Hyper-scaling NFV

and getting a competitive Edge

Christos Kolias

Orange Silicon Valley

Open Source Networking Day

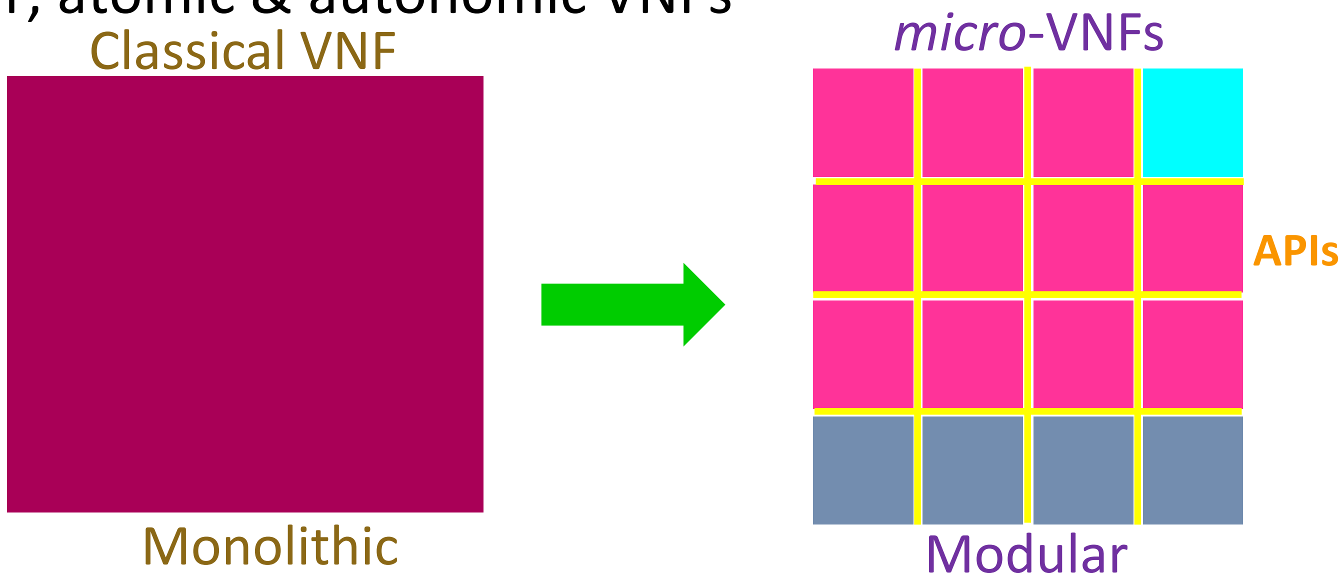
November 1, 2018 – Santa Clara, CA

Legacy (aka old-fashioned) VNFs

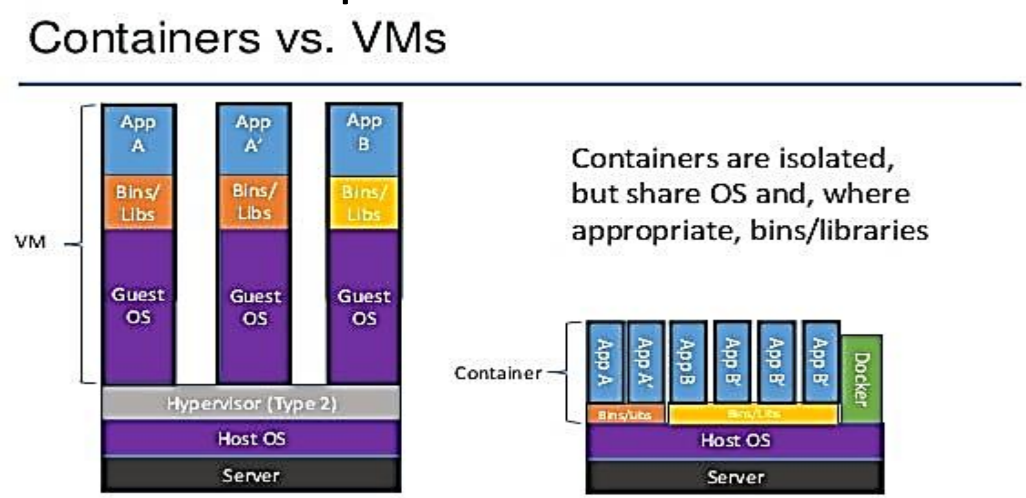
- First VNFs were simply the software version of the hardware equivalent (physical appliance)
- Quite monolithic
  - designed (originally) for specialized hardware
  - same service logic
  - required many resources (CPU, storage)
  - not optimized for cloud operations and virtualized environment, often complex
  - absence of (open) APIs
  - still very vendor-oriented
- Not easily or very scaleable (up and out)
- Simply fit the classic way of architecting networks
- However, in order to take full advantage of NFV (and SDN) new approaches (and thinking) are needed

