Intermediaries for the World-Wide Web: Overview and Classification

Marios Dikaiakos

Department of Computer Science
University of Cyprus
mdd@ucy.ac.cy
Outline

- Definition and Motivation

- Intermediaries as an extension of the WWW-access model.

- Populating the network with Intermediaries.

- A Classification Framework and Taxonomy.

- Summary and Conclusions.
Definition

- “Software entities that intervene to the flow of information from clients to origin servers at the application level of the WWW.”

- From simple relaying and caching to complicated transformations, such as filtering, indexing, and transcoding.

- Deployed on Internet hosts of the wireline and wireless Web between origin servers and client systems.

- Provide a reusable and expandable set of services and functions needed by many applications to function well in a networked environment: middleware components.
Motivation

• Intermediaries are important:
  – A useful abstraction for designing and studying emerging software infrastructures for the WWW.
  – Will permeate the Internet because of the increasing demand for personalization, localization, and ubiquity.

• Our goal:
  – Overview a wide range of intermediary systems and identify common characteristics and functional properties.
  – Examine the requirements and identify key components of intermediary systems.
  – Define a framework for comparing and designing intermediaries.
Extending the WWW-access model
Intermediaries at the origin server

• Enhancing dynamic-content provision by reducing Web-server load and improving QoS.

• HTTP accelerators: distribute requests, cache replies.

• Composition of dynamic content: IBM’s ABR framework, INRIA’s Weave. To this end, we need:
  – High-level abstractions for Web-site structure: object-models, graphs, or declarative languages.
  – Abstractions for specifying content composition: object dependence graphs with embedded trigger monitors; declarative specification of SQL queries and runtime policies.
  – Caching content or fragments of dynamic content.
Intermediaries on the net

• Web proxies and Content-Distribution Networks.

• Notification Systems (aka “Publish-Subscribe”):
  – SIFT (Stanford University).
  – AIDE by AT&T Research.
  – Grand Central Station by IBM.
  – FIGI (U. Cyprus).

• Issues and Features:
  – Description of profiles.
  – Profile execution.
  – Caching and versioning.
  – Server location and scalability.
Intermediaries for Mobility and Ubiquity: requirements

- Optimize C/S communication over the wireless medium.
- Support seamless access from a variety of devices.
- Customize content to different devices.
- Enable the provision of multiple formats to the same device over the same link.
- Support both synchronous and asynchronous interaction modes.
- Optimize the amount of useful content reaching the user (filtering).
Intermediaries for Mobility and Ubiquity: approaches

- Characteristic solutions:
  - IBM’s WebExpress: C-I-S model.
  - WAP Gateways.
  - Blazer for PalmOS Handspring devices.
  - Web clipping system of Palm Inc.

- Features and Issues:
  - Single-proxy vs. end-to-end.
  - Open vs. proprietary design.
  - Centralized vs. distributed architecture.
  - Modularity and extensibility.
Intermediary Infrastructures

- Intermediary solutions need to support, additionally:
  - Large numbers of simultaneous end-users.
  - A large heterogeneity of end-user devices.
  - High throughput of requests for service.
  - Highly bursty workloads.
  - High-availability, robustness, and incremental scalability.
  - Definition and deployment of new services.

- Requirements:
  - Shift of computation, storage and complexity from clients, origin servers, mobile base-stations, or mobile hosts, into the network.
  - Distributed, cooperating, network-centered modules.
  - Support for programmability and/or reconfigurability.
Towards Intermediary Infrastructures

- Web Browser Intelligence or WeB Intermediaries (WBI), IBM Almaden.
  - Information Streams.
  - Requests editors, generators, document editors, monitors, autonomous functions

- iMobile by AT&T Research.
  - iProxy and let engine
  - devlets, infolets, applets

- Ninja
  - bases, active proxies, units, paths
  - design patterns: wrap, pipeline, combine, replicate
eRACE
Classifying Intermediaries

- Software architecture:
  - Structure: centralized vs. distributed.
  - Location of components: network, client side, origin-server side.
  - Caching and Archiving support.
  - Programmability and Configurability.

