LiteLab: Efficient Large-scale Network Experiments

Liang Wang, Arjuna Sathiaseelan, Jon Crowcroft, Jussi Kangasharju*
Presented by Otto Waltari*

University of Cambridge, UK  University of Helsinki, Finland*
Content

• Motivation
• Background
• Architecture
• Evaluation
• Discussion
• Conclusion
Motivation - Challenges

• Deploying network systems in developing regions is a very difficult task, even for those mature and well-tested systems.

• There are both technical and non-technical barriers such as
  • extreme environment,
  • inefficient transportation,
  • lack of local technicians,
  • limited equipment,
  • poor infrastructure,
  • and many other socio-economical challenges.
Motivation - Requirements

• "Trial and error" method will simply fail in the rural context due to its high cost. To reduce both investment risk and maintenance overhead, we need to obtain a comprehensive understanding of the system.

• Due to the large parameter space, researchers usually need to run thousands of experiments with different parameter combinations.
  • Replayability is critical in modern network experiments.
  • Flexibility in the platform: easy to configure and extend.
  • Take advantage of the computation and storage resources in the cluster.
  • Reduce the experiment complexity and speeds up experiment life-cycle.
  • Provide satisfying accuracy.
Background – Multiple Options

• There are generally two methodologies to evaluate a system: model-based evaluation and experiment-based evaluation.

• Model-based: apply analytical models, mathematically tractable; but the complexity may explode as a system becomes more complicated.

• Experiment-based: take advantage of the ever-growing computation power to explore the problem space; suitable for large and complex systems; simulation, emulation, and real testbed.
Three Exemplars

• Simulators: NS2 and NS3
  • one of the most famous among general purpose simulators;
  • has many tunable parameters to allow more realistic settings;

• Emulator: Emulab
  • tries to integrate simulation, emulation and live network into a common framework; the configuration and setup can be quite complicated.

• Internet: PlanetLab
  • a platform for live network experiments. The traffic goes through the real Internet and is subject to real-life dynamics.

• Many others, specialised in different.
LiteLab Architecture

• LiteLab consists of two subsystems: Agent Subsystem and Overlay Subsystem.

• All experiments are jobs in LiteLab and are defined by a job description archive provided by a user.

• A user submits the job to LiteLab which processes the job description archive, determines needed resources and allocates necessary physical nodes from the available nodes.

• LiteLab informs the selected nodes and deploys an instance of the Overlay Subsystem on them.

• LiteLab starts the experiment, and the logs are saved.
LiteLab Architecture
Agent Subsystem

• The Agent Subsystem provides a stable and uniform experiment infrastructure. It hides the communication complexity, resource failures and other underlying details from the Overlay Subsystem.

• There are three main components in Agent Subsystem:
  • NodeAgent represents a physical node;
  • JobControl manages all the submitted jobs in LiteLab;
  • Mapping maps virtual resources to physical resources;
Resource Allocation – Static Mapping

• Resource allocation focuses on the mapping between virtual nodes and physical nodes, and it is the key to platform scalability.
• The mapping maximises the resource utilisation, guarantee there is no violation of physical capacity.
• Four metrics are taken into account as the constraints: CPU load, network traffic, memory usage and use of pseudo-terminal devices.
• The mapping is formulated as an optimisation problem.
Static Mapping – Inputs and Outputs

Node States

<table>
<thead>
<tr>
<th>PHY #</th>
<th>CPU</th>
<th>Memory</th>
<th>Traffic</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>500</td>
<td>129</td>
<td>332</td>
<td>57</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>556</td>
<td>221</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>900</td>
<td>227</td>
<td>799</td>
<td>35</td>
</tr>
</tbody>
</table>

Job Requirement

<table>
<thead>
<tr>
<th>SR #</th>
<th>CPU</th>
<th>Memory</th>
<th>U_BW</th>
<th>V_BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>56</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Mapping Module

LP solver

Deployment Matrix

<table>
<thead>
<tr>
<th>PHY</th>
<th>SR 0</th>
<th>SR 1</th>
<th>SR 2</th>
<th>SR 4</th>
<th>SR 6</th>
<th>SR 7</th>
<th>SR 8</th>
<th>SR n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Constraints
1) CPU constraints
2) Memory constraints
3) Traffic constraints
4) Natural constraints
Resource Allocation – Dynamic Migration

- The static mapping cannot efficiently handle the dynamics during an experiment. An overloaded node may impact the experiment results.
- Dynamic migration is implemented as a sub-module in NodeAgent. It keeps monitoring the on its host.
- If NodeAgent detects a node is overloaded, some tasks will be moved onto other machines without restarting the experiments.
- Migration is not able to completely mask the effects from other users, but can alleviate the worst problems.
Overlay Subsystem

• Overlay Subsystem constructs an experiment overlay by using the resources from Agent Subsystem.

