FlexNets: Evaluating Flexibility in Softwarized Communication Networks

Wolfgang Kellerer
Technical University of Munich
Zagreb, June 29, 2017

ConTEL 2017
Introduction

• Networking today: **new requirements** from vertical industries, dynamically changing user behavior, and global digitalization

• Less (explicitly) addressed: **flexibility** and hence **adaptation**

• In this talk, I will …
  … present our definition of a measure for network flexibility …
  … give concrete examples of how to apply …
  … raise more questions
Outline of this Talk

• On flexibility in softwarized networks

• Proposal for a *flexibility measure*

• Use Cases
  • The Function Placement Problem
  • Dynamic Controller Placement
  • HyperFlex: a flexible SDN Hypervisor solution
The Internet

… is able to adapt its resources
… somehow (best-effort, TCP elasticity, BGP, OSPF)

early-days simplicity
→ complex and ossified network system

very slow adaptation to new requirements
→ reaction to dynamic changes hardly possible

source: SFB MAKI
New concepts such as …

**Network Virtualization (NV), Software Defined Networking (SDN) and Network Function Virtualization (NFV)**

…*promise* to create and adapt networks and functions on demand in *software*
New concepts such as …

Network Virtualization (NV), Software Defined Networking (SDN) and Network Function Virtualization (NFV)

…promise to create and adapt networks and functions on demand in software

→ Softwarized Networks
All problems solved?

• Are we fully flexible already?

• How far can we go? What is the right network design?

We need

• a fundamental understanding of how to provide flexibility
• a quantitative measure for flexibility pro and contra certain designs

For networks, flexibility = ability to support new requests to change design requirements (traffic pattern, latencies,...)

This work is part of a project that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 program grant agreement No 647158 – FlexNets (2015 – 2020).
A simple measure

For networks, flexibility = ability to support new requests to change design requirements (traffic pattern, latencies, ...) via adaptation of resources (topology, capacity, ...) if needed

\[ \phi(S) = \frac{|\text{supported new requests}|}{|\text{total number of given new requests}|} \]

- fraction of the number of new requests that can be supported of all given requests
- \( \phi(S) \in [0,1] \) „percentage“
What is missing?
The time aspect of flexibility

What Robert de Niro says on *flexibility*

in HEAT (1995) as Neil McCauley:

“Don’t get attached to anything you can’t walk out on in 30 seconds flat if you feel the heat around the corner."

Not only the number of options, but the time matters for *flexibility*!
Flexibility Measure – proposed definition

\[ \varphi_T (S \mid \text{state } i) = \frac{|\text{supported new requests within } T|}{|\text{total number of given new requests}|} \]

- fraction of the number of new requests that can be supported in a time interval \( T \) of all given new requests

\[ \varphi_{T \to \infty} (S) = \frac{|\text{supported new requests}|}{|\text{all given new requests}|} \]

\( T \)
A simple illustration (1)

- **New request** to an SDN-network: Controller Capacity (cc) is increased
- Can such new request be supported?  
  e.g. *by migrating the controller to a node with higher capacity (NC)*
- BUT: migration time cannot exceed “1 hop“ (T)

\[ \varphi = \frac{|1 \text{ new request supported}|}{|\text{all given new requests}|} \]

*new cc = 2*

\[ \text{nc = 1} \]

\[ \text{nc = 1} \]

\[ \text{nc = 3} \]

\[ \text{nc = 2} \]

new request can be supported
A simple illustration (2): more requests

\[ \varphi_{T=1\text{hop}} = \frac{|1 \text{ new request supported}|}{|3 \text{ given new requests}|} = \frac{1}{3} = 33\% \]

\[ \varphi_{T \to \infty} = \frac{|2 \text{ new request supported}|}{|3 \text{ given new requests}|} = \frac{2}{3} = 66\% \]
Flexibility a new measure? - Yes

no single quality indicator for a **Quality of Flexibility (QoF)**
- similar to QoS
- to be regarded by case (requirements, design goals, system)

we propose: **flexibility aspects** [1, 2]
- similar as we do with QoS (rate, delay, throughput, jitter,...)
- shall allow us to quantitatively compare two different system designs

Flexibility Aspect example 1: Flow steering and reconfiguration

Parameters (for change requests):

- number of flows,
- granularity (forwarding, duplicating,…),
- time to change
Flexibility Aspect example 2: Function Placement

Parameters:

- set of possible locations,
- number of supported requirements (latency, …),
- time of placement (static, dynamic)
Use Case 1: *The Function Placement Problem*

