# A little REST and Relaxation

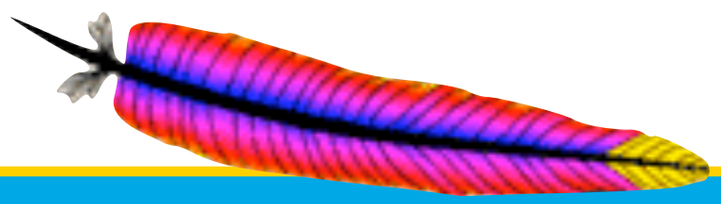

Roy T. Fielding, Ph.D.

Chief Scientist, Day Software

V.P., Apache HTTP Server


http://roy.gbiv.com/talks/200804_REST_ApacheCon.pdf

ApacheCon



# Representational State Transfer

## Web retrospective

Understanding Architecture

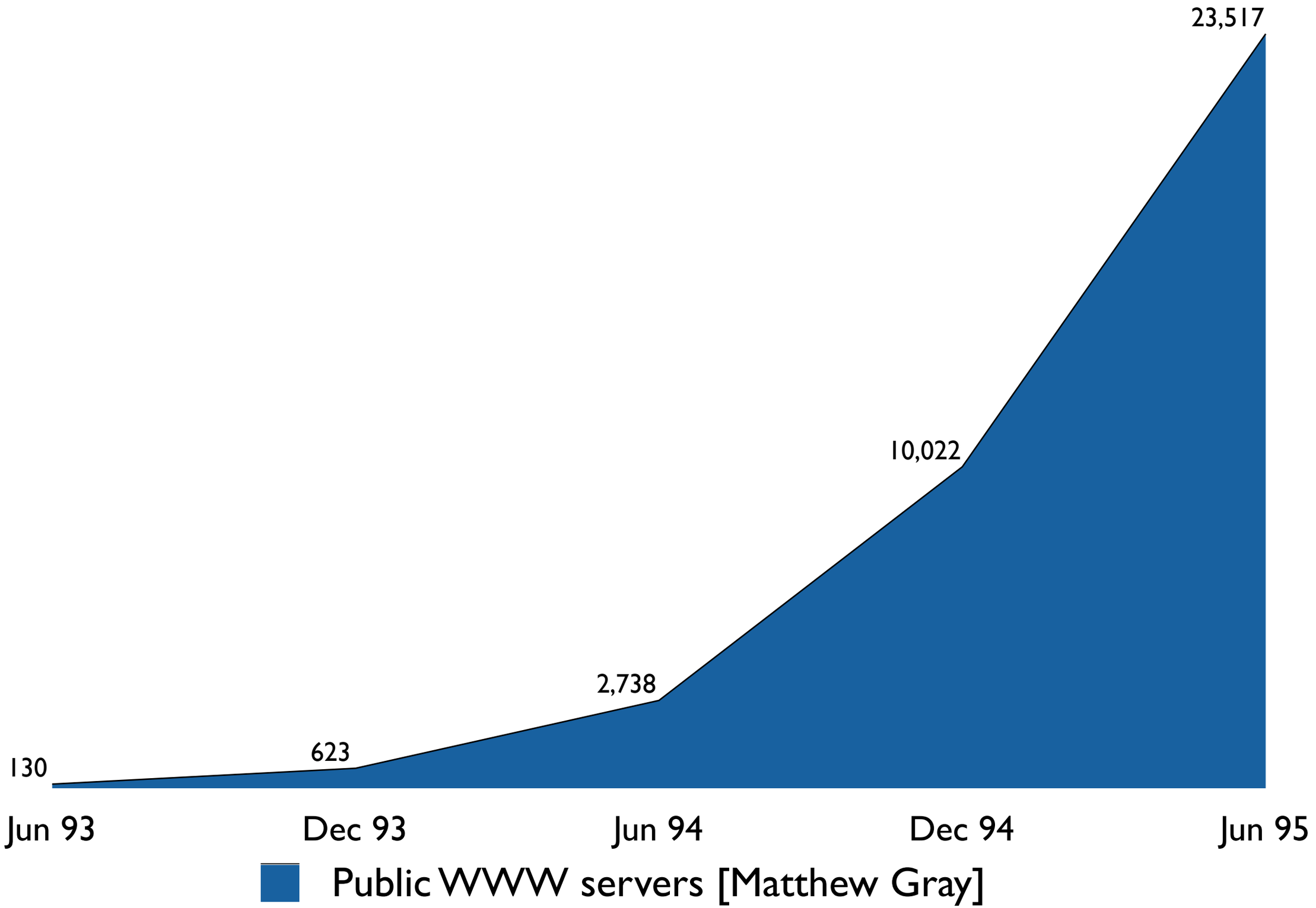
What is REST?

Why REST?

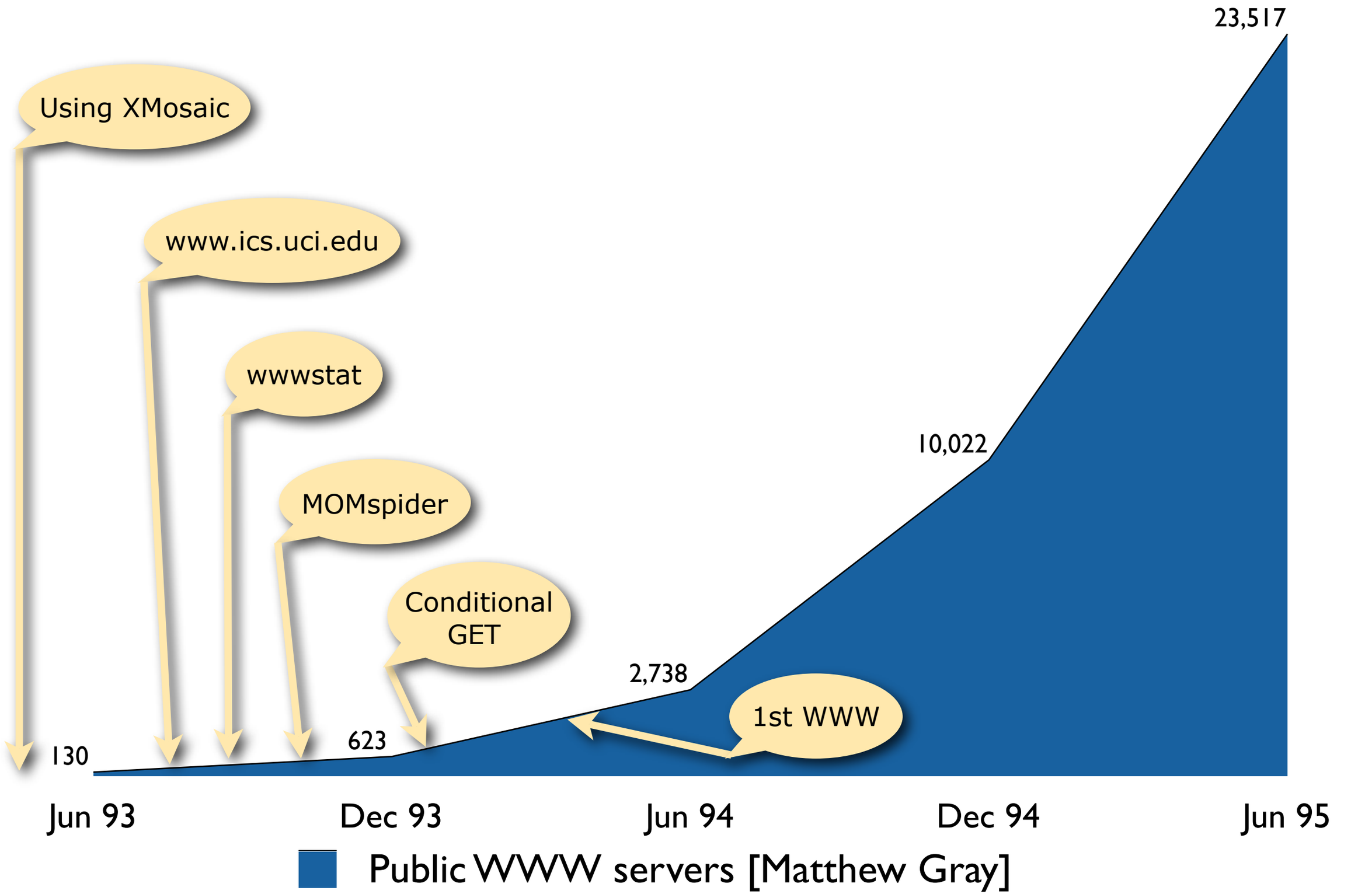
REST at Day

Q & A

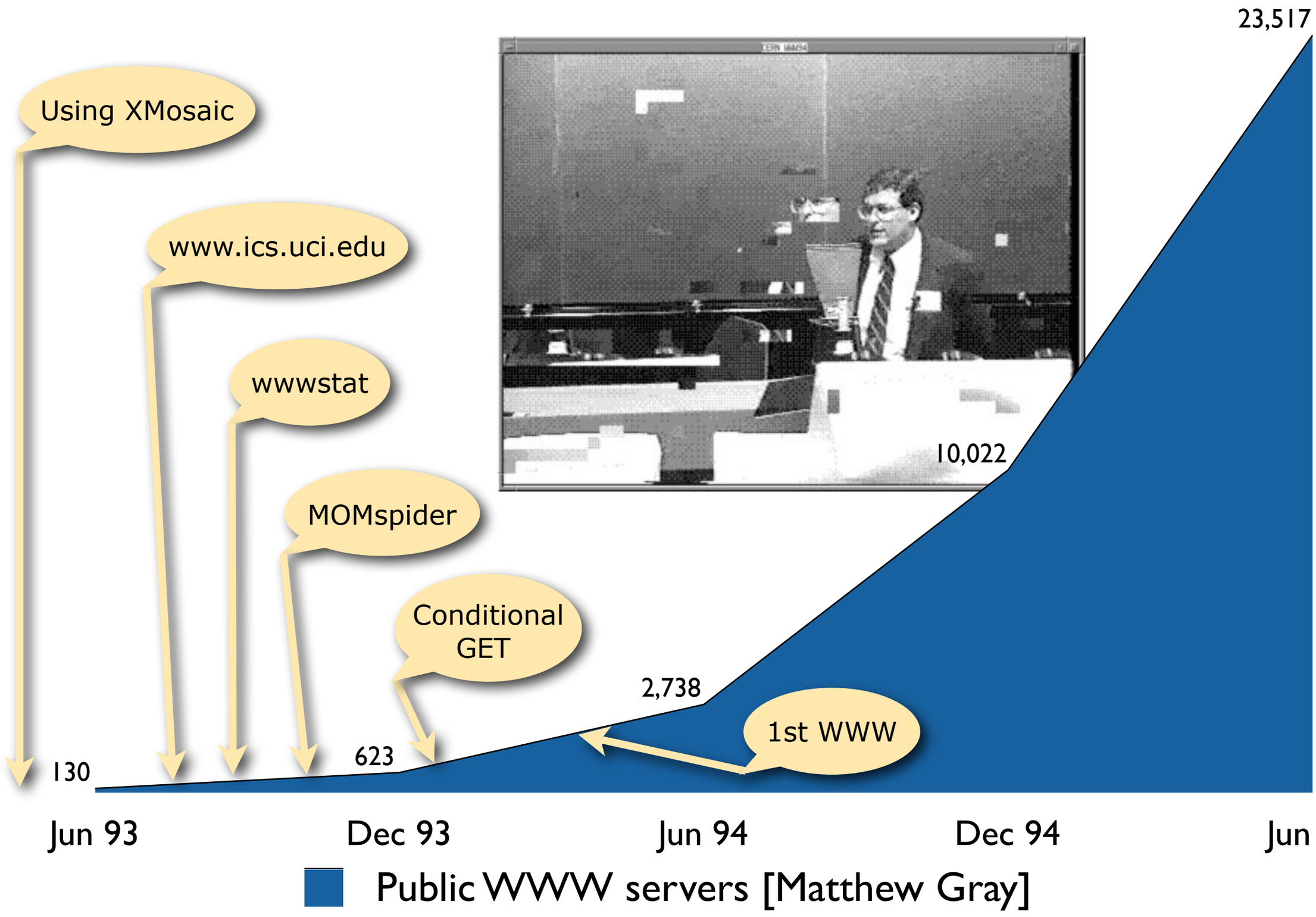
# Context


Mar 08 = 162,662,052 (6,917x)
23,517
10,022
2,738
623
130
Jun 93
Dec 93
Jun 94
Dec 94
Jun 95
Public WWW servers [Matthew Gray]

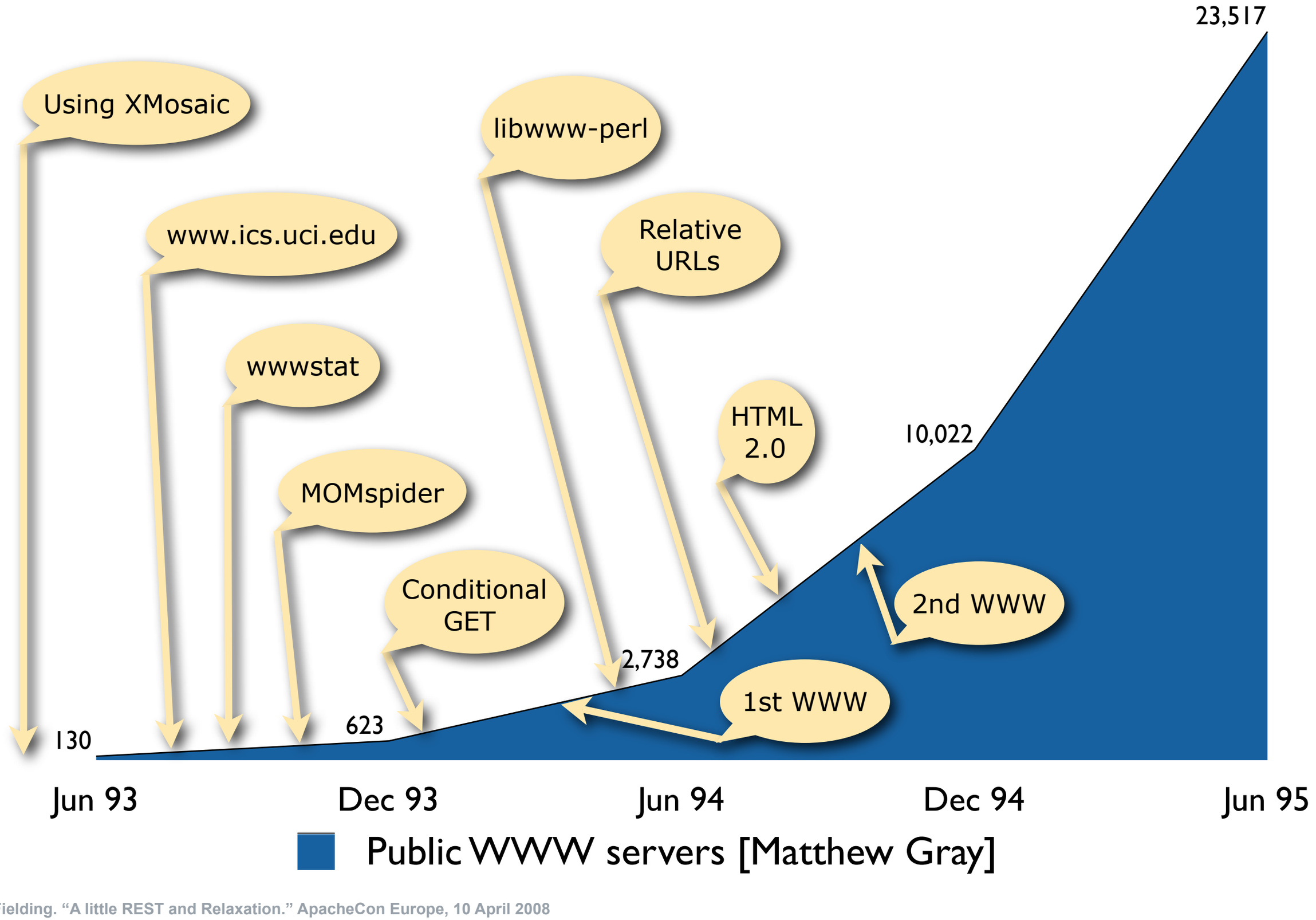
# Context


Mar 08 = 162,662,052 (6,917x)
23,517
10,022
2,738
623
130
Using XMosaic
www.ics.uci.edu
wwwstat
MOMspider
Conditional GET
1st WWW
Jun 93
Dec 93
Jun 94
Dec 94
Jun 95
Public WWW servers [Matthew Gray]

