Sparse Control and Data plane Telemetry features for BGP anomaly detection

Jose Cordova-Garcia
ESPOL University, Ecuador
Motivation: Anomaly detection

- Timely detection of network failures is crucial to support daily operations.
- Core (e.g. a data center) networks should rely on automatized detection.
- Problems?
  - Detection based on monitoring features, which features?
  - High-resolution/high-dimensionality monitoring data become available, which nodes should provide data?
  - Operators favor visual inspection, can we work with raw features?

- Traditional monitoring:
  - Visual inspection at NOCs
  - CLI, scripts, active polling
    - Polling methods: SNMP
A motivating network

- Data center network running BGP (DCN)
- Network anomalies
  - Network topology changes
  - Policy changes
  - Misconfigurations
  - Attacks
  - Device failure

https://github.com/cisco-ie/telemetry/
How to detect anomalies

<table>
<thead>
<tr>
<th>Work</th>
<th>Data Collection</th>
<th>Features used</th>
<th>Detection Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huang’07</td>
<td>10 minutes</td>
<td>1 feature</td>
<td>PCA</td>
</tr>
<tr>
<td>Deshpande’09</td>
<td>5 minutes</td>
<td>4 features</td>
<td>Normality test</td>
</tr>
<tr>
<td>Al-Rousan’12</td>
<td>1 minute</td>
<td>17 features, select 10</td>
<td>Naive Bayes</td>
</tr>
<tr>
<td>Al-Roustan’12</td>
<td>1 minute</td>
<td>37 features, select 10</td>
<td>SVMs, HMM</td>
</tr>
<tr>
<td>Ding’16</td>
<td>1 minute, bin</td>
<td>37 features</td>
<td>LSTM</td>
</tr>
<tr>
<td>Cheng’16’19</td>
<td>1 minute, bin</td>
<td>33 features</td>
<td>Multistage-LSTM</td>
</tr>
<tr>
<td>Nguyen’19</td>
<td>3 minute bins</td>
<td>55 features</td>
<td>VAE</td>
</tr>
</tbody>
</table>
What about a Telemetry network?

- A Telemetry-enabled network:
  - Nodes automatically stream their status data
  - Seconds’ resolution
  - Data available is pre-configured\(^1\)

- When a node fails, BGP will say something!
  - Referred works apply

- What would Telemetry say?
  - Which features?
  - From where?
  - Transform?

\(^1\) https://github.com/cisco-ie/telemetry/
Proposed Scheme

- **Most common:**
  - Separate FE from Learning Model
  - **UPDATE** features inherited from literature

- **Now, Telemetry provides more features**
  - Proposal: Integration

```
Raw Telemetry data
M samples
T nodes, F IFs
```

```
Pre-processing
N = C_p + D_p F
```

```
L1 Sparse Classifier S
```

```
Instance extraction
```

```
YANG Models
C_p, D_p
```

```
RFE
```

```
Detection model
```

```
Score
```

```
Sparse-RFE YANG features ready for detection
```

```
K Sparse-RFE features
```

```
Reduce S
```

```
Score
```

```
S sparse features
```

```
N data features
```

```
C
```

```
K
```
Telemetry data

- More than BGP **UPDATE**s
- Vendor provides data based on YANG models
- Model: encoded tree
  - Data/Control plane trees available.
  - Leaf: **YANG feature**
- Operator configures *Model Driven Telemetry (MDT)* © at Telemetry nodes
  - Each node provides its own instances of YANG features

In the network: **Telemetry nodes=15**, 12 fabric, 3 edge
  - IFs: leaf/spine 34, edge 74
Telemetry data

- Nodes provide $N$ data features to the detection model

- Network-wide monitoring:
  - Collect $N=17000+$ features
  - Which data features are important?
  - Moreover, which YANG leafs are important?

\[
N = C_p + D_p F
\]

Diagram:

- Raw Telemetry data
- $M$ samples
- $T$ nodes, $F$ IFs
- Pre-processing
- $N = C_p + D_p F$
- YANG Models
- $C_p, D_p$
Stage 1: Sparse features

- Let the machine decide what is important for detection
  - Start with data features (node-level instances of YANG leafs)
  - Sparse classifier for feature selection:
    - Lasso, reg. SVM, Coordinate Lasso
  - S sparse features
Stage 1: Sparse features

