Network Edge structure, latency & ONAP services Considerations

Pasi Vaananen, Red Hat

30 May, 2018

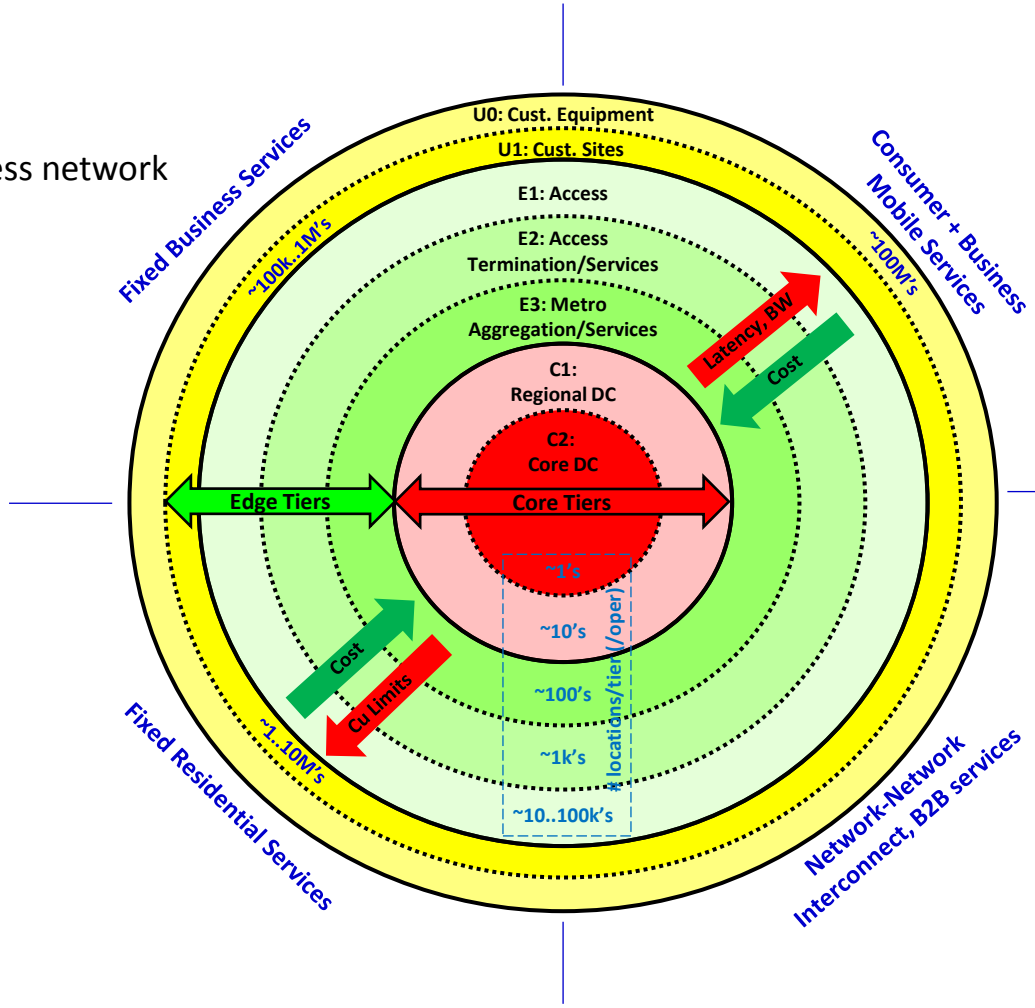
Outline

- Network Structure and its implications
- Why stuff is required in / near the edge
- Latency basics
- Latency components
- Latency by process – application vs. control / management
- Work divisions Application / VIM / ONAP
- FM/AvM example
- ONAP and “real-time” operations & control loops
- What to distribute
- Recap / Takeways

Generic Network Structure

- Cu → Optical (xPON, ptpt, WDM)
- L2 (ELAN, ETREE, ELINE) & L2.5 (MPLS) access network
- L3 (IP – public & private; SD-WAN overlays)
- QoS / SLA guarantees → OAM
- Voice, Security, …
- CPEs → vCPEs (thin, thick)
- Typ. Wireless & Ethernet mix. @premises
- 100k's to Million+ locations

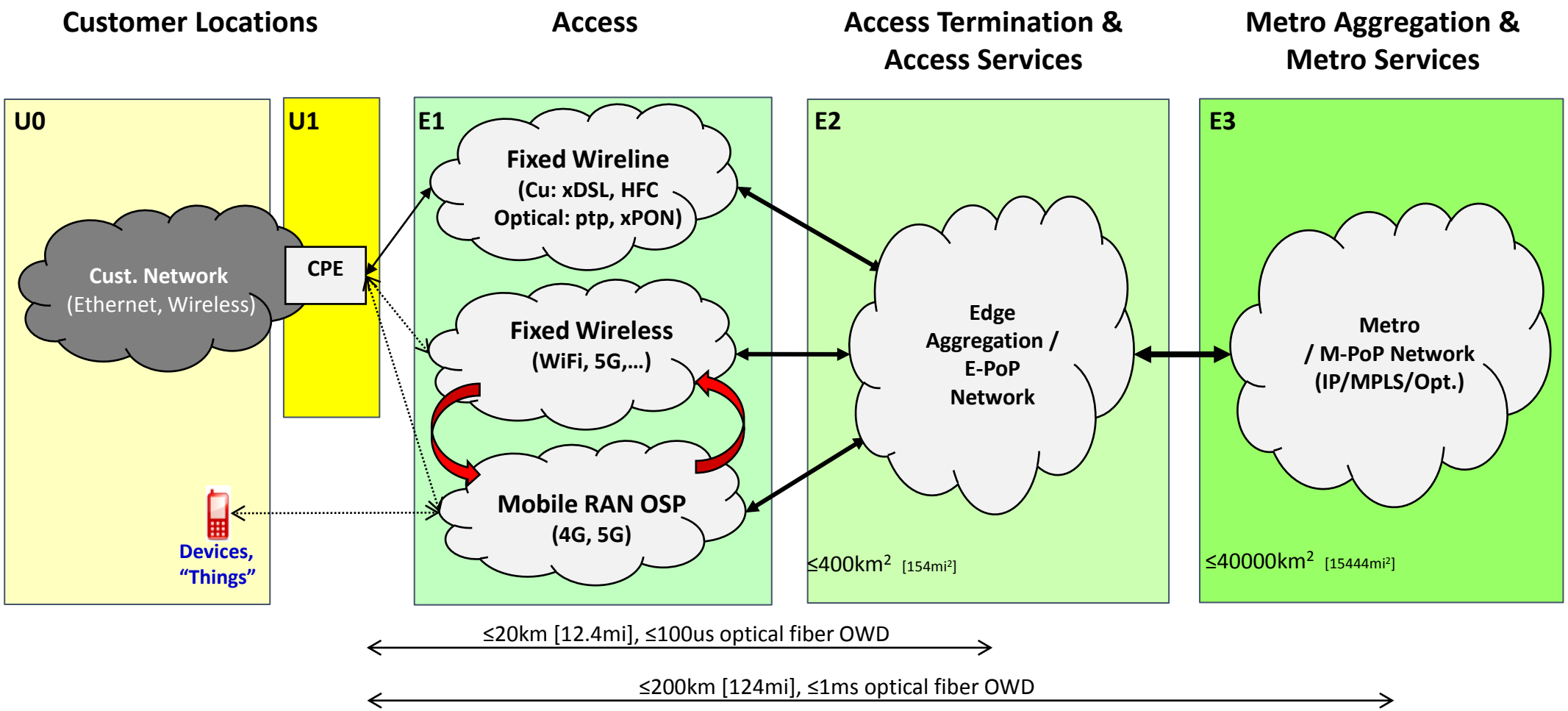
- Telcos: Cu(xDSL)→Shorter Cu→xPON
- CableCos: Cu(HFC)→ Shorter Cu→xPON
- Fixed Wireless (solve last drop cost)
- L2 in access network (ELINE, ETREE)
- Services mostly over IP (typ NAT'ed)
- IP Multicast in access network (Video)
- Voice (VoIP/POTS), Video, Data
- CPEs: NIDs, STBs, WAPs, routers
- Complex @home networks
  - Wireless preferred
  - Coax: MoCA; twisted pair: xDSL
  - Structured Cu: Ethernet
- 1's to 10's Millions locations



