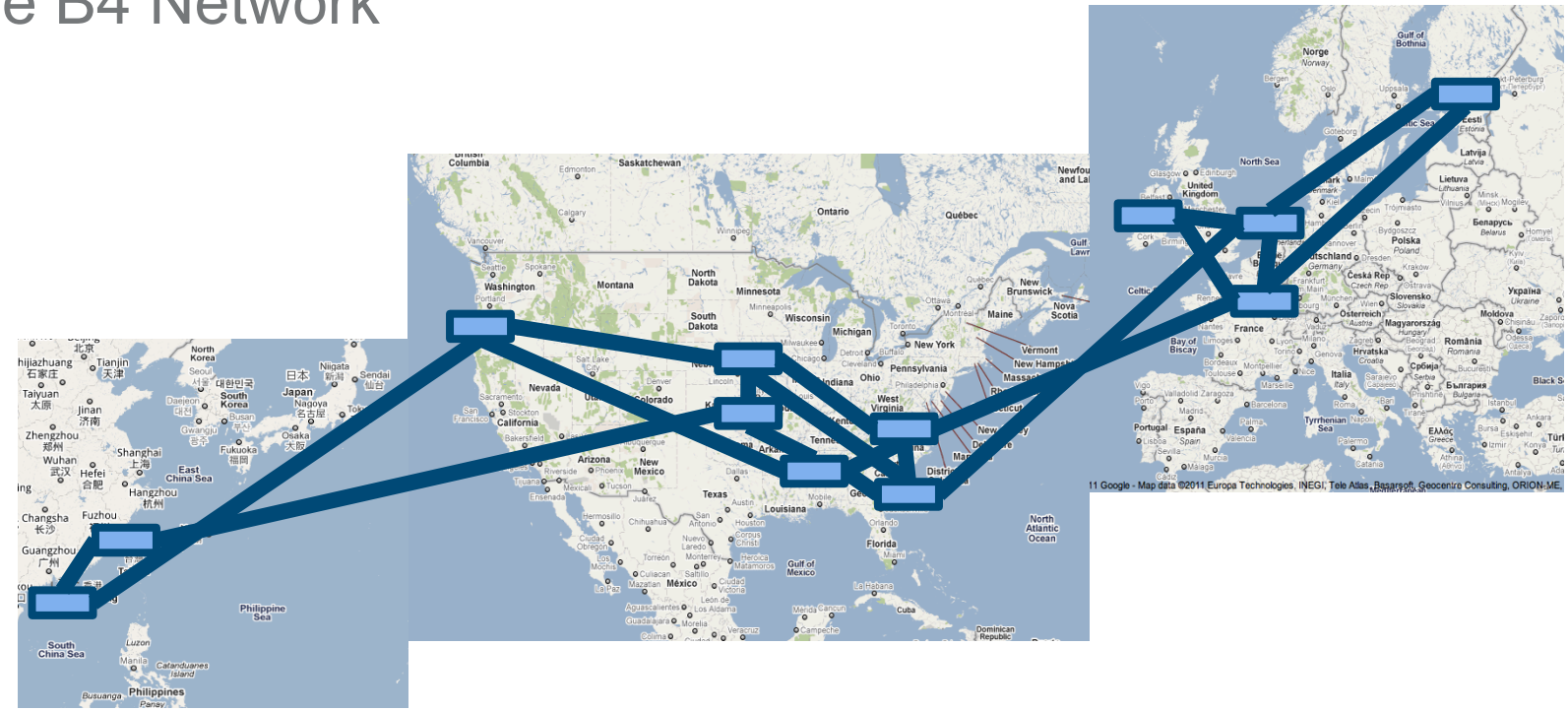
A new architecture allows us to retain the benefits of SDN/NFV while minimizing some of the challenges, costs, and unpleasantries of earlier approaches.