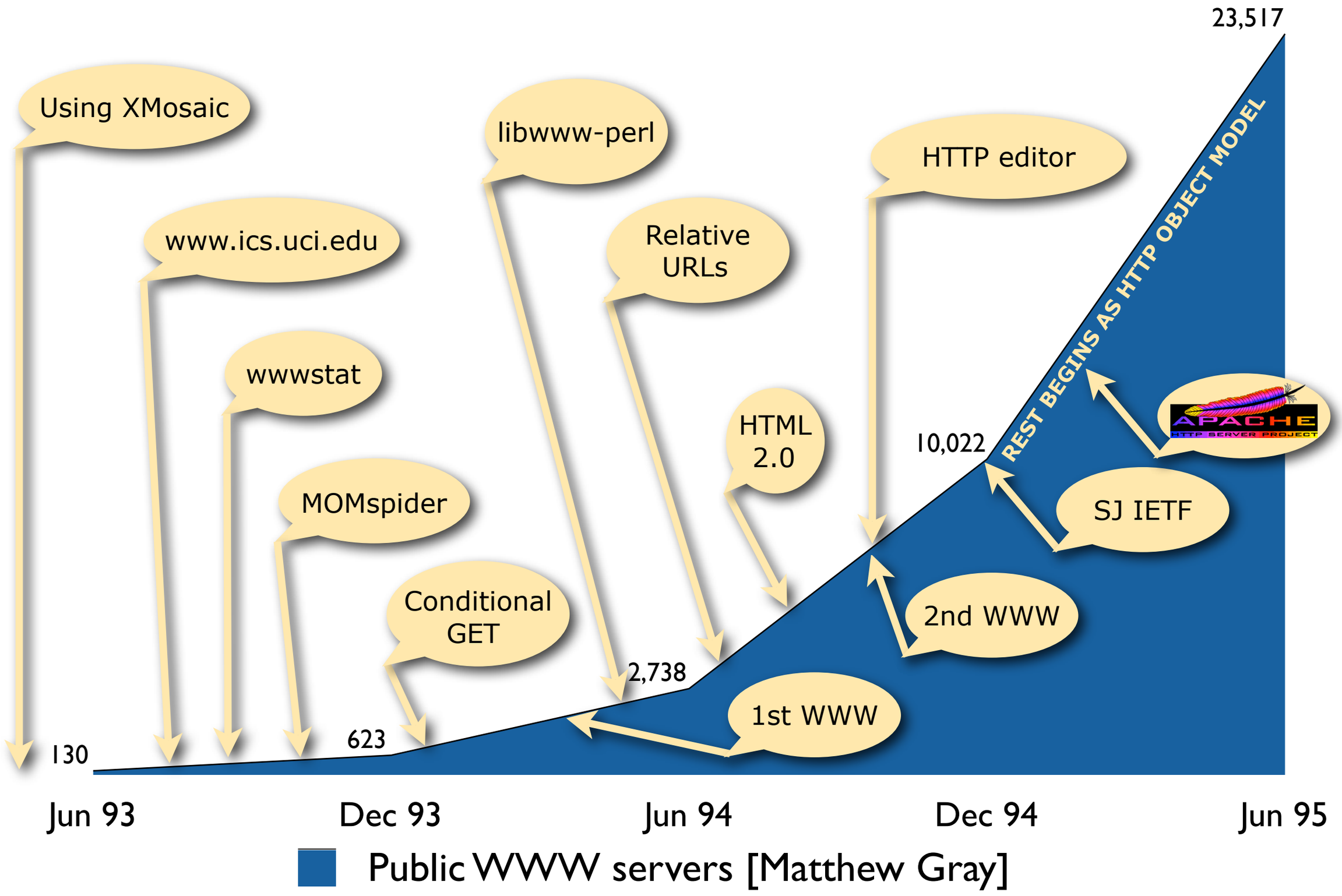
# Context


Mar 08 = 162,662,052 (6,917x)
23,517
10,022
2,738
623
130
Using XMosaic
www.ics.uci.edu
wwwstat
MOMspider
Conditional GET
1st WWW
Jun 93
Dec 93
Jun 94
Dec 94
Jun 95
Public WWW servers [Matthew Gray]

# Context


Mar 08 = 162,662,052 (6,917x)
23,517
10,022
2,738
623
130
Using XMosaic
www.ics.uci.edu
wwwstat
MOMspider
Conditional GET
libwww-perl
Relative URLs
HTML 2.0
2nd WWW
1st WWW
Jun 93
Dec 93
Jun 94
Dec 94
Jun 95
Public WWW servers [Matthew Gray]

# Context


Mar 08 = 162,662,052 (6,917x)
23,517
10,022
2,738
623
130
Using XMosaic
www.ics.uci.edu
wwwstat
MOMspider
libwww-perl
Relative URLs
Conditional GET
HTML 2.0
HTTP editor
REST BEGINS AS HTTP OBJECT MODEL
APACHE
HTTP SERVER PROJECT
SJ IETF
2nd WWW
1st WWW
Jun 93
Dec 93
Jun 94
Dec 94
Jun 95
Public WWW servers [Matthew Gray]

# The Web Problem (circa 1994)

## Early architecture based on solid principles

- URLs, separation of concerns, simplicity
  - lacked architectural description and rationale

## Protocols assumed a direct server connection

- no awareness of caching, proxies, or spiders
- many independent extensions

## Emerging awareness of the Web

- exponential growth threatened the Internet
  - commercialization meant new stakeholders with new (selfish) requirements

## A modern Web architecture was needed

- but how do we avoid breaking the Web in the process?

# Web Requirements & Properties

# Web Requirements & Properties

## Low entry barrier

- Hypermedia User Interface
- Simple protocols for authoring and data transfer

▸ must be **Simple** and **Reusable**; want **Extensible**

# Web Requirements & Properties

## Low entry barrier

- Hypermedia User Interface
- Simple protocols for authoring and data transfer

▸ must be **Simple** and **Reusable**; want **Extensible**

## Distributed Hypermedia System

- Large data transfers
- Sensitive to user-perceived latency

▸ must be **Data-driven** and **Streamable**; want **Performant**

# Web Requirements & Properties

## Low entry barrier

- Hypermedia User Interface
- Simple protocols for authoring and data transfer

▸ must be **Simple** and **Reusable**; want **Extensible**

## Distributed Hypermedia System

- Large data transfers
- Sensitive to user-perceived latency

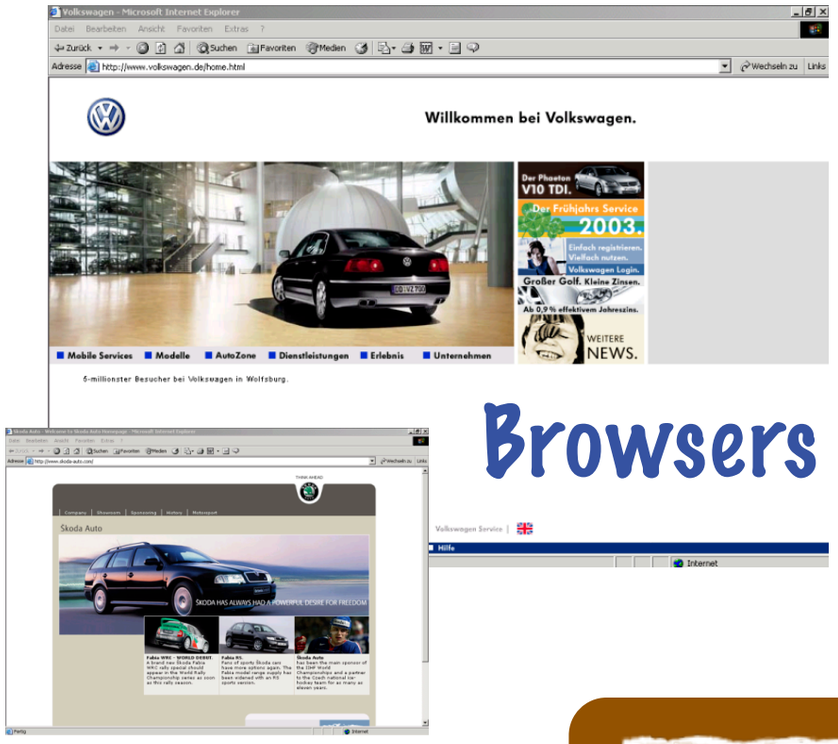
▸ must be **Data-driven** and **Streamable**; want **Performant**

## Multiple organizational boundaries

- Anarchic scalability
- Gradual and fragmented change (deployment)

▸ must be **Scalable**, **Portable**, **Evolvable**; want **Reliable**, **Visible**, **Customizable**, **Configurable**, **Extensible**, …

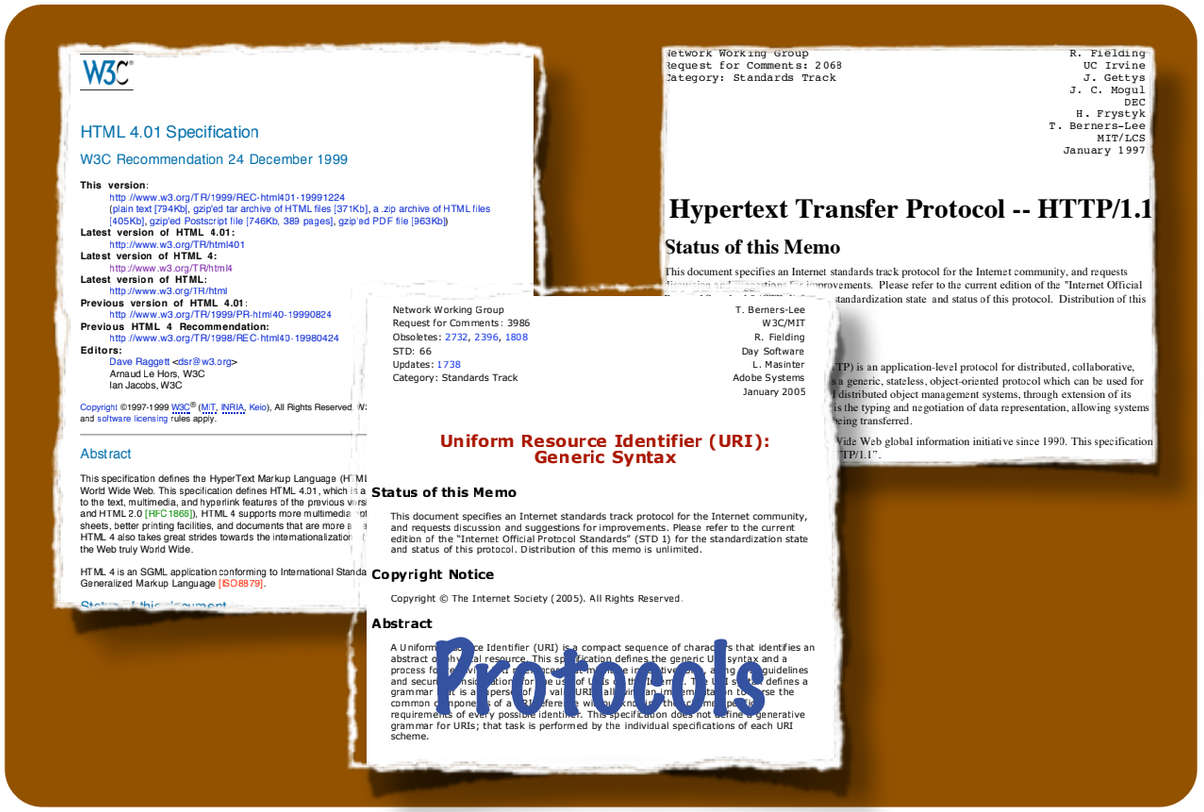
# What is the Web, really?