• One overlay instance corresponds to a job, therefore LiteLab can have multiple overlay instances running in parallel at the same time.

• The most critical component in Overlay Subsystem is SRouter, which is a software abstraction of a realistic router.
Innards of SRouter
Flexible Configurations

• Users can configure many parameters of SRouters, e.g., link properties (delay, loss rate, bandwidth), queue size, queueing policy (Droptail, RED), and so on.
  • Queues: iqueue, equeue, cqueue;
  • Processing Chain: similar concept as the chains of rules in `iptables`;
  • VID: a logical ID (VID) to identify a Srouter, neutral to any naming scheme;
  • Routing: is based on VID, can plug in different routing algorithms;
  • User Application: ihandler provides a passive way to interact with SRouters.
Evaluation - Accuracy

**TABLE I:** Accuracy of SRouter’s bandwidth control as a function of link bandwidth and packet size.

<table>
<thead>
<tr>
<th>Bandwidth (Kbps)</th>
<th>Packet Size</th>
<th>Observed Value bw (Kbps)</th>
<th>% err</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>64</td>
<td>55.77</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>57.62</td>
<td>2.89</td>
</tr>
<tr>
<td>384</td>
<td>64</td>
<td>382.56</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>387.96</td>
<td>1.03</td>
</tr>
<tr>
<td>1544</td>
<td>64</td>
<td>1539.23</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>1546.32</td>
<td>0.15</td>
</tr>
<tr>
<td>10000</td>
<td>1518</td>
<td>9988</td>
<td>0.12</td>
</tr>
<tr>
<td>45000</td>
<td>1518</td>
<td>44947</td>
<td>0.12</td>
</tr>
</tbody>
</table>

**TABLE II:** Accuracy of SRouter’s delay at maximum packet rate as a function of 1-way link delay and packet size.

<table>
<thead>
<tr>
<th>OW Delay (ms)</th>
<th>Packet Size</th>
<th>RTT</th>
<th>Observed Value stddev</th>
<th>% err</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>64</td>
<td>0.190</td>
<td>0.004</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>0.221</td>
<td>0.007</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>10.200</td>
<td>0.035</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>10.230</td>
<td>0.009</td>
<td>2.30</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>20.212</td>
<td>0.057</td>
<td>1.06</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>20.185</td>
<td>0.015</td>
<td>0.92</td>
</tr>
<tr>
<td>50</td>
<td>64</td>
<td>100.209</td>
<td>0.060</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>100.218</td>
<td>0.031</td>
<td>0.22</td>
</tr>
<tr>
<td>300</td>
<td>64</td>
<td>600.189</td>
<td>0.083</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>1518</td>
<td>600.273</td>
<td>0.034</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Evaluation - Scalability

Figure 4 shows that the time to construct network increases linearly as the number of nodes increases in random networks. However, the growth of time is slower in scale-free networks because the nodes with high degree dominate the construction time.

Fig. 4: Time to construct synthetic networks of different type.
Heuristic in Resource Allocation

**Algorithm 1**: Heuristic to improve mapping efficiency

**Input**: job requirements $R$, physical nodes $L$

**Output**: minimum physical node set $S$

Calculate overall job requirement $R$.

Order nodes from lightest to heaviest load into $L$.

**foreach** node $N$ in $L$ do

- Add $N$ to $S$.
- Calculate capacity of $S$: $C$.
- if $R < C$ then
  - Solve LP.
    - if optimal solution exists then break.
    - else $R \leftarrow 2 \times R$.
  - end
- end

**foreach** node $N$ in $L$ do

- Select minimum node set satisfying job requirement.

- **Deployement Matrix**

Fig. 5: Reduce deployment matrix size by selecting minimum physical node set that satisfies the job requirement.
Discussion

• LiteLab aims at a flexible, easy-to-deploy experiment platform and in this goal, it makes tradeoffs between accuracy and performance.

• LiteLab cannot completely eliminate external effects from other processes running on the test platform.

• SRouter’s processing power is another limitation as it can only process about 10000 packets per second. Adding more user-defined modules will further slow down SRouter.
Conclusion

• LiteLab which aims to reduce the deployment risk by providing a light-weight platform for both network scientists and practitioners to efficiently evaluate their novel system designs.

• LiteLab combines the benefits from both emulation and simulation: ease of use, high accuracy, no complicated hardware settings, easy to extend and interface with user applications, etc.

• LiteLab is open sourced and is available for download for others.
Thank you!

Questions?