CONTEXT AND TRENDS
WHERE WE COME FROM…
QUASI-STATIC BITPIPES, **RECOVERY TIME <= 50 MS**
WHERE WE COME FROM
PHYLOSOPHIES OF THE PAST

- KISS = Keep It Simple & Stupid
- Do NOT touch anything that works
SELF-DRIVING CARS

- Self-driving cars often taken as example for strong delay requirements
  - Car @ 144km/h = 40 m/s = 4 cm/ms
  - Is end-to-end delay of <= 1ms meaningful?
  - What about the recovery time?
    - 50 ms → 2m
    - What loss can you tolerate during recovery time?
INTERNET OF THINGS

- Astronomic number of end-devices
  - 10 000s of sensors in a single factory
- One or a few gateways to cover whole country / city.

- Concentration of MANY devices per function
  - In steady-state probably ok
  - In case of failure: what about session mgmt. etc (resulting in storms that take a long time to be resolved before ending up again in a steady-state)?
RAS
RELIABILITY, AVAILABILITY AND SERVICEABILITY

- Wikipedia:
  - Reliability can be defined as the probability that a system will produce correct outputs up to some given time \( t \)
  - Availability means the probability that a system is operational at a given time
  - Serviceability or maintainability is the simplicity and speed with which a system can be repaired or maintained
  - Note the distinction between reliability and availability: reliability measures the ability of a system to function correctly, including avoiding data corruption, whereas availability measures how often the system is available for use, even though it may not be functioning correctly.
    - For example, a server may run forever and so have ideal availability, but may be unreliable, with frequent data corruption.

- Translated to our context
  - Availability: classical bitpipe connection broken \( \rightarrow \) improve with network recovery
  - Reliability: QoS/performance issues
    - Softwarization makes things less predictable
  - Serviceability
    - Softwarization and increased complexity require shift in what failures to handle, philosophy in operations, etc
    - Automation in DevOps approach should help a lot.
SDN / NFV / SFC
SOFTWARE DEFINED NETWORKING (SDN)
SEPARATES NETWORK CONTROL FROM THE FORWARDING PLANE
NETWORK FUNCTIONS VIRTUALIZATION (NFV)
VIRTUALIZING SOFTWARE-BASED NETWORK FUNCTIONS

Source: ETSI, “Network Functions Virtualization Whitepaper,”
SERVICE FUNCTION CHAINING (SFC)
BUILDING SERVICES USING VIRTUAL NETWORK FUNCTIONS (VNFS)
ADVANTAGES OF SDN, NFV, AND SFC

- Replace dedicated hardware with generic hardware and software-based functions
- Dynamically scale network, computing and storage resources based on service requirements
- Maximize resource utilization and optimize energy usage
- Faster and easier deployment, configuration, and updating of network functions
- Reduced time to market for services
- Flexibly manage service deployment based on traffic patterns and user mobility
RAS IN SOFTWARIZED NETWORKS
CONTROLLER ISOLATION
NO RESTORATION; AFTER FAILURE REPAIRED BOOTSTRAPPING FROM SCRATCH
RECOVERY TIME IN SOFTWARE APPLICATIONS
WHERE IS THE 50 MS ??

Recovery Time
On average, how long does recovery take when an application fails?

- 22% Less than 10 minutes
- 22% 10 to 30 minutes
- 26% 31 to 60 minutes
- 22% More than an hour but less than a half day
- 8% Half a day to less than a full day
- 6% A full day or more
- 4% Don’t know

Data: UBM survey of 300 IT professionals involved in applications, January 2017
PROTECTION IN NFV

- VNF functionality vs. configuration vs. state?
  - What part to pre-instantiate for backup resources?
  - Pre-load function but not configuration or state?
  - Pre-load function, configuration but not state?

- How to synchronize state?
  - copy on-demand vs. continuous copying of state
DETERMINISTIC PERFORMANCE?