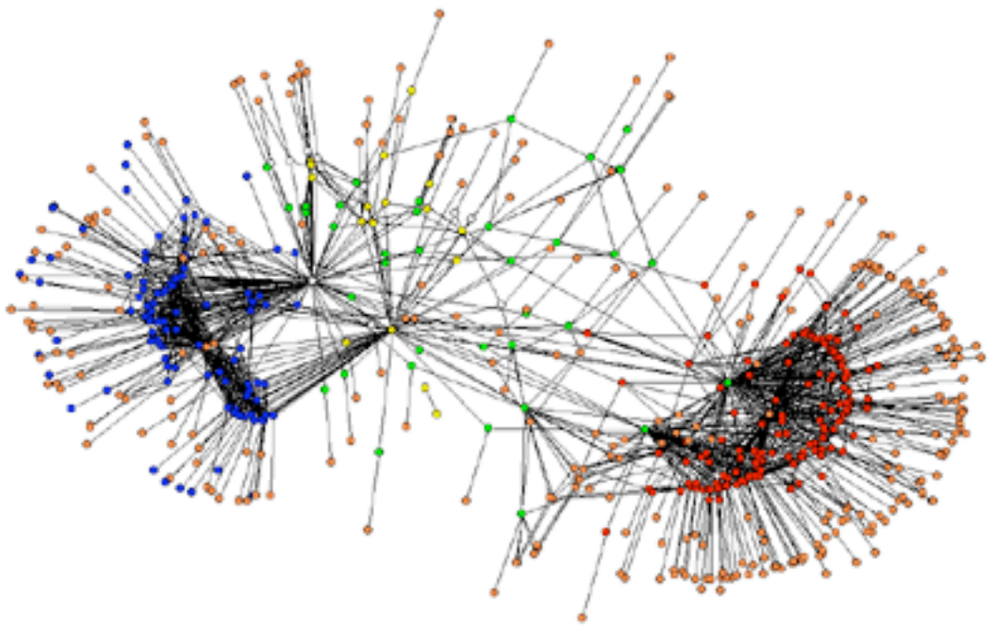

Browsers

Information

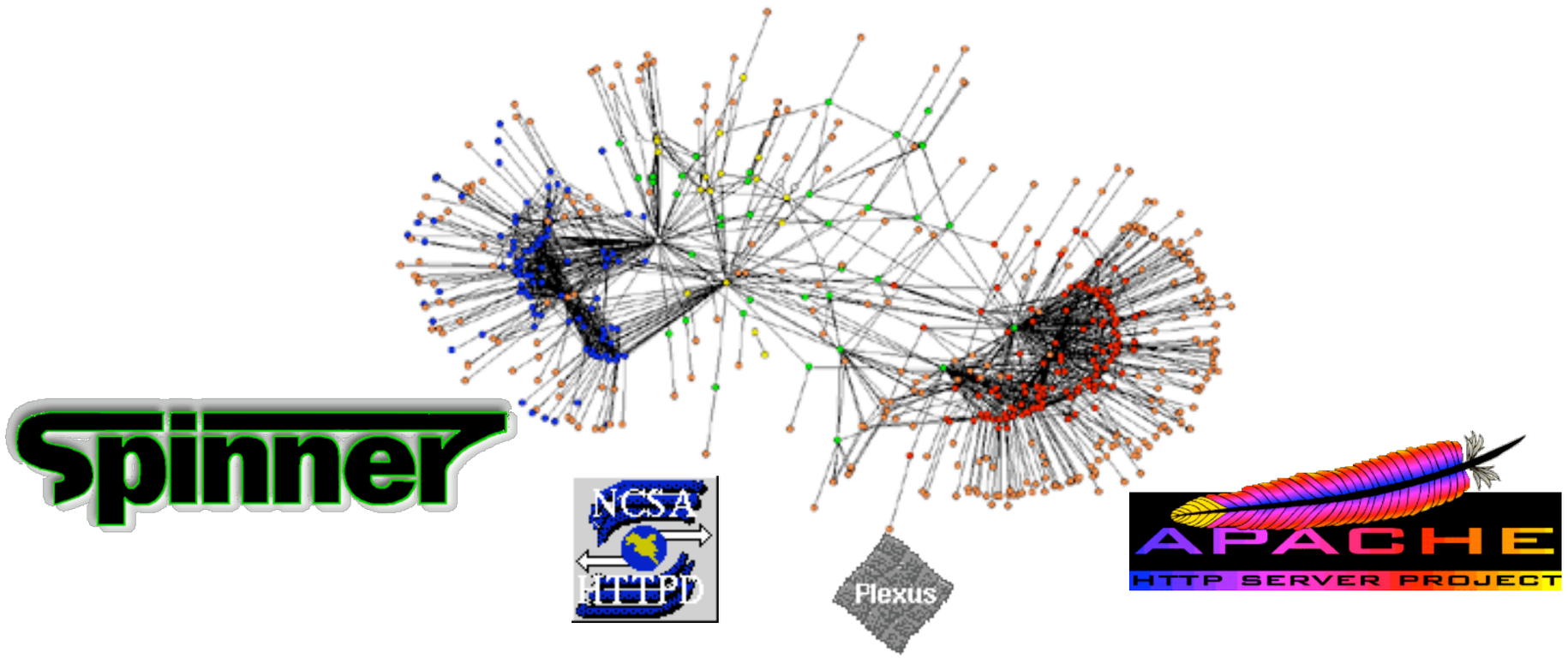
Protocols



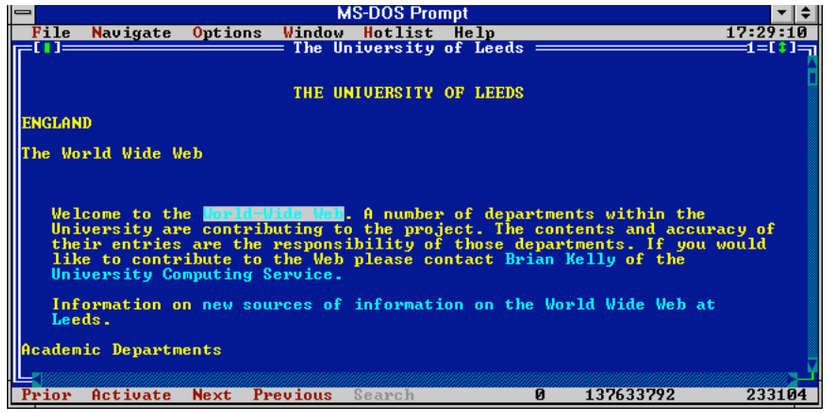
# Web Implementation (user view)