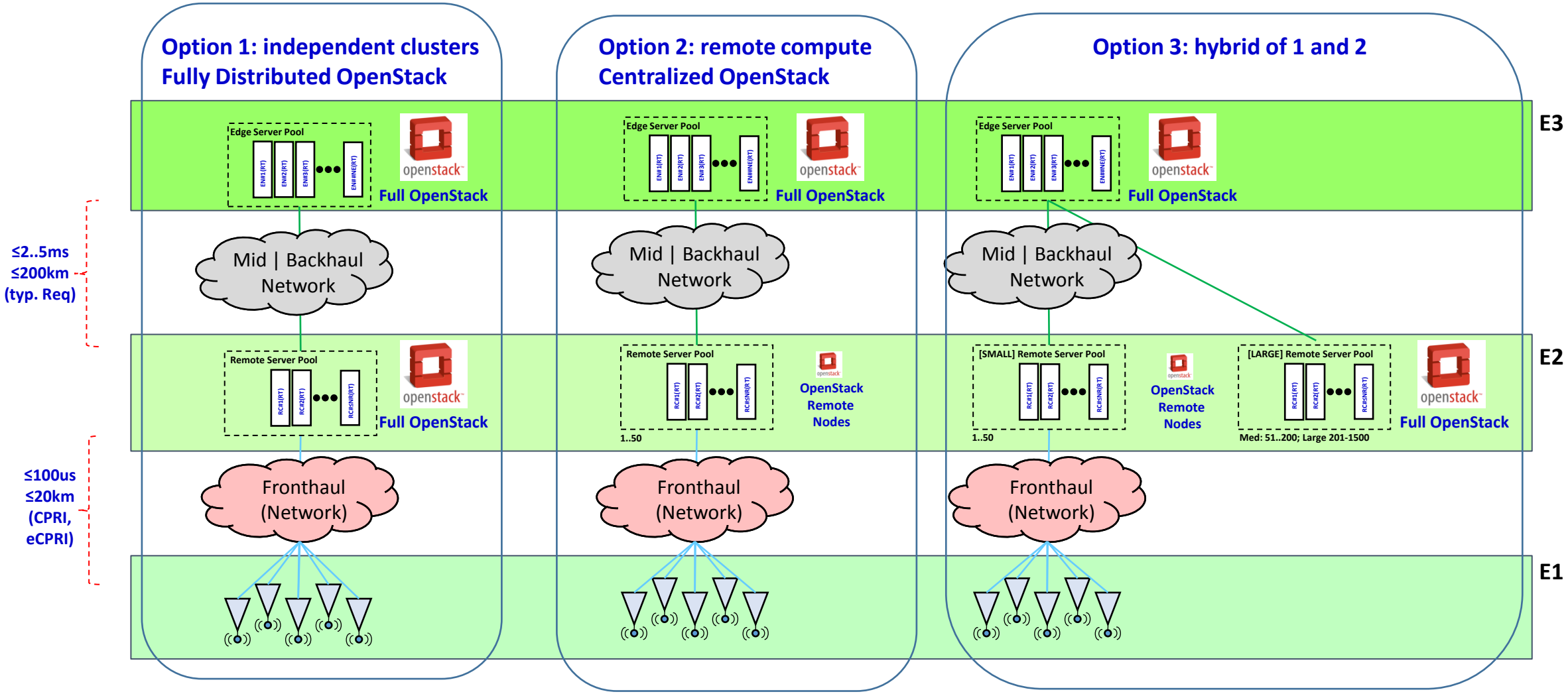
- 4G → 5G
- ≤3Ghz →+ ≤6Ghz & ≥28+ Ghz
- Few large cells → +many small cells
- ELINE, ELAN;
- tight QoS/latency SLOs
- Data, Voice, SMS
- Services mostly over IP
- People: 10's – 100M's subscribers
- IoT: 100M's of "things" (devices)

- Typically from core tiers
- Internet peering points
- Large customer connections
- Public cloud provider edge
- Network-Network interfaces (to peer operator networks)
- All about speeds
- Publicly routable IP connectivity
- Optional services (e.g. On MPLS)
- Service interconnect / roaming with bi-lateral agreements
- Transport services; simplicity & high speeds
- ≥100GBe; wavelength based services

Edge tiers vs. access technologies overview



OpenStack options @edge (vRAN use cases)



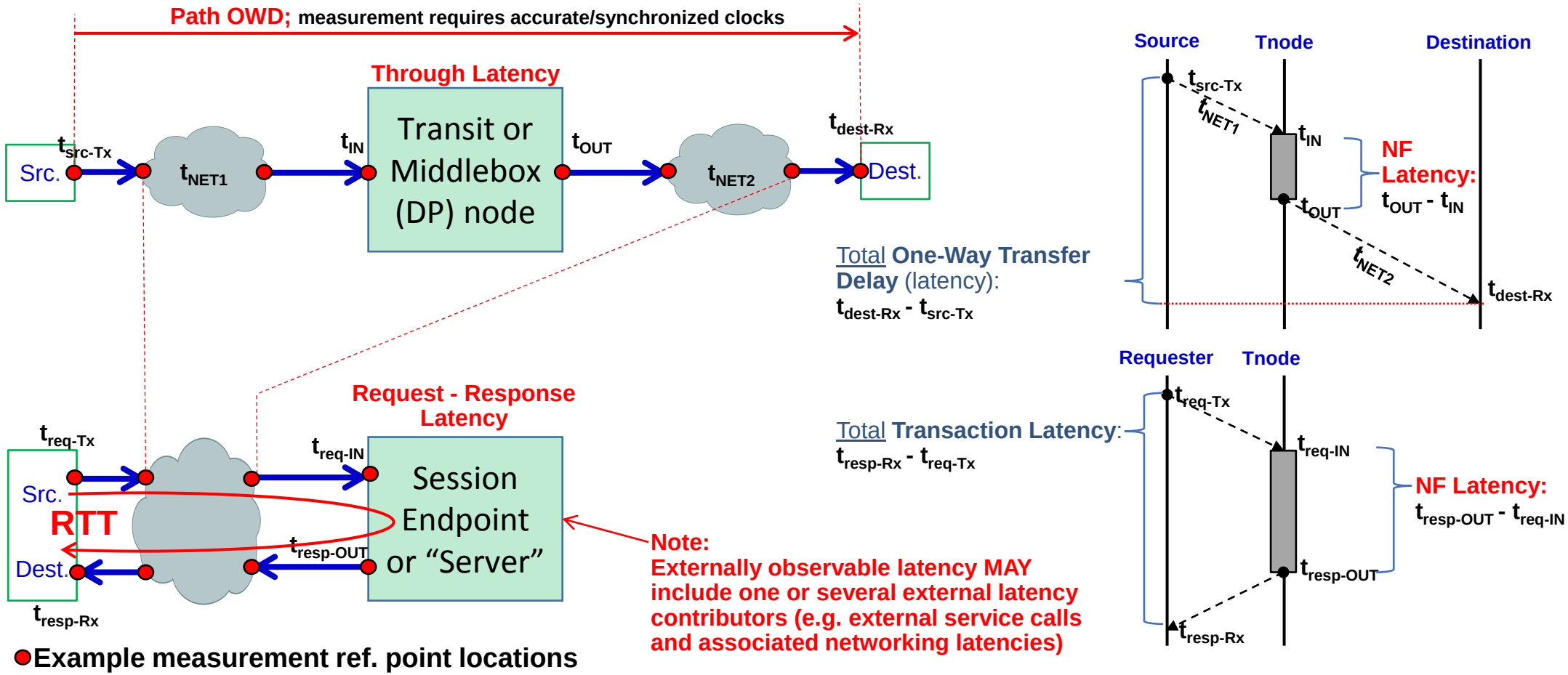
Why move stuff towards the edge ?

