Roy Fielding's PHD Dissertation

Chapter's 5 & 6 (REST)

Architectural Styles and the Design of Network-based Software Architectures

Roy Fielding

University of California - Irvine

2000

Chapter 5

Representational State Transfer (REST)

Deriving REST

- Walkthrough of the process of deriving rest
- Two Perspectives on Architectural Design
  - Blank Slate
  - Whole System Needs
    - Emphasizes Restraint and System Context
    - REST

Starting with the Null State

- Null State is the system without constraints
- The WWW is the Null state for REST
- No distinguishing boundaries between components (architecturally)

Figure 5-1. Null Style

Client - Server Constraints

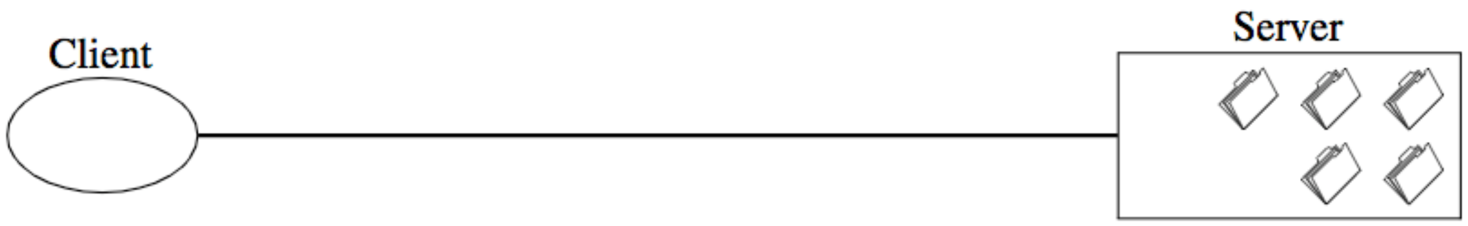


Figure 5-2. Client-Server

- Separation of Concerns
  - User Interface vs. Data Storage
  - Improves portability and scalability
  - Allows components to evolve independantly

Statelessness

- Communication must be stateless
- Session state kept entirely on client
- Improves:
  - Visibility
  - Reliability
  - Scalability
- Design Trade-offs
  - Possible decrease in network performance
  - Reduces server control over application behavior
    - Depends on correct implementation of semantics across multiple clients

Client - Stateless - Server

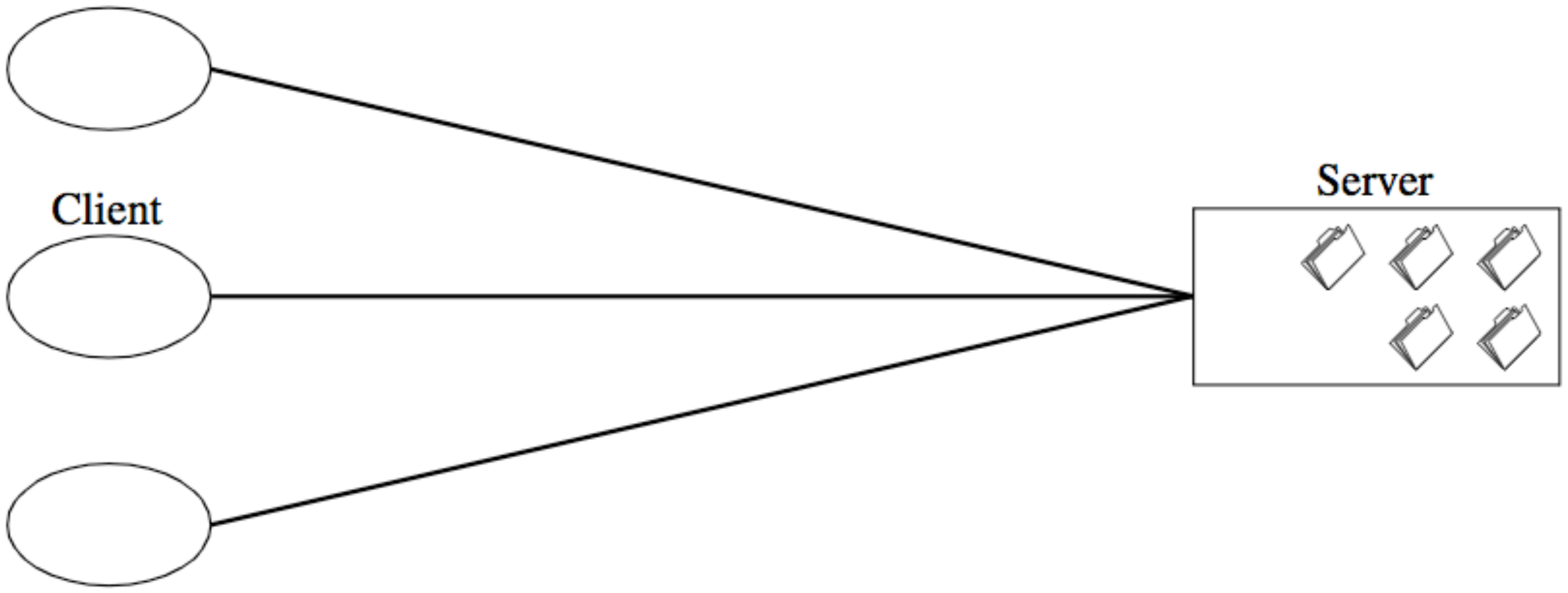


Figure 5-3. Client-Stateless-Server

Caches

- Requires data responses to be labled cacheable or not
- Improves
  - Network efficiency
  - Reduces average latency
- Design Trade-offs
  - Can reduce reliability
    - Stale data
    - Major changes in the server not updated in the cache