MANY THINGS IMPACT PERFORMANCE

Source: https://en.wikipedia.org/wiki/Memory_hierarchy#/media/File:Hwloc.png
VNF PROFILING
INCLUDING UNANTICIPATED / WORST-CASE WORKLOADS

**Profile Function A**
- KPI limit
- Workload parameters

Increase workload until the KPI limit is reached

Measure resource usage

**Profile Function B**
- Resource limit
- Workload parameters

Increase resources until maximum budget is reached

Measure KPIs
Can lead to a total integrated platform that supports:
- deployment of VNF services
- dynamic scaling of the needed resources

Elastic Service Management Function

Service Provider

Service Users
generate workload

Monitor Service QoS requirements (KPI's)
Monitor Workload

QoS assurance scaling actions

scaling actions:
- resource scaling
- decomposition
- migration

workload prediction

deploy VNFs and setup links

Orchestration

Infrastructure Provider

Monitor Infrastructure resources

monitor data repository
Learning environment
VNF Profile repository
VNF Decomposition repository

Elastic Service Support Systems

Service 1
Service 2
Service N
OPEN SOURCE SOFTWARE AND STANDARDIZATION

- Quality
  - A few high-quality projects
  - Some more decent projects
  - Tons of projects that are not ready to use

- Excellent project gov/mgmt. is CRUCIAL
- Several competing projects (same functionality) is inefficient

- Standardization should become more agile and aligned with the relevant opensource projects
  - OSM as an ETSI-hosted project
  - ONF acquired ONLab (ONOS controller)
DEVOPS
THE DEVOPS CHALLENGE

Collaborating across teams and functions is a difficult and challenging task

Input by Ignacio Labrador Pavón (ATOS)
DevOps is not a single tool, platform or framework in the regular sense. Instead, it is rather a set, consisting of multiple tools ("DevOps toolchain") used to perform the different aspects of the software development and delivery process:

1. Code — Code development and review, version control
2. Build — Continuous integration tools
3. Test — Testing tools
4. Package — Artifacts repository
5. Release — Change management, release approvals
6. Configure — Infrastructure configuration and management
7. Monitor — Applications performance monitoring tools
DEVOPS PROCESS: MODULARIZATION
DEVOPS PROCESS AND TOOLCHAIN

Input by Ignacio Labrador Pavón (ATOS)
## DevOps Toolchain

<table>
<thead>
<tr>
<th>Container Orchestration</th>
<th>Rancher</th>
<th>Kubernetes</th>
<th>CoreOS</th>
<th>Mesosphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container Engines</td>
<td>Docker</td>
<td>Rocket</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mini OSs</td>
<td>RancherOS</td>
<td>Atomic</td>
<td>Photon</td>
<td>CoreOS</td>
</tr>
<tr>
<td>Clouds with Docker support</td>
<td>Openstack</td>
<td>Joyent Triton</td>
<td>OpenShift</td>
<td>Cloud Foundry</td>
</tr>
</tbody>
</table>

This is not a comprehensive list. The DevOps tool landscape is constantly changing and there are new and updated tools introduced frequently.

Input by Ignacio Labrador Pavón (ATOS)
High-performing teams deploy more — and faster.

46X more frequent code deployments
That means high performers deploy multiples times per day instead of once a week or less.

440X faster lead time from commit to deploy
That means high performers have a lead time of less than an hour instead of more than a week.

96X faster mean time to recover from downtime
That means high performers recover in less than an hour instead of several days.

5X lower change failure rate
That means high performers’ changes fail 7.5% of the time instead of 38.5%.

AUTOMATION CRUCIAL FOR SUCCESS

High-performing teams love automation. They do less manual work and have automated:

- 33% more of their configuration management
- 27% more of their testing
- 30% more of their deployments
- 27% more of their change approval processes

CONCLUSIONS
CONCLUSIONS

- Softwarization = more complex from a RAS perspective
  - Long recovery times compared to the classic 50 ms.
  - Performance is difficult to control in a softwarized context
- DevOps
  - Cultural change = difficult
  - Automatization crucial in keeping downtime under control
embracing a better life