- Moving towards the edge has three categories of reasons:
  - Meeting constraints / reduction of **latency** (e.g. vRAN / low latency EC/MEC)
  - Reduction of backhaul / transport **bandwith** (e.g. CDN close to customers)
  - Distribution of load / improvement of resiliency (autonomy, smaller impact zones)
- Why NOT move stuff towards the edge ?
  - Extreme moves start increasing the cost (smaller equipment, reduction / elimination of pooling potential – ultimately “back to box” constraints)
  - Moving too much can start have adverse consequences – e.g. moving CDN servers too far start affecting the content cache hit rates to the level that it becomes uneconomical
  - If the latency is the driver, then move just as far “out” as required to satisfy the latency constraints
- “Distribute what you can, centralize what you must”

Latency types

- Latency is always to be specified as a time between two reference events, as observed at the specified reference measurement point(s)
  - e.g. “10ms latency+ by itself does not specify anything…
- Main cases:
  - Through latency (e.g. through the transmission channel, VNFCI, PE, …)
  - Response time latency; time from req to corresponding resp (e.g. signaling message pairs, msg res/ack, latency to “serve” something, …)
- Through latency component types:
  - Unidirectional: one-way latency of specific path (One-Way-Delay, OWD)
  - Bidirectional: Round Trip Time (RTT), composed of two uni-directional paths (and responder latency – typically embedded on remotely observed resp. time latency)
  - Note that RTT component OWD paths and/or associated latencies are not necessarily the same for both directions

Latency types illustration



Latency sensitivity (i.e. tolerance to added latency components) needs to be considered in the context of the application type and **total latency from endpoint's view** (either total transaction latency or transfer delay).

Common component latencies

Latency Type	Description	Fixed/Variable	Comments
Serialization / Deserialization	Time to send / receive packet over transmission channel	Fixed	Depends on the transmission rate of the channel
Coding	Time to add / remove medium specific codings	Typ. Fixed	Esp. for noisy / low quality channel using mechanisms such as FEC
Channel access	Time to gain access to transmission channel after having PDU to transmit	Variable	Applies to shared channels (xPON, DOCSIS, wireless, …)
Transmission	Propagation delay through the channel	Fixed	For single mode fiber, ~5us/km
Switching / multiplexing	Latency through the switch elements	Fixed (excl. queuing)	Typ store and forward latencies
Queuing	Waiting time in queues	Variable	Depends on load
Processing	Time to process a request / packet	Fixed (excl. scheduling / queuing)	May include external calls for req-response transactions

BOLD: generally applicable to all NFV use cases, others are primarily associated with access networks

Management-Control Continuum (MCC) ?

4.1. The Management-Control Continuum (MCC) concept

Source: TMF IG1118

In the FMO architecture, it is considered that the traditional demarcation between management (OSS/NMS/EMS) and (network-based) control functions is no longer valid. Even in today's solutions, management functions that cooperate with control functions are deployed both outside and inside the network[7]. Thus, there is nothing that forces management to be centralized or control to be distributed. The components of the management-control continuum should be deployed to whatever degree of distribution is required and this may vary over time etc. The distribution will involve varying degrees of collaboration, etc.

Other than annoying things like speed of light…

At least the document ack's the Basic facts

As an example, the Management-Control Continuum can spread from current OSS level driven configuration of policies for per-packet processing and potential packet discard, through 50ms protection switch, to actively rebalancing the network to allow for expected future demand. Each of these activities involves management or control in that there is a communication of requests for change based upon analysis of conditions.

Notes:

- yes, the boundaries between management and control are getting increasingly intermixed / irrelevant
- no, it does not mean that they do **not** exist – performance and latency related constraints and implementation related implications remain real (and in the many cases are reflected in the design of the associated interfaces etc.)