μ-VNF

- μVNF: scaled-down, lightweight version of a regular VNF
- Small piece of code that can run in a small, compact factor
  - hardware
    - small single-board computer, i.e. Raspberry Pi
    - microserver/SoC, multicore
    - smart mobile device, IoT device, etc.
  - software & virtualization
    - containers
    - linux
- Based on a completely modular, decentralized architecture, deconstruct and functionally decompose monolithic VNF into smaller, atomic & autonomic VNFs

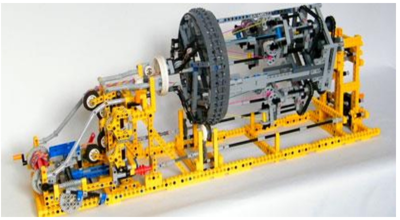


- Properties
  - execute a very specific task, perhaps only one
  - transient in nature (potentially short-lived)
  - scale dynamically and independently
  - don't have to be collocated
  - easily permutable, re-usable , replicable
  - may not be shared amongst service chains
  - seamless & stateless execution
  - could *mutate*, morph at run time (depending on current conditions)
  - a regular VNF can be broken down into as many µVNFs the designer may desire (and is feasible)
- Purposeful customization (per app, user, device, etc)
- Fine-grained computation and parallelism



hyper-scaling with μVNFs

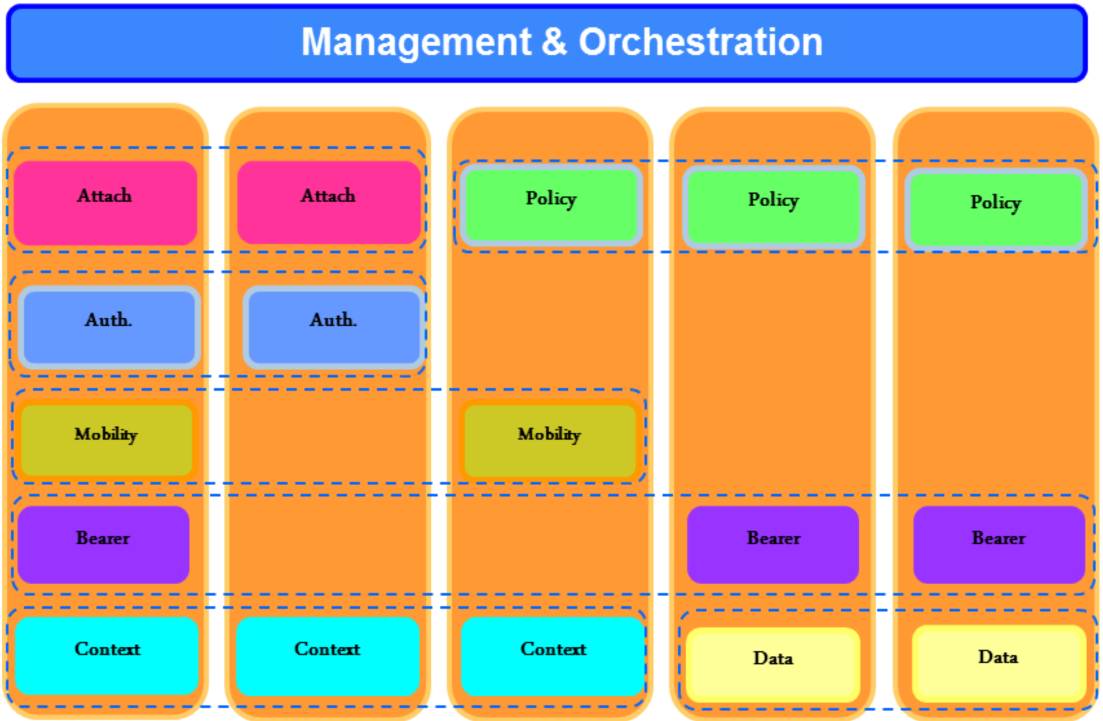
- Very dynamic chaining of μVNFs (compose/decompose)
  - coupled only during run time
  - per service instantiation (customizeable)
- Definition of VNF expanded to include functions such as policy control (QoS, TE, traffic offloading, etc), analytics,…