- Interaction with clients and origin servers:
  - Proxy-Server Protocols: HTTP, NNTP, SMTP, WAP, etc.
  - Client-Proxy Protocols: HTTP, UDP, GSM/SMS, WAP, etc.
  - Supported Media: wireline, wireless.
  - Access Model: push vs. pull.
  - Communication Mode: synchronous vs. asynchronous.
Classifying Intermediaries (ctd.)

- Functionality:
  - Customization.
  - Filtering.
  - Annotation.
  - Transcoding.
  - Aggregation.
<table>
<thead>
<tr>
<th></th>
<th>Palm Clippings</th>
<th>SIFT</th>
<th>AIDE</th>
<th>WBI</th>
<th>TACC</th>
<th>eRACE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Software Architecture</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structure</td>
<td>central.</td>
<td>central.</td>
<td>central</td>
<td>distr.</td>
<td>distr.</td>
<td>distr.</td>
</tr>
<tr>
<td>Component Location</td>
<td>network &amp; client</td>
<td>network</td>
<td>network client, server</td>
<td>network server</td>
<td>network</td>
<td>network</td>
</tr>
<tr>
<td>Caching</td>
<td>limited</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Crawling support</td>
<td>−</td>
<td></td>
<td>✓</td>
<td>−</td>
<td>−</td>
<td>✓</td>
</tr>
<tr>
<td>Archiving</td>
<td>−</td>
<td></td>
<td></td>
<td>✓</td>
<td>−</td>
<td>✓</td>
</tr>
<tr>
<td>Programmability</td>
<td>−</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Configurability</td>
<td>✓</td>
<td>−</td>
<td>−</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Client-Intermediary-Server Interaction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proxy-Server Protocol</td>
<td>HTTP</td>
<td>NNTP</td>
<td>HTTP</td>
<td>HTTP</td>
<td>HTTP</td>
<td>HTTP, NNTP, SMTP</td>
</tr>
<tr>
<td>Client-Proxy Protocol</td>
<td>UDP and compressed msgs.</td>
<td>SMTP</td>
<td>HTTP</td>
<td>SMTP</td>
<td>HTTP</td>
<td>wireless, HTTP, SMTP, GMS/SMS</td>
</tr>
<tr>
<td>Medium</td>
<td>wireless</td>
<td>wireline</td>
<td>wireline</td>
<td>wireline</td>
<td>wireline</td>
<td>wireline</td>
</tr>
<tr>
<td>Access Model</td>
<td>pull</td>
<td>pull/push</td>
<td>pull/push</td>
<td>pull/push</td>
<td>pull</td>
<td>pull/push</td>
</tr>
<tr>
<td>Communication Mode</td>
<td>synch.</td>
<td>async.</td>
<td>async.</td>
<td>synch.</td>
<td>async.</td>
<td>async.</td>
</tr>
<tr>
<td><strong>Intermediary Functionalities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customization</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Filtering</td>
<td>−</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>−</td>
<td>✓</td>
</tr>
<tr>
<td>Annotation</td>
<td>−</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transcoding</td>
<td>✓</td>
<td>−</td>
<td>−</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aggregation</td>
<td>−</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusions

• Intermediaries represent a useful abstraction for designing, developing, analyzing and comparing emerging software infrastructures for the wireline and wireless Web.

• Classification and comparison of different intermediary systems can be perform along three main dimensions, which capture the basic properties of an intermediary.

• Design of future systems has to be established upon distributed software modules, with an explicit information architecture, communicating via pass-by-value semantics, with modular design enabling them to support new communication protocols and to achieve incremental scalability.

• Open issues: Programming and/or configuration of intermediaries, interoperability of different platforms, reusability of components.