Learning Transfers via Transfer Learning

Md Arifuzzaman and Engin Arslan
University of Nevada, Reno
Data transfers often experience performance issues due to various reasons:
- I/O interference, network congestion, packet loss, misconfigured end hosts, etc.

Performance issues can have severe impacts on application performance:
- Packet loss of 0.1% leads to 5-7x decrease in throughput
- Real-time streaming services affected negatively by increased network jitter
Finding the **root causes of anomalies** is key to quickly recover from anomalies.

Different anomaly types can result in similar outcomes such as throughput decrease.

<table>
<thead>
<tr>
<th>Transfer Condition</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>612 Mbps</td>
</tr>
<tr>
<td>Delay Jitter</td>
<td>285 Mbps</td>
</tr>
<tr>
<td>Packet Duplication</td>
<td>327 Mbps</td>
</tr>
<tr>
<td>Packet Reorder</td>
<td>428 Mbps</td>
</tr>
<tr>
<td>Packet Loss</td>
<td>154 Mbps</td>
</tr>
<tr>
<td>Packet Corruption</td>
<td>198 Mbps</td>
</tr>
</tbody>
</table>
Related Work: Anomaly Detection

• Statistical Model
  • Separate anomalous and normal transfers
  • Example: Change-point detections, principal component analysis

• Supervised Model
  • Binary classification models

• Unsupervised Model
  • Time-series analysis, clustering

Limitations
• No support for anomaly diagnosis
• Can be sensitive to data drift
Related Work: Anomaly Diagnosis

• Supervised Model
  • Multi-label classifications
  • Predict types of anomalies such as packet loss, reordering or duplication

• Heuristic Solution
  • Decision tree-like classification tree but developed heuristically

Limitations
  • Sensitive to data drift
  • Domain dependent
Proposed Solution

1. Train ML classifiers to predict the root causes of performance issues using socket statistics

2. Mitigate the need for labeled data collection in each network by eliminating/transforming domain dependent features

Random Forest Classifier

- Packet Loss
- Jitter
- Packet Reorder

Socket statistics
Data Collection and Modeling
Data Collection

- Created 9 networks with different bandwidth and delay settings in Emulab
  - e.g., 100Mbps and 100ms, 5000 Mbps and 30ms
- Five anomaly types reproduced as *loss, corruption, reorder, duplicate*, and *jitter* using *Linux traffic control module (tc)*
- Executed 600 sample transfers per anomaly type in each network
- Used **ss** command to capture socket statistics

Source  

*Bridge*: Emulate network and anomalies  

Destination
Socket Statistics (ston)

- utility tool to investigate sockets
- Reports 30+ flow-levels metrics
- Metrics are instantly available
- Collected metrics at 10th second to wait for transfer performance to stabilize
Feature Engineering

• Eliminate redundant features
  • Reduce training cost and improve accuracy

• Selected features with combined 95% importance score
  • Ended up with *nine* features
Model Evaluation (Emulab)

- Trained separate ML models for each network
- Hyperparameters are tuned with Scikit-Optimize library
- F-measure used as the performance metric
Transfer Learning

Separate Random Forest classifier trained for each network

Each classifier is evaluated in other networks

Train custom models for each network?

ML models fail to achieve good performance when transferred to different networks.
Transfer Learning – Proposed Solution

- Standard normalization results in different distributions in transfer learning

\[ Z = \frac{x - \mu}{\sigma} \]

**Feature Transformation**: Remove network dependency from collected metrics.

- E.g., divide retransmitted packet count to total packets to convert it to the ratio.

![Graph showing packet loss vs transfer ID for different conditions](image)
Transfer Learning – Proposed Solution

Significant improvement in transfer learning performance. From less than 60% to over 85% in Emulab testbeds
## Evaluation in Production Networks

<table>
<thead>
<tr>
<th>Train Dataset</th>
<th>Test Dataset</th>
<th>F-measure (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulab (Salt Lake City)</td>
<td>HPCLab (Reno)</td>
<td>81.9 ± 2.8</td>
</tr>
<tr>
<td>HPCLab (Reno)</td>
<td>Emulab (Salt Lake City)</td>
<td>81.1 ± 3.7</td>
</tr>
<tr>
<td>Austin-Chicago</td>
<td>Austin-San Diego</td>
<td>81.2 ± 1.6</td>
</tr>
<tr>
<td>Austin-San Diego</td>
<td>Austin-Chicago</td>
<td>79.8 ± 1.8</td>
</tr>
</tbody>
</table>

* Please see networks specifications in the paper
Impact of Congestion Control Algorithms

BBR significantly outperforms Cubic and HTCP

BBR has more distinguishable feature values
Explainable Models

- Random Forest or XGBoost models consist of hundreds of decision trees
- Debugging is severely challenging
- The Decision tree method with the original seven features still build very complicated trees (as many as 19 depths)
- Eliminated least important 3 features
- Performance drop 2-5%, but makes model interpretable
ML models perform well diagnosing anomalies but require labeled data collection for each network due to poor transfer learning performance.

Feature transformation eliminates domain dependency and significantly improves transfer learning performance.

BBR based model perform better compared to other congestion control algorithms.
Thanks!

This project is funded by National Science Foundation under award number 2007789