- Building customizable, personalized apps & services
  - Lego-lization of NFV: interlocked building blocks
  - for instance, a mobile app can be part of a service chain
- A μVNF-enabled element essentially becomes a programmable device
  - policies pushed by the network controller
- For scaleability, only those μVNFs that need to be scaled are replicated, not entire system
  - redundancy becomes easier, more efficient (and less expensive)

- Will require different type of orchestration as to deal with the potential enormity of μVNFs
  - possibly in a hierarchical fashion
- Automation
  - on-demand instantiation
  - auto-scale elastically
  - self-monitoring
  - self-learning
  - self-healing
  - auto-recovery
  - zero touch provisioning & deployment

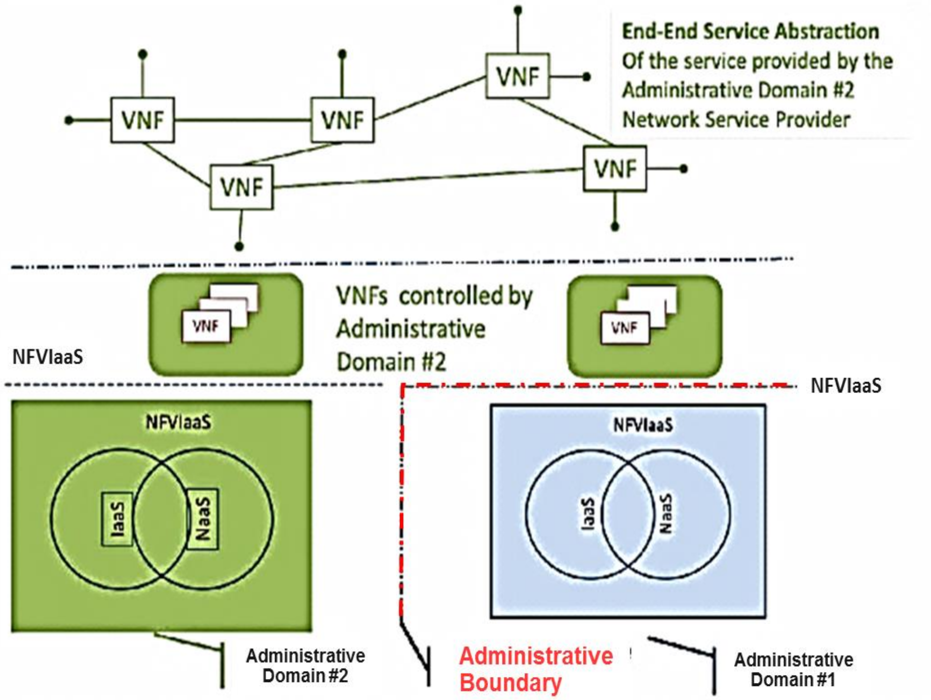
- Example: vEPC
  - node disaggregation
  - service-oriented than stack-oriented



- Example: vIMS
- Example: SD-WAN & vCPE
- Example: micro-probes for telemetry, QA
  - like benevolent *viruses*, micro-organisms flowing around in the network
- Example: TCP optimization, video caching