- NFV = virtualize & move function (= everything) to DC

Example: mobile core network functions

```
RAN                                      Core                                      PDN

MME                                      HSS                                      OCS

SGW                                      PCRF                                     PGW

High volume data traffic                High speed packet processing
```
Function Realization based on NFV

- Virtualization of GW functions [1] → NFV

Data-plane latency?

depends on the DC placement

Network load?

traffic transported to DC
(longer path → cost)

Virtualized GW

Function Realization based on SDN: move functions back

- Decomposition of GW functions [1] via SDN

data-plane latency?
additional latency is avoided

Control load? SDN control load!
depends on API (e.g. OpenFlow)

Interdependencies → Function chains (mixed design)

- Propagation latency depends on function chain = path SGW - PGW

Can be more complex for other use cases

Function Placement shall address:

- Function (de-)composition
- Function chaining

(c) SGW Virtualized
PGW Decomposed

(d) PGW Virtualized
SGW Decomposed
Virtualize all GWs? decompose all? mixed deployment?

Which GWs should be virtualized? decomposed? DC(s) placement?

- minimize core load
- satisfy data-plane latency

Some Evaluation Studies

The Functions Placement Problem

[4] A. Basta, W. Kellerer, M. Hoffmann, H. Morper, K. Hoffmann, Applying NFV and SDN to LTE Mobile Core Gateways; The Functions Placement Problem, AllThingsCellular14, Workshop ACM SICGOMM, Chicago, IL, USA, August 2014
3 design choices (= systems) to compare [5]:
(1) SDN design
(2) NFV design
(3) mixed SDN/NFV design

Parameter in focus:
• Flexibility to support different latency requirements for
  - control plane latency and data plane latency
    e.g.: \{5, 10, 15, \ldots, 45, 50\} ms

[1] W. Kellerer, A. Basta, A. Blenk,
*Using a Flexibility Measure for Network Design Space Analysis of SDN and NFV*, SWFAN’16, IEEE INFOCOM Workshop, April 2016.
Design Choices

Legacy LTE core design: Gateways (GW) as dedicated middleboxes

(1) SDN design: separation of control and data plane for GWs

(2) NFV design: all functions (data and control) run in a cloud

(3) Mixed SDN/NFV design:

only control to cloud

control and data to cloud
Flexibility measure and evaluation setup

Flexibility measure:

\[ \varphi_{\text{placement}}(\text{design. } x) = \frac{\left( \sum_i \sum_j \text{feasibleSol}_{i,j} \cdot w_{i,j} \right)}{\sum_i \sum_j w_{i,j}} \]

Function placement problem formulated as a MILP [6]

- SGW and PGW (VNF) placement
- constraints on data and control plane latency
- weights

\[ w_{i,j} = \frac{\alpha}{\text{dataLatency}_i} + \frac{\beta}{\text{controlLatency}_j} \]

Results [5]

With respect to the support of latency requirements in function placement:

- mixed SDN/NFV is more flexible for a logically centralized data center infrastructure
- for distributed data centers all three design choices are equally flexible

What are the costs of a design for flexibility?

- in terms of signaling overhead, number of data centers,…

Possible relationship (to be confirmed):

Nothing is for free: Cost of Flexibility
Use Case 2: Dynamic Controller Placement Problem

- place 1 ..n SDN controllers for time varying traffic input
  → controller migration/reconfiguration

- Evaluation parameters
  - Abilene network topology (11 nodes, 14 links)
  - new requests: 100 different flow profile requests over time (random)
  - $N = 1,\ldots, 4$ controllers (design choices for comparison)
  - Algorithm finds optimal controller placement and flow to controller assignment
    optimization goal: minimize avg. flow setup time (performance)
  - How many controllers can be migrated (incl. control plane update) in time $T$?
    (success ratio → Flexibility)
  - Migrations and reconfigurations → Cost


Simulation Results

Use Case

**Flexibility**
- **success ratio**

**Performance**
- **avg. flow setup time**
- **reconfigurations**

**Cost**
- **reconfigurations**

---

**Number of controllers** $N$

**migration time threshold** = 803 ms

**$T$** is very short (800 ms is transmission delay of 1 controller)
Simulation Results

Use Case

Flexibility
- success ratio

Performance
- avg. flow setup time

Cost
- reconfigurations

Number of controllers $N$

migration time threshold = 804 ms

$T$ is very short (800 ms is transmission delay)
Simulation Results

Use Case

Flexibility

success ratio

Performance

avg. flow setup time

Cost

reconfigurations

Number of controllers N

migration time threshold = 805 ms
Simulation Results

Use Case

Flexibility

- Success ratio

Performance

- Avg. flow setup time

Cost

- Reconfigurations

Migration time threshold = 806 ms

1 controller has highest flexibility at low cost
But: performance is not good (flow setup time)
Simulation Results