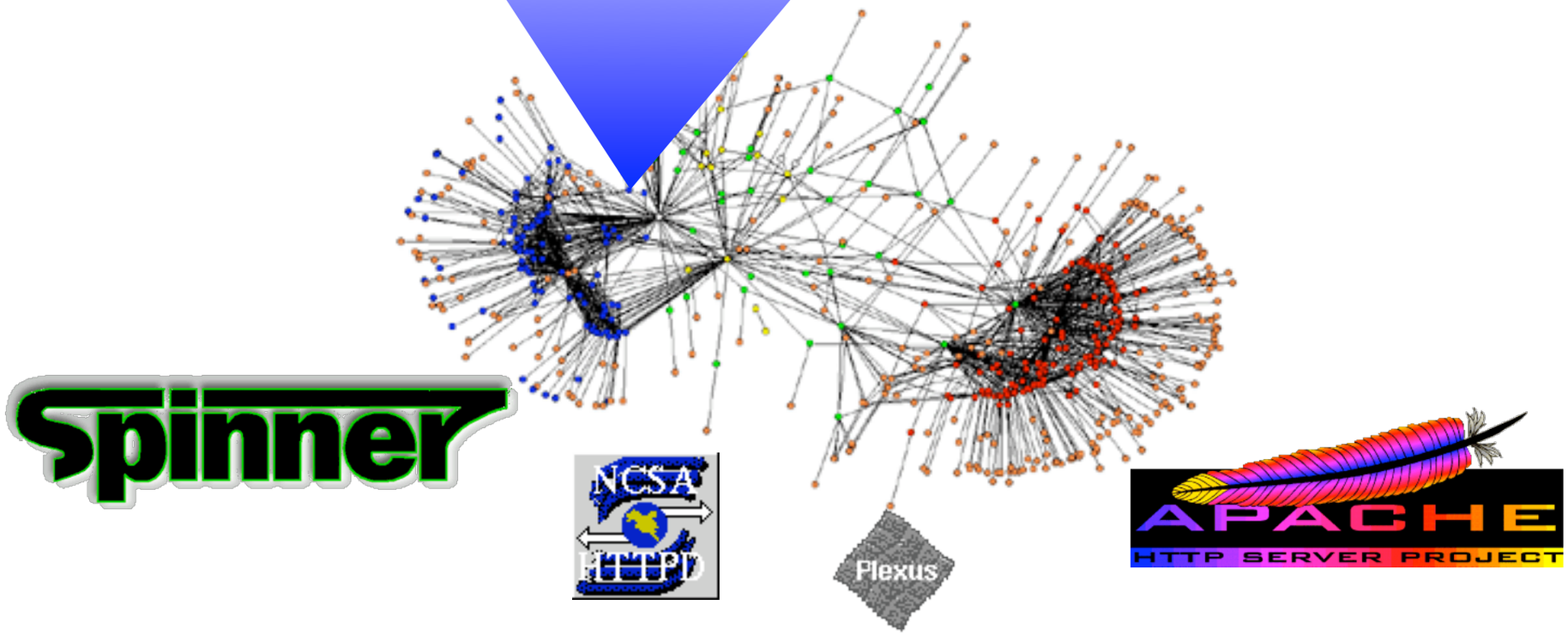


# Web Implementation (user view)

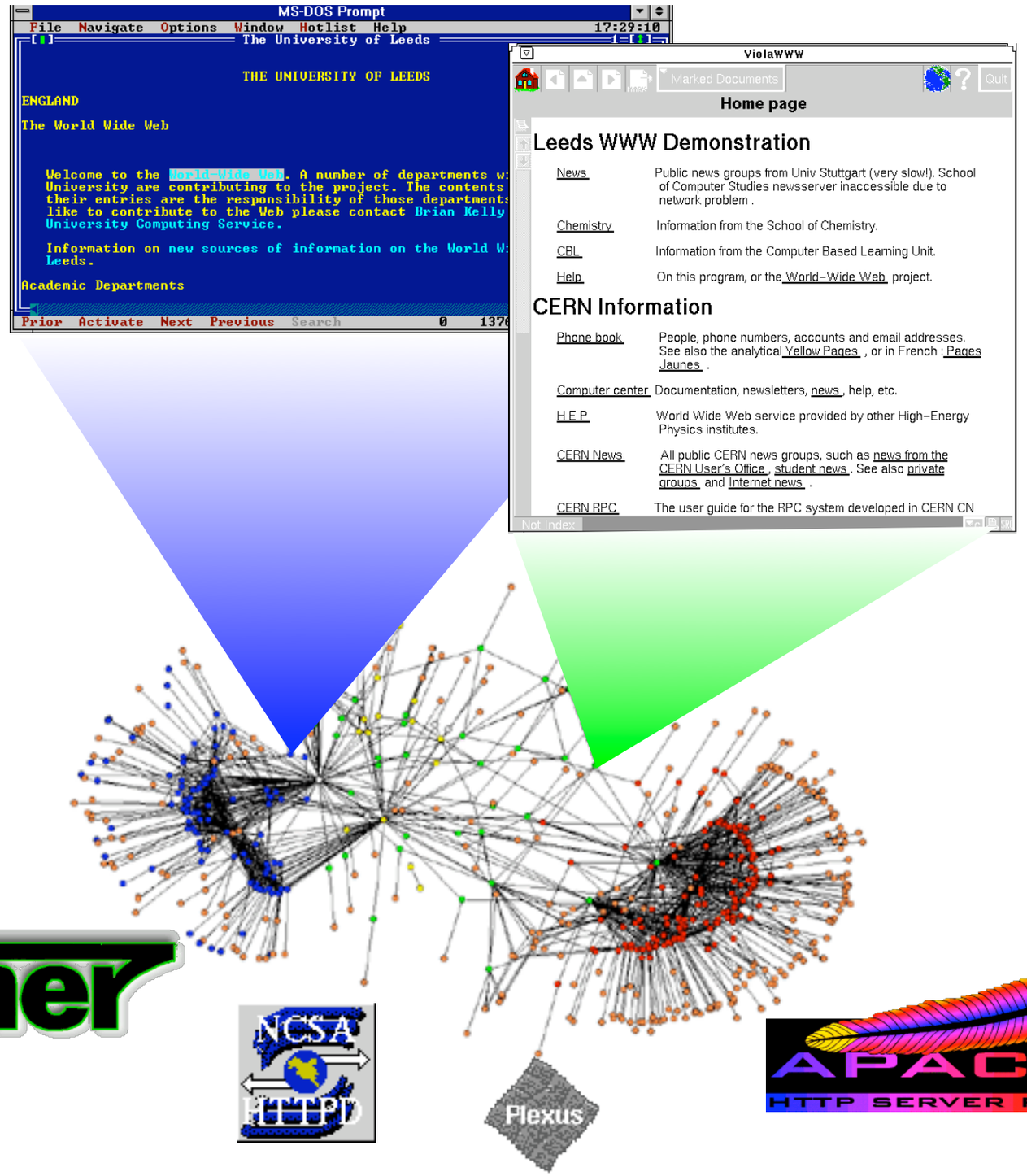


# Web Implementation (user view)

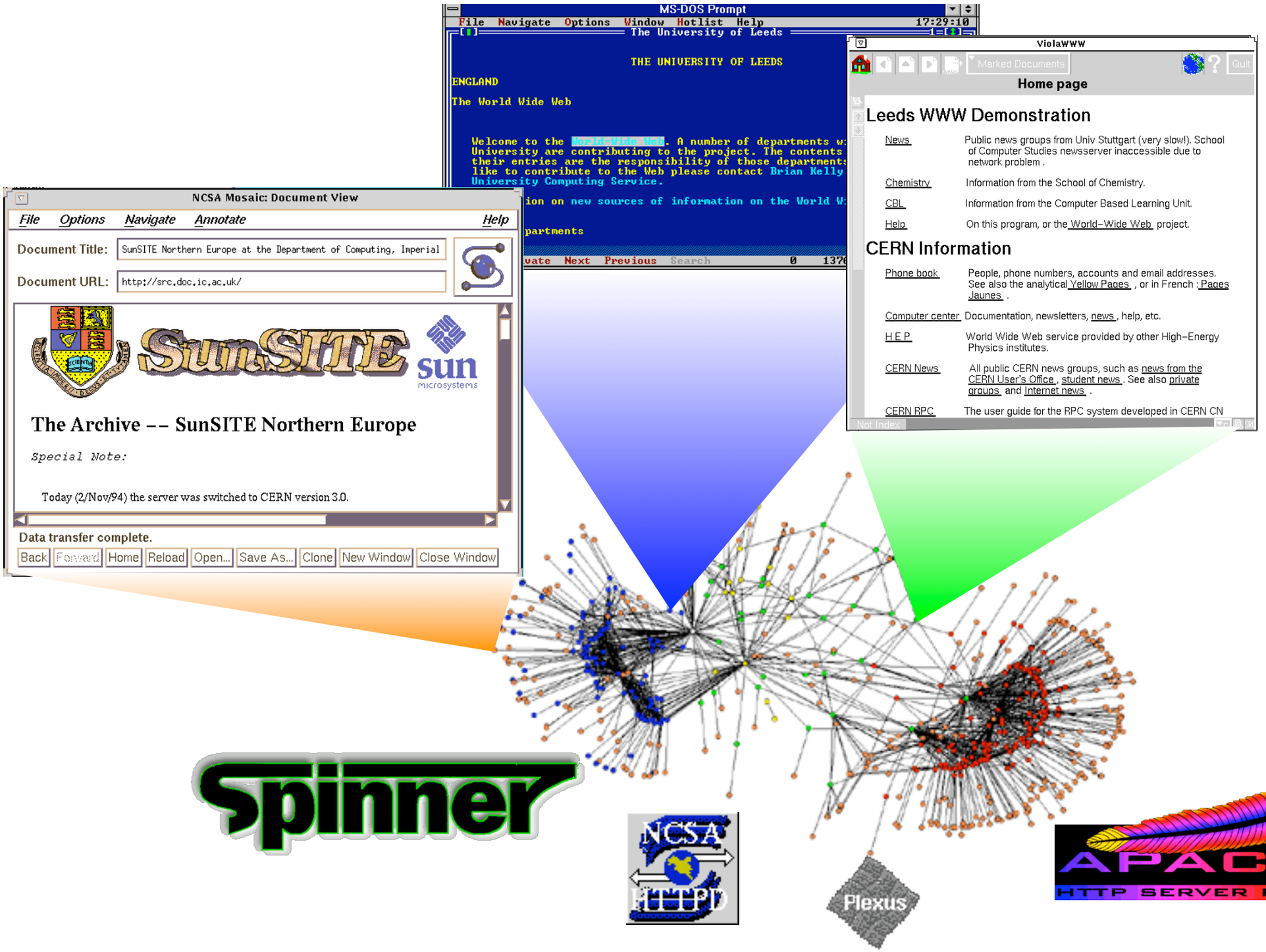


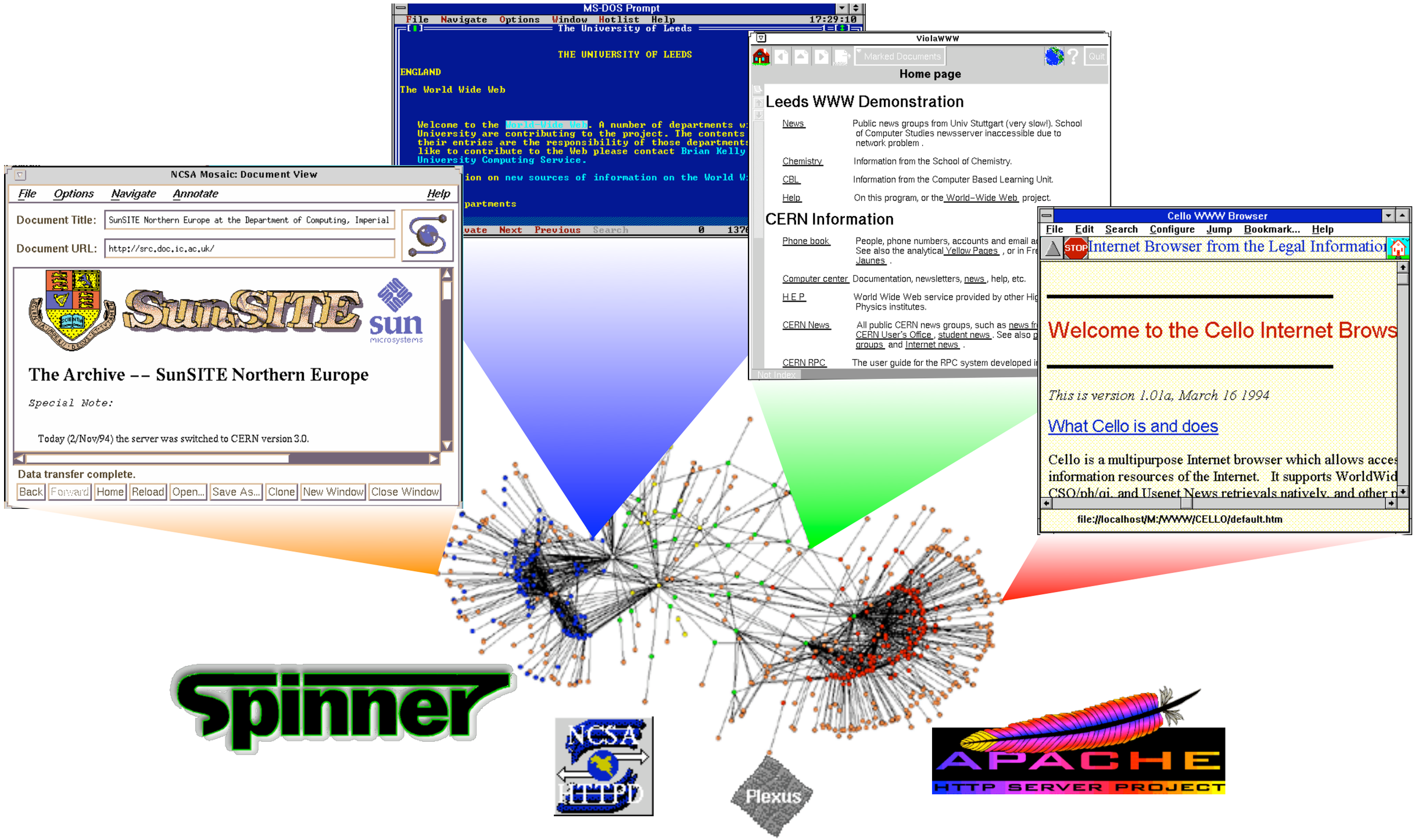
# Web Implementation (user view)



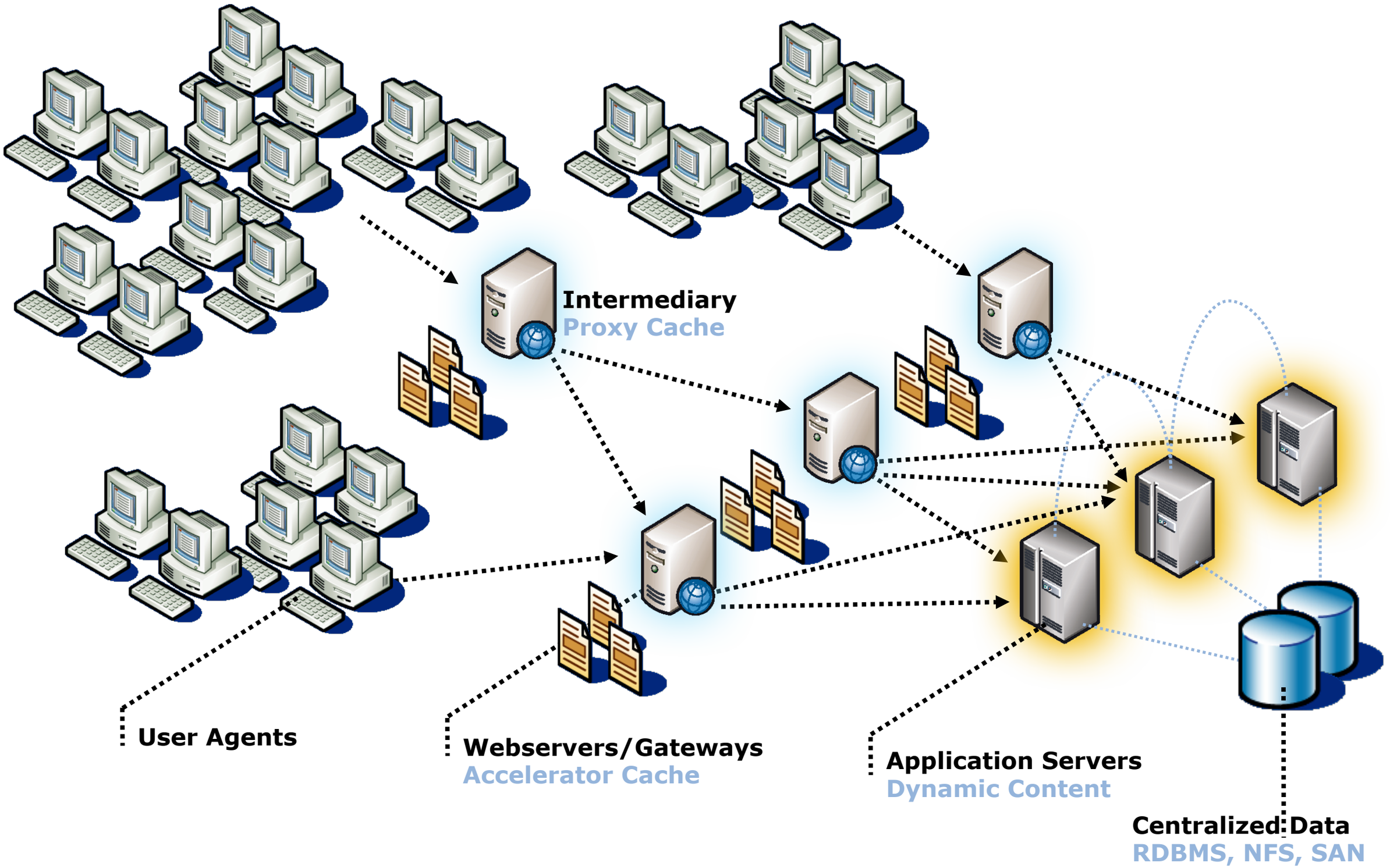
# Web Implementation (user view)



# Web Implementation (user view)

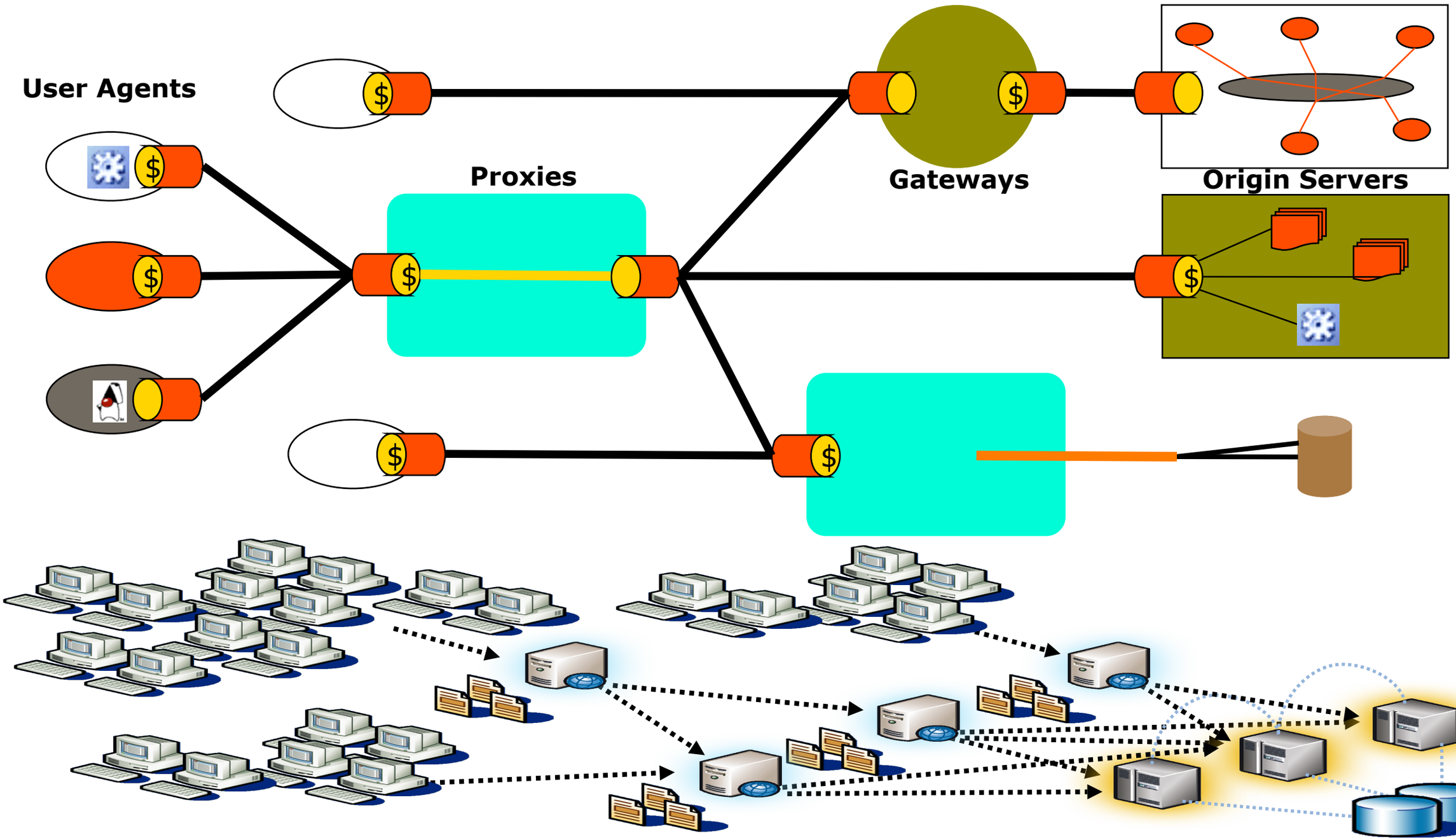


# Web Implementation (origin view)



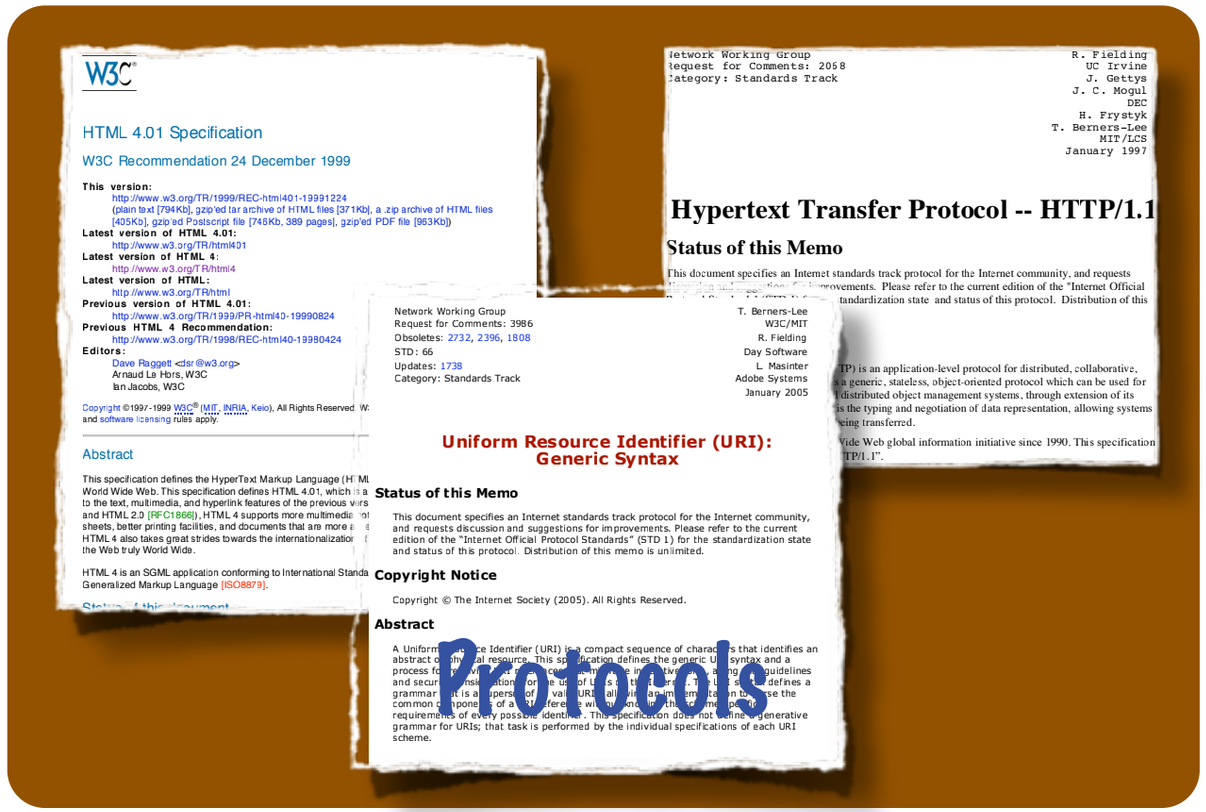
# Web Architecture

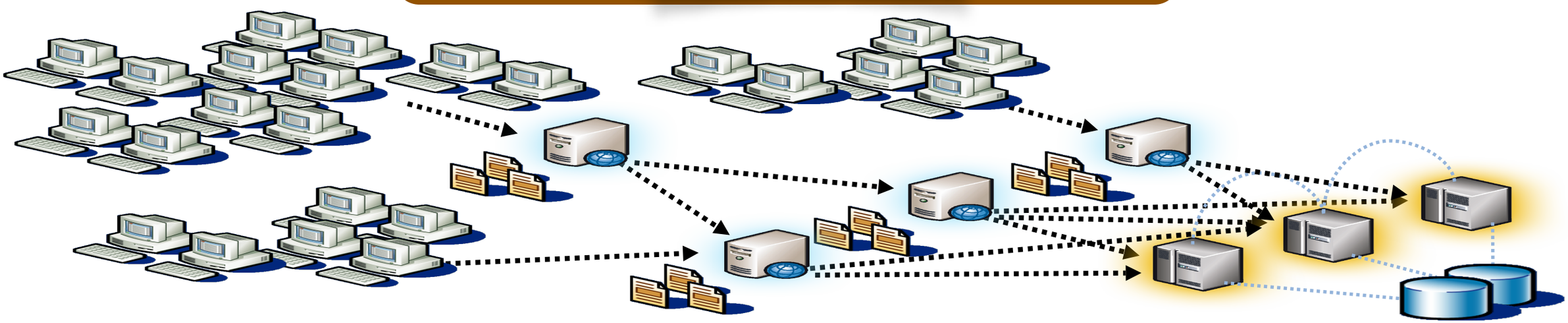
## A **vertical abstraction** on implementation



# Web Architecture

## A **vertical abstraction** on implementation



# Web Architecture

A **vertical abstraction** on implementation

Components

- User agents, Intermediaries, Servers
- Browsers, Spiders, Proxies, Gateways, Origin Servers

Connectors

- **HTTP**: a standard transfer protocol to prefer over many

Data

- **URI**: one identifier standard for all resources
- **HTML**, **XML**, **RDF**, **PDF**, **JPEG**, **JSON**, ...
  - common representation formats to describe and bind resources

# Agenda

Web retrospective

Understanding Architecture

What is REST?

