Flexibility Matters
On the Design and Evaluation of Softwarized Networks

Wolfgang Kellerer
Technical University of Munich

IFIP Networking 2018
Zurich, Switzerland
May 14-16, 2018

with PhD students and postdocs
A. Basta, A. Blenk, P. Babarczi, M. He
P. Kalmbach, M. Klügel, A. Martinez,
J. Zerwas

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

- Networking today: new requirements from vertical industries and dynamically changing behavior of users and tenants
- Novel techniques to softwarize networks
- Less explicitly addressed: flexibility and hence adaptation

Today, we will …

… present our definition of network flexibility and a flexibility measure, …
… give concrete use cases of how to apply …
… and show methods to speed up adaptations with ML.
Towards softwarized networks

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

eyear-days simplicity
→ complex and ossified network system

very slow adaptation to new requirements
→ reaction to dynamic changes hardly possible
New concepts such as ...

**Network Virtualization (NV),**

**Network Function Virtualization (NFV),** and

**Software Defined Networking (SDN)**

...promise to create and adapt networks and functions on demand in software
All problems solved?

- Are we fully flexible already?
- How far can we go? What is the optimal 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, ...) in a timely manner via adaptation of resources (topology, capacity, ...) if needed

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).
Why do we think flexibility analysis is important?

- A survey on 5G technology [1] reports “flexible and scalable network” as the top motivation for technology investment of 297 companies.

- Enables operators to cover the future!
  - react to regulatory changes and fast arrival of new technologies

- A key decision factor between network designs
  - can be a tie-breaking decisive advantage for a certain network design

- For research and development
  - which technical concepts lead to more flexibility in network design?
    → optimize networks for flexibility
    → design guidelines for more flexible networks

- SoA: lack of a concrete definition and a quantitative analysis!

How to define network flexibility? [3]

For network systems, **flexibility** = ability to support new requests to change design requirements (traffic pattern, latencies, …) in a *timely* manner via adaptation of resources (topology, capacity, …) if needed.

**System?**
- communication network (topology, flows, node functions, resources) serving a certain objective (e.g. highly reliable communication)

  *Note: in most cases, flexibility is not the objective*

**Request?**
- “new challenges”, e.g., new flows, new (virtual) topology or new latency requirements

  *Note: explicit list or via a distribution (e.g. flow arrivals)*

So: *the more requests are supported the more flexible a system is?*

**Time?**

What Robert de Niro on *flexibility* in HEAT (1995):

“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 requests, but the *time* matters for flexibility!

*how to consider for flexibility?*

*how to speed up adaptation?*
How to define network flexibility?

For network systems, **flexibility** = ability to support new requests to change design requirements (traffic pattern, latencies, ...) in a timely manner via adaptation of resources (topology, capacity, ...) if needed.

**System?**
- communication network (topology, flows, node functions, resources) serving a certain objective (e.g. highly reliable communication)

**Request?**
- “new challenges”, e.g., new flows, new (virtual) topology or new latency requirements

**Time?**
- the network may need to adapt the state of the topology, flows, functions, or resources → it should meet a time constraint
- the network may be designed in a way that it simply accommodates requests without adaptations → meets any time constraint

no single quality indicator for a Quality of Flexibility (QoF)

- similar to QoS: to be regarded by case

we propose: flexibility aspects [1, 3]

- similar as we do with QoS (rate, delay, throughput, jitter,…)

Flexibility aspects to technologies mapping

- SDN: is about flow control, also supports network resources scaling

- NFV: targets flexible placement, degrees of freedom in configuration and function scaling

- NV: targets flexible (virtual) topologies, also provides degrees of freedom for configuration and scaling of these (virtual) networks

<table>
<thead>
<tr>
<th>Aspect (see Sec. III-B)</th>
<th>SDN</th>
<th>NFV</th>
<th>NV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Configuration: flow steering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Configuration: function programming</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter Configuration: change function parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Placement: distribution, placement, chaining</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource and Function Scaling: processing and storage capacity, number of functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology Adaptation: (virtual) network adaptation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost vs. Flexibility

- Flexibility has to be also evaluated against cost
- It is not clear if flexibility adds more cost overhead
- A flexible system can also achieve cost savings on the longer run

→ trade-off needs to be studied and evaluated

- We need to consider all different cost factors

<table>
<thead>
<tr>
<th>Resources (CAPEX)</th>
<th>Operation (OPEX)</th>
<th>Adaptation/Migration</th>
<th>SLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource overhead</td>
<td>control, data plane throughput</td>
<td>synchronization overhead</td>
<td>fines</td>
</tr>
<tr>
<td>network complexity</td>
<td>control, data plane latency</td>
<td>configuration latency</td>
<td>flow interruption</td>
</tr>
<tr>
<td>software complexity</td>
<td>energy consumption</td>
<td>topology adaptation latency</td>
<td>network interruption</td>
</tr>
</tbody>
</table>

