Graph-Based Analysis of IP Data

Emden R. Gansner
Balachander Krishnamurthy
Wealth of IP data

Sources

- Proxy logs
- Server logs
- NetFlow
- Packet traces
- Packet monitoring
- Active probes
Data mining opportunities

- Can we identify user access patterns?
- Can we characterize sites based on usage?
- Are there signs of impending problems?
- Does the ability to answer these suggest new services?
Too much data

- Inhibits interactive exploration
- Complicates real-time analysis
- Still only dealing with a portion of data
- Typical solutions limit:
  - Portion of network
  - Span of time
  - Accuracy
A different approach

- Use domain and problem knowledge to identify relevant structure and attributes
- Extract from the raw data a graph that models the desired information
- Use graph as basis for analysis
  - Statistical and graph-based analysis
  - Active probing
  - Domain knowledge
  - External data
- Refine the graph model
Why graphs?

- Equivalent abstractions, such as relational databases
- Expertise in manipulating graphs
  - Speed and in-core algorithms
  - Simple to generate code optimized for query
  - Available tools
  - Potential external memory algorithms
- Some IP data artifacts map naturally to graphs, e.g. weighted degree
  - Makes implicit structure in flat data explicit
- Alternate, more “graphical” view of data
  - Graph characteristics, especially anomalies, can suggest new directions to explore
Graph library

- Constructed as a boiler-plate graph library
  - Common algorithms and I/O
  - Common utility data structures
  - Tailor data, then mix-and-match
  - In C, using Vo’s libcdt

- Serves as a platform for our approach

- Most programs are short

- Complexity $\leq O(n \log n)$, of necessity
Example: Characterizing DNS traffic

Analysis by Cranor, Gansner, Krishnamurthy and Spatscheck

Identify sites as clients, local DNS and authoritative DNS with high probability

Why?

- Reduce number of mapped IP addresses
- Chose best local DNS server dynamically
- Identify anomalous hosts
Deriving the model

Nodes correspond to unique IP addresses
Edges correspond to a request or response between nodes, with one node (head) tied to port 53
### Reduction

<table>
<thead>
<tr>
<th></th>
<th>Edge count</th>
<th>KBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data</td>
<td>N/A</td>
<td>&gt; 1 Terabyte</td>
</tr>
<tr>
<td>Daily UDP data</td>
<td>257,339,100</td>
<td>7,832,896</td>
</tr>
<tr>
<td>Daily TCP data</td>
<td>22,435,114</td>
<td>691,057</td>
</tr>
<tr>
<td>Graph model</td>
<td>76,224,714</td>
<td>914,694</td>
</tr>
</tbody>
</table>

- Typical analysis query takes several minutes, as opposed to days/forever on the raw data.
Example: Classifying P2P data

Analysis by Krishnamurthy and Wang

Sort NetFlow-generated data into various categories: belonging to ATT, ATT-customers, ATT-customers-customers, ATT-peers, ATT-peer’s-customers, etc.

- Use graph library to derive connected components; zoom in on the heaviest component in terms of bytes exchanged between its members
- Use weighted indegree and outdegree information to separate nodes in heaviest component into data and control sections
- Examine just the top few addresses in the data part to identify potential P2P servers and the set of busy clients from control section
- Classify heavy entities (IP addresses, prefixes, AS numbers) extracted from heaviest component
Future work

- See if it would help in anomaly detection (work in progress: Krishnamurthy, Sen and Zhang)
- Apply this technique to other protocols and questions, e.g., multimedia data
- Extend operations in graph library
- Construct a tool for generating optimized code
- Squeeze out additional performance, and prepare for more data
- Determine if dynamics can be captured, along with reasons for changes (work in progress: Gansner, Krishnamurthy, Wang and Willinger)
- Integrate with other tools and approaches