Why REST?

REST at Day

Q & A

# NOT “Enterprise Architecture”

Half of what you read about Architecture in Software Industry trade rags is **WRONG**

- **it usually isn’t even about architecture**
  - **strategic vision**
  - **resource planning**
  - **requirements analysis**
  - **stakeholder reviews**
  - **staffing & purchasing**
  - **software structure**
  - **libraries/frameworks**
  - **buzzword-compliance**

**The same folks will be selling REST**

- **there will be lots of miscommunication (in and out)**

# NOT “Enterprise Architecture”

Half of what you read about Architecture in Software Industry trade rags is **WRONG**

- **it usually isn’t even about architecture**
  - **strategic vision**
  - **resource planning**
  - **requirements analysis**
  - **stakeholder reviews**
  - **staffing & purchasing**
  - **software structure**
  - **libraries/frameworks**
  - **buzzword-compliance**

skyscrapr where architects become ARCHITECTS

This MSDN site panders in the worst way...

“Solutions Architect”

“Infrastructure Architect” (CTO/sysadmins)

“Enterprise Architect” (CIO/manager)

# The same folks will be selling REST

- **there will be lots of miscommunication (in and out)**

# Software Architecture

A software architecture is an **abstraction** of the run-time elements of a software system during some phase of its operation.

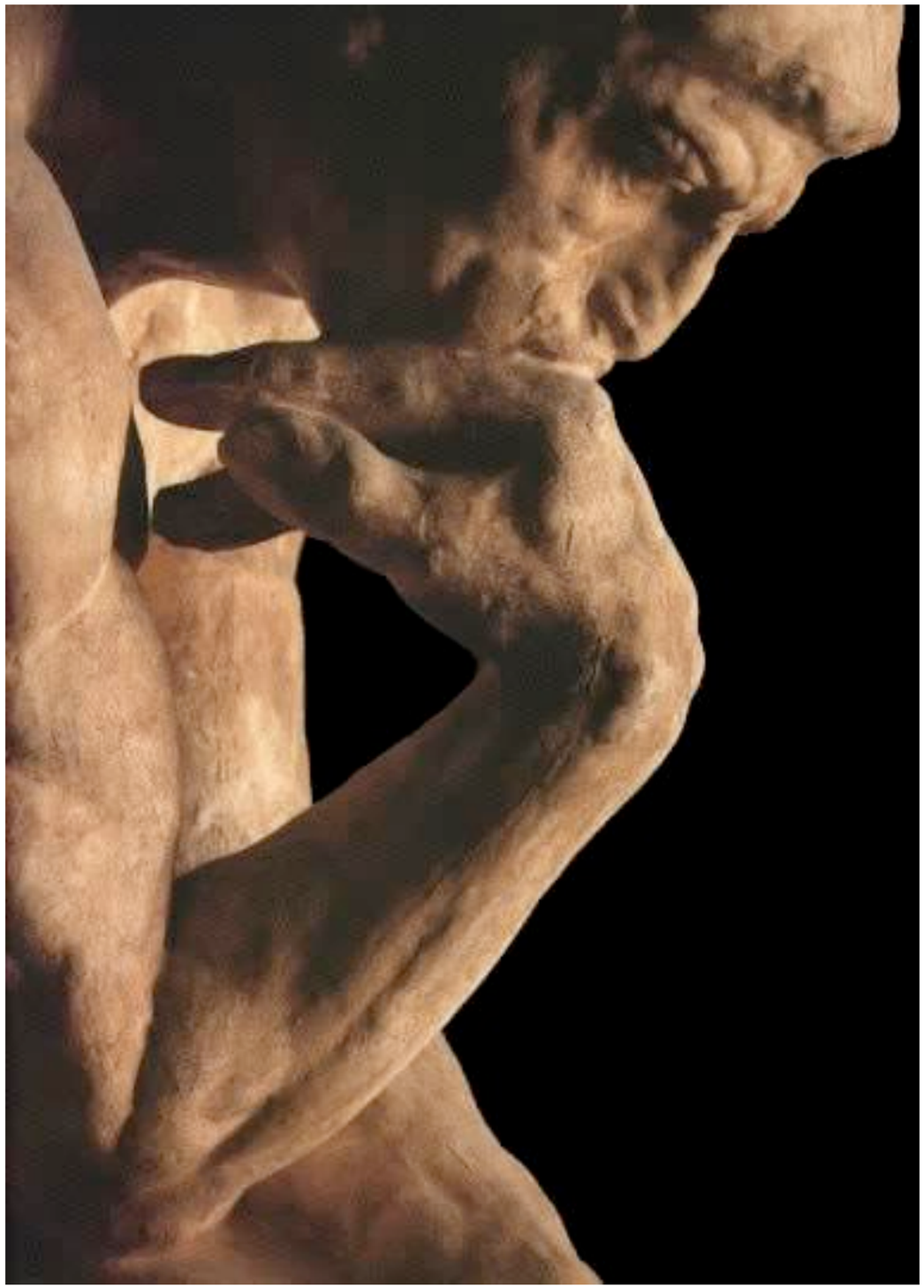
- A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.
- A software architecture is defined by a configuration of architectural elements—components, connectors, and data—constrained in their relationships in order to achieve a desired set of architectural properties.
  - A configuration is the structure of architectural relationships among components, connectors, and data during a period of system run-time.

# Architectural Styles

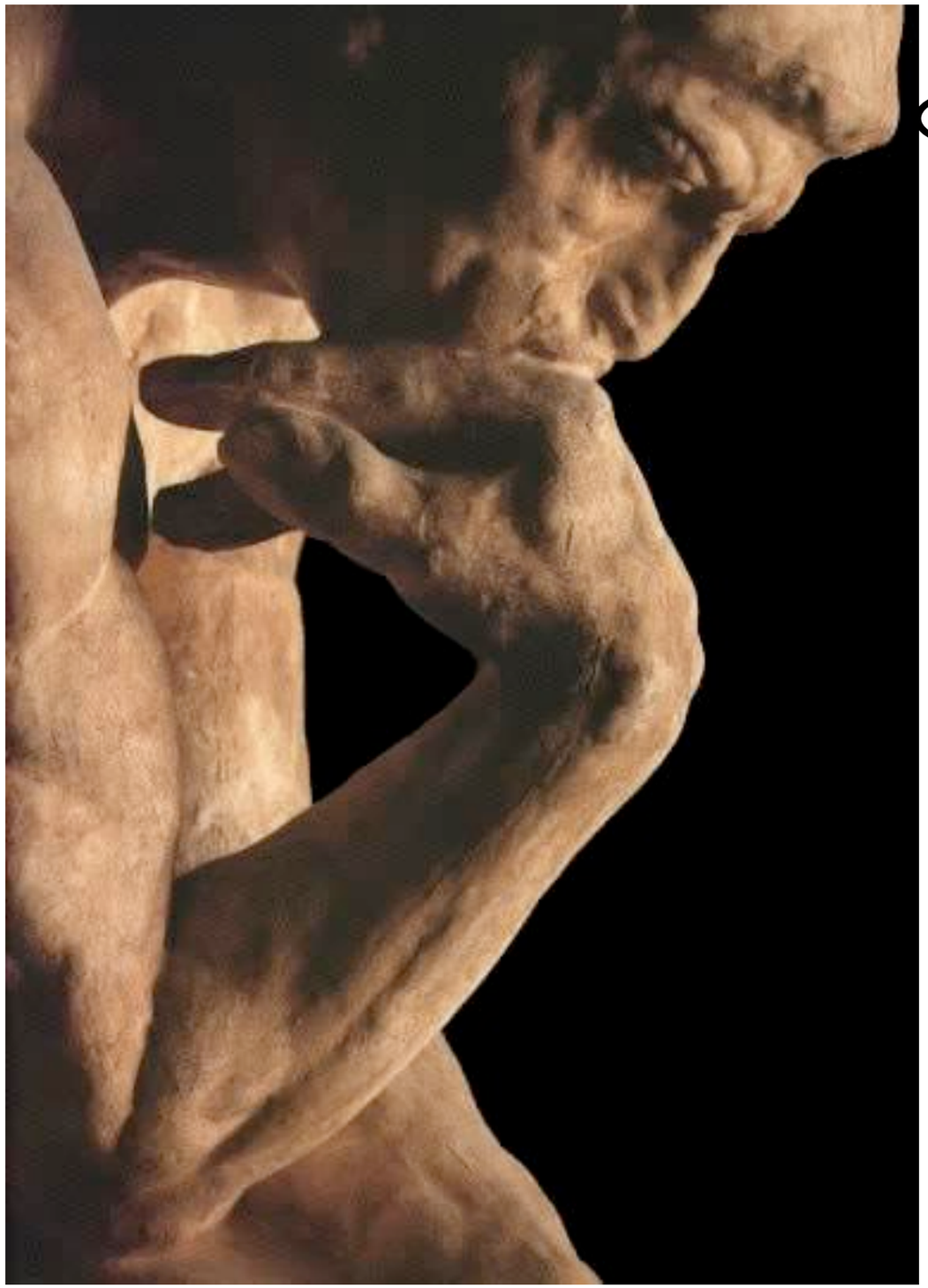
An architectural style is a **coordinated set of architectural constraints** that restricts the roles and features of architectural elements, and the allowed relationships among those elements, within any architecture that conforms to that style.

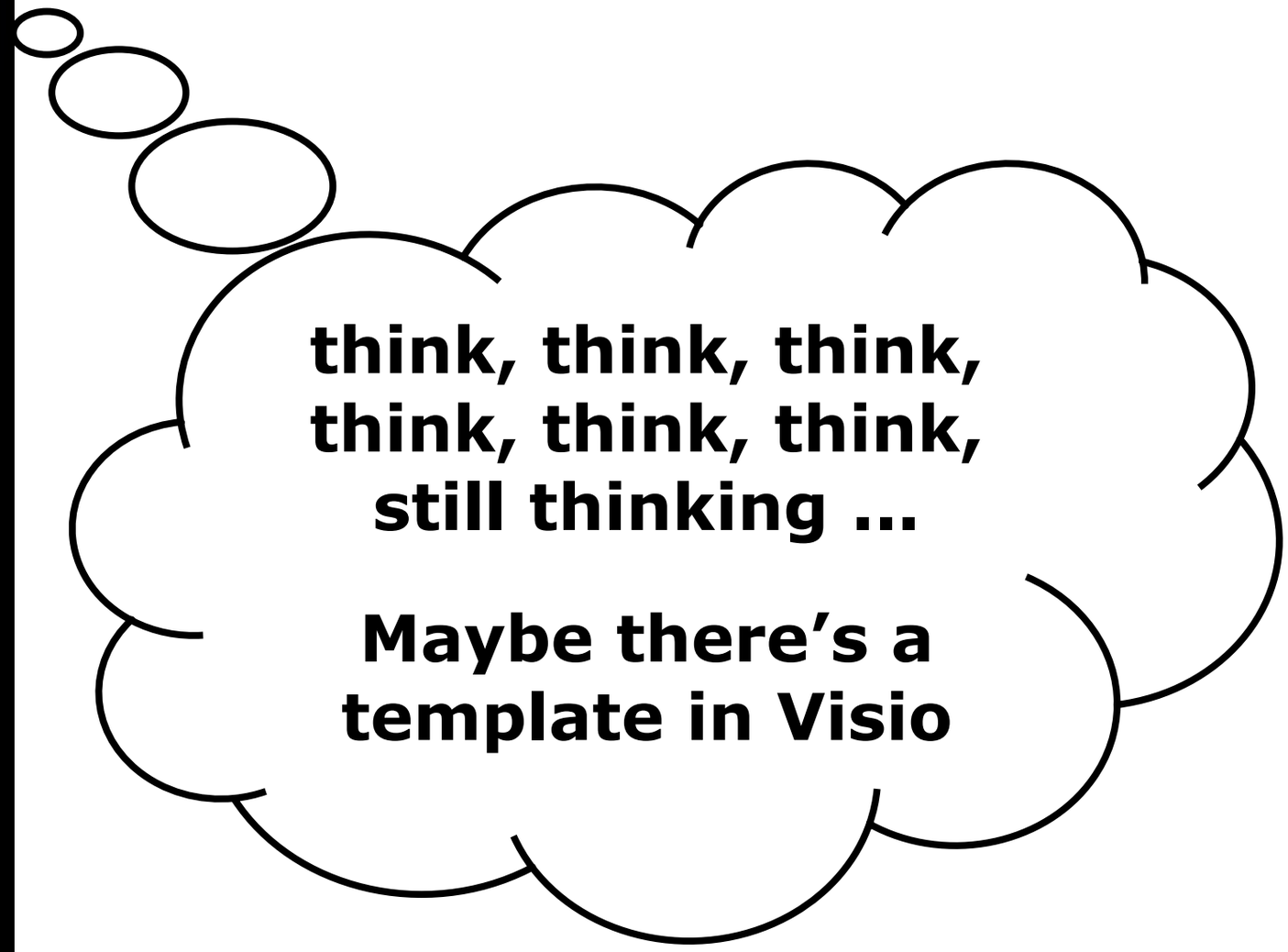
- A style can be applied to many architectures.
- An architecture can consist of many styles.