Common Application (VNF) level rates / latencies in NFV

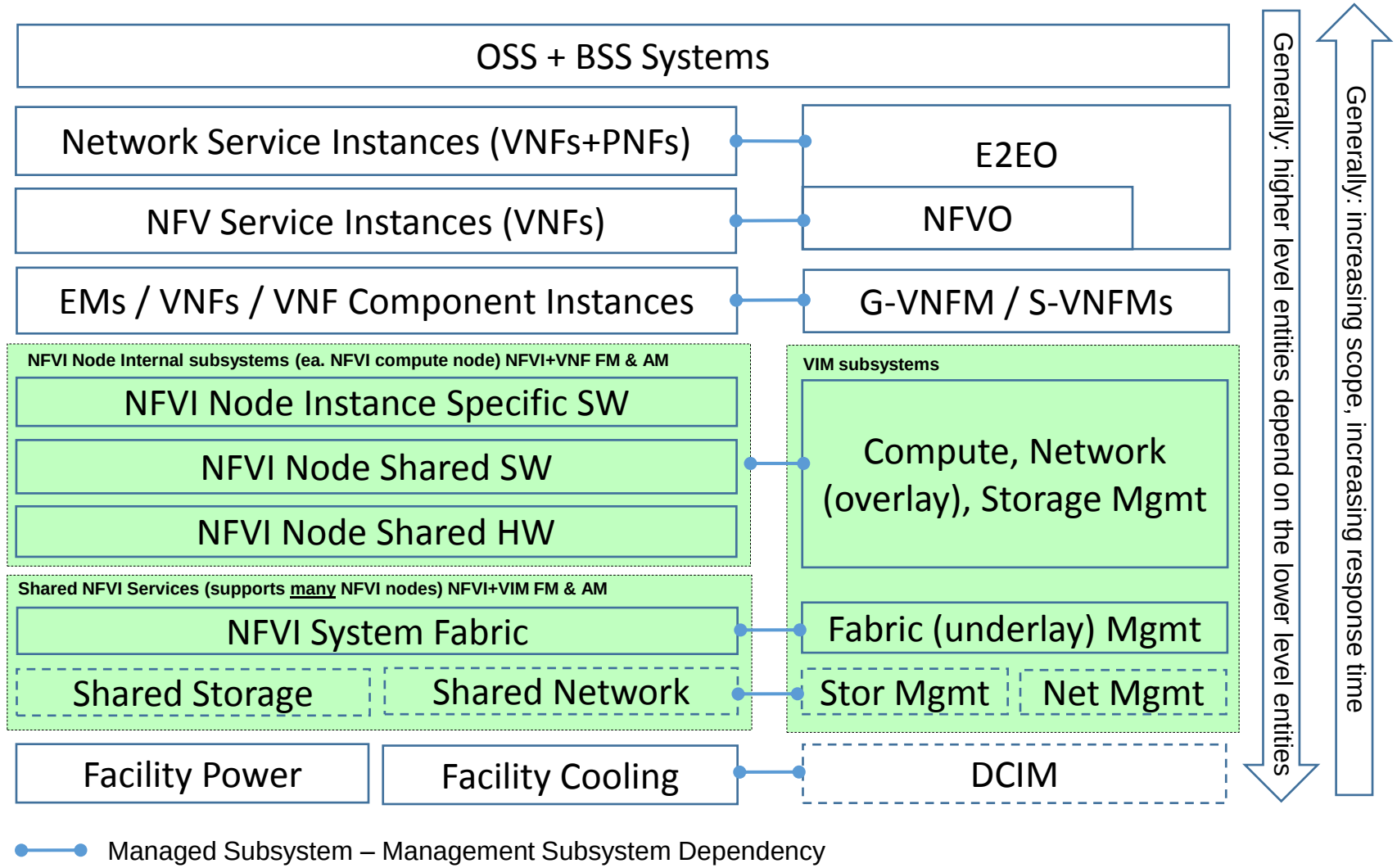
Type	Description	Rate / Latency (per interface or instance)	Comments
Management Plane Operations	Management system driven changes	1k / O(ms's-seconds)	Typically ASCII / Restful encodings on protocols / APIs
Control Plane Operations	Protocol driven changes	10k-100k / O(ms's)	Typically binary encodings on protocols
Flow level operations	Traffic driven flow-level operations	100k-10M / O(us's-ms's)	For each new traffic flow & flow state retirement
Packet level operations (e.g. VNF DP ops)	Time to process a single packet arrival	1M-100Mpps / O(ns-us's)	Run-to-completion process

- Arrival rates and processing latencies are related, but decoupled
- For control plane interactions, aggregate rates can be very high when user population originated (e.g. mobile network interactions), typically less frequent for the network internal controls
- For packet level processing, while typically run-to completion, processing latency is determined by parallelism in SW (HW) implementations (i.e. single thread performance is key determinant on single pkt level latency in SW)
- For control and esp. management plane processes, relationship is more complicated – includes both parallelism of the serving process level but also potentially many other external component interactions as well (uServices)

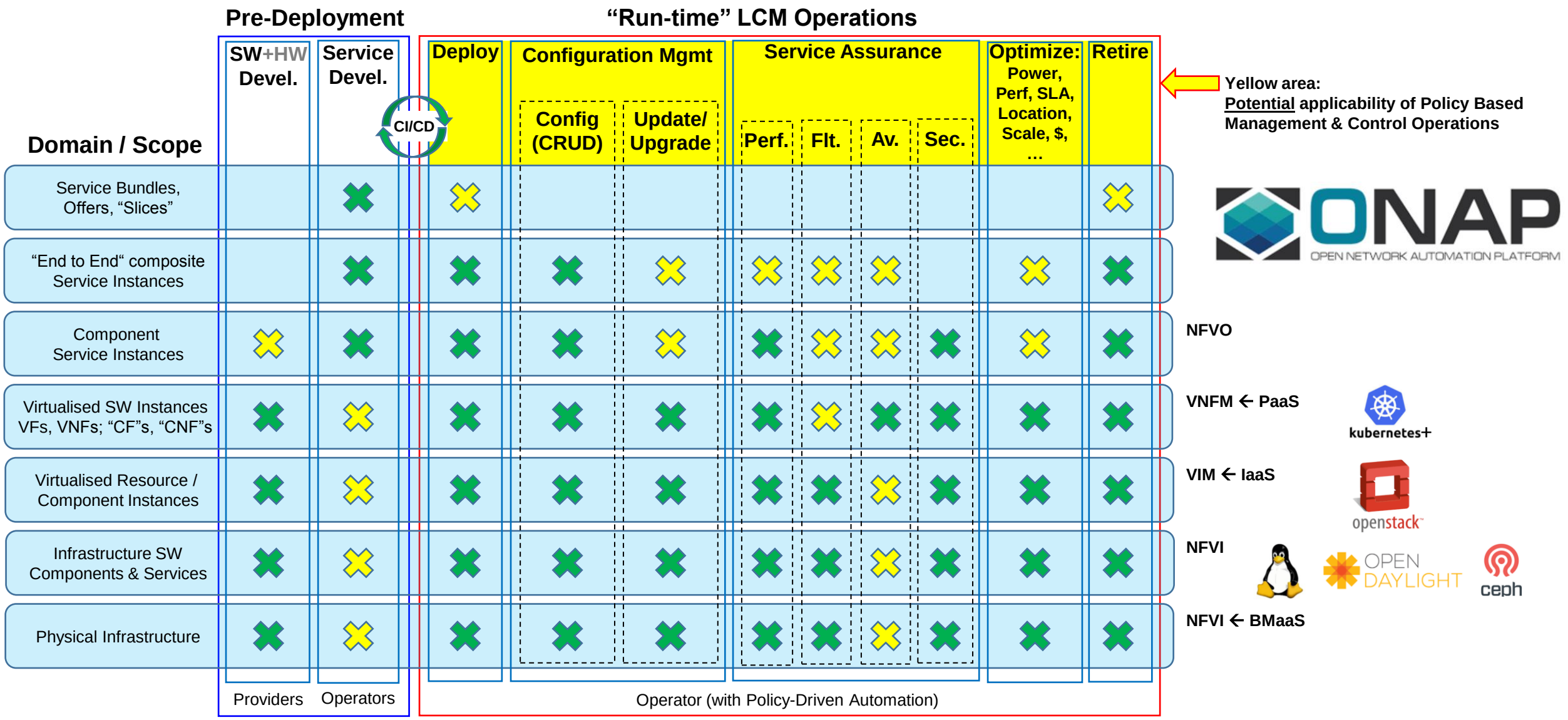
Latency recap

- Latency has to be associated with SOMETHING i.e. scenario w/ reference points
- Application level latencies
  - Primarily applications problem, but:
  - Latency constrained VNF/VNFCI placement (e.g. how close to access an entity needs to be placed) requires exposure to capability to place entities **based on some latency constraints**
  - Above IMPLIES that there needs to be some information available about latency characteristics within the system to be feasible
  - Measured latencies: latency between reference points is based on active measurements by the infrastructure elements (but ref. points may be outside of the managed scope in the worst case – e.g. UE to eNB L1/L2 processing VM) !?
  - Implied latencies: use topological relationships as a proxy (e.g max # nodes from…)
- Infrastructure / ONAP service latencies
  - This should be the primary focus; related but decoupled from the application level latencies
  - Need to get some ideas on targets in place to be able to do meaningful designs
  - Depends on the service and scope (increasing scope while moving “up” on hierarchy, but also increasing time)
  - Focus on the control loop latencies first (loop tightening has **lots** of implications on component service design and placement)