Stratum SwitchOS

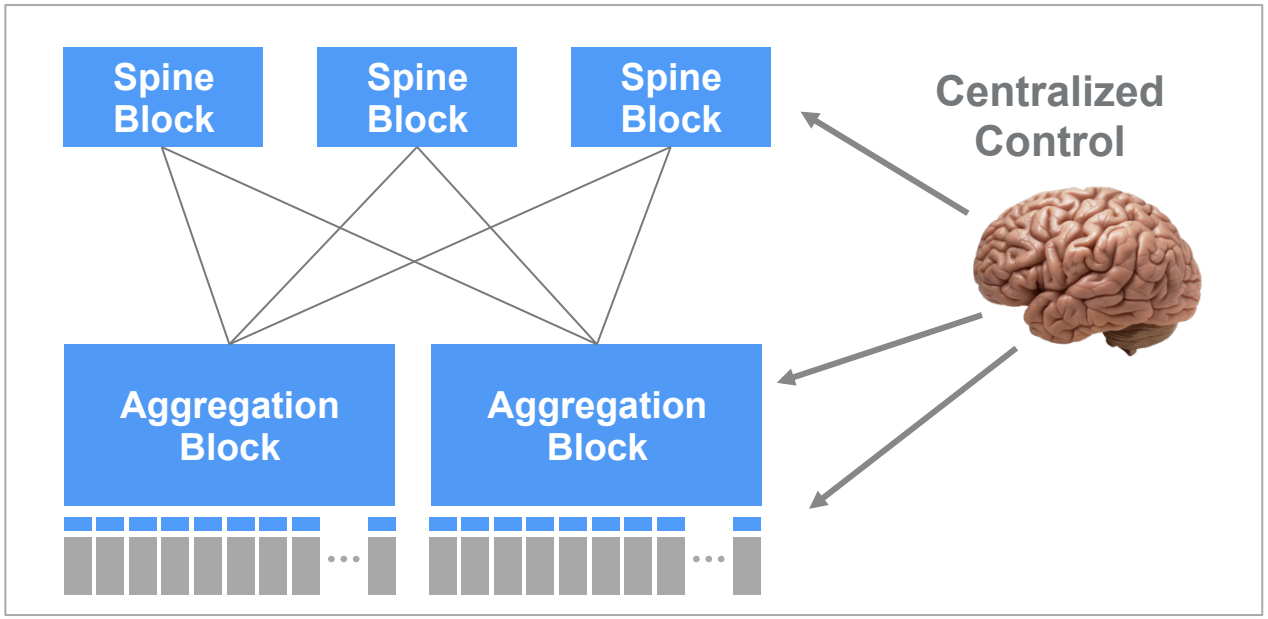
Alireza Ghaffarkhah, Devjit Gopalpur, Brian O'Connor, Jim Wanderer

Google

Google B4 Network



Centralized SDN Control Plane



- Scale
- Performance
- Control
- Failure handling
- ...

Enables control and optimizations difficult or impossible with traditional networking.