Client-Cache-Stateless-Server

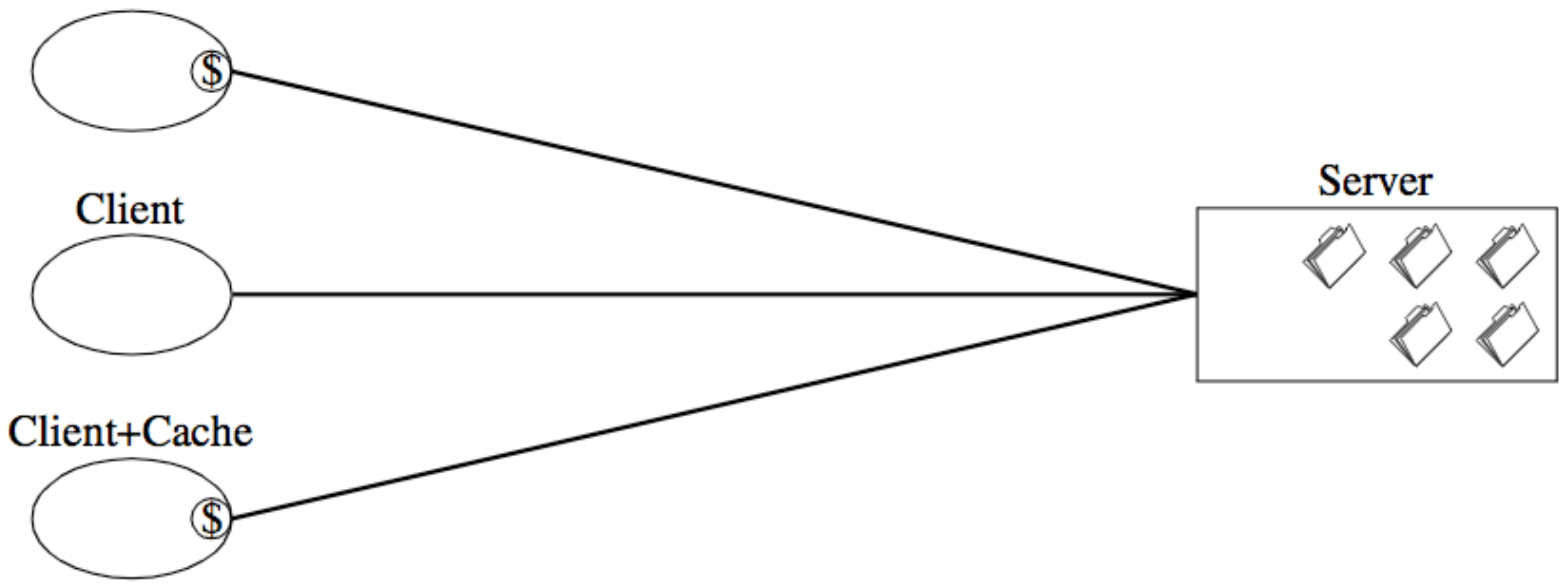


Figure 5-4. Client-Cache-Stateless-Server

State of the Early Web

- Web pre-1994
- Developers quickly exceeded early design
  - Dynamically generated responses
  - Server-side scripts

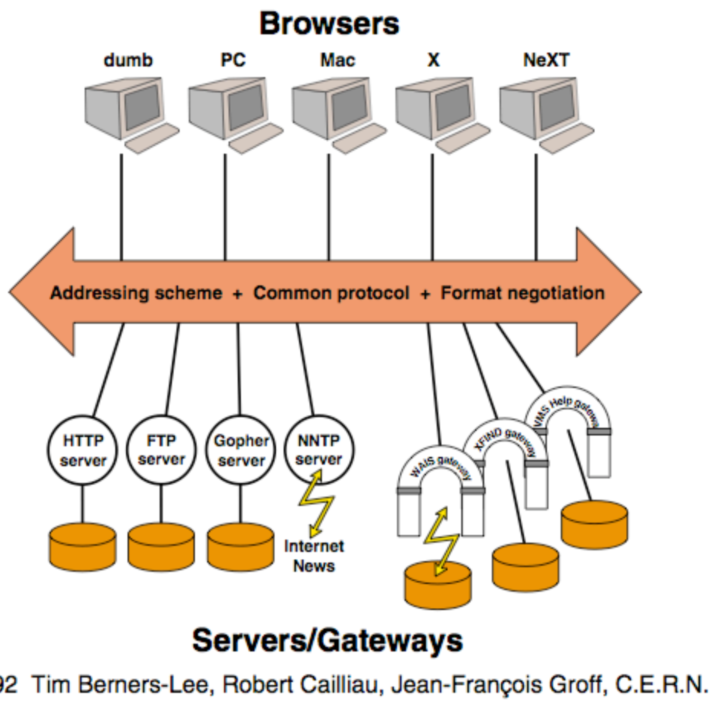


Figure 5-5. Early WWW Architecture Diagram

Uniform Interface

- Distinguishes REST from other network based styles
- Implementations decoupled from services
- Additional Constraints
  - Identification of resources
  - Manipulation of resources through representations
  - Self-descriptive messages
  - Hypermedia as the engine of application state