NFV System Dependencies (simplified view)

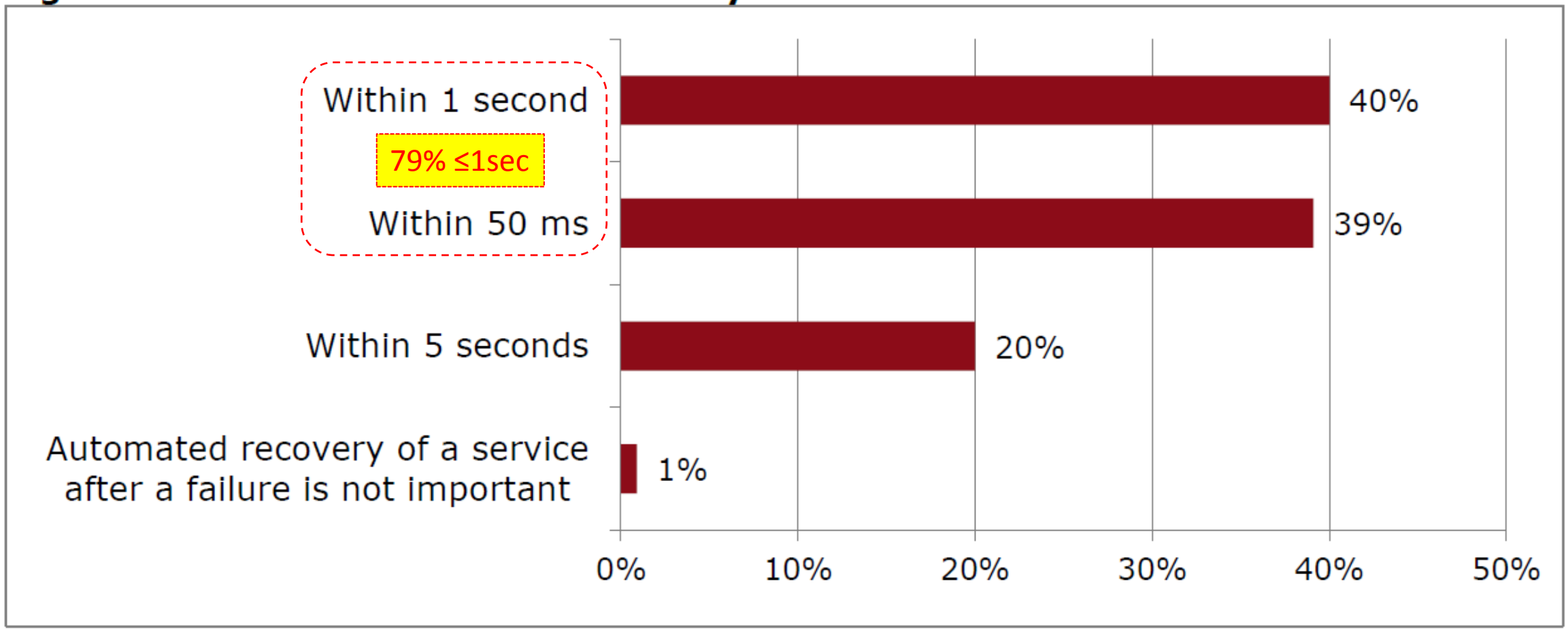


LCM overview



FM/AvM timelines example – an operator survey

Figure 13: Automated Service Recovery



Source: Heavy Reading NFV operator survey of 128 service providers, “Telco Requirements for NFVI”, November 2016

FM timelines example – ETSI REL / Verizon proposal

- "We assume that each High Availability (HA) layer depicted in Figure 1 has independent HA mechanism (i.e. failure recovery). Therefore, each layer has its own failure recovery timer as described below. The failure recovery time is the summation of the time for failure identification and the time for switching from failed entity to the health entity."
  - The failure recovery time for INF-L1 is **T11;**
  - The failure recovery time for VM-L is **T2;**
  - The failure recovery time for VNF-L is **T3;**
  - The failure recovery time for INF-L2 is **T4;** and
  - The failure recovery time for **CONNECTION-L** is **T5,** where
  - **T11 <T4<T2<T3<T5 (EQ.1)"**
- "EQ.1 may or may not hold all the time. The main objective here is to ensure that the timers are configured such that there is no race condition among layers. In other words, there must be adequate time gap between failure recovery timers of layers. These relationships between timers are further explained in Figure 5."
- "Time intervals between the timers above are desired to be within 100 msecs. Therefore, the desired relationships among the timers are:
  - **T4=T11+100 msecs; T2=T4+100 msecs; T3=T2+100 msecs; T5=T3+100 msecs (EQ. 2)"**

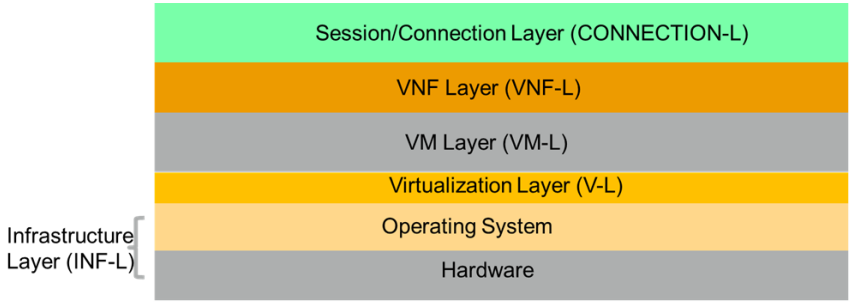
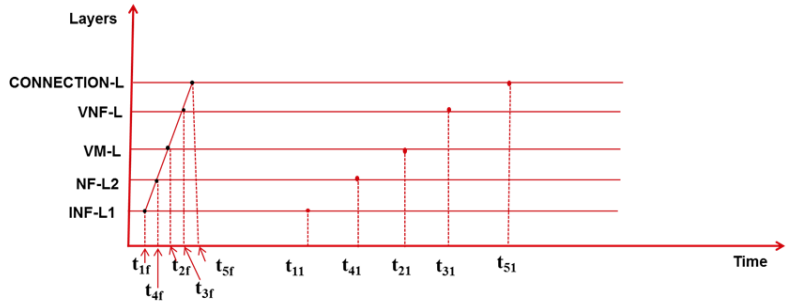