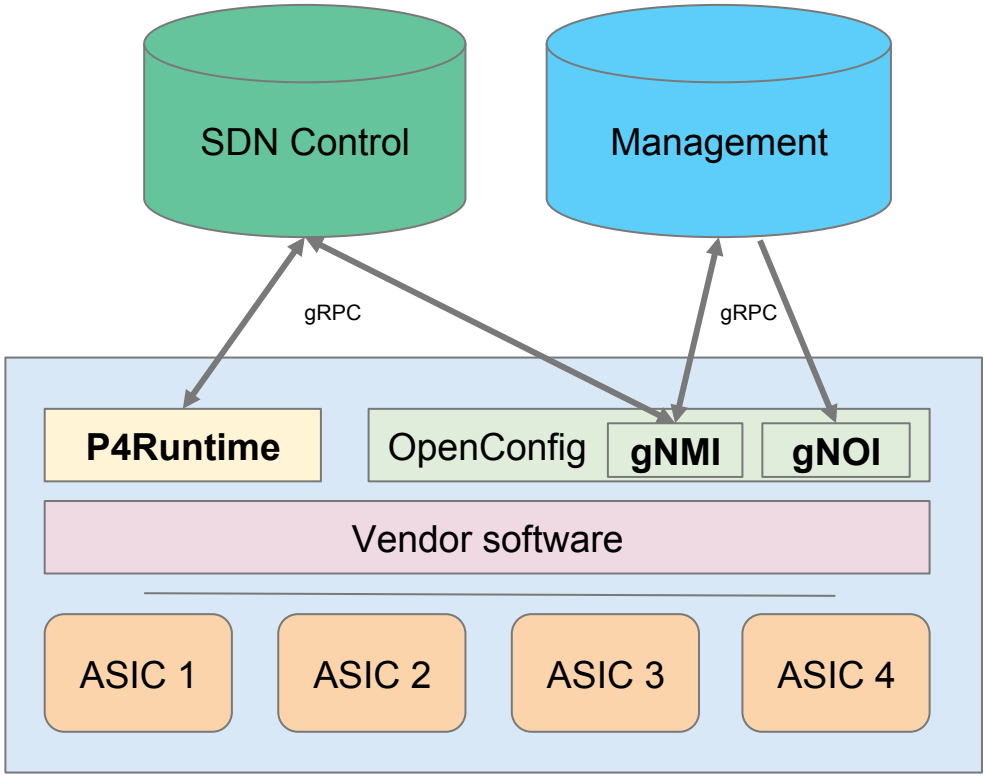
Use of OpenFlow

- Foundation for all of our SDN development
- Big win enabling centralized SDN

- Difficult to exploit full capabilities of HW
- Required customizations and extensions
- Growing complexity
- Poor match for programmable chips

- Proprietary implementation
- Can’t get solutions from industry

P4Runtime + OpenConfig



- Standard APIs
- Vendor abstraction
- Exploit HW capabilities
- Fixed function chips
- Programmable chips

- Drive industry direction
- Open, minimal, production-ready distribution
- Team of operators and vendors
- Drop into Google SDN fabrics

NG ONOS

Rationale & Tenets

Thomas Vachuska - ONF

ONOS Today

- ONOS provides a stable platform with nice characteristics:
  - easy app development
    - SDK, distributed stores/primitives, app archetypes, etc.
  - easy deployment as a distributed cluster
    - Docker containers, Kubernetes, etc.
  - automatic service injection
  - super-fast
    - service calls are just method calls
  - lots of existing apps and extensions
    - protocol extensions, device drivers, utilities, etc.
    - support for both legacy protocols and next-gen SDN interfaces

ONOS Today

- ONOS architecture also has some caveats and limitations:
  - limited isolation mechanism
    - core & apps share same resources
  - unable to have tenant-specific apps
    - only tenant-aware ones
  - apps limited to Java or JVM-based languages
    - e.g. Scala, Jython, Groovy
  - horizontal app/service scaling is difficult
    - enforced cluster symmetry
  - difficult to migrate components off-platform
    - e.g. control-plane modules embedded on switch

Looking Ahead

- With ONOS 2.0 being a stable platform for some time to come, now is the time to consider next generation architecture
- With UPAN reference design starting to materialize with Stratum being its DP, now is the time to consider its CP
- Goal is to establish the next generation SDN controller architecture
  - completely in the open and with the help of the ONOS community
  - kick of at start of 2019
- Continue to curate ONOS 1.x & 2.x maintenance and releases
  - core team to focus solely on bug fixes, code reviews and release engineering
  - ONOS community to continue new feature development