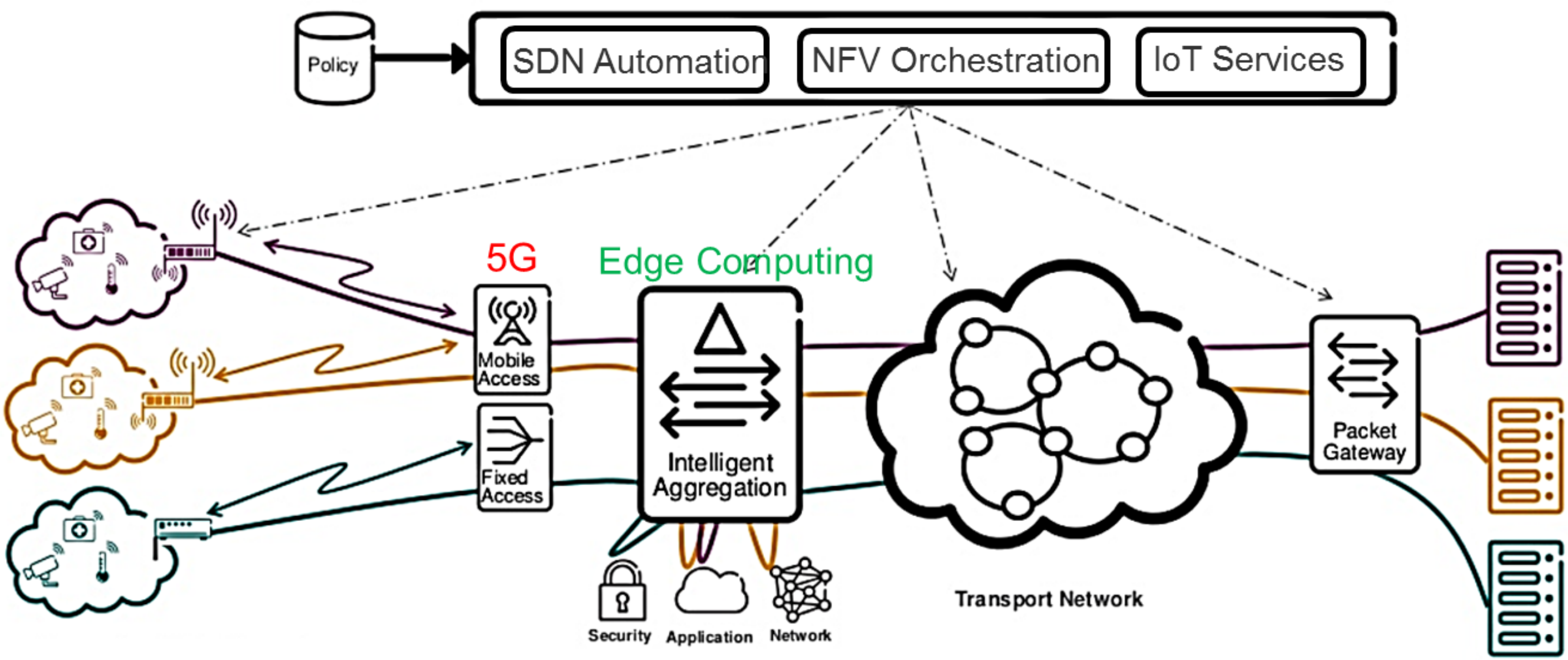
μVNF for mobile

- Deploy μVNFs at the mobile edge
  - can perform various functions: security, DPI, filtering, monitoring, load balancing, QoS, etc
- Deploy μVNFs on the mobile UE itself
  - filtering, QoS, load balancing
  - can be application-based
- VNF Roaming
  - use another provider's VNFs: *NFVIaaS*
- vEPC, vRAN functions
- Ideal fit for MEC
  - especially if you want to install it on a light post!



μVNF for IoT

- An IoT end-point (device) can run one or more μVNFs
- NFV orchestration can play an important role in orchestrating heterogeneous IoT elements
- A dedicated IoT network (slice) could have its own “cloned” micro vEPC instance, potentially over a shared vRAN platform