- Design trade-offs
  - Degrades efficiency
    - Information is not in a form specific to the application
  - Designed to work well for the Web (large-grain) hypermedia data transfer
    - May not be optimal for other situations

Uniform-Client-Cache-Stateless-Server

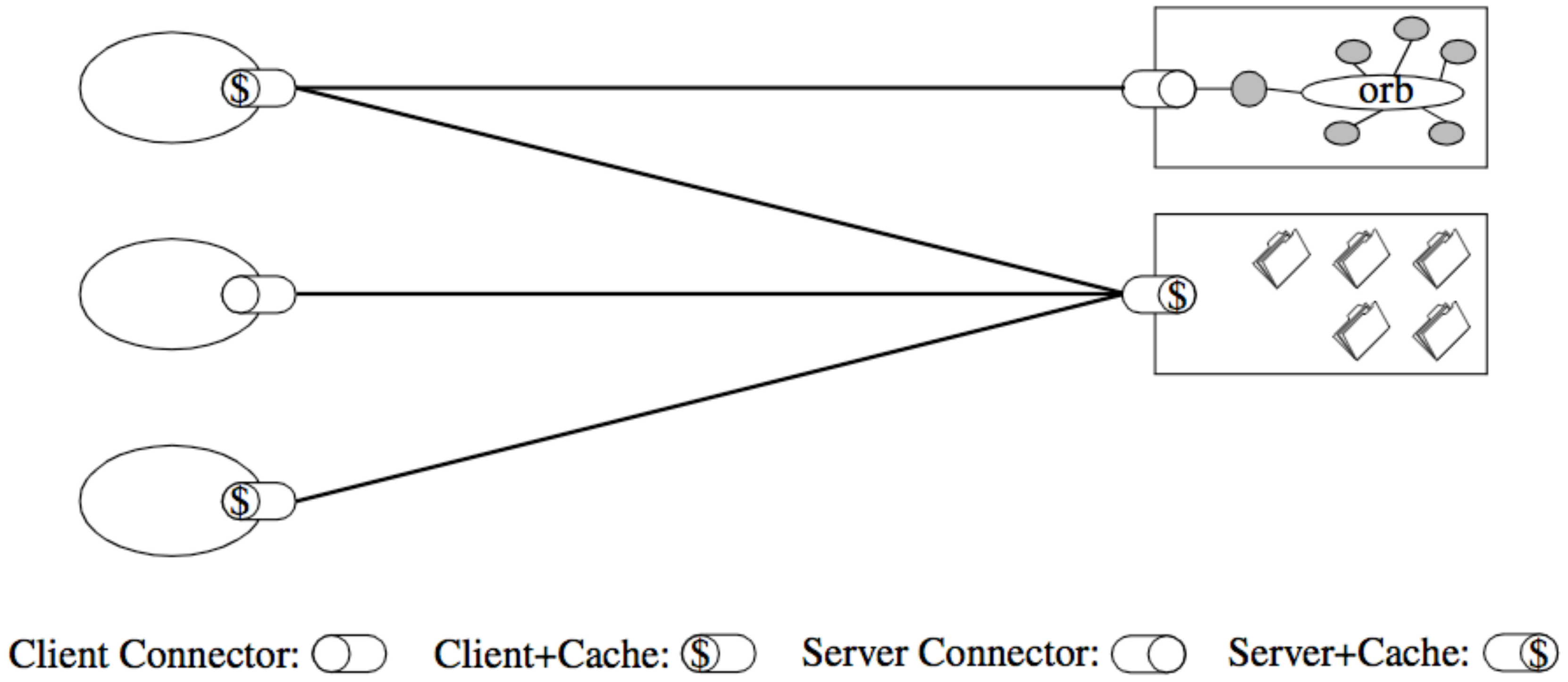


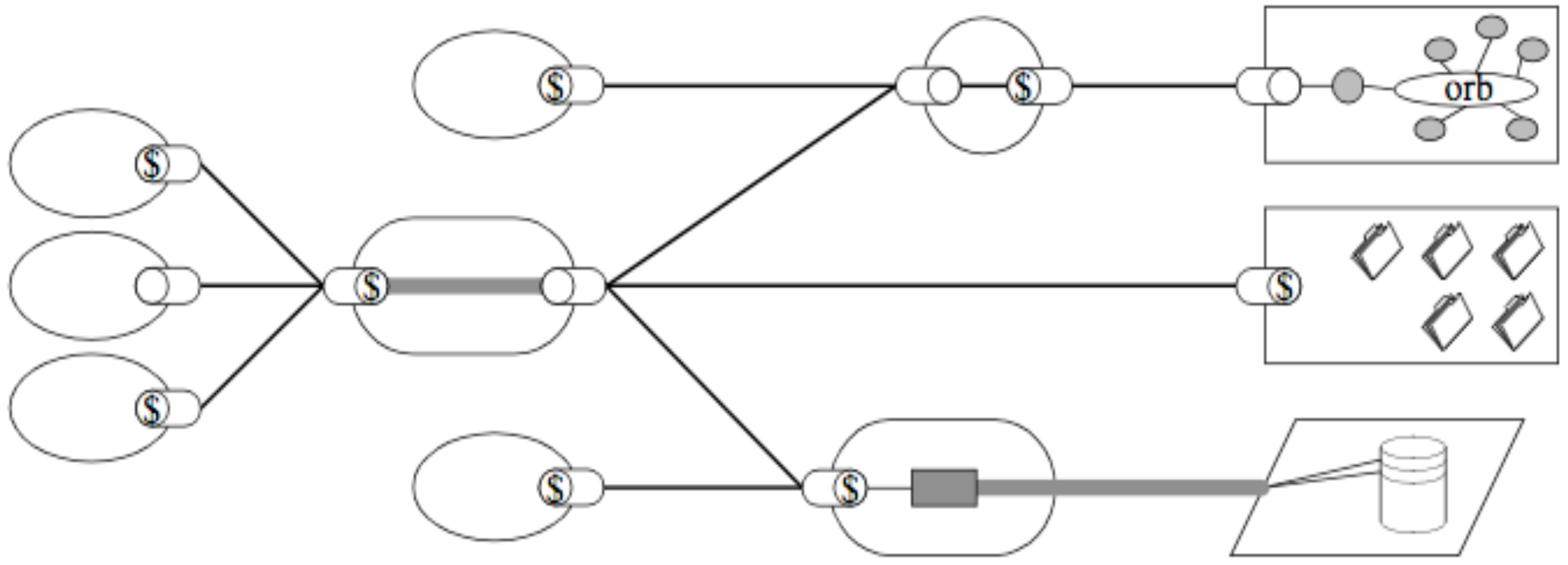
Figure 5-6. Uniform-Client-Cache-Stateless-Server

Layered System

- Adds hierarchical layers
  - Creates a bound on overall system complexity
  - Promotes substrate independence
- Provides encapsulation
- Improves Scalability
  - Load balancing

- Design Trade-offs
  - Adds overhead and latency to the processing of data increasing user perceived latency
    - This can be mitigated with shared caches on organizational boundaries

Uniform-Layered-Client-Cache-Stateless-Server



Client Connector: ⊂○⊃ Client+Cache: ⊂$⊃ Server Connector: ⊂○⊃ Server+Cache: ⊂$⊃

Figure 5-7. Uniform-Layered-Client-Cache-Stateless-Server

Code-On-Demand

- Optional Constraint
- Allows extension of client functionality
  - Reduces the number of pre-implemented features
  - Improves system extensibility
- Trade-off
  - Reduces visibility