- **S sparse features**

<table>
<thead>
<tr>
<th>S</th>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>130</td>
<td>Lasso</td>
<td>94.65%</td>
<td>94.11%</td>
<td>93.20%</td>
</tr>
<tr>
<td>116</td>
<td>SVM-L1</td>
<td>95.06%</td>
<td>94.17%</td>
<td>94.17%</td>
</tr>
<tr>
<td>1945</td>
<td>Coord. Lasso</td>
<td>95.06%</td>
<td>96.90%</td>
<td>91.26%</td>
</tr>
</tbody>
</table>

![Diagram showing L1 Sparse Classifier and data flow](image)
Stage 2: Sparse-RFE

- Push forward → Let the machine decide which important features contribute to better detection
  - Embed model and performance into feature selection
  - Select $K$ Sparse-RFE features
Stage 2: Sparse-RFE

- $K$ Sparse-RFE features
- (data features)

<table>
<thead>
<tr>
<th>$K$</th>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Lasso</td>
<td>94.23%</td>
<td>94.95%</td>
<td>91.26%</td>
</tr>
<tr>
<td></td>
<td>SVM-l1</td>
<td>91.35%</td>
<td>88.67%</td>
<td>91.26%</td>
</tr>
<tr>
<td></td>
<td>Coord. Lasso</td>
<td>90.12%</td>
<td>96.47%</td>
<td>79.61%</td>
</tr>
<tr>
<td>20</td>
<td>Lasso</td>
<td>94.65%</td>
<td>93.26%</td>
<td>94.17%</td>
</tr>
<tr>
<td></td>
<td>SVM-l1</td>
<td>95.35%</td>
<td>95.04%</td>
<td>93.20%</td>
</tr>
<tr>
<td></td>
<td>Coord. Lasso</td>
<td>95.47%</td>
<td>96.93%</td>
<td>92.23%</td>
</tr>
<tr>
<td>30</td>
<td>Lasso</td>
<td>94.65%</td>
<td>95.00%</td>
<td>92.23%</td>
</tr>
<tr>
<td></td>
<td>SVM-l1</td>
<td>94.65%</td>
<td>93.26%</td>
<td>94.17%</td>
</tr>
<tr>
<td></td>
<td>Coord. Lasso</td>
<td>96.29%</td>
<td>95.19%</td>
<td>96.11%</td>
</tr>
<tr>
<td>40</td>
<td>Lasso</td>
<td>95.47%</td>
<td>96.00%</td>
<td>93.20%</td>
</tr>
<tr>
<td></td>
<td>SVM-l1</td>
<td>95.88%</td>
<td>94.28%</td>
<td>96.11%</td>
</tr>
<tr>
<td></td>
<td>Coord. Lasso</td>
<td>96.29%</td>
<td>95.19%</td>
<td>96.11%</td>
</tr>
<tr>
<td>52</td>
<td>Lasso</td>
<td>93.82%</td>
<td>93.13%</td>
<td>92.23%</td>
</tr>
<tr>
<td></td>
<td>SVM-l1</td>
<td>94.65%</td>
<td>94.11%</td>
<td>93.20%</td>
</tr>
<tr>
<td></td>
<td>Coord. Lasso</td>
<td>96.70%</td>
<td>97.02%</td>
<td>95.14%</td>
</tr>
</tbody>
</table>

*a Selected through crossvalidation.

Performance doesn’t degrade greatly
Stage 2: Sparse-RFE

- **$K$ Sparse-RFE features**
  - Data features decreased to $K$
  - Comparable performance
  - YANG features??

[Diagram showing the process of Sparse-RFE with different values of $K$ (30, 40, auto) and the comparison of Lasso, SVM-I1, and Coord. Lasso for each case. The diagram also includes a section labeled Yang features.]
What Telemetry data are important?