NG ONOS Architectural Tenets

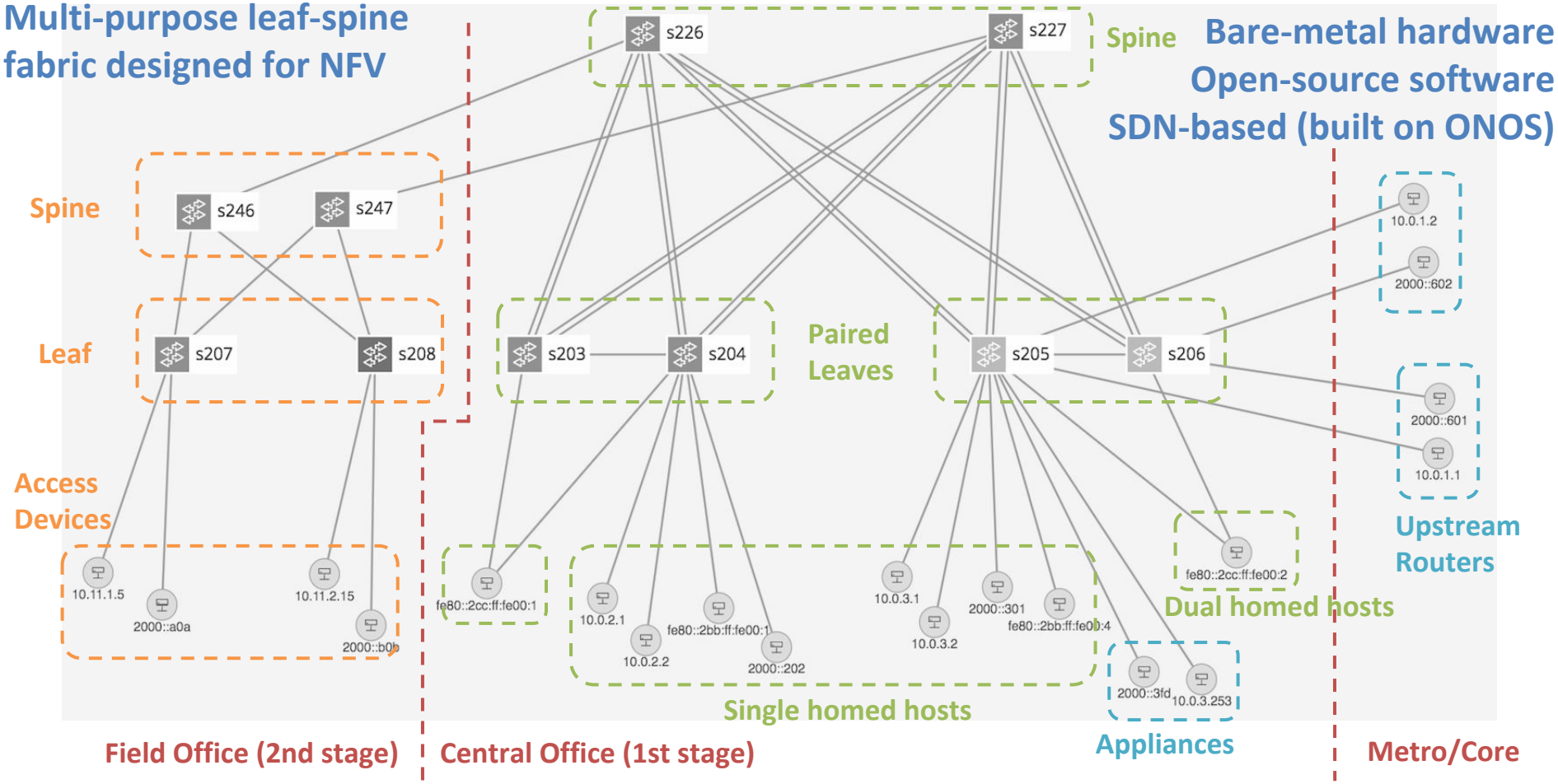
- Use gRPC-centric interfaces
  - gNMI, gNOI, P4Runtime, OpenConfig, etc.
- Follow micro-services principles
  - horizontal scaling of services, support for tenant apps, etc.
- Rely on existing orchestration platforms
  - e.g. Kubernetes, Helm charts
- Reuse code as appropriate
  - e.g. Atomix, GUI, protocol libraries
- Focus on features required for production deployments
  - live update, diagnostics, monitoring, integrations with orchestrators, etc.
- Allow components written in different languages
  - Java, Go, Python, etc.