REST

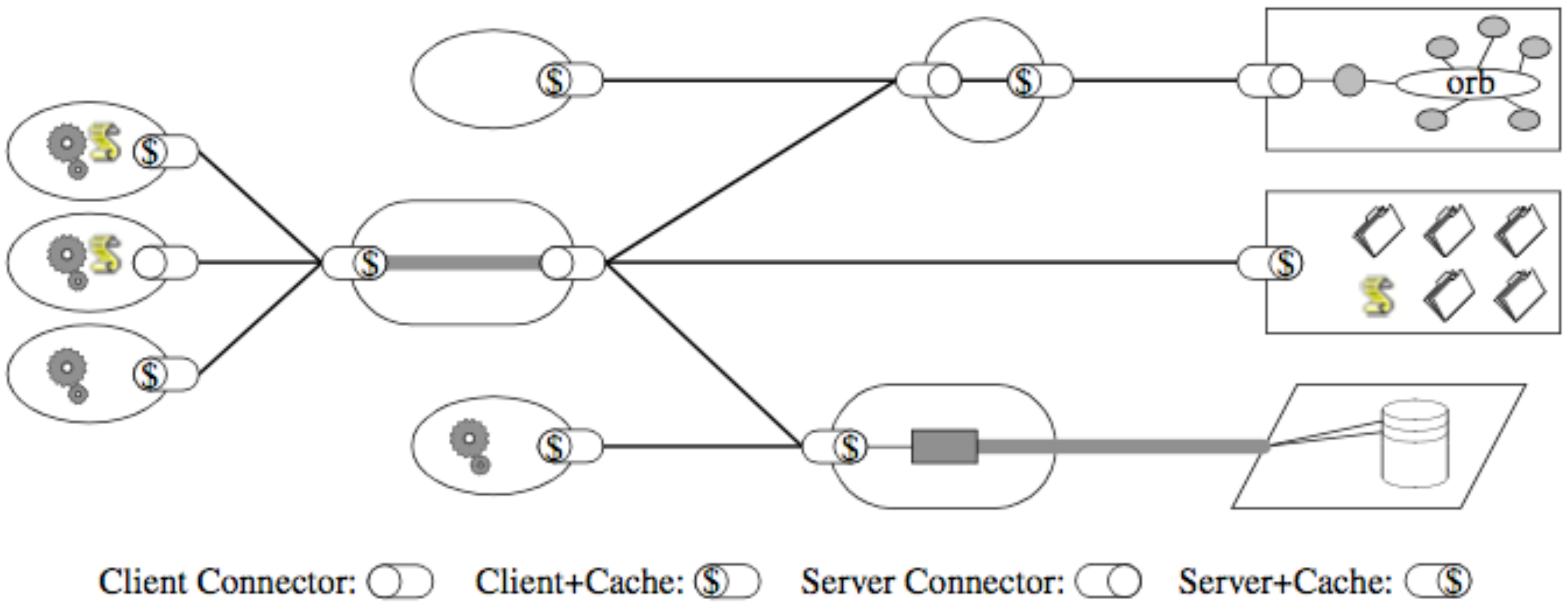


Figure 5-8. REST

REST Derivation by Constraints

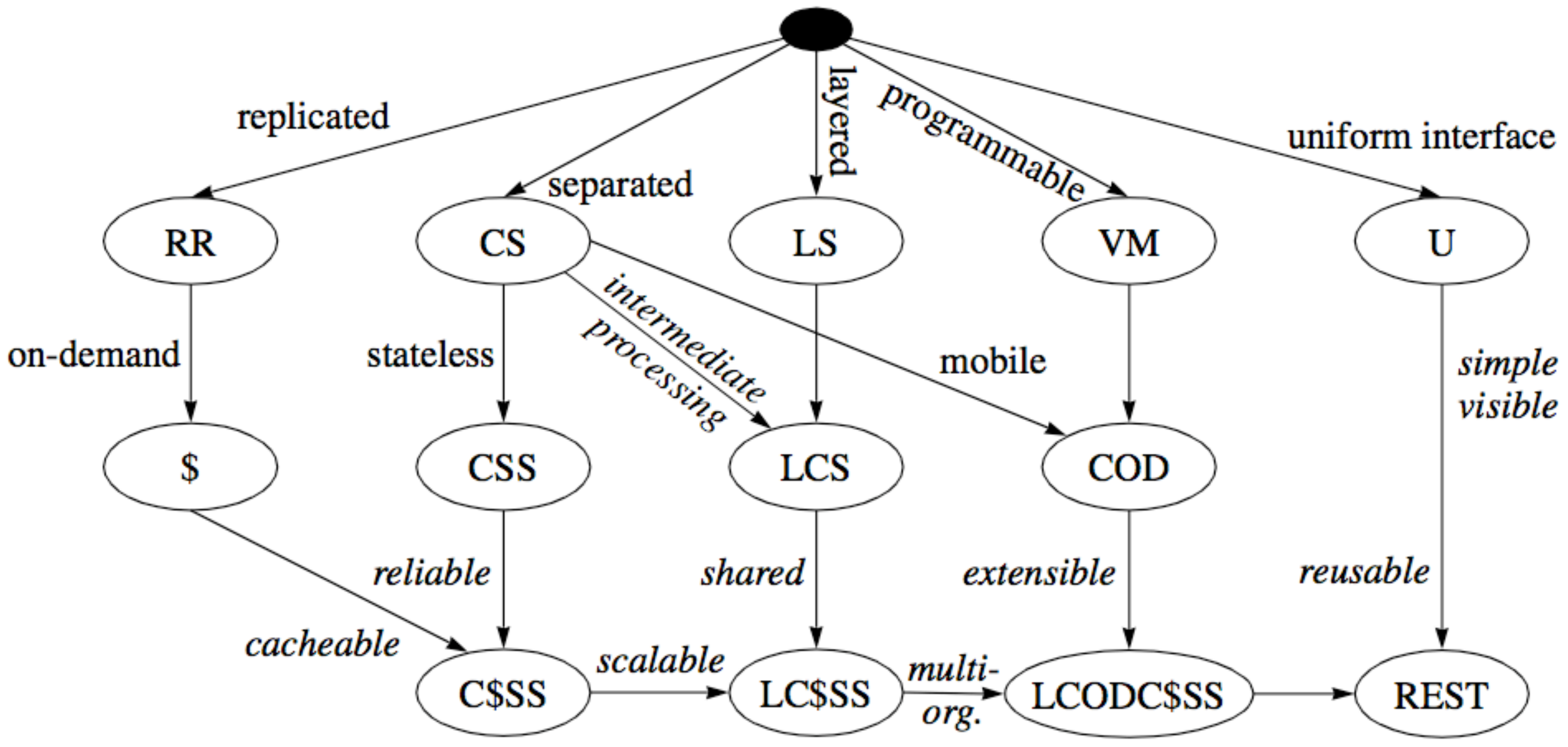


Figure 5-9. REST Derivation by Style Constraints

REST Architectural Elements

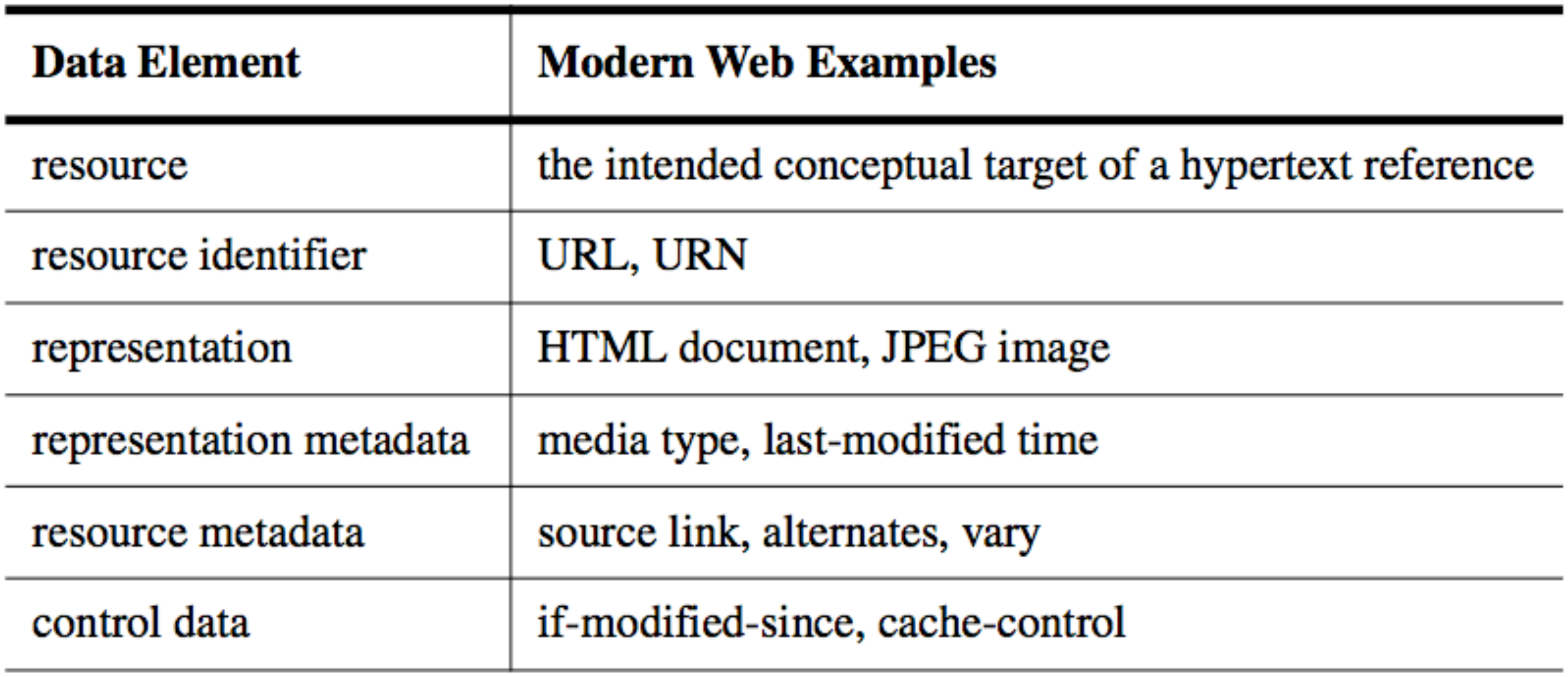
- REST focuses on
  - The roles of components
  - Constraints upon component interaction
  - Component’s interpretation of significant data elements

Data Elements

- The nature and state of data is a key aspect of REST
- REST uses a shared understanding of data types with metadata, but limits the scope of what is revealed to the interface
- Components communicate by transferring a representation of a resource

Data Elements

Table 5-1. REST Data Elements



Data Element	Modern Web 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, vary
control data	if-modified-since, cache-control

Resources and Resource Identifiers

- Any information that can be named can be a resource
  - Resource R is a temporally varying membership function M R(t), which for time t maps to a set of entities, or values, which are equivalent
- A resource identifier is chosen to best fit the nature of the concept being identified

Representations

- A representation is a sequence of bytes plus representation metadata
- May also include resource metadata
  - Information about the resource not specific to the representation
- Data format of a representation known as a *media type*
  - Design of a media type may influence user perceived latency

Connectors

- Encapsulate the activities of accessing resources and transferring resource representations
  - Provide clean separation of concerns
  - Provide substitutability by hiding implementations and allowing them to be replaced
- Remember REST is stateless