Figure 1: Layers of an NFV based system



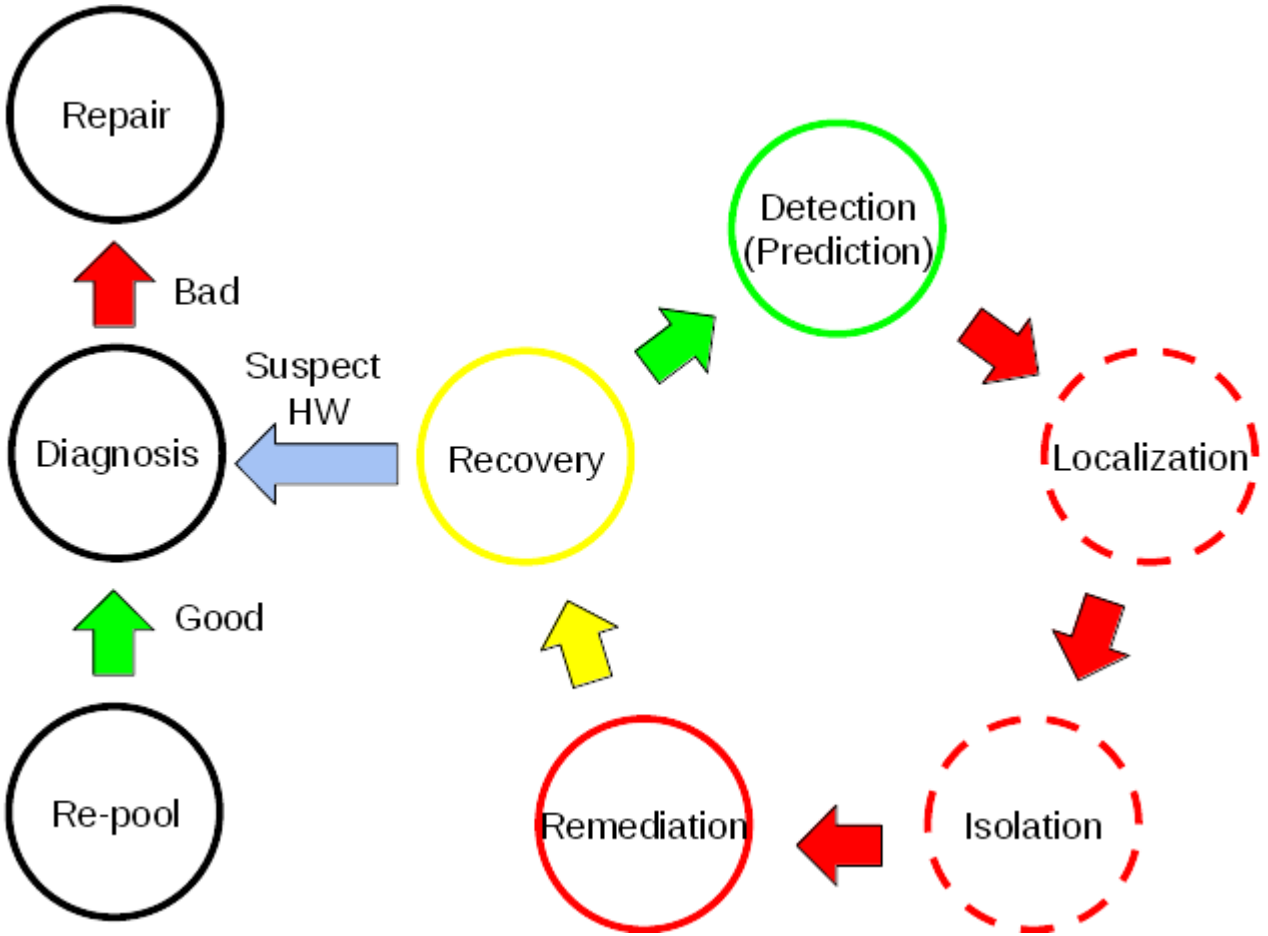
$T11=t_{11}-t_{1f}$; $T4=t_{41}-t_{4f}$; $T2=t_{21}-t_{2f}$; $T3=t_{31}-t_{3f}$; $T4=t_{41}-t_{4f}$; $T5=t_{51}-t_{5f}$

Figure 5: Relationships between Failure Recovery Timers of a Virtual System/Service supporting **EQ.1**

Notes:

- **While the times and "layering" (i.e. dependencies) are subject to discussion, overall this contribution represents a good illustration of the common general design pattern of coordinated multi-layer network recovery timing ("lowest layer first").**
- Source (Mehmet Toy / Verizon – ETSI REL): https://docbox.etsi.org/ISG/NFV/REL/05-CONTRIBUTIONS/2017//NFVREL(17)000120r3_CLAUSE_4_OF_REL008__ARCHITECTURE_FOR_ERROR_HANDLING.docx

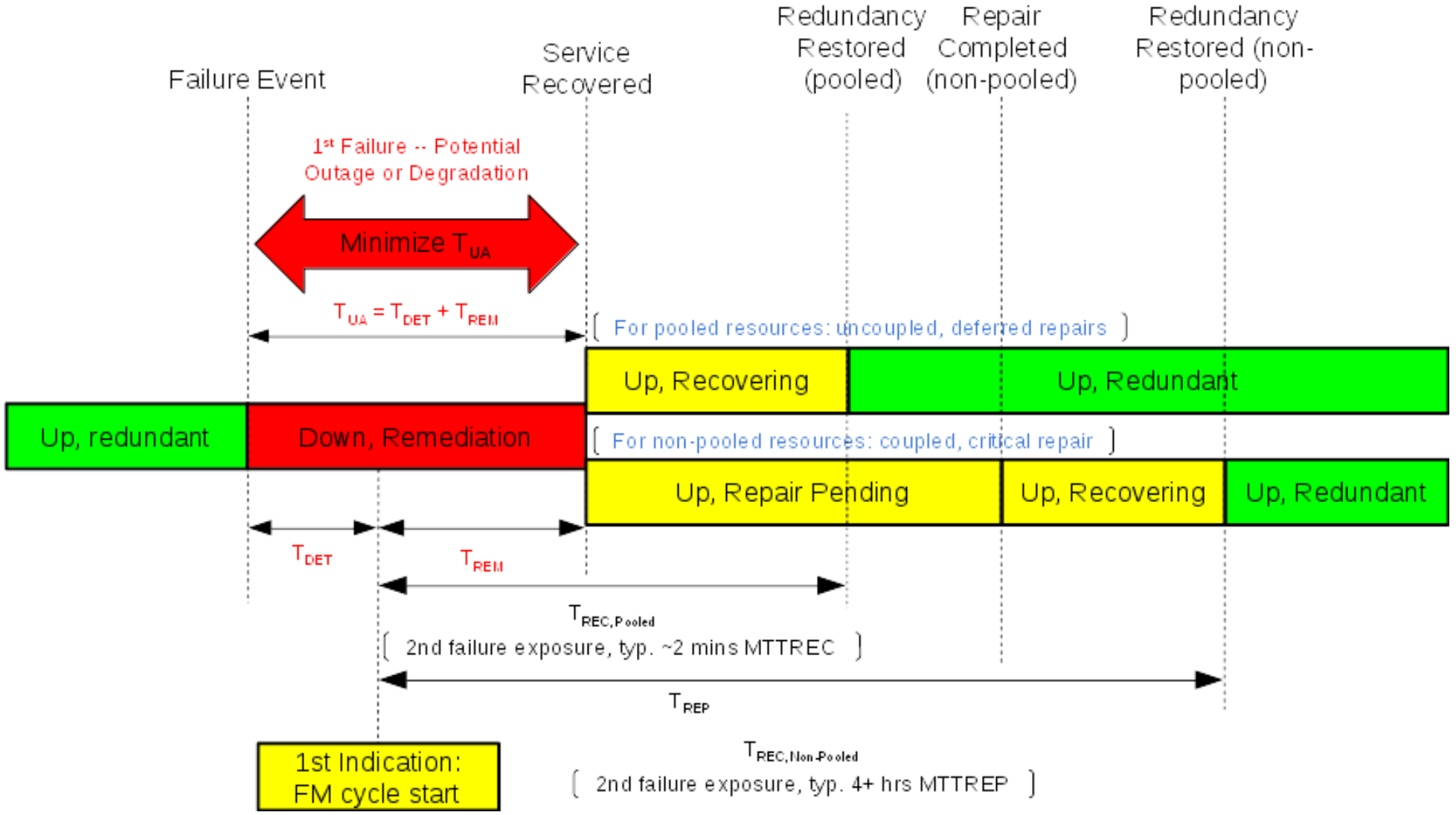
FM cycle


Repair
Bad
Diagnosis
Suspect
HW
Recovery
Detection
(Prediction)
Localization
Good
Re-pool
Remediation
Isolation

