GTNA 2.0
Graph-Theoretic Network Analyzer

A Framework for Rapid Prototyping and Evaluation of Routing Algorithms

Benjamin Schiller, P2P Networks, TU Darmstadt
SummerSim’13 / SCSC / 10.07.2013
Motivation

• Systems for network analysis and routing simulation
  – Graph-theoretic network analysis (topological characteristics)
  – Round-based simulators (tailored to specific problem)
  – Event-based simulators (high level of details)

• Good for dynamics and highly complex systems
  – Can become complex even for simple systems
  – Not feasible for quickly testing an idea
  – Do not provide means for analyzing topologies

• GTNA 2.0 closes this gap
  – Enables quick prototyping
  – Combines routing simulation and graph-theoretic analysis
Outline

• Introduction

• Methodology for routing evaluation

• Framework components

• Example

• Summary & Outlook
Routing in Distributed Systems

- Distributed systems
  - Wireless sensor networks, Peer-to-Peer systems
  - IP networks, Named Data Networking

- Nodes provide services
  - Lookup, storage, update of:
    - Web pages, video streams, measurements, profile information

- Main challenge
  - Find the provider of a requested service
  - No global knowledge, only local information
Decision Tree of a Local Routing Algorithm

Service provided?

Term. forced?

Yes

Delegation possible?

SUCCESS

No

FAILURE

No

FAILURE

Yes

DELEGATE
Recording a routing attempt

Input: source s, target t

Route r

Routing Algorithm

\[ \text{Route } r \]

\[\text{Input: source } s, \text{ target } t\]
Components of GTNA

- Plots
  - generates
- Plotter
  - input
- Data
  - comp.
- Network
  - generates
- Graph
  - input
- Metric
  - param.
- Setup
  - adds
- Graph Property
  - input
- Routing Algorithm
  - input
Workflow for Evaluating Routing Algorithms

RoutingAlgorithm RA
SourceSelection S
TargetSelection T
Routing Metric

s = S.nextSource()
t = T.nextTarget()

Route r = RA.route(s, t)

Route[] routes
Data d₁  Data d₂  Data d₃  ...

service provided?
No
Term forced?
Yes

SUCCESS
Yes
No
FAILURE
Neighbor found?
Yes
No
FORWARD
FAILURE

repeat
Example: A Proof of Concept

- Routing on the autonomous system (AS) level of the Internet
  - Routing tables: Border Gateway Protocol (BGP)
  - Routing decision: longest-prefix matching of IPs

- Routing in Names Data Networks (NDN)
  - Requests are forwarded based on resource name
  - Data items cached on the way back
  - Simple version: IP-based routing with caching

- Networks
  - AS topologies obtained from CAIDA (between 2007 and 2013)

- Setup
  - Add node IDs, create routing tables, add data stores

- Measuring during simulation
  - Route length distribution, replica distribution
Example: IP Routing Algorithm

**Input:** current route, target identifier, current node

```java
// add current node to route
route.add(current);

// terminate if service can be provided
if (isEndPoint(current, target))
    return new Route(route, true);

// terminate with a failure in case ttl exceeded
if (route.size() > ttl)
    return new Route(route, false);

// failure if no next hop can be determined
int nextHop = getRoutingTable(current).getNextHop(target);
if (nextHop == current || nextHop == RoutingTables.noRoute)
    return new Route(route, false);

// forward request to next hop according to routing table
return this.routeToTarget(route, nextHop, target, nodes);
```
Example: NDN Routing Algorithm

**Input**: current route, target identifier, current node

```java
// perform IP-based routing
Route route = super.routeToTarget(start, target);

// replicate data on the whole route if successful
if (route.isSuccessful())
    for (int node : route.getRoute())
        getStorageForNode(node).addReplica(target);
return route;
```
Example: Simulation Execution

// setup
Transformation t1, t2, t3;
t1 = new NodeIds();
t2 = new NodeIdsRoutingTable();
t3 = new NodeIdsDataStorage(new LruDataStore(0, 30), 1);
Transformation[] t = new Transformation[] { t1, t2, t3 };

// creating network instances
Network[] nw = new Network[filenames.length];
for (int i = 0; i < filenames.length; i++)
    new ReadableFile("AS", "caida", filenames[i], t);

// creating metric instances
SourceSelection ss = new ConsecutiveSourceSelection();
TargetSelection ts = new DataStorageRandomTargetSelection();
Metric r1 = new Routing(new IpRouting(), ss, ts);
Metric r2 = new Routing(new NdnRouting(), ss, ts);
Metric ds1 = new DataStorageMetric();
Metric ds2 = new DataStorageMetric();
Metric[] metrics = new Metric[] { r1, ds1, r2, ds2 };

// generating networks, simulate routing, compute metrics
Series[] s = Series.generate(nw, metrics, 100);

// plot the results
Plotting.multi(s, metrics, "multi/");
Plotting.single(s, metrics, "single/");
Example: Results
**Evaluation: Runtime & Memory**

<table>
<thead>
<tr>
<th>Snapshot</th>
<th>Nodes</th>
<th>Edges</th>
<th>IP</th>
<th>NDN</th>
<th>Routing Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-09-13</td>
<td>12,190</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008-01-02</td>
<td>14,038</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008-07-02</td>
<td>14,128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009-01-03</td>
<td>15,205</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009-07-02</td>
<td>15,427</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010-01-01</td>
<td>16,695</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010-07-01</td>
<td>17,143</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011-01-02</td>
<td>18,427</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011-07-01</td>
<td>19,281</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-01-02</td>
<td>20,098</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-07-03</td>
<td>20,393</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013-01-02</td>
<td>21,525</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary, Conclusion, & Outlook

• GTNA 2.0
  
  – A framework for rapid prototyping and evaluation of routing algorithms
  
  – Closes the gap between network simulators and graph analysis tools
  
  – Less complex than full-fledged network simulators
  
  – Small memory footprint, quick evaluation of new ideas
  
  – Open source (GPL) at https://github.com/BenjaminSchiller/GTNA

• Future work
  
  – Enhancements of multi-threading capabilities
  
  – Support for modeling parallel routing requests
GTNA 2.0
Graph-Theoretic Network Analyzer
Routing Algorithms

- Greedy, GreedyBacktracking, 10 variations
- Lookahead, 3 variations
- MultiPhase
- RandomWalk
- RoutingTables, NDN
- HighestDegreeNeighbor
Routing Parameters

• Source selection
  – Consecutive
  – Random

• Target selection
  – RandomDataStorageElement
  – RandomIdentifier
  – RandomRepresentativeIdentifier
Identifier Spaces

- Ring

- Plane

- MultiDimensional

- DataStores (FIFO, LRU, Unlimited)

- LookaheadElements

- NodeData/Identifier
Metrics

- RoutingLengthDistribution, DataStorage
- ClusteringCoefficient, DegreeDistribution, ShortestPaths
- CommunityDetection (8), Roles (2)
- RichClubConnectivity, StrongConnectivity, WeakConnectivity
- EdgeCrossings
- StrongFragmentation, WeakFragmentation
- IdentifierSpaceDistances (8)
- 3-Motifs, 4-Motifs
- PlacementCoverage
- AdjacencyMatrix