Connectors

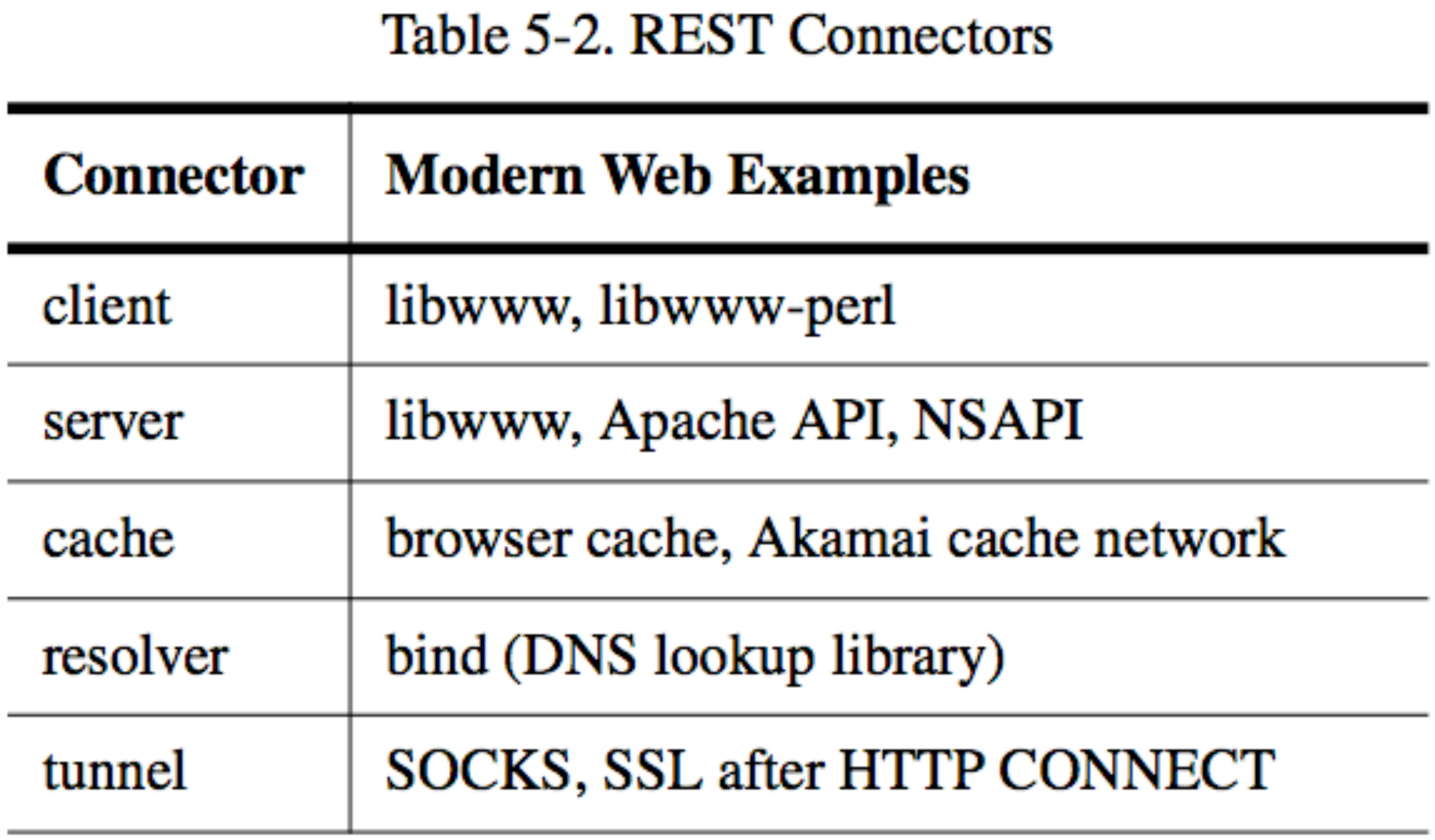


Table 5-2. REST Connectors

Connector	Modern Web Examples
client	libwww, libwww-perl
server	libwww, Apache API, NSAPI
cache	browser cache, Akamai cache network
resolver	bind (DNS lookup library)
tunnel	SOCKS, SSL after HTTP CONNECT

