Fault Tolerant Service Function Chaining

MILAD GHAZNAVI, ELAHEH JALALPOUR, BERNARD WONG, ALI MASHTIZADEH, RAOUF BOUTABA
Fault Tolerant Service Function Chaining

Keep a service function chain running after $f \geq 1$ number of its service functions fail

Extend IETF network service header (NSH) to support fault tolerance
Introduction
Fault Tolerant Chaining
NSH for Fault Tolerant Chaining
Conclusion
Introduction
Fault Tolerant Chaining
NSH for Fault Tolerant Chaining
Conclusion
Service Functions (Middleboxes)

Network Address Translator (NAT)  Firewall

Alice
192.168.10.10

Bob
192.168.10.20

129.97.12.14

Internet
Contributing to 43% of high-severity incidents “Demystifying the dark side of the middle: a field study of middlebox failures in datacenters.” IMC 2013.
Service Function Fault Tolerance

NAT Connection State

<table>
<thead>
<tr>
<th></th>
<th>Alice &lt;-&gt; Apple</th>
<th>Bob &lt;-&gt; Bing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- Alice and Bob are connected to a NAT device.
- The NAT device connects to the Internet.
- Apple and Bing are connected to the Internet through the NAT device.
- A table showing NAT connection states: Alice <-> Apple and Bob <-> Bing.
Service Function Fault Tolerance

NAT Connection State

| Alice ⇔ Apple | Bob ⇔ Bing |

Replica

| Alice ⇔ Apple | Bob ⇔ Bing |

Alice

Bob
Service Function Fault Tolerance

NAT Connection State

<table>
<thead>
<tr>
<th></th>
<th>Alice ⇔ Apple</th>
<th>Bob ⇔ Bing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Replica

<table>
<thead>
<tr>
<th></th>
<th>Alice ⇔ Apple</th>
<th>Bob ⇔ Bing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Most of existing solutions are snapshot based

- **Pico Replication**, SoCC 2013
- **FTMB**, SIGCOMM 2015
- **REINFORCE**, CoNEXT 2018
Service Function Chains (Chains)
Fault Tolerance for a Chain

EXISTING SNAPSHOT-BASED APPROACHES

OUR APPROACH: FAULT TOLERANCE FOR AN ENTIRE CHAIN

Throughput Drop

<table>
<thead>
<tr>
<th>Chain Length</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop Percentage</td>
<td><img src="chart.png" alt="Bar Chart" /></td>
<td><img src="chart.png" alt="Bar Chart" /></td>
<td><img src="chart.png" alt="Bar Chart" /></td>
<td><img src="chart.png" alt="Bar Chart" /></td>
<td><img src="chart.png" alt="Bar Chart" /></td>
</tr>
</tbody>
</table>

Fault Tolerant Chain Structure:

- $SF_1$
- $SF_2$
- $SF_m$
Introduction

→ Fault Tolerant Chaining

NSH for Fault Tolerant Chaining

Conclusion
Design Choices

State piggybacking

In-chain replication
Design Choices – State Piggybacking

EXISTING APPROACHES

SF

Data traffic
State dissemination

FTC’S APPROACH

SF

Data traffic +
State dissemination
Design Choices – In-Chain Replication

EXISTING APPROACHES

Replica

R₁

R₂

\ldots

R_f

SF

FTC’S APPROACH

SF₁

SF₂

\ldots

SF_{f+1}
Fault Tolerant Chain Protocol

\[ SF_1 \quad SF_2 \]
Fault Tolerant Chain Protocol

SF_1
R_1

SF_2
R_2

R_3
Fault Tolerant Chain Protocol

Replicate SF₁ state in R₁ and R₂ to tolerate its failure (f=1)
Fault Tolerant Chain Protocol

Replicate SF_2 state in R_2 and R_3 to tolerate its failure (f=1)
Fault Tolerant Chain Protocol

A replica intercepts packets
Fault Tolerant Chain Protocol

A packet piggybacks its state updates
Fault Tolerant Chain Protocol
FTC’s Performance

3.5x higher throughput
Introduction

Fault Tolerant Chaining

→ NSH for Fault Tolerant Chaining

Conclusion
Network Service Header – RFC 8300
A service function forwarder (SFF) as a replica
### Service Function Forwarder As a Replica

<table>
<thead>
<tr>
<th>SUPPORTED BY ORIGINAL NSH</th>
<th>OUR CONTRIBUTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet forwarding through a chain</td>
<td>Extensions to NSH</td>
</tr>
<tr>
<td></td>
<td>◦ State management API</td>
</tr>
<tr>
<td></td>
<td>◦ State replication</td>
</tr>
<tr>
<td></td>
<td>NSH support in Click modular router</td>
</tr>
</tbody>
</table>
Network Service Header Format

NSH header to piggyback state
Network Service Header Format – Type 2

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|C|R|R|R|R|R|R| Length | MD Type=0x2 | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Service Path Identifier | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Variable Length Context Headers (opt.) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Network Service Header Format – Type 2

Variable length context headers to piggyback state

Variable Length Context Headers (opt.)
Variable Length Context Headers

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Metadata Class       |      Type     |U|    Length   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Variable-Length Metadata                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: NSH TLV Format

- **Metadata Class (MD Class):** Defines the scope of the Type field to provide a hierarchical namespace.
- **Type:** Indicates the explicit type of metadata being carried. The value is one from the Network Service Header (NSH) TLV Type registry (Section 7).
- **Unassigned bit:** One unassigned bit is available for future use. This bit MUST NOT be set, and it MUST be ignored on receipt.
- **Length:** Indicates the length of the variable-length metadata, in bytes. In case the metadata length is not an integer number of 4-byte words, the sender MUST add pad bytes immediately following the last metadata byte to extend the metadata to an integer number of 4-byte words. The receiver MUST round the Length field up to the nearest 4-byte-word boundary, to locate and process the next field in the packet. The receiver MUST access only those bytes in the metadata indicated by the Length field (i.e., actual number of bytes) and MUST ignore the remaining bytes up to the nearest 4-byte-word boundary. The length may be 0 or greater. A value of 0 denotes a Context Header without a Variable-Length Metadata field.

4. **NSH Type 2 TLVs**

In [RFC8300] defined that Metadata Class 0x0000 as IETF Base NSH MD Class. In this draft, metadata types are defined for the IETF Base NSH MD Class.

4.1. **Forwarding Context**

This TLV carries a network-centric forwarding context, used for segregation and forwarding scope. Forwarding context can take several forms depending on the network environment. Commonly used data includes VXLAN/VXLAN- GPE VNID, VRF identification or VLAN.
Context Headers to Piggyback State

SUPPORTED BY NSH

Packet encapsulation
Variable length metadata

OUR CONTRIBUTIONS

Extensions to NSH
- State piggybacking using NSH metadata
- Secure state piggybacking
Introduction
Fault Tolerant Chaining
Evaluation
→ Conclusion
FTC keeps a service function chain running after $f \geq 1$ of its service functions fail

- State piggybacking
- In-chain replication

Extending network service header protocol to support fault tolerant service function chaining