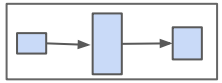
Trellis 2.0

Saurav Das
Director of Engineering, ONF

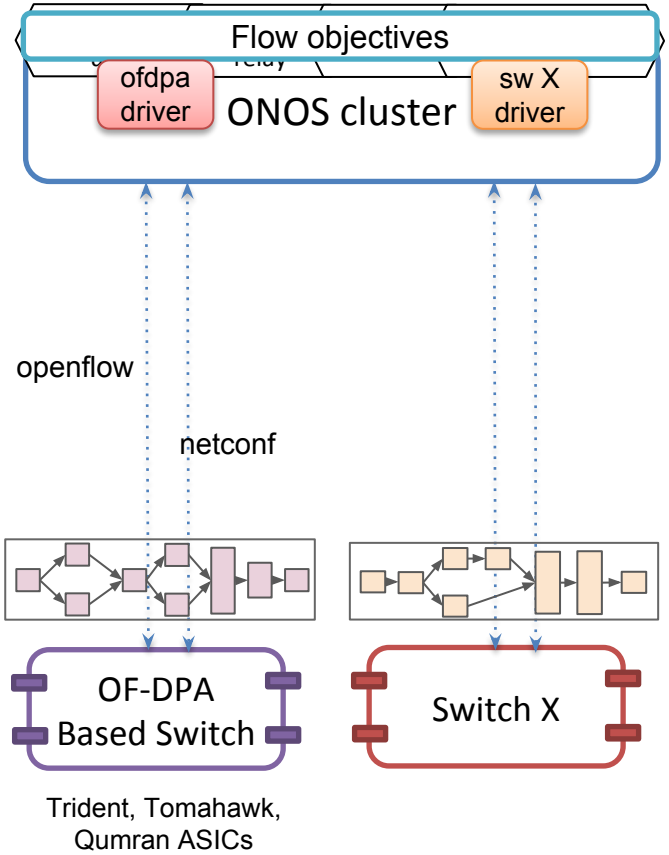
December 5th, 2018

Trellis Overview

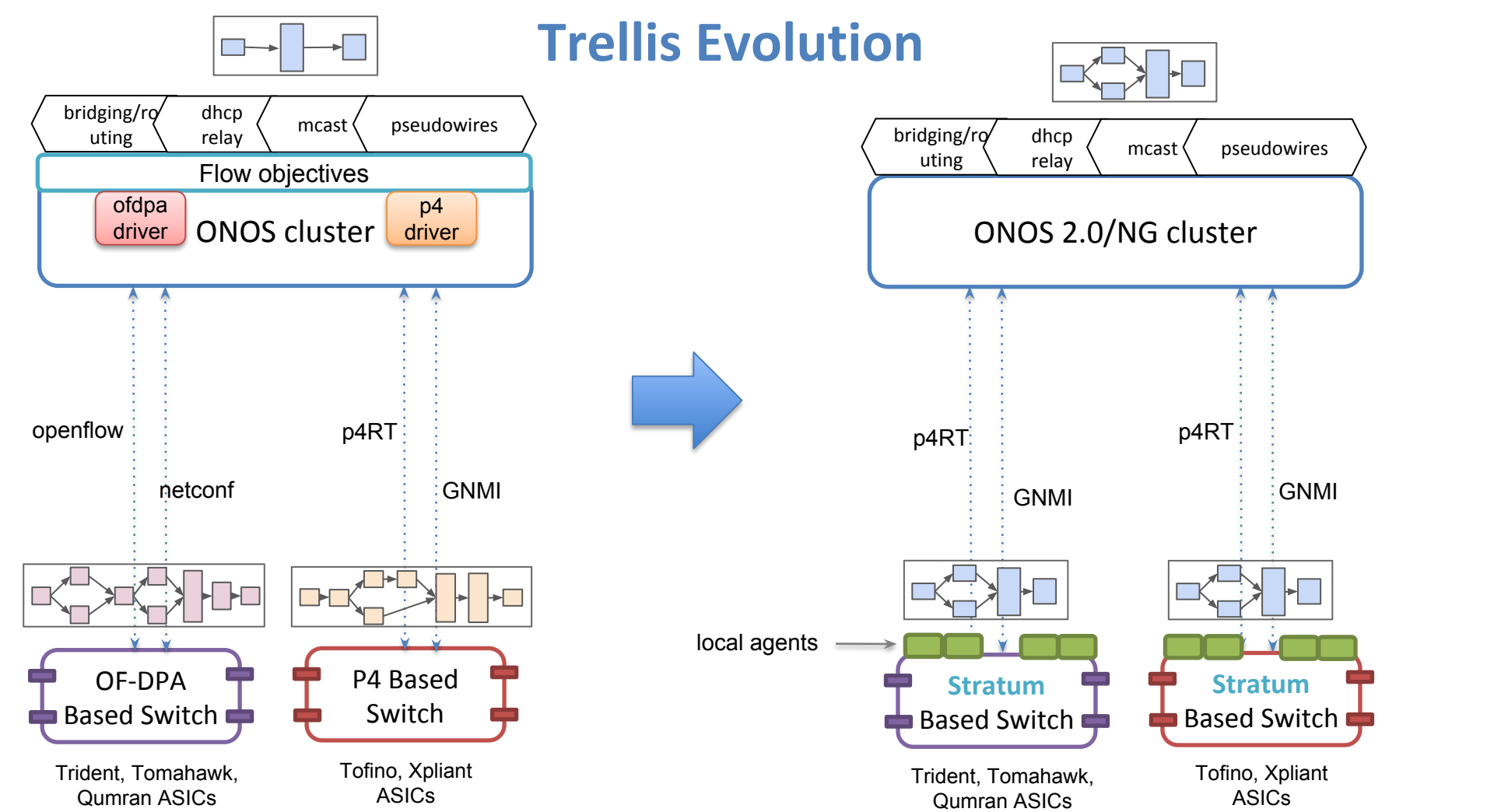




Trellis Evolution



Trellis Evolution



UPAN PANEL SESSION

INPUT FROM DT AT ONF CONNECT 2018

