**Drift-Bottle:** A Lightweight and Distributed Approach to Failure Localization in General Networks

Xudong Zuo, Qing Li*, Jingyu Xiao, Dan Zhao, Yong Jiang

Tsinghua Shenzhen International Graduate School
Failures in Computer Networks

- **Network Failures**: link failures, link corruptions, node failures, misconfiguration of flow tables...

- **Harm of Network Failures**: impairs network performance by affecting latency and throughput of data transmission

- It is essential for network operators to detect and localize the failed or corrupted links as quick as they can to mitigate the damage
The topologies of general networks are irregular, which may fail the solutions in DCN
Failure Localization in General Networks

- It is hard to deploy monitoring modules on end hosts
Failure Localization in General Networks

- Existing switch-based solutions may introduce too much overhead to network bandwidth
Switches can perceive the occurrence of failures by flow monitoring.
Failure Localization by Multiple Switches

- Multiple switches + multiple flows + data paths = failure location

\[
A_{host} = \begin{pmatrix}
    l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \\
    1 & 1 & 0 & 0 & 0 & 0 \\
    1 & 0 & 1 & 0 & 1 & 1 \\
    0 & 1 & 1 & 0 & 1 & 1 \\
    0 & 0 & 1 & 1 & 0 & 0 \\
\end{pmatrix}
\]

- $h_1 \leftrightarrow h_2$
- $h_1 \leftrightarrow h_3$
- $h_2 \leftrightarrow h_3$
- $h_3 \leftrightarrow h_4$
Failure Localization by Multiple Switches

- Multiple switches + multiple flows + data paths = failure location

![Diagram showing network topology with switches and hosts]

\[ A_{\text{switch}} = \begin{pmatrix}
    l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \\
    1 & 0 & 0 & 0 & 1 & 0 \\
    0 & 1 & 0 & 0 & 1 & 0 \\
    0 & 0 & 1 & 0 & 0 & 1 \\
    0 & 0 & 0 & 1 & 0 & 1 \\
    1 & 0 & 0 & 0 & 1 & 1 \\
    0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \quad \begin{aligned}
    h_1 & \leftrightarrow s_2 \\
    h_2 & \leftrightarrow s_2 \\
    h_3 & \leftrightarrow s_2 \\
    h_4 & \leftrightarrow s_2 \\
    h_1 & \leftrightarrow s_3 \\
    h_4 & \leftrightarrow s_3
\end{aligned} \]
Inference Aggregation by ‘Drift’

- Collect the inference along the data path to localize potential failures
Overview of Drift-Bottle

Introduction
Motivation
Overview
Design
Evaluation
Flow Monitoring Module

- **Goal**: find the flows influenced by potential failures
- **Why Decision Tree**: easy to be transformed into entries of match-action tables on the data plane
- Operators can customize different flow classifiers

Flow Status: Normal or Abnormal
Flow Monitoring Module

- Measures and Features
Flow Monitoring Module

- Definition of measures and features

Control Plane

**Timer**

Saved Features

Data Plane

Feature Extractor

<table>
<thead>
<tr>
<th>Measure</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_packet</td>
<td>number of received packets</td>
</tr>
<tr>
<td>len_all</td>
<td>total size of received packets</td>
</tr>
<tr>
<td>len_max</td>
<td>size of the largest packet</td>
</tr>
<tr>
<td>len_last</td>
<td>size of the last packet</td>
</tr>
<tr>
<td>n_burst</td>
<td>number of bursts</td>
</tr>
<tr>
<td>pos_burst</td>
<td>position of the last burst</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Feature</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_flow</td>
<td>RTT</td>
<td>RTT of monitored flow</td>
</tr>
<tr>
<td></td>
<td>len_path</td>
<td>length of data path of flow</td>
</tr>
<tr>
<td></td>
<td>n_interval</td>
<td>n of intervals to cover a RTT</td>
</tr>
<tr>
<td>f_avg</td>
<td>avg_n_packet</td>
<td>avg. n_packet of intervals in last RTT</td>
</tr>
<tr>
<td></td>
<td>avg_len_all</td>
<td>avg. len_all of intervals in last RTT</td>
</tr>
<tr>
<td></td>
<td>avg_len_max</td>
<td>avg. len_max of intervals in last RTT</td>
</tr>
<tr>
<td></td>
<td>avg_len_last</td>
<td>avg. len_last of intervals in last RTT</td>
</tr>
<tr>
<td></td>
<td>avg_n_burst</td>
<td>avg. n_burst of intervals in last RTT</td>
</tr>
<tr>
<td></td>
<td>avg_pos_burst</td>
<td>avg. pos_burst of intervals in last RTT</td>
</tr>
<tr>
<td>f_last</td>
<td>last_n_packet</td>
<td>n_packet in last interval</td>
</tr>
<tr>
<td></td>
<td>last_len_all</td>
<td>len_all in last interval</td>
</tr>
<tr>
<td></td>
<td>last_len_max</td>
<td>len_max in last interval</td>
</tr>
<tr>
<td></td>
<td>last_len_last</td>
<td>len_last in last interval</td>
</tr>
<tr>
<td></td>
<td>last_n_burst</td>
<td>n_burst in last interval</td>
</tr>
<tr>
<td></td>
<td>last_pos_burst</td>
<td>pos_burst in last interval</td>
</tr>
</tbody>
</table>
**Inference Generation Module**

- **Goal**: generate the local inference of potential failures with abnormal flows and their data paths

- **Inference Format:**

  - The times of aggregation (for inference aggregation)
  - Serial number of link
  - Weight of link

  ![Diagram](image)

  - Total length: 9B for k = 4

---

Introduction | Motivation | Overview | Design | Evaluation
Inference Generation Module

