AjaxTracker: Active Measurement System for High-Fidelity Characterization of AJAX Applications

Myungjin Lee†, Ramana Rao Kompella†, Sumeet Singh‡
†Purdue University, ‡Cisco Systems
Wind of changes

Asynchronous Javascript and XML (AJAX)
AJAX vs. classical web applications
AJAX vs. classical web applications

AJAX allows applications to interact with servers **asynchronously and/or autonomously**.

Diagram showing the interaction between the Web Browser, User Interface, Web Server(s), and Time.
Why characterize AJAX applications?

- Limited understanding about the impact of AJAX apps on the network
- Comprehensive study on AJAX apps is critical
  - Enterprises
    - Employee productivity may be affected by these apps
    - Need to monitor the performance of these apps continuously
  - Network operators
    - Need to project how application popularity changes may affect network traffic growth
    - Need to monitor for new threats and security vulnerabilities
Key goals of our work

- **Goal 1**: Characterization of full application sessions
  - # of flows/servers
  - Request/response distributions
  - Inter-request time distributions
  - Predict application characteristics under different network conditions (e.g., low bandwidth, high RTT)

- **Goal 2**: Characterizing per-operation network activity
  - Mail applications consist of click inbox, read mail, attach file, etc.
  - Maps applications consist of dragging, zoom in/out, etc.
Characterization approach

- Classic approach: Traces in the middle of the network
  - E.g. Schneider et al. in [PAM08]
  - **Limitation 1**: Cannot easily differentiate traffic that belongs to a given application
  - **Limitation 2**: Cannot isolate network activity for individual operations
  - **Limitation 3**: Cannot study application under different network characteristics

- Our approach: End-host based characterization
  - Run the application on an end host in isolation
  - Produces interference-free access to ground truth
  - Characterize individual operations easily
AjaxTracker

- **Key idea:** Mimic human interactions with AJAX applications and collect network trace at end-host
  - Inject events such as scrolling, drag-and-drop to a Web browser externally
  - Model to simulate human think-time between operations
  - Relies on network sniffers (tcpdump) at the end host to collect packet traces
  - Characterization support for individual operations with the help of per-operation logs with timestamps
  - Control network conditions with delay shaper and bandwidth throttler
Components of AjaxTracker

<Procedure>
1. Search Form
2. Search Button
3. Menu Buttons
4. Map Rendering Area

XML Scenario File

Event Generator

Event Log

Trace

Packet Sniffer

Traffic Shaper

Analyzer

Web Servers
Scenario file

- Guide how event generator injects events to a browser
  - Emulate user session
  - Written as XML

- Two navigation modes
  - Static mode
    - Follow exact sequence of events in the order specified in the scenario file
  - Random mode
    - Shuffle the order of events
    - Randomize the location (screen coordinates) where events occur
    - Randomize human-think time with various models (Weibull & Pareto)

- **Scenario file has high flexibility to describe events**
Example of a scenario file (1/2)

<SCENARIO>
  <NAME> Google Maps </NAME>
  <PRE_EVENTS>
    ...
  </PRE_EVENTS>
  <MAIN_EVENTS TYPE="static" EXEC_COUNT="2">
    <EVT_REF IDREF="navigate_map" />
    <EVT_REF IDREF="search_city" />
  </MAIN_EVENTS>
  <POST_EVENTS>
    ...
  </POST_EVENTS>
  <EVENT ID="navigate_map">
    ...
  </EVENT>
</SCENARIO>
Example of a scenario file (2/2)

String is used for logging event
Simulate human think time
Describe object and possible actions
Describe operation
Other components

- Off-the-shelf browser
  - E.g., Firefox

- Event generator
  - Parse XML-based scenario file
  - Implemented using C++, GTK+, X library and Xerces-C++ parser

- Traffic shaper
  - Run Click modular router as a kernel module

- Packet sniffer
  - Collect packet traces
  - E.g. tcpdump
Evaluation

- How representative are traces generated by AjaxTracker?
  - Comparing our results with a passive campus trace
- Show the characterization of individual operations
  - Based on the help of the causality analysis component of our tool
  - E.g., ‘click’ and ‘drag-drop’ in two Ajax applications - Google Maps and Mail
- Perform macroscopic characterization of full application sessions
  - With a focus on results about Ajax application traffic characteristics under different network conditions
Comparison with a real trace (1/2)

- A real trace of Google Maps users
  - Collected from a campus switch of Purdue University
  - 24 hours worth of client activity

- Calibration for comparing Inter-Request Time (IRT)
  - Run AjaxTracker with different bandwidth conditions
    - 500Kbps, 1Mbps, 5Mbps, 10Mbps
  - Use linear combinations of different bandwidth traces
AjaxTracker can be calibrated easily to reflect real life pattern
Characterizing individual operations (1/2)

- **Selected operations**
  - Google Maps: drag map, zoom in, zoom out, click search button

- **Inferring causality between operation and network traffic**
  - Use long inter-operation time to avoid interference between operations (e.g., 60 sec.)
  - Consider all the traffic after an operation is initiated and before a new operation is invoked
Characterizing individual operations (2/2)

PDF

Number of connections

- click search button
- drag map
- zoom in
- zoom out
Impact of different network conditions

- Constraint I: Bandwidth cannot be set larger than that of a bottleneck link along the end-to-end path
- Constraint II: Delay is added to the round trip time of the end-to-end path
Characterization of full sessions (2/2)

![Graph showing bandwidth (Kbps) vs. average IRT (second) for Google Mail and Google Maps. The graph indicates a decrease in average IRT as bandwidth increases.]
Summary

- An active measurement system to automatically interact with AJAX-powered Web applications
  - Available at [http://www.cs.purdue.edu/synlab/ajaxtracker](http://www.cs.purdue.edu/synlab/ajaxtracker)

- Allows users to characterize AJAX applications in the different level of details
  - Session-level and Operation-level

- Enables the characterizations of AJAX applications under different network conditions
Questions?