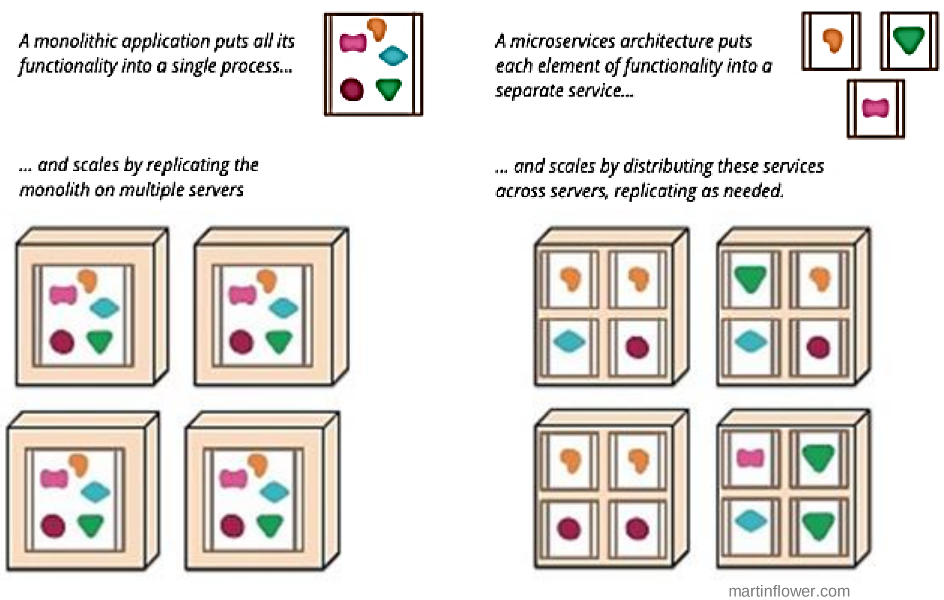
source: Ericsson

μVNF and AI/ML

- Deploy μVNFs throughout the network (core, edge, access) that perform ML tasks
  - μVNFs are "live" entities that learn and improve on by themselves
- A *neural network* of μVNFs
  - security, forensics (root analysis, fraud detection, etc)
  - edge computing (predictive maintenance, etc)
  - mobile networks (vEPC optimization, etc)
- Orchestrating and synchronizing the μVNFs
  - parallel and distributed process (think of a MapReduce approach for data traffic)
  - results are fed into the policy decision engine (that could reside in the network controller)
    - determines new, adapted policies, as required
    - allows for local (i.e., edge) decisions to be made (for latency and urgency reasons)
  - could be deployed around dynamically and on-demand

microservices

- Microservices: lightweight, autonomous, modular software components that run in a distributed fashion on cloud (ie, across multiple servers)
  - typically run in containers (Linux, Docker, etc)
  - communication via language-agnostic APIs
  - agile approach for DevOps. Easy to update.
- Opposite of monolithic architectural style that eg, traditional VNFs (as ported from physical systems) run

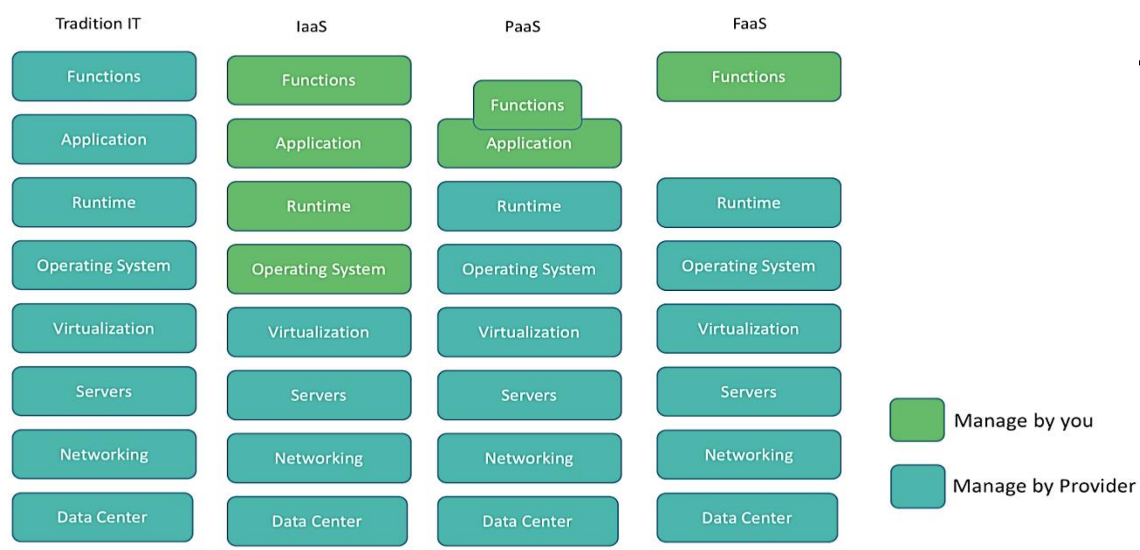


- Very modular architecture

- NFV and microservices are well suited for each other BUT are totally independent.
- A *micro* service chain (μSC) is synthesized of tens of μVNFs
- Micro service chains can support highly customized/personalized services
  - *dynamic* μVNFs: run for a specific purpose & time
    - ephemeral/instantaneous (set up & tear down)
    - spawn on demand, rapidly
    - eg: provisioning, voice/video session (HD, 4k, etc)
  - based on various criteria/preferences, such as user profile, location, device, connectivity, etc
- Applications are not monolithic any longer
  - micro service chains easily adaptive to changes in µVNFs
- Kubernetes for scaling microservices and for micro-orchestration
  - open-source platform for automating deployment, scaling and operations of app containers across clusters of hosts
- Example: *Quagga* router

VNFaaS

- Granular computing
  - very large numbers, swarms, of very small, tiny tasks (few micro secs duration) that have to be executed concurrently, very fast
  - mimic large-scale parallel processing
  - requirements: high throughput (low latency), cluster scheduling (orchestration), thread management, etc
- Serverless mode:
  - no provisioning/management of servers/VMs
  - cloud provider manages allocation of resources
  - dynamic pricing: pay for what you use (run as software)
- Build highly a n microservices
- VNFaas provi
  - examples: Azure Function, A