# Architectural Design Process

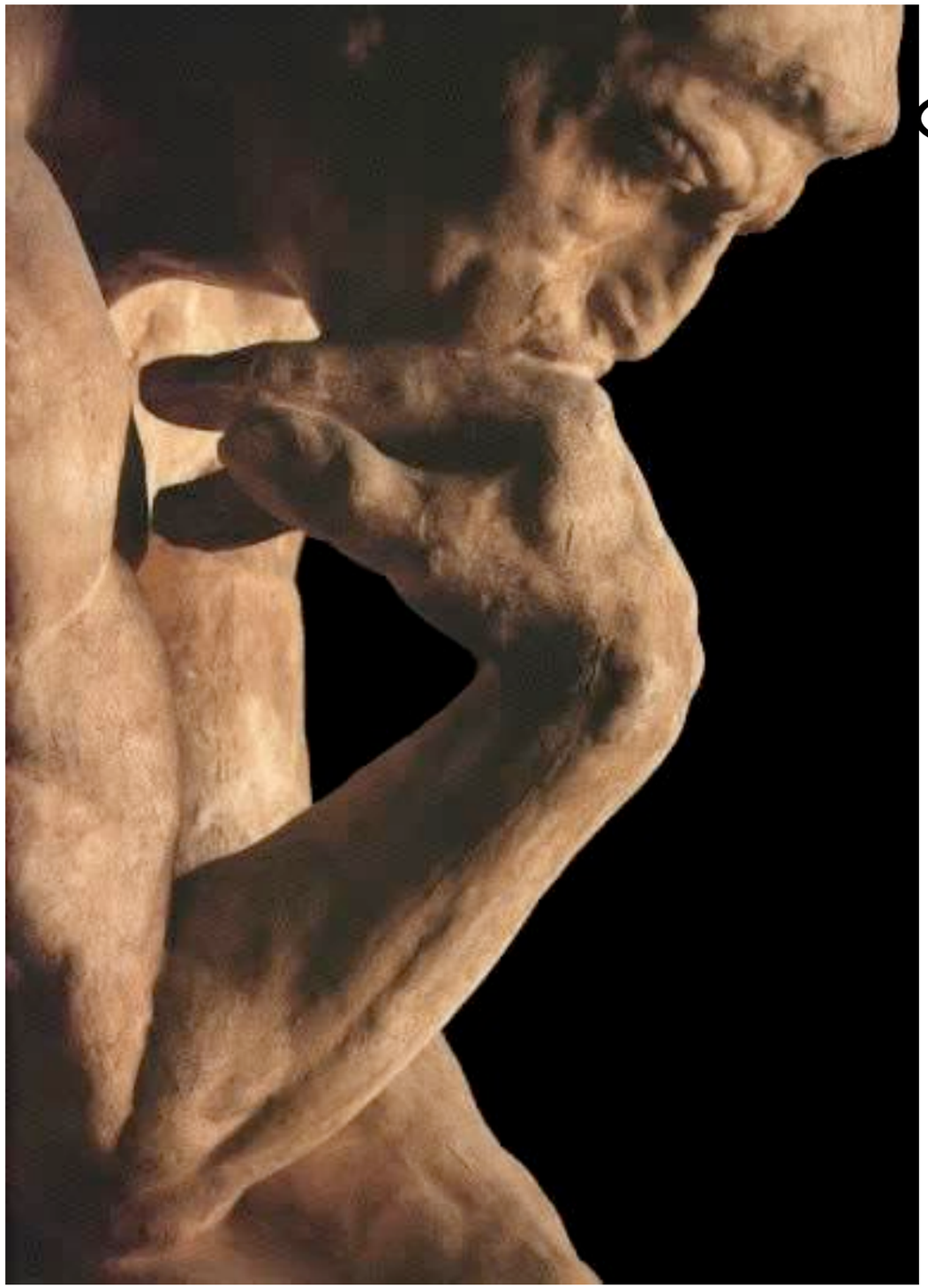


# Architectural Design Process





# Architectural Design Process



**Look at what works in practice, identify styles, and see how they can be combined to obtain properties**


ABEL


Bleu Marine
020 4194141


VERSCHOOR

# Architectural Styles

A **horizontal abstraction** on architecture

- that's one too many abstractions for most folks
- a way of naming architectural patterns in implementation

An architectural style is a set of **constraints**

- unfortunately, constraints are hard to visualize
  - kind of like gravity or electromagnetism
  - observed only by their effect on others
- and they are **voluntary**
  - there are no architecture police, but there are many architecture critics

Constraints induce **architectural properties**

- both desirable and undesirable properties
  - a.k.a., software qualities and design trade-offs

# Styles of Architectural Design

Design at the right level of abstraction

- Styles help architects communicate architecture
- Architecture determines potential system properties
- Implementation determines actual system properties

Sometimes known by other names

- Systems Engineering (when it includes software)
- Architectural Patterns (styles with common recipes)

Just because it's called architecture ...

# Agenda

Web retrospective

Understanding Architecture

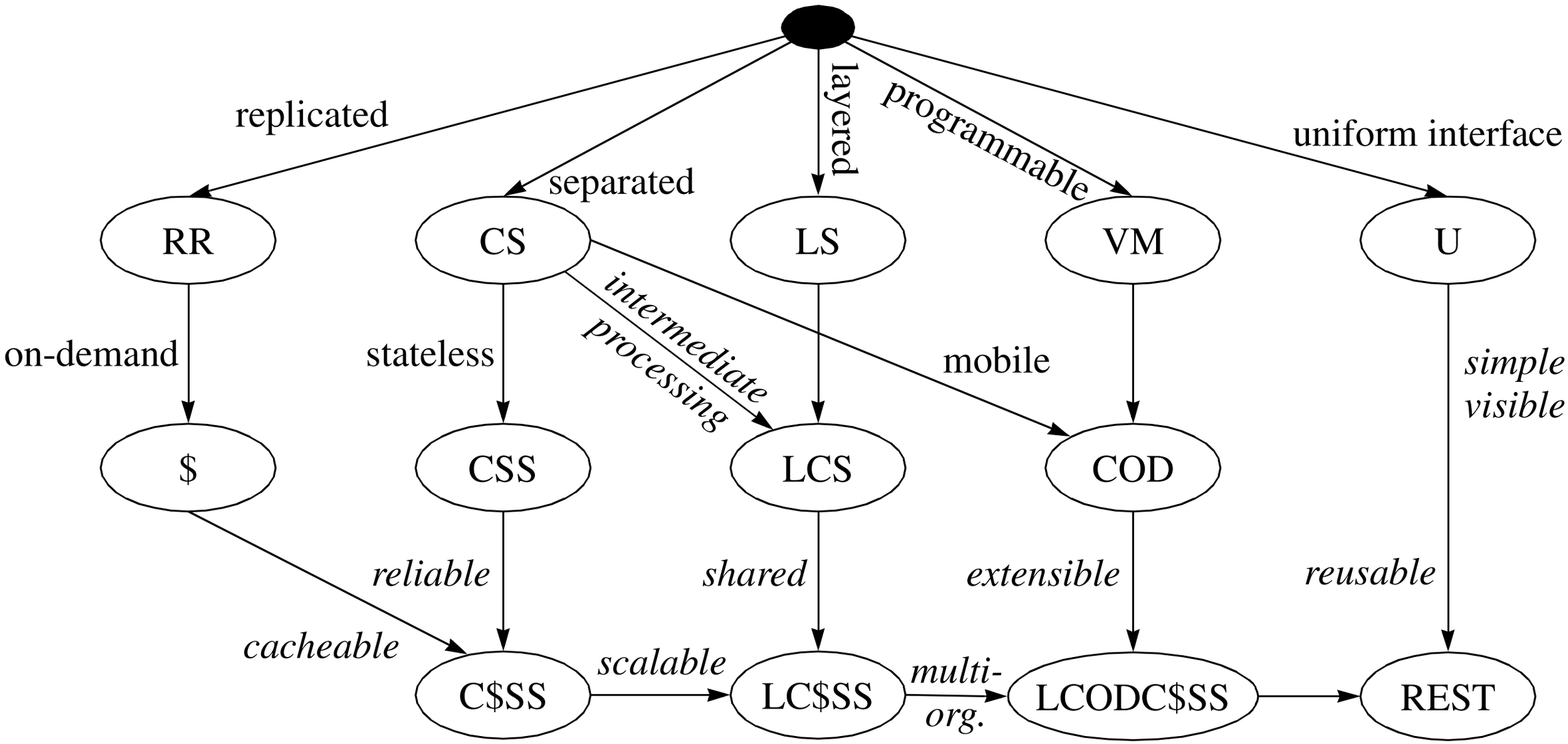
What is REST?

Why REST?

REST at Day

Q & A

# REST on a slide

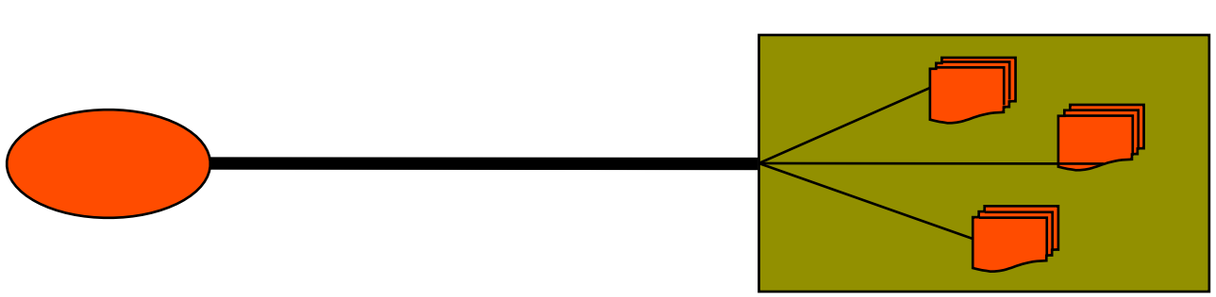


# Style = nil

Starting from a condition of no constraints…

# Style += Client/Server

Apply separation of concerns: Client-Server



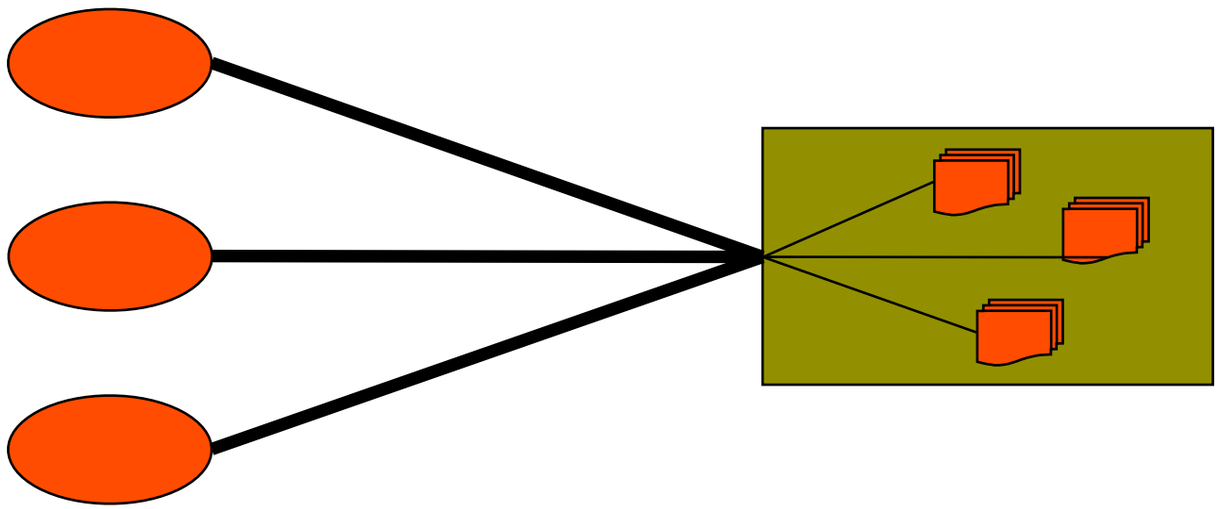
improves UI portability

simplifies server

enables multiple organizational domains

# Style += Stateless

Constrain interaction to be stateless...