Components

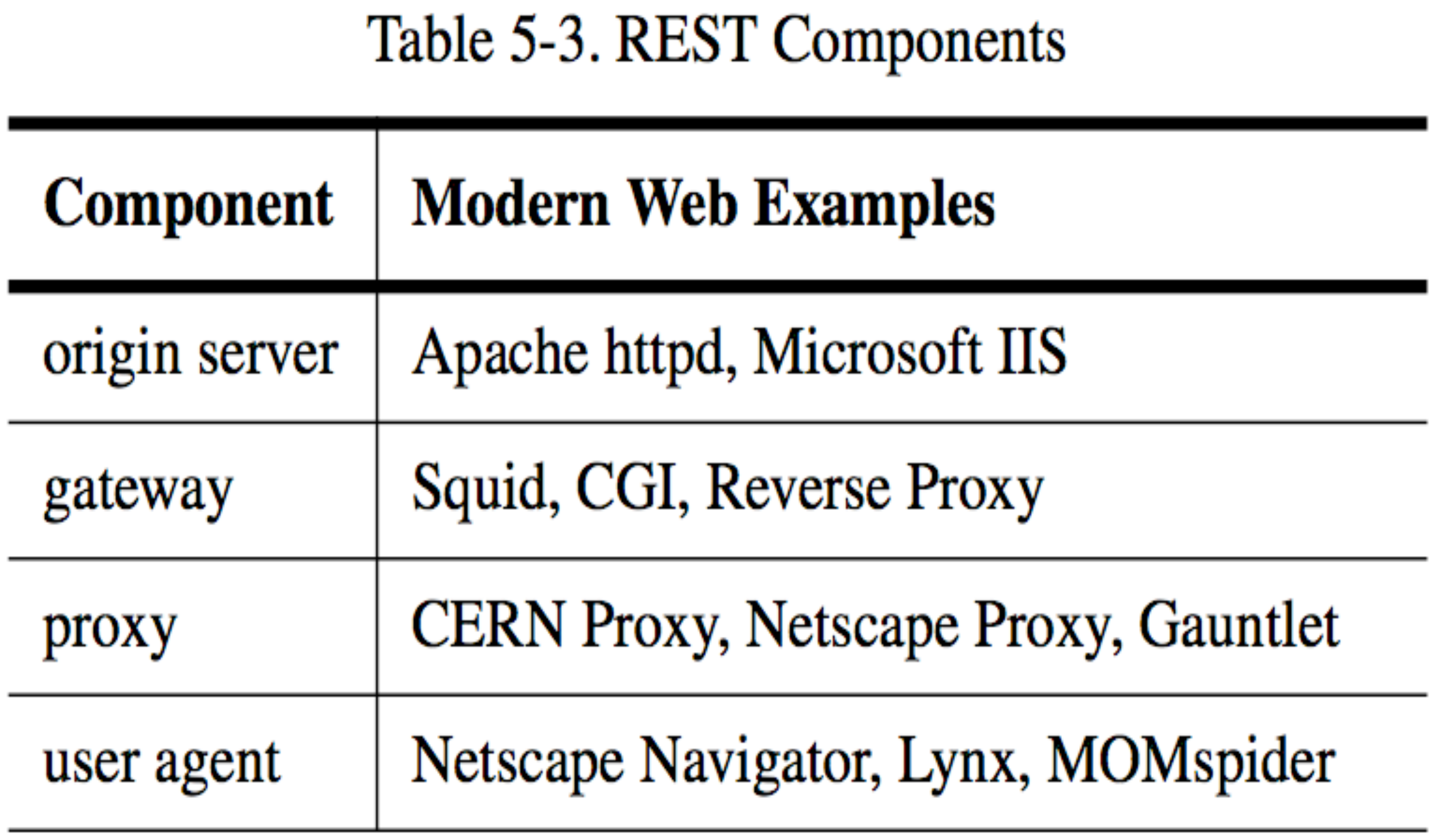


Table 5-3. REST Components

Component	Modern Web Examples
origin server	Apache httpd, Microsoft IIS
gateway	Squid, CGI, Reverse Proxy
proxy	CERN Proxy, Netscape Proxy, Gauntlet
user agent	Netscape Navigator, Lynx, MOMspider

Process view of REST

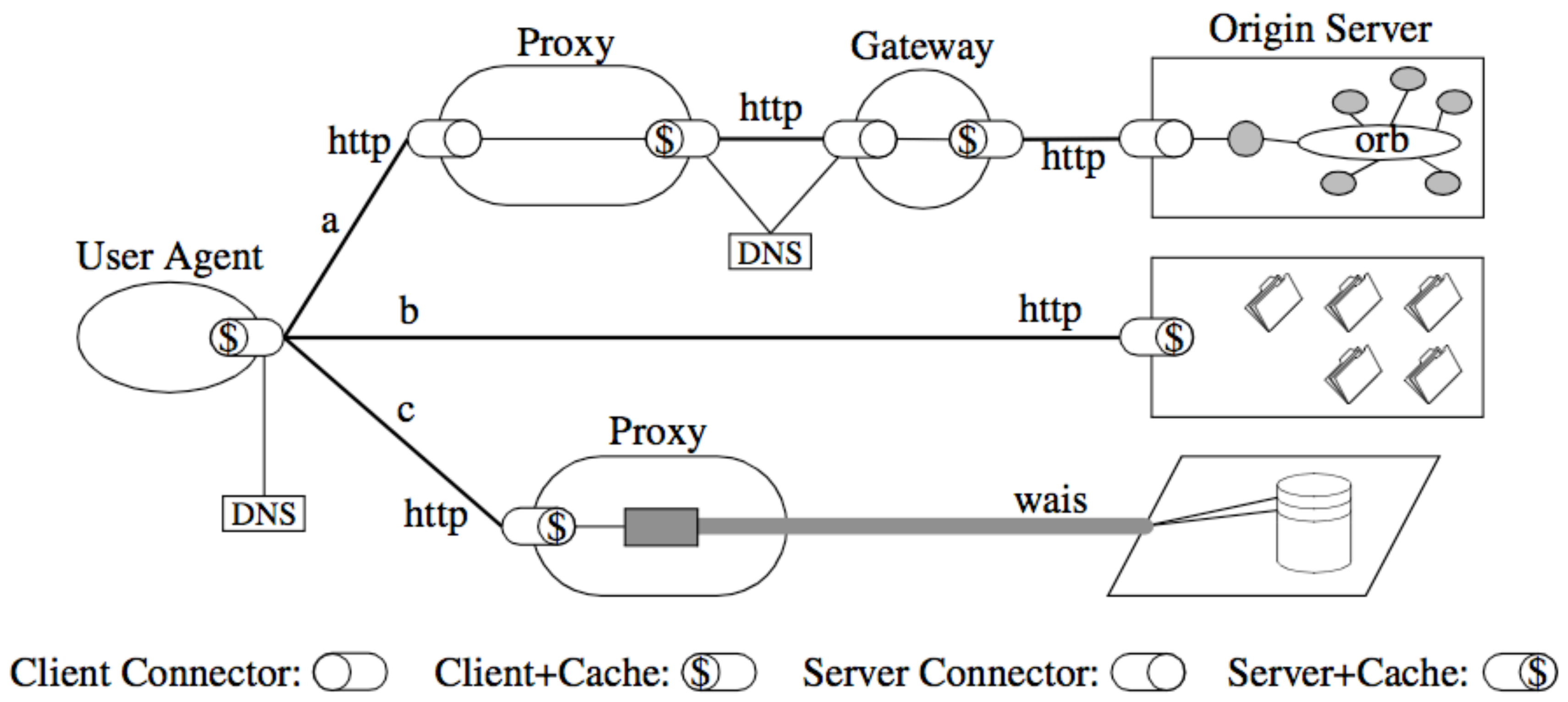


Figure 5-10. Process View of a REST-based Architecture

Connector View of REST

- The mechanics of communication
- Clients examine resource identifier in order to determine communication mechanism
  - REST does not restrict communication protocol

Data View of REST

- Control state concentrated into the representations received in response to interactions
  - Steady state reached when there are no more outstanding requests
- Application state stored and controlled by the user agent