Principled Design of the Modern Web Architecture

By Roy Fielding and Richard Taylor

Presentation by K.V.Crittenden

Essence of the article

- REST is built/based on the modern web architecture which supports a large scale distributed hypermedia system

- Overview of REST's architectural principles, elements, and views

Outline

- Introduction
- REST
  - Principles
  - Architectural elements
  - Architectural views
- REST vs. Web Architecture
- Conclusion

Introduction

- Definition of architectural style: ".. coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements …"

- REST is inspired by WWW architecture/domain characteristics:
  - Interface generality and simplicity (hypermedia)
  - Usability (Reduced interaction latency)
  - Scalability (handling flash crowds)
  - Flexibility (independent components, low entry barrier)

- "How do we introduce a new set of functionality to an architecture that is already widely deployed, and how do we ensure that its introduction does not adversely impact, or even destroy, the architectural properties that have enabled the Web to succeed?"

- …. the REST architectural style is born…

REST
Representational State Transfer

- The web is an instance of REST

- Principles:

  - Minimize latency and network communication

  - Maximize the independence of components

  - Maximize the scalability of components

Architectural Elements

- Data Elements
- Connectors
- Components

Data Elements

Data Elements	Examples
resource	The intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	Media type, last modified time
resource metadata	Source link, alternates, varies
control data	If-modified-since, cache-control

Data Elements cont'd...

– Handle the main design issues of moving information

- Separating concerns of client and server
- Information hiding (generic interface)
- Provides for a diverse set of functionality through downloadable feature-engines.

Connector Elements

Connectors	Examples
client	Libwww, libwww-perl
server	Apache API, libwww
cache	Browser cache, Akamai cache network
resolver	Binding (DNS lookup library)
tunnel	SOCKS, SSL after HTTP CONNECT

Connector Elements cont'd…

– REST connectors support stateless interactions

– Four reasons why this is good:

1. No need for connectors to retain application state between requests
2. Interactions can be processed in parallel
3. Intermediaries can understand requests in isolation
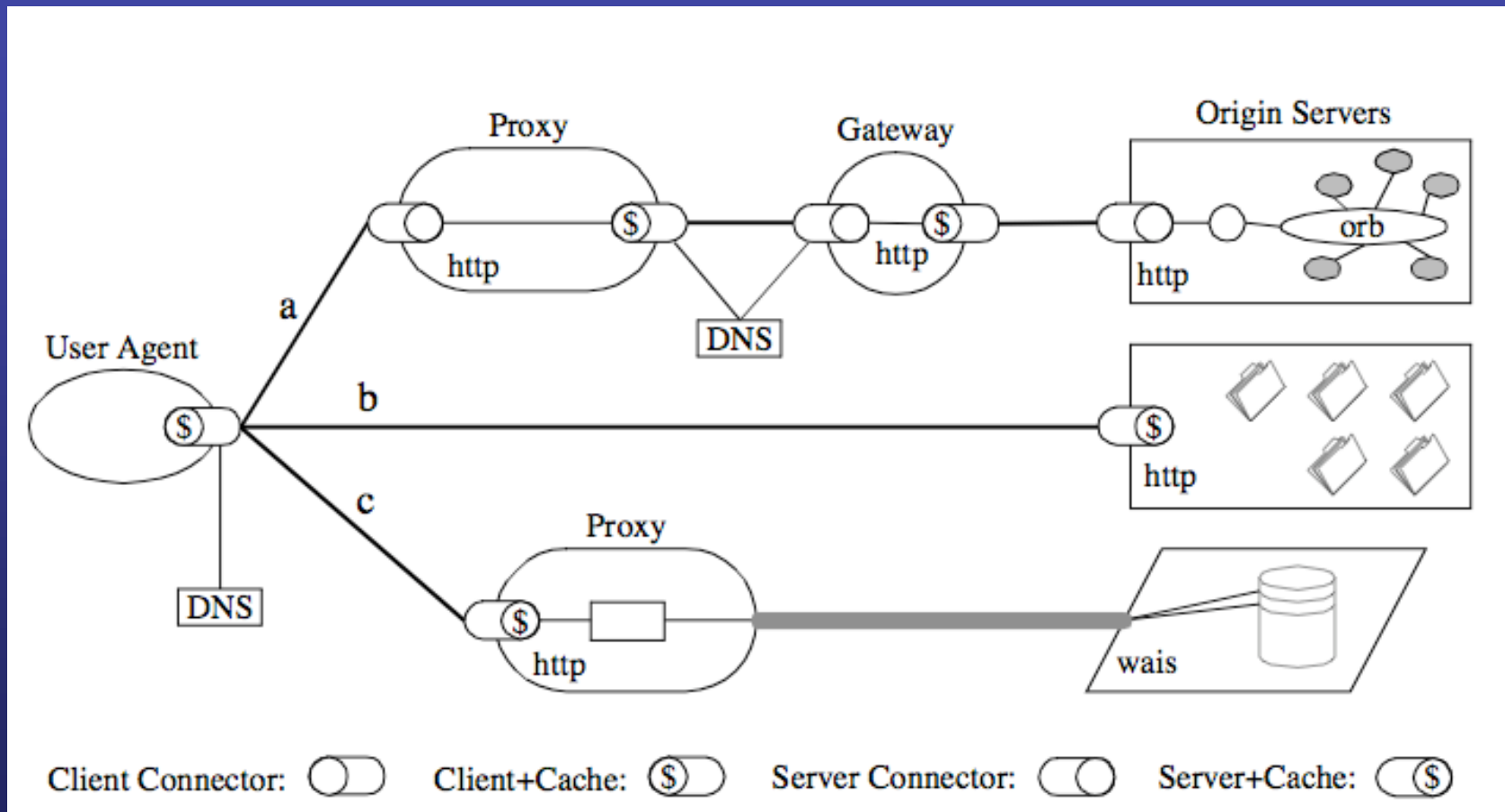4. Information that factors into cache behavior is present

Component Elements

Component	Example
origin server	Apache httpd, Microsoft IIS
user agent	A web browser
proxy	Netscape proxy
gateway	CGI, reverse proxy

Architectural Views
(how the elements work together to form an architecture)

- – Process view
- – Connector view
- – Data view

Process View

Proxy   Gateway   Origin Servers

http   http   orb   http

DNS

a

User Agent

b

http

c

DNS

Proxy

http   wais

Client Connector: ◯   Client+Cache: ⑤   Server Connector: ◯   Server+Cache: ⑤

Connector View

- – "concentrates on the mechanics of the communication between components"
- – For REST, their focus is on the constraints that define the generic resource interface

Data View

- – "reveals the application state as information flows through the components"
- – Application state defined by
  - Pending requests
  - Topology of connected components
  - Active requests
  - Data flow of representations
  - User agent processing of representations
- – Steady state
  - No outstanding requests, or requests received to the point of being treated as a representation data stream
  - User-perceived performance is effected by the latency between steady states

REST vs. Web Architectural Style
(How the web architecture diverges from that of REST)

- – Cookies can lead to confusion when moving between representation states
- – HTTP protocol does not distinguish between representation metadata and message control information
- – HTML frames -- one application can be wedged within the subcontext of another application

Conclusion

– REST is derived from the key architectural principles of the world's largest distributed application.

– Developed to address optimum network communication and component interactions

– Main attributes:

  • Intermediate processing (due to generic connector interfaces)

  • Standardized interfaces

  • Caching and reuse of interactions

  • Dynamic substitutability of components

  • Stateless interactions

  • Component communication via resource representation transfer

Thanks, Roy