- Weight assignment scheme without the information from normal flows

$h_1 \rightarrow h_9 \quad \sqrt{\rightarrow} \times$
$h_2 \rightarrow h_9 \quad \sqrt{\rightarrow} \times$
$h_3 \rightarrow h_9 \quad \sqrt{\rightarrow} \times$
$h_4 \rightarrow h_9 \quad \sqrt$

$\ldots$

$h_8 \rightarrow h_9 \quad \sqrt$

$W_1 = 3$
$s$
$W_2 = 2$

$h_9 \rightarrow h_1 \times$
$h_{10} \rightarrow h_1 \times$
Inference Generation Module

- Weight assignment scheme with the information from normal flows

\[ W_1 = -2 \quad W_2 = 2 \]
Inference Generation Module

- Algorithm

**Algorithm 1: Local Inference Generation**

**Input:** \( F \) - set of monitored flows, \( P \) - upstream data paths of flows, \( L \) - set of links, \( S \) - status of monitored flows, \( k \) - length of inference

**Output:** \( I \) - local inference about failures

1. \( IF \leftarrow 0 \) on the data plane;
2. for \( f \in F \) do
   3. \( path_f \leftarrow \) upstream data path of \( f \) from \( P \);
   4. \( status_f \leftarrow \) status of \( f \) from \( S \);
   5. if \( status_f = \) abnormal then
      6. \( IF \leftarrow \{(l_i, 1) \mid \forall l_i \in path_f \} \);
   7. else
      8. \( IF \leftarrow \{(l_i, -1) \mid \forall l_i \in path_f \} \);
   9. end if
10. \( IF \leftarrow IF \cup \{I_f\} \);
11. end for
12. Upload \( IF \) to the control plane;
13. \( I \leftarrow \{(l_i, 0) \mid \forall l_i \in L \} \) on the control plane;
14. for \( I_f \in IF \) do
15. \( I \leftarrow I \oplus I_f \);
16. end for
17. Remove \( (l_i, w_i) \) from \( I \) if \( w_i = 0 \);
18. Sort \( I \) = \( (l_i, w_i) \) in descending order by \( w_i \);
19. Truncate \( I \) to the \( k \)-th \( (l_i, w_i) \);
20. Send \( I \) to the data plane;
21. return \( I = \{(l_i, w_i)\} \)

**Aggregation Operator** \( \oplus \): adds the weight of the same links from two inferences, maintains the others
Inference Aggregation Module

- **Goal**: uses normal packets in the network to aggregate inferences from different switches

- **Inference Processing Logic**:

```
<table>
<thead>
<tr>
<th>Inference</th>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
</table>
```

Data Plane

```
Inference ⊕ Inference = Inference
```

Local

```
P4 Switch
```

```
Warning Alarm
```

Introduction | Motivation | Overview | Design | Evaluation
Inference Aggregation Module

- Switch keeps its local inference unchanged in order to avoid over aggregation

- Over Aggregation:

```
S3  S1  S2
```

```
i1+i2 pkt1 i2
```

```
i1+2i2 pkt2 i2
```

```
i1+3i2 pkt3 i2
```

...
Inference Aggregation Module

● Warning Raising Mechanism

\[ \text{condition 1} \]

\[ \text{condition 2} \]

\[ \text{condition 3} \]

\[ \text{hop}_{\text{now}} \geq \text{hop}_{\text{min}} \]

\[ w_1 \geq \alpha \times \text{hop}_{\text{now}} \]

\[ w_1 \geq \beta \times w_2 \]

\[ \text{hop}_{\text{min}} \text{ and } \alpha \text{ are preset thresholds related to the scale of the network} \]

\[ \text{The selection of } \beta \text{ is irrelevant to the topology. Read our paper for more details} \]

Introduction | Motivation | Overview | Design | Evaluation
Evaluation Setup

• Simulation by Mininet on 4 chosen topologies
• Generate random traffic with the injection of link failures and corruptions
• Statistics of the chosen topologies:

<table>
<thead>
<tr>
<th>Topology</th>
<th>Node</th>
<th>Link</th>
<th>VAR. of link latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geant2012</td>
<td>40</td>
<td>61</td>
<td>14.12</td>
</tr>
<tr>
<td>Chinanet</td>
<td>42</td>
<td>66</td>
<td>8.09</td>
</tr>
<tr>
<td>Tinet</td>
<td>53</td>
<td>89</td>
<td>247.64</td>
</tr>
<tr>
<td>AS1221</td>
<td>104</td>
<td>151</td>
<td>9.39</td>
</tr>
</tbody>
</table>
Length of Inference

Geant2012

Chinanet

Introduction | Motivation | Overview | Design | Evaluation
Weight Assignment Scheme

Drift-Bottle: \((1, -1)\)
Non-Negative: \((1, 0)\)

007-Drifted: \((1/n, 0)\)
007-Modified: \((1/n, -1/n)\)
Single Failure Scenario (Chinanet)
Multiple Failures Scenario (Chinanet)

- Multiple link failures caused by a single node failure
Multiple Failures Scenario (Chinanet)

- Random multiple failures

![Graphs showing metrics vs number of failures for link and node failures.](image-url)
Warning Locality

Chinanet (link)

Chinanet (node)
Conclusion

• We introduce Drift-Bottle, a lightweight and distributed approach to failure localization in general networks.
• Drift-Bottle utilizes the in-network intelligence technique to detect flow-level anomalies on switches, then generates concise inferences about potential failures with information of data paths.
• Instead of a centralized mechanism, Drift-Bottle uses a distributed mechanism for inferences aggregation, which avoids high bandwidth overhead and additional infrastructural modification in networks.
Thanks!