- i.e. which YANG leafs have been instantiated?
  - 12 CP+DP common YANG features

<table>
<thead>
<tr>
<th>Plane</th>
<th>Instances</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>deleted-routes-count</td>
<td>Node</td>
</tr>
<tr>
<td></td>
<td>global-established-neighbors-count*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>protocol-route-memory</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>free-application-memory</td>
<td>Node</td>
</tr>
<tr>
<td></td>
<td>free-physical-memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>incomplete-adjacency-packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no-route-packets</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ram-memory*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>system-ram-memory*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total-cpu-fifteen-minute*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total-cpu-five-minute*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total-cpu-one-minute*</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>input-data-rate*</td>
<td>Interface</td>
</tr>
<tr>
<td></td>
<td>input-load*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>input-packet-rate*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>output-data-rate*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>output-drops*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>output-load*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>output-packet-rate*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reliability*</td>
<td></td>
</tr>
</tbody>
</table>

*K* Common to any choice of $K$. 

![Diagram showing YANG Models and Sparse-RFE features](image-url)
Monitoring using Sparse-RFE

- **Which features?** 12 CP+DP YANG leafs are important for detection

- **From where?**
  - Data features could be down to small $K \rightarrow$
  - Keep $T$ small:
    - For visualization
    - To avoid transformation
  - Any node?
    - Avoid failure nodes

- Pick a ML model, test with:
  - A: 1 spine, 1 leaf, 1 edge node
  - B: 1 edge, 2 leafs nodes
  - C: 3 fabric nodes
  - D: 3 edge nodes

<table>
<thead>
<tr>
<th>Set</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>YANG CP+DP</th>
<th>Total features</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>95.46</td>
<td>95.98%</td>
<td>93.20%</td>
<td></td>
<td>1028</td>
</tr>
<tr>
<td>B</td>
<td>95.88</td>
<td>96.03%</td>
<td>94.17%</td>
<td></td>
<td>1028</td>
</tr>
<tr>
<td>C</td>
<td>94.23</td>
<td>94.95%</td>
<td>91.26%</td>
<td></td>
<td>756</td>
</tr>
<tr>
<td>D</td>
<td>93.41</td>
<td>93.93%</td>
<td>90.20%</td>
<td>12</td>
<td>1644</td>
</tr>
</tbody>
</table>
What did we learn?

- Telemetry data can be big, careful preprocessing is needed for visualization and interpretability.
- Automatically selected YANG features can reduce the number of monitoring nodes and data processed.
  - Operators can select the predictors to inspect using Sparse-RFE
- Not all CP/DP leafs are needed but they enable other applications, e.g. localization.
Thank you

- Currently working on:
  - Localization through multi-class/multi-label techniques.
  - Exploring larger YANG trees.
  - Efficient Telemetry placement.

- Contact:
  Jose Cordova-Garcia
  jecordov@espol.edu.ec
Summary

- Vast literature on anomaly detection,
  - Steady interest on BGP anomalies
- Based on ML techniques:
  - PCA [Huang'07]
  - SVMs [Al-rousan'12]
  - DBScan [Putina’18]
  - LSTM [Chen’19]

- Revise the data collection and features used.
  - Data collection → Telemetry → Large monitoring data
  - Telemetry features for detection of node failures:
    - Which features?
    - From where?
    - Transform?

https://tinyurl.com/yyu5j73
Extra info on other methods
Features used in automated methods

- Al-Rousan’12: 17 features of volume and AS-path: numbers, means, and maximum values. Samples are 1 minute resolution. 10 features are selected and use Naive Bayes.

- Al-Rousan’12: 20 new features based on common values of AS-path lengths and edit distances are added and use SVMs and HMMs.
Features used in automated methods

- Based on BGP update messages, commonly accepted [Ding’16].
  - Volume: # of BGP announcements/withdrawals
  - AS-path: length and edit distance
  - Ding’16: 37 features from BGP updates, 1 minute resolution
Features used in automated methods

- Deshpande’09: 4 features from BGP updates, but may be less, 5 minute resolution, inspect per-feature and analyze correlations.
Features used in automated methods

- **Cheng’16**: 33 BGP traffic updates, aim to find the proper scale for processing features by averaging $p$ samples, i.e. in a way finding the right bin size. Once found, use LSTM for detection. Considers minute resolution data points.
Features used in automated methods

- 53 aggregated features: mean and std. Deviation of average packet size, entropy of destination ports, etc. [Nguyen’19]
  - Stats from netflow data binned in 3 minutes flows VAE
  - Bins with less than 10 netflow records are removed.

- Flow-based methods rely on proper stats and granularity (bin/window size) gathered in the flow and source nodes in that flow, not really polling status.