Generic Fault Management Cycle Phases (ETSI terminology)

- **Detection** – Low-latency, low-overhead, low false positives / false negatives mechanisms
- **Localization** – Physical/Virtualized resources to resource consumer(s) mapping within the context of fault trees
- **Isolation** – Remove the ability of the failed component to affect service state of un-affecged instances
- **Remediation** – Service restoration through failover to redundant resource / component, or component restart
- **Recovery** – Restoration of intended redundancy configuration

FM Cycle Timeline; key phases (generic)



FM cycle targeting

- $T_{DET} + T_{NOT} + T_{REM} \leq 50ms$ (for lowest layers in hiearchy – “network”)
- T_{DET} – Detection Time
- T_{NOT} – Notification Time

Minimize; but – single fault event (e.g. node failure) may require correlation and multiple notifications (e.g. one notification per manager of affected VNF)

- T_{REM} – Remediation Time

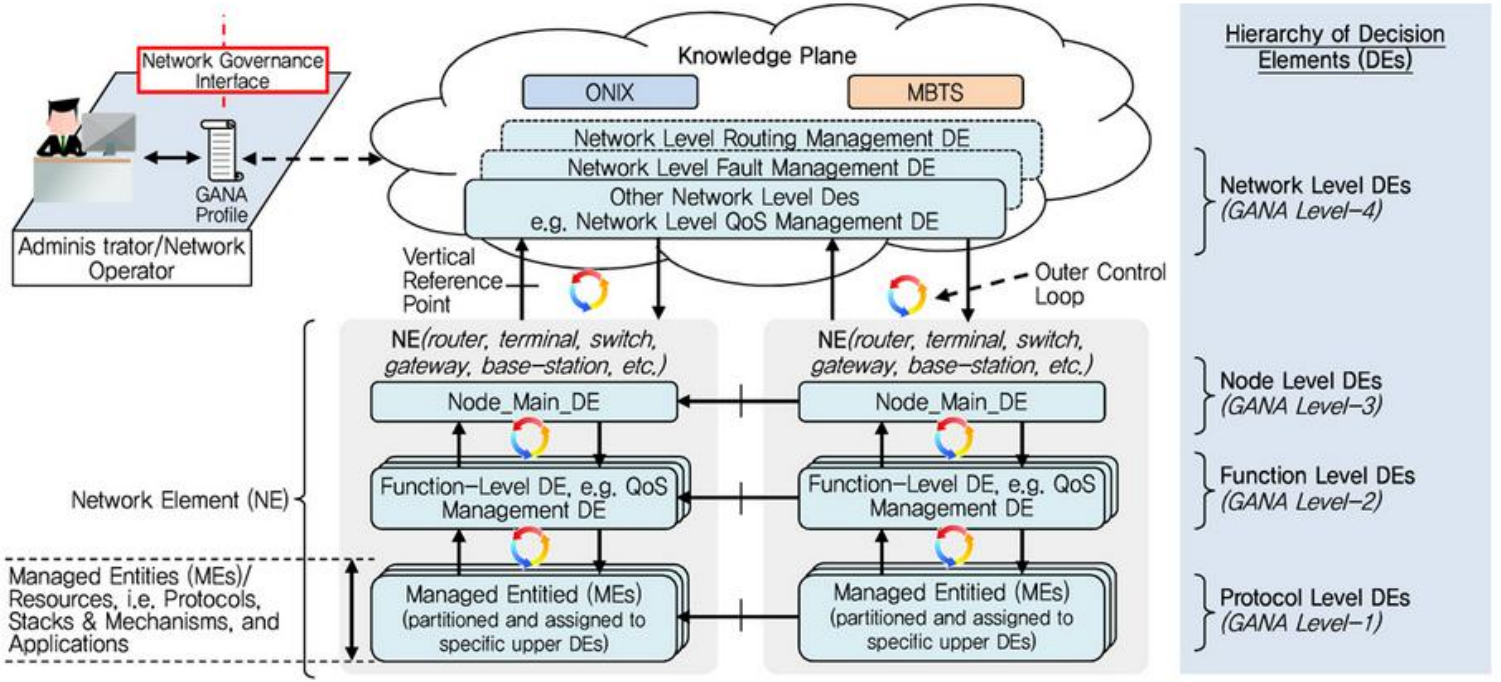
Often the longest process, not fully under control of infrastructure, may require multiple interactions from VNF to VIM levels

The high level goal is to minimize timeline from fault to service restoration – i.e. minimize the length of the observable service outage. Service **Availability** could in principle be managed by infrastructure services, but Service **Continuity** requires some participation by the associated VNFCI and availability management processes.

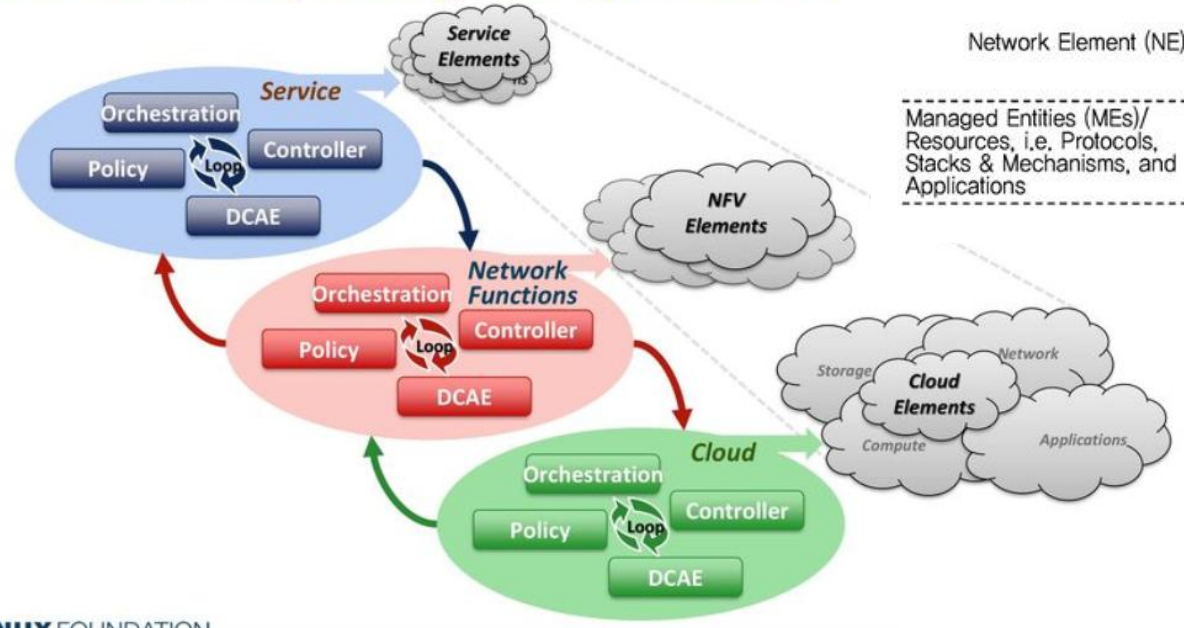
Some earlier work with the right basic idea ?

