ConfigTron: Tackling network diversity with heterogeneous configurations

Usama Naseer, Theophilus Benson
Duke University
Diversity in network conditions

<table>
<thead>
<tr>
<th>Region</th>
<th>Avg. RTT in ms</th>
<th>Avg. packet loss %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans-Atlantic</td>
<td>74</td>
<td>0.05</td>
</tr>
<tr>
<td>Europe</td>
<td>11.25</td>
<td>0.05</td>
</tr>
<tr>
<td>North America</td>
<td>35</td>
<td>0.05</td>
</tr>
<tr>
<td>Intra-Japan</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Trans-Pacific</td>
<td>102.8</td>
<td>1</td>
</tr>
<tr>
<td>Asia Pacific</td>
<td>102.6</td>
<td>1</td>
</tr>
<tr>
<td>Latin America</td>
<td>135</td>
<td>1</td>
</tr>
<tr>
<td>EMEA to Asia Pacific</td>
<td>120.5</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>RTT in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>UK to Oceanic</td>
<td>~400</td>
</tr>
<tr>
<td>US to Southeastern Asia</td>
<td>~260</td>
</tr>
<tr>
<td>UK to East Asia</td>
<td>~350</td>
</tr>
<tr>
<td>North-America to India</td>
<td>~380</td>
</tr>
</tbody>
</table>

http://www.verizonenterprise.com/about/network/latency/#latency
Content serving stack

Web App

HTTP

TLS

TCP

What we can have

Compression strategies

HTTP/1.1  HTTP/2

v1.0, v1.1, v1.2, v1.3

Cubic, Reno, Vegas, BBR, etc.

ICW, RTO, FastOpen etc.

Choice of queuing FIFO, FQ etc.

Web App

HTTP/2

QUIC

UDP

Server OS
Content serving stack

What we actually have $\rightarrow$ **one size fits all!**
In the light of divergent network characteristics, is the **one-size-fits-all** networking stack optimal?
Are the configurations being used optimal for everyone?

- Verus (NSDI ‘15) - TCP versions

- How quick is QUIC? (IEEE ICC ‘16) - QUIC vs HTTP2 vs HTTP1.1

- Overclocking the Yahoo!: CDN for faster web page loads. (ACM SIGCOMM '11)

A broader study to understand the impact of reconfiguration across different layers.
Experimental setup

• Controlled website replay  
  - MahiMahi

• Network emulation  
  - NetEM + TC

• Realistic workloads  
  - Alexa top 100

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>● RTT</td>
</tr>
<tr>
<td></td>
<td>● Bandwidth</td>
</tr>
<tr>
<td></td>
<td>● Loss rate</td>
</tr>
<tr>
<td>TCP/IP stack</td>
<td>● TCP ICW</td>
</tr>
<tr>
<td></td>
<td>● Congestion control</td>
</tr>
<tr>
<td></td>
<td>● Slow_start_after_idle</td>
</tr>
<tr>
<td></td>
<td>● Low_latency</td>
</tr>
<tr>
<td></td>
<td>● Auto_corking</td>
</tr>
<tr>
<td></td>
<td>● Packet scheduling</td>
</tr>
<tr>
<td>Web App</td>
<td>● HTTP protocol</td>
</tr>
<tr>
<td>Website</td>
<td>● Web object size</td>
</tr>
<tr>
<td></td>
<td>● # of objects</td>
</tr>
</tbody>
</table>
Reconfiguration improves performance!

Loss > 2.5%
BW < 1Mbps
Higher number of objects
Given the benefits of reconfiguration, how can we use network conditions reconfigure on the go?
ConfigTron optimizes web performance by **systematically reconfiguring** web servers in a principled manner.
ConfigTron architecture

**Request Router**
- Forwards user’s request to the reconfigured server.

**Configuration Manager**
- Predicts optimal config.
- Propagates information.

**Web server**
- Gets mappings from config. manager.
- Reconfigures servers stack.

**Config. Agent**
- Not required for every request.
Configuration Manager

Learning Database

Learning Function
Learning Database
• Generates data related to performance of different configurations.

Synthetic data:
• Mahimahi based web emulation
  • Pros
    • Can be generated offline.
  • Cons
    • Requires lots of resources.
    • Requires lots of execution time.

Real world data:
• A/B tests
  • Pros
    • Dynamic.
    • Adjusts with time.
  • Cons
    • Impacts real users.
    • Needs mechanism for selection of real users. (Kraken OSDI ‘16)
Learning Function

- Aggregates data from learning database to predict optimal configurations.
- Requires a learning algorithm to learn mappings from network conditions to configurations.
- Machine Learning (Decision trees, SVM), Deep Learning (Reinforcement learning), Statistical (Markov process)

Decision Trees

- One decision tree per website category.

Configuration Manager

Learning Database | Learning Function
Challenges in reconfiguring today’s stack

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Values</th>
<th>Configuration Granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP stack</td>
<td>TCP ICW</td>
<td>Per connection (IP tables)</td>
</tr>
<tr>
<td></td>
<td>Congestion control</td>
<td>Per connection (setsockopt)</td>
</tr>
<tr>
<td></td>
<td>Auto_corking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low_latency</td>
<td>Kernel wide (proc interface)</td>
</tr>
<tr>
<td></td>
<td>Slow_start_after_idle</td>
<td></td>
</tr>
</tbody>
</table>

- TLS and HTTP layers are more flexible due to user space.
- Challenges:
  - Requires rewriting chunks of kernel.
  - User space TCP stacks are limited in configuration diversity (eg, only cubic or reno).
ConfigTron architecture (VM based)
ConfigTron architecture (Socket based)
VM based:

Pros:
- VMs can be reconfigured in isolation.
- Fine-grained control over resource usage.

Cons:
- Higher resource usage.
- Traps resources.
- High startup times. May need to have VMs on standby.

Socket based:

Pros:
- Less resource intensive.
- Can be created on-the-go.

Cons:
- Not all TCP parameters are reconfigurable by setsockopt.
Related work

- Overclocking Yahoo! CDN (ACM SIGCOMM ‘11) looks at configuring TCP at end hosts in a datacenter environment where controller has a view of entire network.

- We are broader in the sense that we are looking at more parameters across different layers.

- OpenTCP (Hot Topics in networks, ACM ‘12.) reconfigures TCP in data center environment whereas Software Defined Transport (ONS ‘14) presents a programmable centralized platform to define transport mechanisms in data centers with a central controller. Similarly TROLL (IEEE NetSys ’17) uses central controller for protocol suggestions and makes decisions for network core and edge devices.

- ConfigTron is different since it reconfigures multiple layers in CDN server to optimize web performance, without the presence of a centralized controller.

- No control over end hosts (users) in case of web.
Conclusion

• One-size-fits-all configurations can be sub-optimal for a proportion of users.

• Network diversity can be tackled by heterogeneity in configurations.

• Reconfiguration can improve PLT by as much as 40%.
Future work + Discussion

• Inspector Gadget -- infers network stack configuration for web servers by active probing.

• AWS + PlanetLab deployment.

• Extending user-space TCP stacks.

• Network Estimation.
Questions?

unaseer@cs.duke.edu
How many configurations are too many?
No configuration is better than others!

Key:
(tcp, initvmd, auto_coikng, low-latency, slow_start_after_idle)
C1: vegas, 18, 0, 0, 0
C2: vegas, 18, 1, 1, 0
C3: cubic, 18, 0, 0, 0
C4: reno, 18, 0, 1, 0
Network coverage wrt number of configurations