Native VNFs

- VNFs can be packaged as:
  - a virtual appliance
  - cloud-based
- Move away from the classical model of VNFs
- Cloud native VNF: solely cloud-based
  - hardware-agnostic
  - API-driven, leveraging open source
  - designed (from the ground up) to run exclusively on the cloud for elasticity, velocity and agility (EVA principle)
  - typically stateless, on-demand deployment
  - more automation, less human intervention
- Use μVNFs towards building native VNFs
- Cloud-native VNFs: building blocks for the telco edge cloud

Le *Nouveau* VNFs

- Spur further innovation (and imagination) for the "new" VNFs which will have to be application-tailored with intent-driven orchestration
- Examples & Opportunities
  - VR/AR

  - IoT

  - cryptocurrency, digital payment
  - autonomous devices: robots, autonomous vehicles, drones

- Fuelled by 5G
- μVNFs can eliminate the stringent requirement of 6 or even 5 nines availability
  - if a μVNF fails, rapidly spawn another

Open Source VNFs?

- Do we need them? Is there a place for them?
  - ✓ Example: open source vEPC.
- What are the implications/concerns?
  - Security?
  - Support?
- Will they be free?
  - Or quite free as part of a package/platform (s/w, h/w)
- Open (and kind of standard) APIs are more important than ever!
- A NFV *apps store* for chaining VNFs into E2E microservices enabled on a network slice

μVNFs App store & services catalogue

- User-defined microservices: build your own microservice
- Services-on-demand
  - Ephemeral clouds
  - Customizeable
  - **Intent**-driven, fully automated (using AI/ML)
- Example: I want to provide a Private LTE network for 100 UEs & 100 IoT sensors

Add Access Mode

- [x] vCPE
- [ ] vEPC_1
- [x] vEPC_2
- [ ] vBS (CRAN)
- [x] WiFi

Add Cloud Services

- [x] vLB
- [ ] vRouter
- [ ] Video Optimization
- [x] Analytics
- [x] WAN Optimization

Add Security Guard

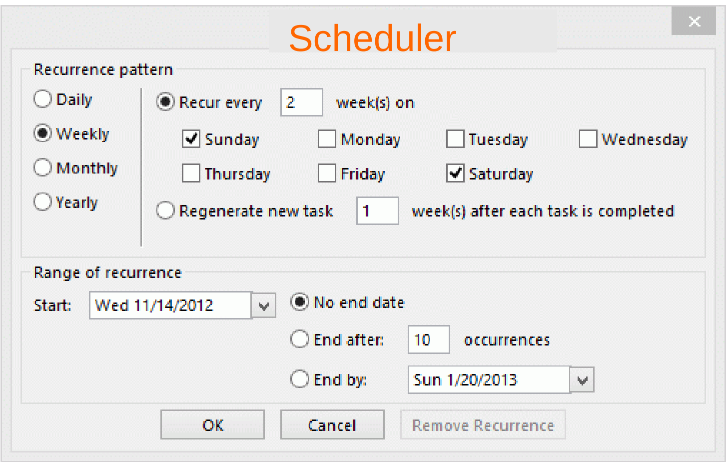
- [x] vFirewall
- [x] vIDS/IPS
- [x] Encryption
- [x] Policy Control
- [x] vDPI

Add Orchestration

- [ ] Orchestrator_1
- [x] Orchestrator_2
- [ ] Orchestrator_3
- [ ] Orchestrator_3
- [ ] OpenStack

Add SLA

- [x] 99.999%
- [ ] 99.99%
- [ ] 99.9%
- [ ] 99%
- [ ] Best Effort



- In an ideal scenario, software (ISVs) could swapped out real-time
- In an ideal world providers could be bidding for offering micro-services

µicro-edging

- NFV has been driving a lot of the edge computing efforts, primarily due to vRAN, vEPC, vCPE.
- Leveraging the Edge for building scaleable services using *µ*VNFs
  - *from the POP to the CO to the Base Station and beyond*
- Compact, agile VNFs running on the network edge
  - generic hardware, eg, x86/ARM-based microserver
- Real-time, dynamic microservices on the intelligent edge
- Customizeable experience for the end-user
- 5G could be the catalyst for edge applications

Ready for *nano-VNFs* ?
and *nano*-services

netsquared studio

christos.kolias@orange.com
www.orangesv.com