**Use Case**

**Flexibility**
- Success ratio

**Performance**
- Avg. flow setup time

**Cost**
- Reconfigurations

Migration time threshold = 807 ms
Simulation Results

Use Case

Flexibility
- success ratio

Performance
- avg. flow setup time

Cost
- reconfigurations

migration time threshold = 808 ms
Simulation Results

Use Case

**Flexibility**
- success ratio

**Performance**
- avg. flow setup time

**Cost**
- reconfigurations

migration time threshold = 809 ms
Simulation Results

Use Case

**Flexibility**

- success ratio

**Performance**

- avg. flow setup time

**Cost**

- reconfigurations

migration time threshold = 810 ms

T is moderate: more controllers $\rightarrow$ higher flexibility at higher cost
Simulation Results

Use Case

**Flexibility**
- Success ratio

**Performance**
- Avg. flow setup time

**Cost**
- Reconfigurations

Migration time threshold = 811 ms

T is moderate: more controllers $\rightarrow$ higher flexibility at higher cost
Simulation Results

Use Case

Flexibility

- Success ratio

Performance

- Avg. flow setup time

Cost

- Reconfigurations

Migration time threshold = 812 ms

T is moderate: more controllers → higher flexibility at higher cost
Some cases: 1 controller is more flexible (short T)
T considerable for adaptation: more controllers → more flexible
There is a cap in gain – cost is rising

migration time threshold = 812 ms
Summary (from [5])

Flexibility

Performance

from fundamental research to practice:

an implementation solution for flexibility
Designing for Flexibility: Network Slicing

- Why do we need network virtualization “slicing“?

- NGMN 5G white paper [7]
  - logical virtual mobile network slices
  - reliable and on-demand slices

- METIS 5G system concept and technology roadmap [8]
  - application and service differentiation
  - logical virtual mobile network slices
  - heterogenous and dynamic slices

5G Slicing: SDN virtualization

- Why do we need SDN virtualization “slicing” in 5G?
- Bring your own controller
- Full flexibility and programmability

• Why do we need SDN virtualization “slicing” in 5G?

Prof. Wolfgang Kellerer | Chair of Communication Networks | TUM
SDN Virtualization Overview

• How to achieve slicing for SDN networks?
  • SDN virtualization layer, i.e., SDN hypervisors
  • e.g. FlowVisor [9], OpenVirteX [10]

• What should an SDN hypervisor do?
  • Virtual SDN abstraction
  • Control plane translation
  • Data and control slice isolation
  • … in a most flexible way


• SDN Slices
  • focus on data plane slices
  • control performance impacts the data plane performance in SDN!

• Management
  • automated slice request is not addressed
  • admission control interfaces are missing

• Deployment
  • no mechanisms to change the deployment on run time
  • e.g., automate adding or removing of a hypervisor instance

Our approach: HyperFlex Features

- Admission Control [12]
  - automated request of virtual SDN slices
  - guarantees for data and control plane performance
  - run time update to slice
  - embedding of virtual links on the physical network

(a) Tenant View

(b) HyperFlex View

HyperFlex Features

• Performance Monitoring [12]
  • monitor the performance of the running hypervisors, e.g., CPU
  • monitor the performance of the SDN slices
    • control plane latency
    • control plane loss rate

HyperFlex Features

- Dynamic Deployment “Orchestration” [12]
- cope with the slice dynamics, e.g., new requirements, time-varying traffic, …
- transparent to tenants, i.e., no interruption and no control loss
- optimal placement of SDN hypervisors

Testbed@TUM: Flexible Application-to-Cloud Softwarized 5G Networks

18x OF switches
Key Takeaways

- Network research is faced with new requirements from emerging networked industries
- These include *flexibility*
- Network softwarization (SDN, NV, NFV) can be used
- Need for
  - a *measure* to analyse flexibility
  - new *flexible concepts* (e.g. HyperFlex)
References for further reading (1)


  
References for further reading (2)


- R. Sherwood et al., Carving research slices out of your production networks with OpenFlow, ACM CCR, 2010

