Position Paper: Software-Defined Network Service Chaining

EWSDN 2014, Budapest, Hungary, 1-3 September 2014

Jeremias Blendin
jblendin@ps.tu-darmstadt.de

Julius Rückert
rueckert@ps.tu-darmstadt.de

Nicolai Leymann
n.leymann@telekom.de

Georg Schyguda
g.schyguda@telekom.de

Prof. Dr. David Hausheer
hausheer@ps.tu-darmstadt.de

This work has been supported in parts by Deutsche Telekom (D-Nets project), the European Union (FP7/317846, SmartenIT and FP7/318398, eCousin) and the German DFG (Collaborative Research Center 1053, MAKI). The authors would also like to acknowledge valuable input and feedback by their colleagues and project partners.
Network Service Chaining (NSC)

SDN/NFV facilitates faster service deployment
- Traditional network service examples: redirects, firewalls, traffic shaping, web proxy, video optimizer

Network service chaining challenge
- Complex services composed of service functions require more flexible deployment models

Existing service chaining approaches
- Gi-LAN today: pre-configured service chains
- IETF Network Service Headers [QGF+14], StEERING [ZBB+13], SIMPLE [QTC+13]

Goal
- Demonstrating the feasibility of dynamic network service chaining using SDN/NFV in a telecom environment

Benefits for ISPs
- Use of standard hardware
- Simplified management and operations
- Faster service deployment for end-users
Service Chaining Terminology and Core Challenges

Abstract definition of a chain consisting of service functions

Service Chain

Service Function

Service Instance

Service Chain Instance

Service Node

Terminology based on IETF [QN13]
Approach

User mapping
- Based in the IP address of the user device

Routing
- MAC address based, inspired by StEERING [ZBB+13]

Traffic identification
- Port based traffic identification
- Use one service instance per user
  - Feasibility of operating a large number of SIs on one server is shown in [BDH+13], which based on clickOS [MAR+13]
  - Proof-of-concept implementation relies on Linux containers

Service instance integration
- Isolation of instances
- Packet conditioning
Service Instance Interface

Network interface
- Traffic identification through switch ports
- Two links per Service Instance to the NFC system: ingress and egress
- Efficient and reliable traffic identification
  - Compared to other approaches using e.g. statistical analysis

Instance isolation
- Block management traffic (e.g. ARP)
- Exchange address information through the management system

Benefits
- Efficient traffic identification
- Simplified deployment
  - No per instances configuration required
  - Overlapping network configurations possible

Diagram:
- Service Instance
  - Packet flow
  - Ingress link
  - Egress link
- OpenFlow switch
- Mobile device
- Internet
- Other parts of the network are not shown
Service Instance Interface

Packet conditioning
- Adapt packets to SI requirements
- Adapt packets to service chaining system requirements
- Result: adapter layer

Example
- User: 20.0.0.1
- Website: 17.1.3.4

<table>
<thead>
<tr>
<th>Service Example</th>
<th>Type</th>
<th>Forwarding</th>
<th>In: requirements</th>
<th>User-&gt;Internet: modified fields</th>
<th>Egress out: applied modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic shaper</td>
<td>L2</td>
<td>bridging</td>
<td>MACdst ≠ MACif</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Firewall</td>
<td>L3</td>
<td>routing</td>
<td>MACdst = MACif</td>
<td>L2 src,dst</td>
<td>L2src = MACingress edge, L2dst = MACnext instance</td>
</tr>
<tr>
<td>Redirect</td>
<td>L4dst</td>
<td>routing</td>
<td>MACdst = MACif</td>
<td>L2 src,dst; L3,L4 dst</td>
<td>type L3 modifications</td>
</tr>
<tr>
<td>NAT</td>
<td>L4src</td>
<td>routing</td>
<td>MACdst = MACif</td>
<td>L2 src,dst; L3,L4 src</td>
<td>L3src = IPuser + type L3 modifications</td>
</tr>
<tr>
<td>HTTP proxy</td>
<td>L7</td>
<td>proxying</td>
<td>MACdst = MACif</td>
<td>L2,L3,L4 src,dst</td>
<td>type L4src modifications</td>
</tr>
</tbody>
</table>

Jeremias Blendin - EWSDN 2014
System Architecture
Conclusion and Next Steps

SDN network service chaining system
- Proof-of-concept implementation
- Approach is technically feasible
- High-level API is proves very useful

Service instance integration
- Traffic conditioning
  - Flexible interface between SIs and NCSs
- Service instance isolation
  - simplified deployment process
- Use one-to-one mapping of users to service instances
  - Supported by service instance isolation
  - Efficient and reliable traffic identification

Next steps
- Switch virtualization technology from Linux containers to e.g. clickOS [MAR+13]
- Analyze the systems scalability properties
Thank You for Your Attention – Questions?

Demo
Today at 14:00h

Contact: Jeremias Blendin
jblendin@ps.tu-darmstadt.de
http://www.ps.tu-darmstadt.de/

Peer-to-Peer Systems Engineering Lab (PS)
References