Flexibility matters!
- new requirements for networking research include flexibility
- network softwarization (SDN, NFV, NV) provides flexibility

Flexibility definition is important!
- for a meaningful system analysis and comparison
- for a trade-off evaluation with performance and cost
- to design for flexibility

We need a measure!
Flexibility qualitative measure exercise

- 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 \(\rightarrow\) probability of requests make a difference
  - maybe both cannot satisfy my requests \(\rightarrow\) infeasible

- When can the re-configurable tool be less flexible?
  - adaptation time \(\rightarrow\) might make the re-configurable object not very useful
Flexibility Measure – Proposal

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

- fraction of the number of \textbf{new requests} that can be supported in a \textbf{time interval } \(T\) of all given new requests \([3]\)

\[ \varphi_T^{\rightarrow \infty} (S) = \frac{|\text{supported new requests}|}{|\text{given new requests}|} \]

Request: “challenge“ constraint on max. adaptation Time

System
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 Node Capacity (nc)*
- BUT: migration time **cannot exceed** “1 hop“ (T)

\[ \varphi = \frac{|1 \text{ new request supported}|}{|1 \text{ given new request}|} \]
A simple illustration (2): more requests

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

\[ \varphi_{T \rightarrow \infty} = \frac{2 \text{ new request supported}}{3 \text{ given new requests}} = \frac{2}{3} = 66\% \]
Case study 1: Dynamic Controller Placement

- Traffic fluctuations require control plane to adapt in order to achieve better control performance → Dynamic Control Plane [4]
  - SDN controller migration
  - SDN switch reassignment

<table>
<thead>
<tr>
<th>Case Study</th>
<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>dynamic SDN controller placement</td>
<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

- Application of the flexibility measure

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

Varying traffic flow profiles

max. adaptation time threshold (will be varied)

SDN controller migration and switch reassignment can be done within \( T \)

- Flexibility \( \rightarrow \) Migration Success Ratio
  - Calculate controller migration and switch reassignment time \( T_{\text{migration}} \)
  - If \( T_{\text{migration}} \) smaller than \( T \) \( \rightarrow \) count as a supported request
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 $T [3]$.  

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

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

- Objective: system recovery  

- Flexibility measure: fraction of recoverable failures

<table>
<thead>
<tr>
<th>Case Study</th>
<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>failure recovery in SDN</td>
<td>flow configuration</td>
<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 protection
  • primary and backup paths pre-calculated
  • backup path is inactive
  • need switching time between primary and backup in case of a failure

• 1+1 protection
  • primary and backup paths pre-calculated
  • primary and backup paths are both active
  • recovery time is almost instantaneous!

• Restoration
  • no backup path in advance
  • switch detect failure → controller informed → re-routes affected flows
  • recovery time is very critical
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

- Protection imposes more than 2x capex overhead than restoration
- Again, if we consider all cost factors, we can reach a trade-off!

intuitive

(a) Flexibility in terms of covered single and dual link failures.

(b) System resources cost (CAPEX) in terms of nodes and links used for reservation.
Key takeaways – part 2

• One way to **measure flexibility** so far only relatively between multiple systems

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

• Flexibility tends to decrease **cost** but also comes at a cost

• Measure can be used to **design for flexibility**

• **Adaptation time** is important for a flexible system
Time matters

• **Adaptation time** is very important for a flexibility measure
• Adaptation examples:
  - Function placement, e.g., SDN controller
  - (re-)embedding of virtual networks/flows, e.g. for resilience

• 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?

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

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

ACM SIGCOMM 2017 Wrksp. on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA), 2017.

Data Available: P. Kalmbach, J. Zerwas, M. Manhart, A. Blenk, S. Schmid, W. Kellerer. Data on "o'zapft is Tap Your Network Algorithm's Big Data!", 2017 [https://doi.org/10.14459/2017md1361589](https://doi.org/10.14459/2017md1361589)
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 [6]

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
Flexibility matters!

- We propose a definition and measure for flexibility
- to compare flexible systems
- to explicitly design for flexibility

- **Adaptation/optimization time** is important for flexible systems
  
  Speedup optimization algorithms through Machine Learning-based preprocessing

- Recent work: **Empowerment** concept to design for flexibility [7]

References