ETSI GANA Reference Architecture;

- **Hierarchy of policy control loops w/ Increasing loop scope & increasing time**



Closed-Loop Repeating Design Pattern

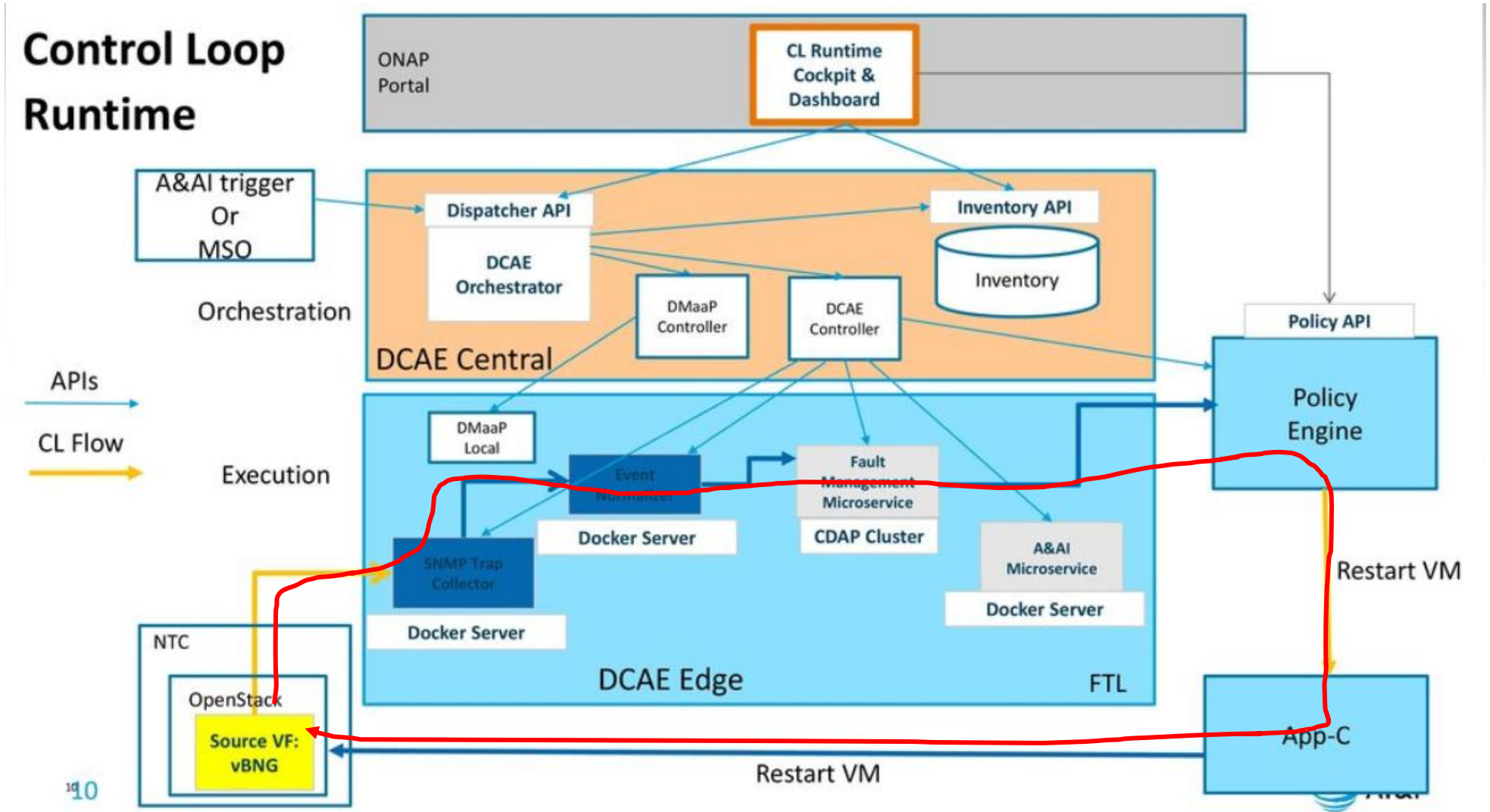


THE LINUX FOUNDATION

One of the early ONAP policy slidewares;

- **Hierarchy of policy control loops (w/ Increasing loop scope & increasing time ?)**

ONAP Control Loop Example (EVT→action)



Critical Path of the ONAP control loop (evt_src → evt collection → normalizer → FM → policy → controller → VIM

Centralization vs. De-Centralization candidates

- To remain in Central ONAP
  - All Design Time components
  - Service Orchestration (at least top level if hiearchical)
- To be distributed ?
  - Basically all **components required for autonomous Closed-Loop Control**
  - DMaaP/MSB service instances (don’t want to go thru central)
  - DCAE
  - Policy components
  - Controllers (SDNC, APPC/VF-C)
- Uncertain …
  - A&AI – Edge; depends on interaction details ?
  - Catalogs (subset only ?)

What are we doing in this area – shortlist

- OPNFV Barometer project: infrastructure events and metrics collection; exposure for higher level entities; node-local recovery policies
  - Goal is to integrate with ONAP policy management infrastructure to close the loop (FM, PM)
  - Need full fault correlation solution (OPNFV Doctor vs. Holmes vs. other stuff ?)
- Testing of Kafka vs. AMQP etc. for the messaging performance (latency)
- FEMDEC distributed Messaging with Inria + Orange collaboration for OpenStack Edge Use cases
- OPNFV VCO project: learn by doing – VCO 2.0 works on (LTE 1st) vRAN case, everything from eNB / EPC / MEC / EPC etc. in a mix of Baremetal, Virtual Machine and Containers; will assess ONAP integration feasibility after Beijing rel. is out.
- Testing of OpenStack configurations with remote compute nodes for edge use cases (focused on RAN use cases / with RAN latencies)
- K8s for the “CNF” LCM feasibility / gaps assessment
- K8s networking, EPA etc. gaps filling in associated communities
- Participation on multiple edge related projects in all domains

Key Take-Aways

- Application & ONAP operation times are related but mostly independent
- The “tightest” loop (in terms of loop time) determines the placement, messaging and timing constraints associated with it’s implementation
  - The supportable loop time determines it’s usability in context of time constrained ops
- For decentralized loops, ALL components associated with the loop need to be decentralized (e.g. messaging, DCAE, policy, controller, VIM)
- Need control processes and control loops at different levels (compute nodes, VIM / VIM subsystems, regions, and network / service levels)
- Many domain specific “policy” processes are in place already, need to be able to manage, not re-invent them
- Complex domain-interactions exist, we do not have adequate models available yet (e.g. availability / service states, power/resource management, perf mgmt, fault management, multi-layer interactions)
- Keep in mind that for uServices, communications latency can become relevant for even centralized (i.e. colocated services due to increasing amount of sub-service transactions)
- Other than latency considerations will also have impact (regional / sub-regional autonomy, bandwith etc.)