degrades efficiency

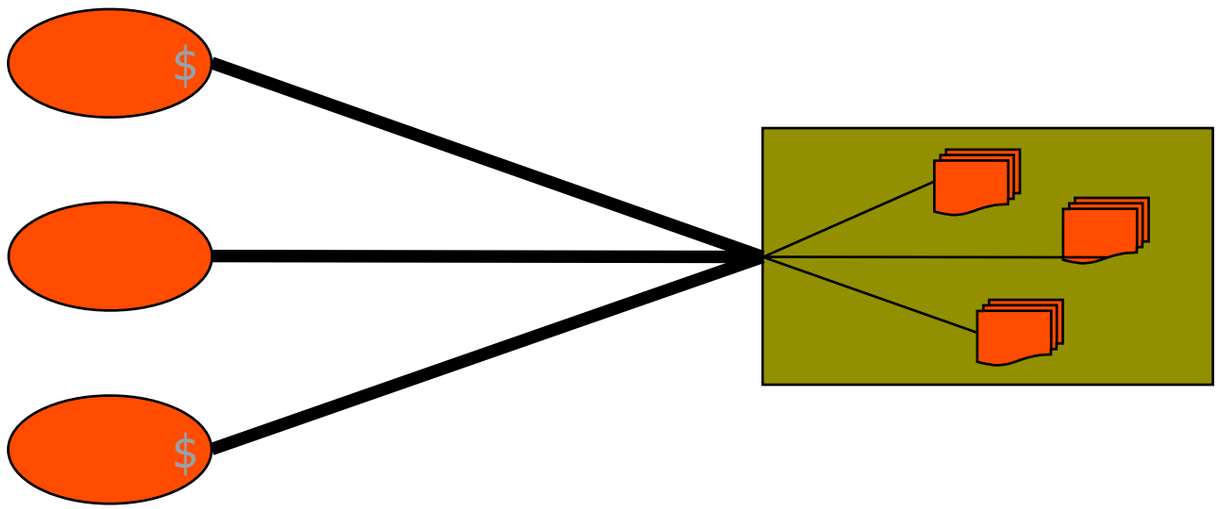
simplifies server

improves scalability

improves reliability

# Style += Caching

Add optional non-shared caching




degrades reliability

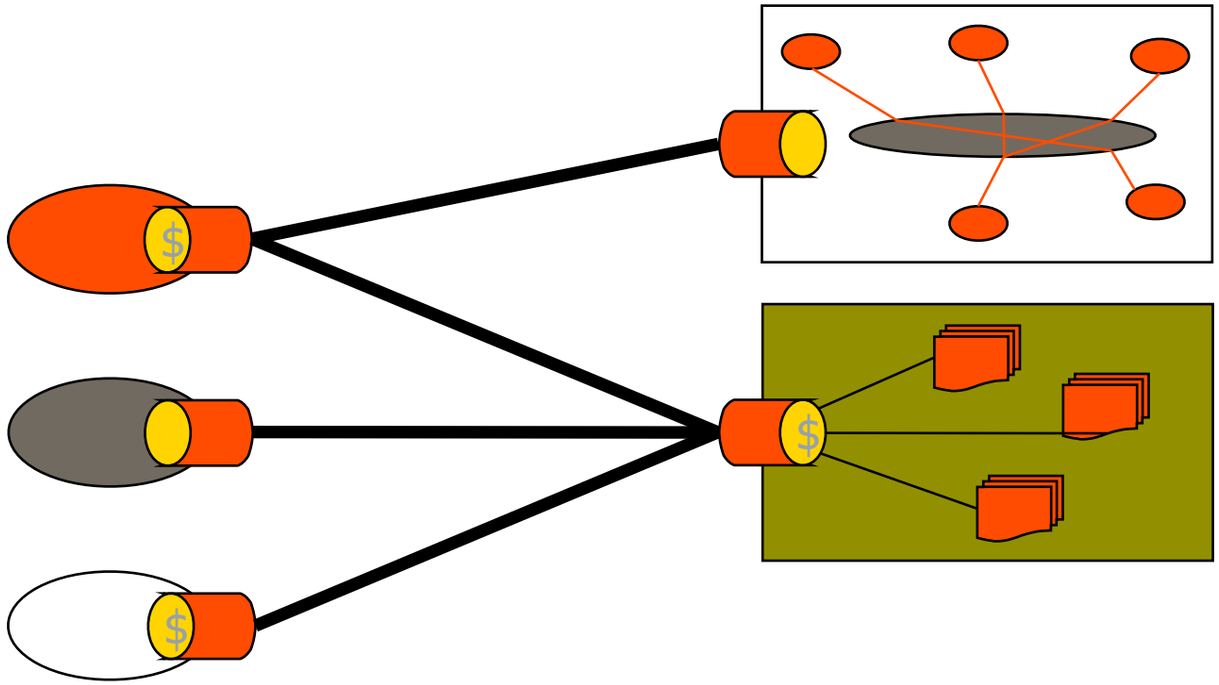
reduces average latency

improves efficiency

improves scalability

# Style += Uniform Interface

Apply generality: uniform interface constraint



improves visibility

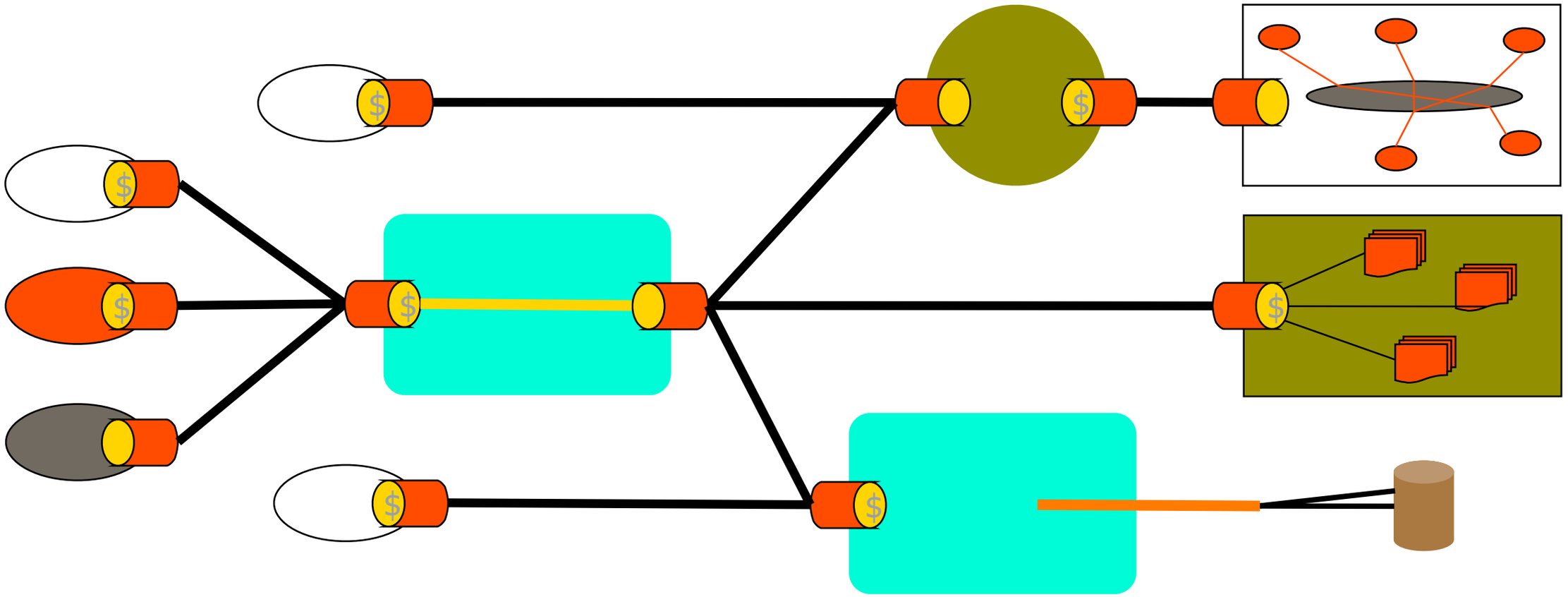
degrades efficiency

independent evolvability

decouples implementation

# Style += Layered System

## Apply info hiding: layered system constraints



adds latency

shared caching

legacy encapsulation

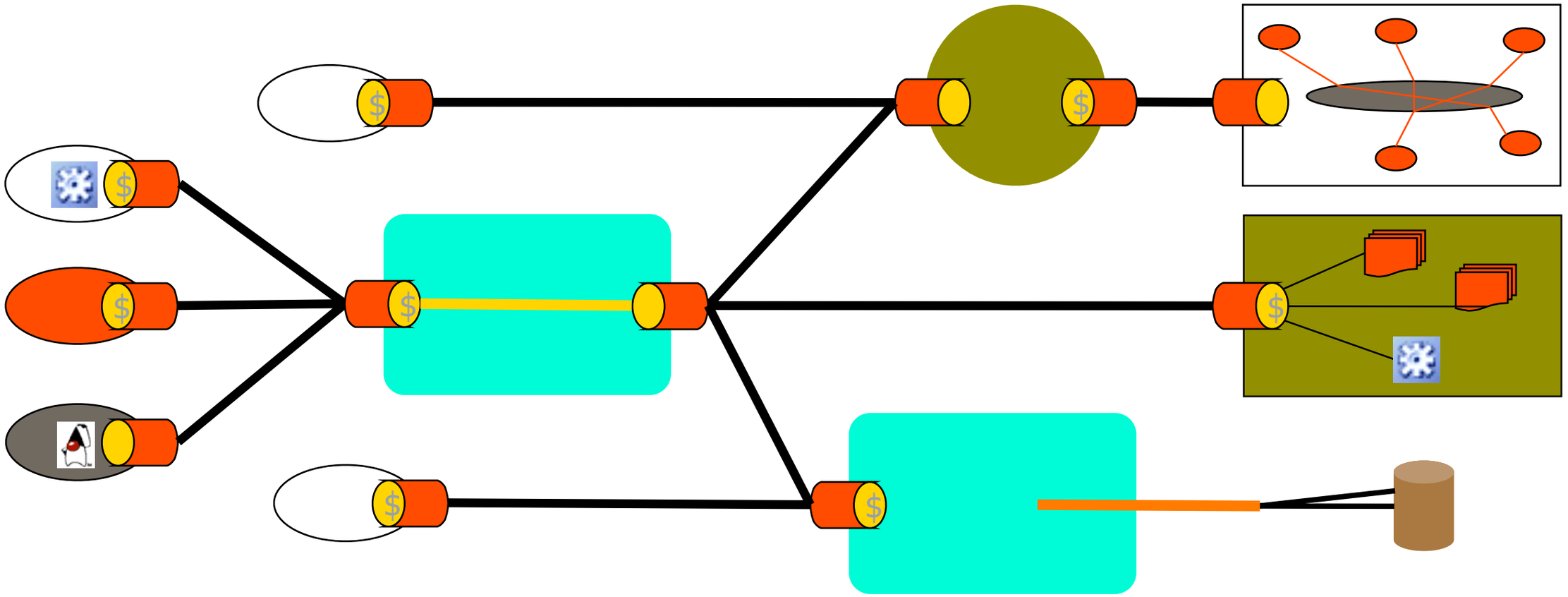
simplifies clients

improves scalability

load balancing

# REST Style

Finally, allow code-on-demand (applets/js)



simplifies clients

improves extensibility

reduces visibility

# REST Uniform Interface

All important resources are identified by one (uniform) resource identifier mechanism

- simple, visible, reusable, stateless communication

Access methods (actions) mean the same for all resources (universal semantics)

- layered system, cacheable, and shared caches

Resources are manipulated through the exchange of representations

- simple, visible, reusable, cacheable, and stateless communication

Exchanged as self-descriptive messages

- layered system, cacheable, and shared caches

# REST Uniform Interface

Hypertext as the engine of application state

- A successful response indicates (or contains) a current **representation** of the state of the identified resource; the **resource remains hidden** behind the interface.
- Some **representations contain links** to potential next application states, including **direction on how to transition** to those states when a transition is selected.
- Each steady-state (Web page) embodies the current **application state**
  - simple, visible, scalable, reliable, reusable, and cacheable
- All application state (not resource state) is kept on client
- All shared state (not session state) is kept on origin server

# Hypertext Clarification

## Hypertext has many (old) definitions

- "By 'hypertext,' I mean non-sequential writing — text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways" [Theodor H. Nelson]
- “Hypertext is a computer-supported medium for information in which many interlinked documents are displayed with their links on a high-resolution computer screen.” [Jeffrey Conklin]

## When I say Hypertext, I mean ...

- The simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.
- Hypertext does not need to be HTML on a browser
  - machines can follow links when they understand the data format and relationship types

# Hypertext Clarification

## Hypertext has many (old) definitions

- "By 'hypertext,' I mean non-sequential writing — text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways." [Theodor H. Nelson]
- “Hypertext is a computer-supported medium for information in which many interlinked documents are displayed with their links on a high-resolution computer screen.” [Jeffrey Conklin]

**Hypertext = non-linear documents**

## When I say Hypertext, I mean ...

- The simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.
- Hypertext does not need to be HTML on a browser
  - machines can follow links when they understand the data format and relationship types

# Hypertext Clarification

## Hypertext has many (old) definitions

- "By 'hypertext,' I mean non-sequential writing — text that branches and allows choices to the reader, best read at an interactive screen. As popularly understood, it is a series of text chunks connected by links which offer the reader different pathways." [Theodor H. Nelson]
- "Hypertext is a computer-supported medium for information in which many interlinked documents are displayed with a high degree of interaction." [Jeffrey Conklin]

**Hypertext = non-linear documents**

**Hypertext = selectable GUI controls**

## When I say Hypertext, I mean ...

- The simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.
- Hypertext does not need to be HTML on a browser
  - machines can follow links when they understand the data format and relationship types

# Hypertext Clarification

## Hypertext has many (old) definitions

- "By 'hypertext,' I mean non-sequential writing — text that branches and allows choices to the reader, best read at an interactive screen. As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways." [Theodor H. Nelson]
- "Hypertext is a computer-supported medium for information in which many interlinked documents are employed with multiple presentation components." [Jeffrey Conklin]

**Hypertext = non-linear documents**

**Hypertext = selectable GUI controls**

## When I say Hypertext, I mean ...

- The simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.
- Hypertext does not need to be HTML on a browser
  - machines can follow links when they understand the data format and relationship types

**Hypertext = data-guided controls**

# Agenda

Web retrospective

Understanding Architecture

What is REST?

Why REST?

REST at Day

Q & A

# Benefits of REST-based Architecture

## Maximizes reuse

- uniform resources having identifiers = Bigger WWW
- visibility results in serendipity

## Minimizes coupling to enable evolution

- uniform interface hides all implementation details
- hypertext allows late-binding of application control-flow
- gradual and fragmented change across organizations

## Eliminates partial failure conditions

- server failure does not befuddle client state
- shared state is recoverable as a resource

## Scales without bound

- services can be layered, clustered, and cached

# Benefits of REST-based Architecture

Simplifies

- hypertext is standardized (fewer UIs)

Simplifies

- identification is standardized (less communication)

Simplifies

- exchange protocols are standardized (fewer integrations)

Simplifies

- interactions are standardized (fewer semantics)

Simplifies

- data formats are standardized (fewer translations)

# Agenda

Web retrospective

Understanding Architecture

What is REST?

Why REST?

Relaxation

REST at Day

Q & A

# Industry Practice

## Meanwhile, in a parallel universe ...

- Monty Python's Architect Sketch
  - Microsoft was selling COM+/DCOM
  - IBM and friends were selling CORBA
  - Sun was selling RMI
  - W3C was developing XML
- Then SOAP was dropped on the shower floor as an Internet Draft
  - and quickly laughed out of the IETF
  - only to be picked up by IBM and renamed "Web Services"
- and REST became the only counter-argument to multi-billions in advertising

# Industry Reaction?


Not very constructive

- proponents labeled as RESTafarians
- arguments derided as a “religion”
- excused as “too simple for real services”

# Service-Oriented Architecture (SOA)

- a direct response to REST
- attempt at an architectural style for WS
  - without any constraints
- What is SOA?
  - Wardrobe, Musical Notes, or Legos?
  - http://www.youtube.com/profile_videos?user=richneckyogi

# Industry Acceptance


Something has changed ...

- People started to talk about the value of URIs (reusable resources)
- Google maps decided to encourage reuse (Mashups)
- O'Reilly began talking about Web 2.0
- Rails reminded people that frameworks can be simple

and REST(ful) became an industry buzzword

**Yikes!**

# Relaxation

Clearly, it's time to start messing with minds

- REST is not the only architectural style
- My dissertation is about Principled Design, not the one true architecture

What do constraints really mean?

- codify a design choice at the level of architecture
  - to induce certain (good) architectural properties
  - at the expense of certain (bad) trade-offs

What if we relax a given constraint?

- Is it really the end of the world?
- Should waka have its own style?

# What if: Non-Uniform Interface ?

If the interface would be resource-specific...

- URI is no longer sufficient for resource identification
  - lose benefit of URI exchange (assumed GET)
  - require resource description language
- Information becomes segregated by resource type
  - walled into gardens (loss of power laws / pagerank)
  - important information must be replicated
- Intermediaries cannot encapsulate services
  - unable to anticipate resource behavior
  - too complex to cache based on method semantics
- No more serendipity
  - mashups must be defined per interface
  - services become tightly coupled

# What if: Non-Uniform Interface ?

If the interface would be resource-specific...

- URI is no longer sufficient for resource identification
  - lose benefit of URI exchange (assumed GET)
  - require resource description language
- Information becomes segregated by resource type
  - walled into gardens (loss of power laws / pagerank)
  - important information must be replicated
- Intermediaries cannot encapsulate services
  - unable to anticipate resource behavior
  - too complex to cache based on method semantics
- No more serendipity
  - mashups must be defined per interface
  - services become tightly coupled

Why would anyone seriously suggest this would be good for the Web?

# What if: Relax client/server ?

## What if we let servers make requests?

- lose implementation simplicity
- potential for confusion with mixed-protocol intermediaries
  - unknown: does it impact session state?

## Trade-offs aren't as severe. Benefits?

- peer-to-peer applications
- shared cache mesh, triggered expiration

## Can we compensate for the trade-offs?

- Make message syntax more uniform
  - Limit server-initiated requests to same-connection
  - Make response messages truly asynchronous
  - Make it possible to ignore requests

# Conclusion

## Use your brains!

- don’t design-by-buzzword
- don’t believe everything you read
- always keep in mind that **change is inevitable**

## Use principled design

- identify desired architectural properties
- select proven architectural styles where appropriate
- constrain behavior to induce properties
- compensate for the design trade-offs

# Agenda

Web retrospective

Understanding Architecture

What is REST?

Why REST?