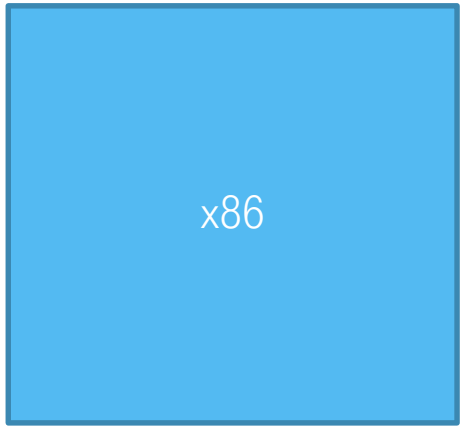
HJ Kolbe, Deutsche Telekom

LIFE IS FOR SHARING.

NETWORK FUNCTION EVOLUTION...

... ONE RING TO RULE THEM ALL?

0 - 2012



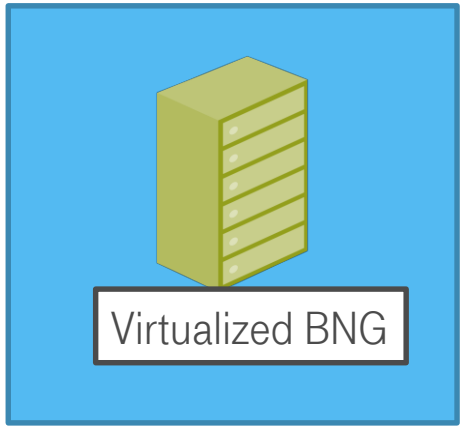
2012 - ~2016

~2016 -

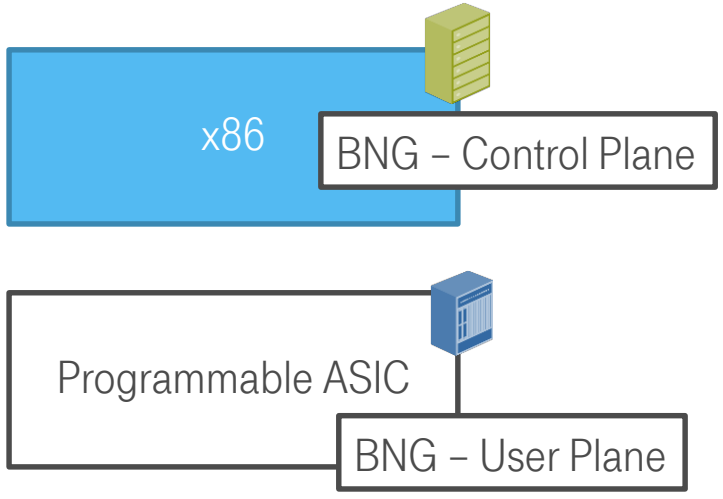
BNG EVOLUTION...

