Analysis and Optimization of Networks for Flexibility

Wolfgang Kellerer
Technical University of Munich, (TUM) Germany

with Peter Babarzci, Andreas Blenk, Mu He, Patrick Kalmbach, Markus Klügel, Alberto Martínez Alba, Johannes Zerwas

CNSM 2018
Rome, Italy, November 7, 2018

www.networkflexibility.org
The rise of flexibility

- Flexibility is gaining increasing **attention** and **importance**

Evolution of the number of publications containing the words **"flexible"** or **"flexibility"** in contrast with those containing **"bandwidth"** or **"capacity"** in four major IEEE journals and magazines on communication, with respect to the number of publications in 1995.
Network Virtualization (NV), Network Function Virtualization (NFV), and Software Defined Networking (SDN) …promise to create and adapt networks and functions on demand in software
Why is flexibility so important?

- Evolution tells us that the more flexible species can better survive.

- What about networks? Will they survive?

- So far less explicitly addressed: **flexibility to adapt to future demands**

- **Considering the Future** is very important for survival
  - enables operators to cover the future
  - key decision factor between network designs
  - optimize networks for flexibility
Are we there already?

- Are we **100% flexible** already (e.g. with Softwarized Networks)?
- How **far** can we go? What is the **optimal network design** for flexibility?

- What is **network flexibility**?

We need
- a **fundamental understanding** of how to provide flexibility
- a **quantitative measure** for flexibility pro and contra certain designs
An exercise on measuring flexibility

- Which tool is more flexible?
  - re-configuration shows more potential to be more flexible

- When can both exhibit the same flexibility?
  - maybe there is no need to change → probability of requests make a difference
  - maybe both cannot satisfy my requests → infeasible

- When can the re-configurable tool be less flexible?
  - adaptation time → re-configurable object might not be handy
  - cost → inefficient
Our approach for *Network Flexibility*

Network **flexibility** = ability to support *adaptation requests (challenges)* (e.g., new requirements or traffic patterns) in a *timely* and *efficient* manner.


[www.networkflexibility.org](http://www.networkflexibility.org)

We provide

- a **definition of network flexibility**
- a **quantitative measure** for flexibility pro and contra certain designs
- **Optimization** for flexibility
- **Empower** networks for flexibility to cover the future

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).

[www.networkflexibility.org](http://www.networkflexibility.org)
Measuring Network Flexibility (our proposal)  
(comparing network designs)

Input: Constraints $T, C$

1. Design sequence $\mathbb{C} = \{s_{i_1,j_1}, s_{i_2,j_2}, \ldots \}$ with $\nu(s_{i,j}) = V$
2. Initialize $\Sigma := 0$
3. FOR $k = 1:K$
   a. Challenge state switch $s_{i_k} \leftrightarrow s_{j_k}$
   b. Observe $\tau_X$ and $c_X$
   c. If $\tau_X \leq T$ and $c_X \leq C$: $\Sigma := \Sigma + 1$
4. END
5. $\varphi(T, C) := \Sigma/K$

Flexibility

$$\varphi(T, C) = \frac{\text{supported requests within constraints } (T, C)}{\text{Number of requests}}$$
Case study 1: Dynamic Controller Placement

- Traffic fluctuations require control plane to adapt in order to achieve better control performance → Dynamic Control Plane
- SDN controller migration & SDN switch reassignment

<table>
<thead>
<tr>
<th>Flexibility Aspect</th>
<th>New Request</th>
<th>Flexibility Measure</th>
<th>System Objective</th>
<th>Cost in focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>function placement</td>
<td>new flow arrival (from distribution)</td>
<td>fraction of successful controller placements</td>
<td>control performance: (min. avg. flow setup time)</td>
<td>operation latency (OPEX): avg. flow setup time</td>
</tr>
</tbody>
</table>
Case study 1: Dynamic Controller Placement

- **Flexibility → Migration Success Ratio**
  - Calculate controller migration and switch reassignment time $T_{migration}$
  - If $T_{migration}$ smaller than $T$ → count as a supported request

\[
\varphi_T(S) = \frac{|\text{supported requests within } T|}{|\text{given new requests}|}
\]

- Varying traffic flow profiles
- max. adaptation time threshold (will be varied)

SDN controller migration and switch reassignment can be done within $T$

$C \rightarrow \infty$ recorded
**Case study 1: Dynamic Controller Placement**

- **More controllers** (larger migration time threshold) $\rightarrow$ higher flexibility
- **Single controller** case: more flexible for **tight time threshold** as probability that single controller stays in optimal location is high

- 1 controller $\rightarrow$ **marginal** performance improvement vs. adaptation T
- 4 controllers $\rightarrow$ **significant** performance improvement vs. adaptation T
- However, if we consider all cost factors, we can reach a trade-off!

---

Case study 2: SDN Resilience

• Flexibility aspect of **flow configuration** for a **resilience** scenario in an SDN network under a given **recovery time** threshold $T$.

