FatTire: Declarative Fault Tolerance for SDN

Mark Reitblatt (Cornell)
Marco Canini (TU Berlin → UC Louvain)
Arjun Guha (Cornell → UMass Amherst)
Nate Foster (Cornell)
In a Perfect World...
But in Reality...
Fault-Tolerance Mechanisms

Traditional Networks

• MPLS local path protection
• Global path protection
• IEEE 802.1ag
• and others...

Software-Defined Networks

• Controller reacts to failures
• Fast failover group actions (OpenFlow 1.1+)
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant

Traffic

Figure 1: Example network.
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
Connectivity from GW to A
SSH traffic traverses IDS
SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant

**Figure 1: Example network.**

- **GW**
- **S1**
- **IDS**
- **S2**
- **A**

**Figure 2: Example ruletables and grouptables.**

- **GW Ruletable and Grouptable**
- **S1 Ruletable and Grouptable**
- **S2 Ruletable and Grouptable**

![Traffic Diagram]

**Traffic**
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant

![Diagram showing network connectivity and traffic flow](attachment:image.png)
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant
• Connectivity from GW to A
• SSH traffic traverses IDS
• SSH is 1-link fault tolerant

![Diagram](image)
Overall, the contributions of this paper are as follows:

- **SSH tra**
- **mentation of FatTire as an extension of the NetCore com-
- **hhave carefully engineered the FatTire compiler to correctly
- **figurations requires a global analysis, and (iii) there can be
- **be significantly more challenging compard to other SDN lan-

![Diagram of OpenFlow Fast Failover](image)

<table>
<thead>
<tr>
<th>Match</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tpDst = 22</td>
<td>Group 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FF</td>
<td>〈Fwd S1, Fwd S2〉</td>
</tr>
<tr>
<td>2</td>
<td>FF</td>
<td>〈Fwd A〉</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Match</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>inPort = GW, tpDst = 22</td>
<td>Group 1</td>
</tr>
<tr>
<td>inPort = IDS, tpDst = 22</td>
<td>Group 2</td>
</tr>
<tr>
<td>inPort = S2, tpDst = 22</td>
<td>Group 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FF</td>
<td>〈Fwd IDS〉, 〈Fwd S2〉</td>
</tr>
<tr>
<td>2</td>
<td>FF</td>
<td>〈Fwd A〉</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Match</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>inPort = IDS, tpDst = 22</td>
<td>Group 1</td>
</tr>
<tr>
<td>inPort = S1, tpDst = 22</td>
<td>Group 2</td>
</tr>
<tr>
<td>inPort = GW, tpDst = 22</td>
<td>Group 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FF</td>
<td>〈Fwd A〉, 〈Fwd S1〉</td>
</tr>
<tr>
<td>2</td>
<td>FF</td>
<td>〈Fwd IDS〉</td>
</tr>
</tbody>
</table>
Why not Frenetic?

- Frenetic provides a declarative language for expressing forwarding policies...
- ... in terms of hop-by-hop forwarding steps
- Example:

  \[(GW \rightarrow S1) + (S1 \rightarrow IDS) + (IDS \rightarrow S2) + (S2 \rightarrow A)\]

- What to do if next hop fails?
Our Approach: FatTire

“Fault Tolerating Regular Expressions”

Key Ingredients:

• Hop-by-hop forwarding → paths
• Deterministic → non-deterministic
• Explicit fault-tolerance constructs

Challenges:

• FatTire programs may specify overlapping paths
• OpenFlow tables are deterministic
• Global analysis to provide fault-tolerance guarantees
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1 link fault tolerant

\[(\text{All} \rightarrow [\text{GW} \ast \text{A}])\]
\[\cap (\text{SSH} \rightarrow [* \text{IDS} *])\]
\[\cup \neg\text{SSH} \rightarrow [*]\]
\[\cap (\text{SSH} \rightarrow [*] \text{with 1})\]
\[\cup \neg\text{SSH} \rightarrow [*]\]
Programming in FatTire

Write programs in terms of regular expressions on forwarding paths

- \([GW \ * \ A]\)
- \([GW \ (S1 \ | \ S2) \ A]\)

Use annotations to specify desired fault tolerance

- \(\text{SSH} \rightarrow [\*] \text{ with } 1\)
- \(\neg\text{SSH} \rightarrow [\*] = \neg\text{SSH} \rightarrow [\*] \text{ with } 0\)
Programming in FatTire

Can combine policies with intersection and union:

- Intersection adds restrictions on paths
  
  \[(\text{All} \rightarrow [\text{GW} \ast A]) \cap (\text{SSH} \rightarrow [*] \text{ with 1}) = \text{SSH} \rightarrow [\text{GW} \ast A] \text{ with 1}\]

- Union loosens restrictions on paths
  
  \[(\text{All} \rightarrow [\text{GW} \text{ S1} A]) \cup (\text{All} \rightarrow [\text{GW} \text{ S2} A]) = \text{All} \rightarrow [\text{GW} (\text{S1} \mid \text{S2}) A]\]
FatTire Compiler

1. Normalize into Disjunctive Normal Form
2. Partition into traffic equivalence classes
3. Compute fault-tolerant forwarding graph
4. Output hop-by-hop Frenetic policy and compile to OpenFlow rules

$$\bigcup \bigcap \ldots \bigcup \bigcap \ldots \bigcup \bigcap$$
$$\text{GW} \quad \text{S1} \quad \text{IDS} \quad \text{GW} \quad \text{S2} \quad \text{IDS} \quad \text{S1}$$

$$((\text{GW} \rightarrow \text{S1}) \oplus (\text{GW} \rightarrow \text{S2}))$$
$$+ ((\text{S1} \rightarrow \text{IDS}) \oplus (\text{S2} \rightarrow \text{IDS}))$$
Implementation

• Full working prototype implemented in OCaml

• Based on an extension of the Frenetic controller with support for OpenFlow 1.3

• Tested on CPqD 1.3 software switch

• See paper for preliminary experimental evaluation using Mininet

• Code available from https://github.com/frenetic-lang/fattire under an open-source license
Future Work

• Extend to handle quantitative path properties
  • Bandwidth
  • Latency
• Provide first-class support for other topology changes such as switch failures
• Investigate applications of non-deterministic network programs
• Investigate other recovery mechanisms
Thank You

FatTire Team:

Mark Reitblatt    Marco Canini
Arjun Guha       Nate Foster

Papers, source code, examples, tutorials, etc.
http://frenetic-lang.org
Backup Slides
Update consistency

• Semantics of failure recovery => per-packet consistency
Regular Expression Derivatives
Path Expressions as verification spec

- Dual use as verification specification?
Interaction of paths

All $\rightarrow$ [S1.FW.S3]
U ALL $\rightarrow$ [S2.FW.S4]