... MORE REALISTIC

- 2012



2012 - ~2016

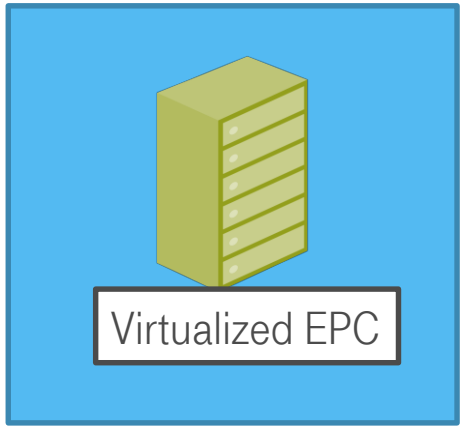


~2016 -

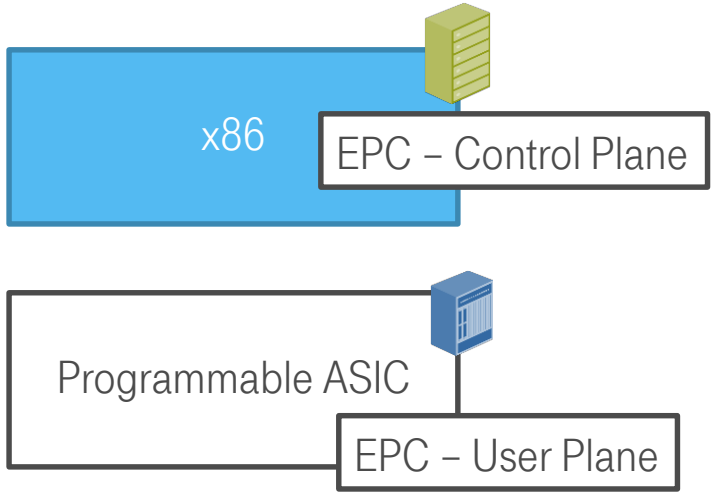
EPC EVOLUTION…

… MORE REALISTIC

- 2012



2012 - ~2016

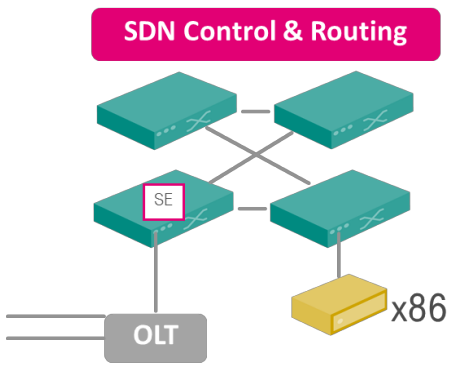


~2016 -

OFFLOADING OF THE USER PLANE
CONCRETE EXAMPLES

Fixed Access

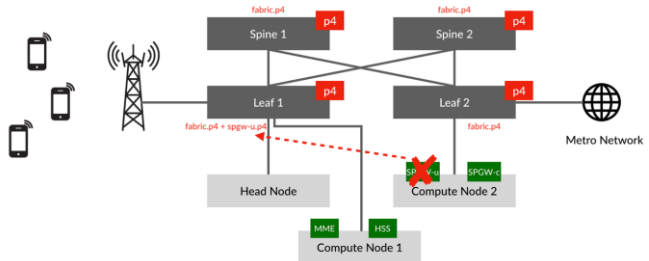
- Offload BNG user Plane to a programmable Switch
  - Ideally TOR switch (“anyway there”)
- Steer traffic according to SEBA blueprint
- Prototypes ready at DT and in early trials
  - P4 code shared at ONF



- Works!
- Integration to SEBA needed
- Productization ahead of us

Mobile Access

- Similar approach based on CP / UP split defined by 3GPP
- ONF demo at MWC 2018



- Challenges ahead -> see presentation from Manuel Paul “Use Cases And Opportunities With M-CORD”

TOWARDS A COMMON SUBSCRIBER EDGE

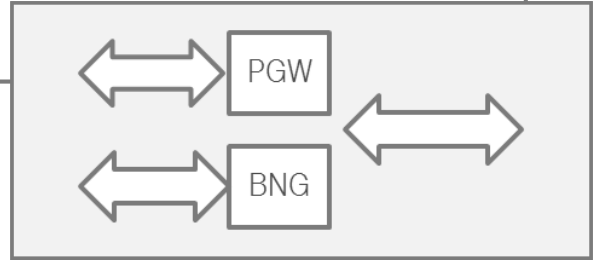
STRUCTURAL CONVERGENCE

FUNCTIONAL CONVERGENCE

Location consolidation

Traffic grooming, local coupling

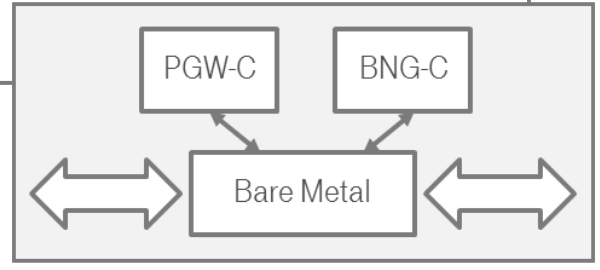
- H-CORD (hybrid)
- Edge Cloud
- Low Latency



User Plane consolidation

Same data path for fixed and mobile user plane (UP)

- Bare Metal-based UP,
- virtualized CP



Control Plane consolidation

converged control plane

- includes slicing
- Following 3GPP+BBF work