• Objective: system recovery

• Compare 3 systems: 1:1 protection vs 1+1 protection vs restoration

• Flexibility measure: fraction of recoverable failures

• New requests: all possible **single and dual link failures**

<table>
<thead>
<tr>
<th>New Request</th>
<th>Flexibility Measure</th>
<th>System Objective</th>
<th>Cost in focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>all possible single and dual failures</td>
<td>fraction of recoverable failures</td>
<td>system recovery: (single and dual failures)</td>
<td>resources overhead (CAPEX): node and link reservation</td>
</tr>
</tbody>
</table>
Case study 2: SDN Resilience

- **1+1** can not reach **full flexibility**
- However, 1+1 is obviously **independent of recovery time**
- Restoration can cover **all failures** if given enough recovery time

### Resources Cost (CAPEX)

<table>
<thead>
<tr>
<th></th>
<th>Node reservation: Avg. number of flow table entries</th>
<th>Link reservation: Number of required links</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 + 1</td>
<td>11.78</td>
<td>13038</td>
</tr>
<tr>
<td>1 : 1</td>
<td>11.78</td>
<td>13038</td>
</tr>
<tr>
<td>Rest.</td>
<td>5.05</td>
<td>5400</td>
</tr>
</tbody>
</table>

(intuitive)

- Protection imposes more than **2x capex overhead** than restoration
- Again, if we consider **all cost factors**, we can reach a trade-off!
Case study 3: FlexRAN (ongoing work)

• Radio Access Network plus SDN/NFV → unexplored flexibility
• our use case: coordinated scheduling
• initial results: PoC

CloudRAN:

pure SD-RAN:

partial SD-(Flex)RAN:

• next: quantify flexibility; flexibility: ratio of successful handling of request
• We can **measure flexibility** so far relatively between multiple systems

• Results can be **less intuitive** than one might think

• Measure can be used to **design for flexibility**
Optimize for Flexibility $\phi$

Use Case example: **Dynamic Controller Placement Problem**

- Requests: traffic profiles with target average flow setup time
- Objective: max. **flexibility** (success: # accomodated traffic profiles)
- Design parameters: # data centers and their locations
Optimize for Flexibility

- **Design of methods to optimize for flexibility**
  - offline optimization

- **online** optimization
  - adaptation time speedup through machine learning
  - *empower* a network to optimize for the future
  - runtime reconfigurability of HW
• **Adaptation time** is very important for flexible networks
• Adaptation examples:
  - function migration, e.g., SDN controller
  - (re-)embedding of virtual networks/flows, e.g. for resilience
  - shift of Radio Access Network functions to a central node

• How can we speedup?
• Yet another heuristic for a specific case study?

We propose:
• Keep your favourite optimization algorithms and
• *Boost your network algorithm with ML preprocessing*
How can we boost the solving of the related optimization problems (leaving you algs. untouched)?

State-of-the-art: Neglects produced data!

Idea: Use problem/solution data generated by algorithms regularly solving problems.

A. Blenk, P. Kalmbach, S. Schmid, W. Kellerer: o’zapft is: Tap Your Network Algorithm’s Big Data!
ACM SIGCOMM 2017 Wrksp. on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA), 2017.

Case Study: Predicting Acceptance Probabilities of VNE Requests

- Supervised learning: use data with accepted and rejected requests! Offline training!
- Recurrent neural network (RNN) for classification
- Filter infeasible and requests with unacceptable algorithm runtime ("no solution")
Can we speed-up optimal algorithms using admission control?

Efficient Filtering of infeasible and unacceptable requests
Efficient saving of model creation time
Latest Results: Neurovine

Hopfield neural network to preprocess (subgraph extraction) VNE algorithms – tailored filtering

- Idea: Extract subgraph with physical nodes close to each other and high available capacities

Neurovine:
Efficiency on Real Network Topologies

- VNE algorithms (GRC, DViNE, RViNE) vs. Hopfield variants (HF-GRC, HF-DViNE, HF-RViNE)

- NeuroViNE accepts more networks with less costs
Optimize for Flexibility

- **Design of methods to optimize for flexibility**
  - offline optimization
  - **online** optimization
    - adaptation time speedup through machine learning
      ➢ we still have a clear objective here to optimize for
      - *empower* a network to **optimize for the future**
Empower your network

optimize for the (unknown) future:
• prepare for possibly unexpected events → flexibility

we need:
• (online) self-optimization
  
  self-driving networks (Rexford, Fearnster): networks which measure, analyze and control themselves in an automated manner, reacting to changes in the environment

• prepare for the unknown

We propose:
• use empowerment for preparedness

Empowering Networks

*empowerment*: quantify the influence of an agent on its environment: agent (several actuators, 1 sensor) restructures networks to maximize options (c) - **not** an objective as in optimization (a) and (b)

Initial result: empowerment-based approaches (EB, EC) can outperform ILP ("0"-line) and heuristics (SA)

Key Takeaways & outlook

• We propose a **definition and measure for flexibility**
  – to compare flexible systems
  – to explicitly **design for flexibility**

• **(online) optimization for flexibility** is supported by
  – Speedup of opt. algorithms through ML-preprocessing
  – **Empowerment** to optimize for flexibility to cover the future
  – Runtime **reconfigurability** of HW with P4

→ Mu He et al.: *P4NFV: An NFV Architecture with Flexible Data Plane Reconfiguration*
  in today’s afternoon session

*join us on* networkflexibility.org
W. Kellerer, A. Basta, A. Blenk, Using a Flexibility Measure for Network Design Space Analysis of SDN and NFV, IEEE INFOCOM Workshop, SWFAN’16, SF, USA, April 2016.