Relaxation

REST at Day

Q & A

# Vision

## 1 Everything is Content

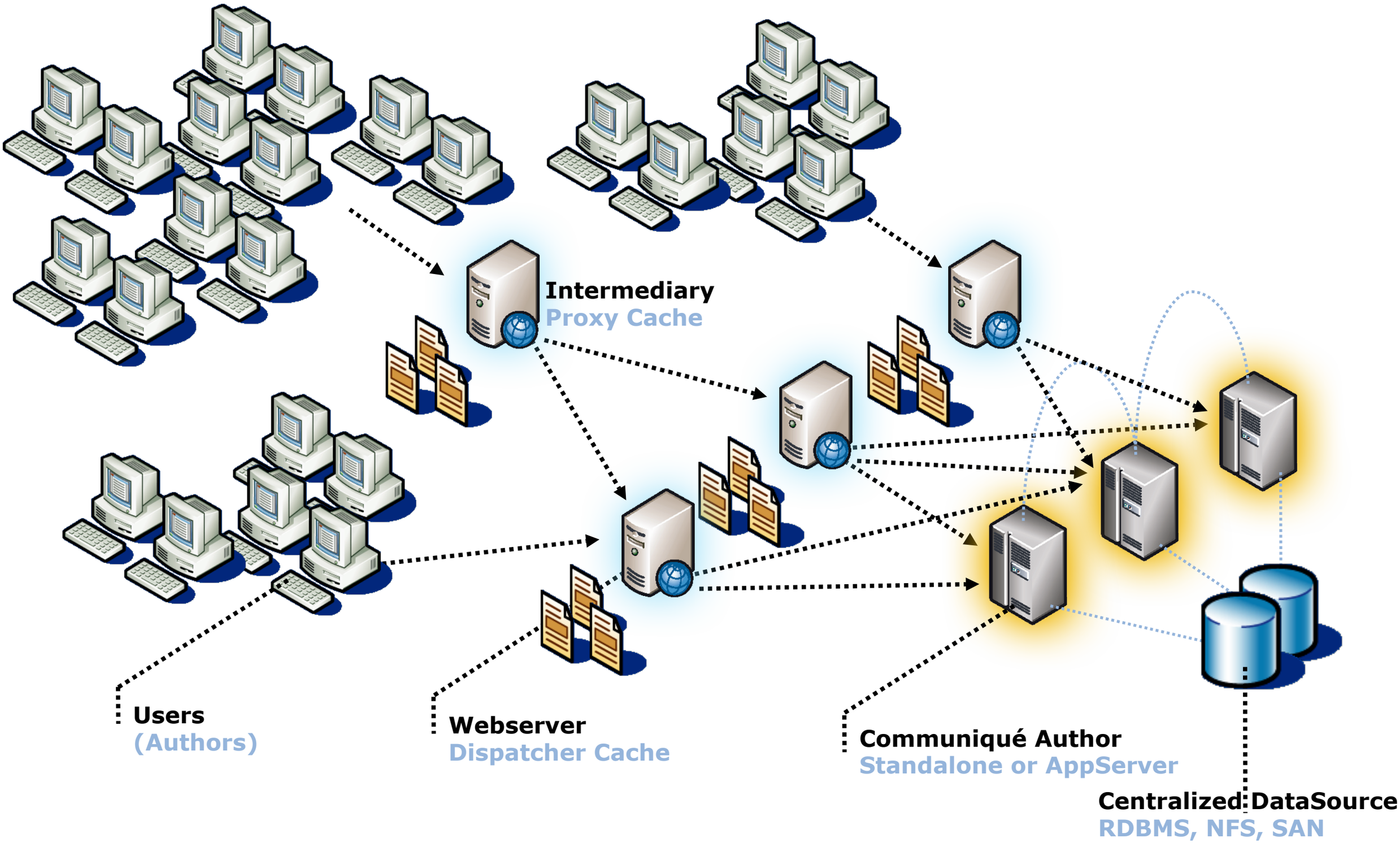
# Vision

# REST

## All important resources have uniform identifiers

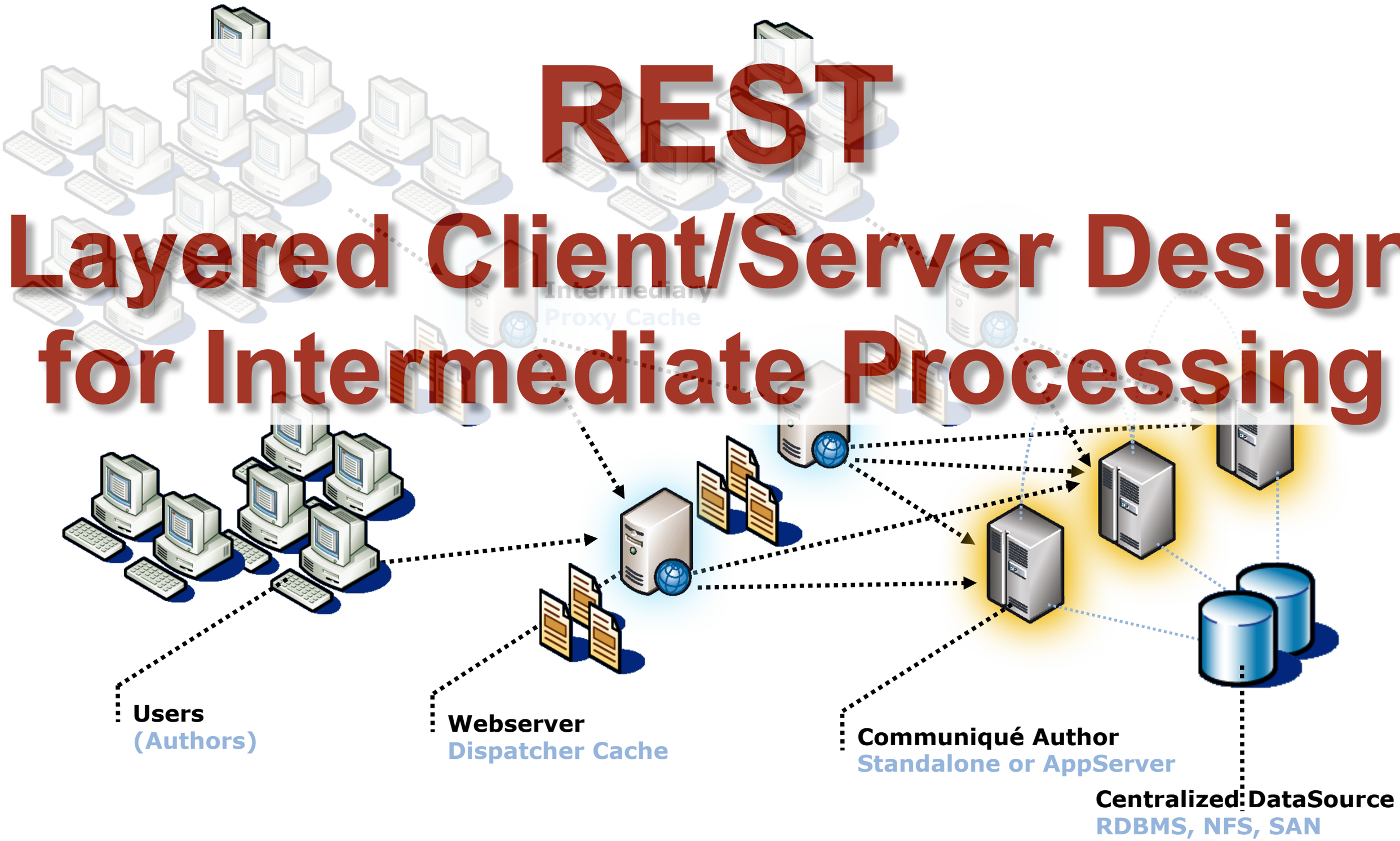
1. Everything is Content

# Intermediary and Cache Friendly

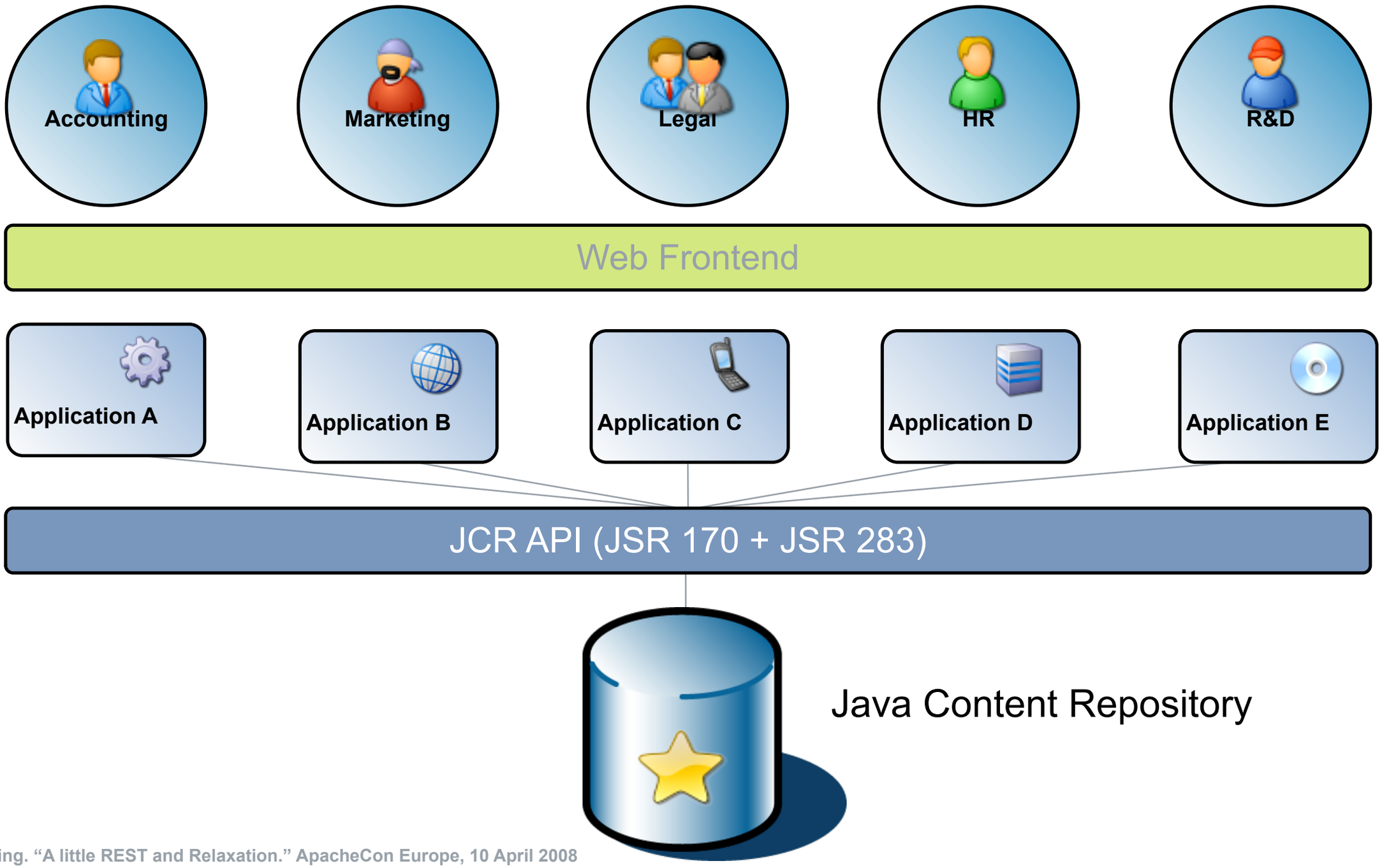


# Intermediary and Cache Friendly

# REST
# Layered Client/Server Design for Intermediate Processing



# Standards


Accounting
Marketing
Legal
HR
R&D
Web Frontend
Application A
Application B
Application C
Application D
Application E
JCR API (JSR 170 + JSR 283)
Java Content Repository

# Standards

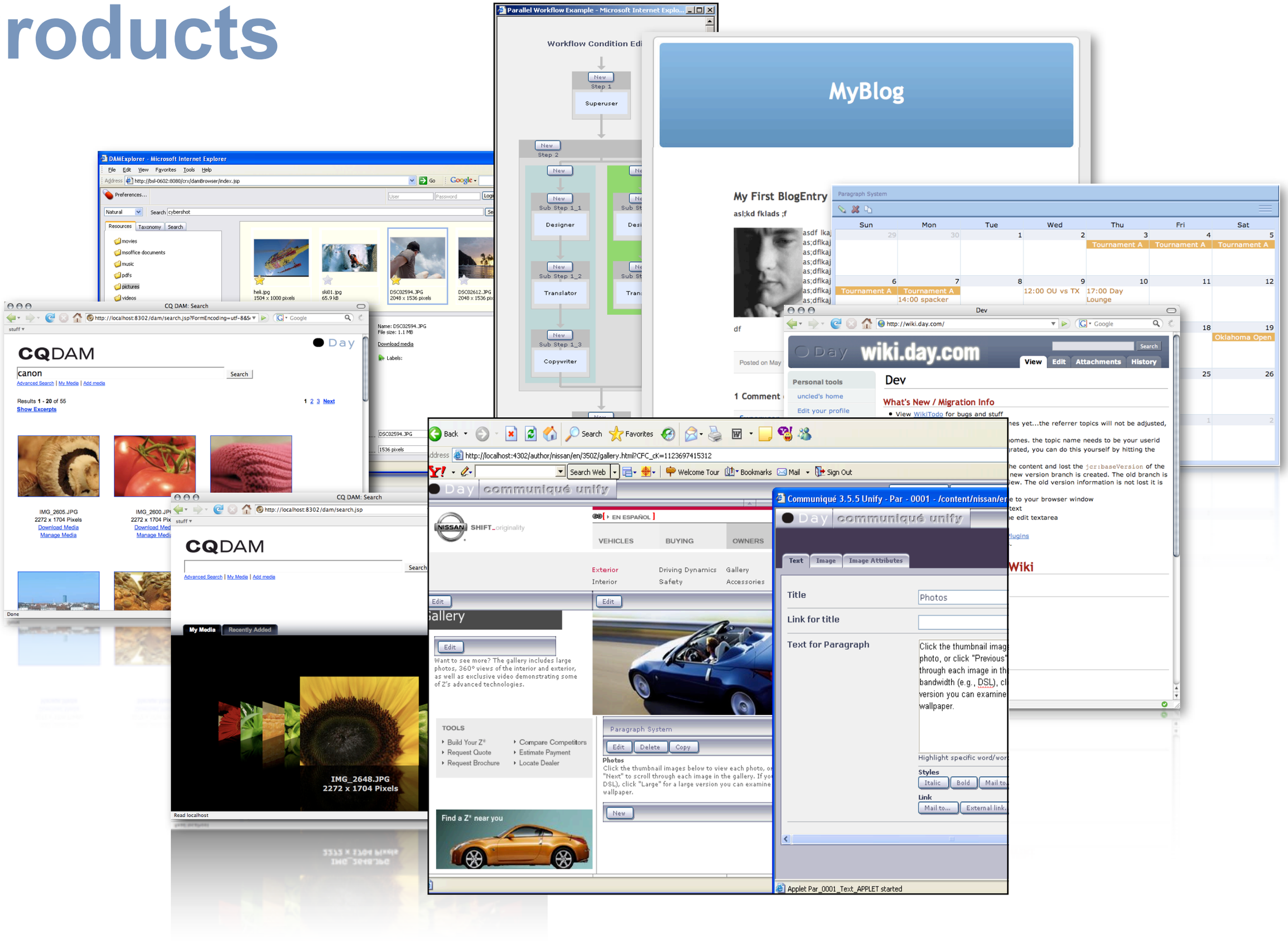
**REST**

**All resources**

**have uniform interface**

# Products

# Products


Parallel Workflow Example - Microsoft Internet Explo...
Workflow Condition Edi
New
Step 1
Superuser
New
Step 2
New
Sub Step 1_1
Designer
New
Sub Step 1_2
Translator
New
Sub Step 1_3
Copywriter
MyBlog
My First BlogEntry
asl;kd fklads ;f
df
Posted on May
1 Comment
Paragraph System
Sun Mon Tue Wed Thu Fri Sat
Tournament A
Tournament A
Tournament A
Tournament A
Tournament A
12:00 OU vs TX
17:00 Day
Lounge
14:00 spacker
Oklahoma Open
DAMExplorer - Microsoft Internet Explorer
Preferences...
Resources
Taxonomy
Search
movies
msoffice documents
music
pdfs
pictures
videos
heli.jpg
1504 x 1000 pixels
ski01.jpg
65.9 kB
DSC02594.JPG
2048 x 1536 pixels
DSC02612.JPG
2048 x 1536 pixels
Name: DSC02594.JPG
File size: 1.1 MB
Download media
Labels:
CQ DAM: Search
http://localhost:8302/dam/search.jsp?FormEncoding=utf-8&s
CQDAM
canon
Search
Advanced Search | My Media | Add media
Results 1 - 20 of 55
Show Excerpts
1 2 3 Next
IMG_2605.JPG
2272 x 1704 Pixels
Download Media
Manage Media
IMG_2600.JP
2272 x 1704 Pix
Download Med
Manage Medi
CQ DAM: Search
http://localhost:8302/dam/search.jsp
CQDAM
Search
Advanced Search | My Media | Add media
My Media
Recently Added
IMG_2648.JPG
2272 x 1704 Pixels
Read localhost
Dev
http://wiki.day.com/
wiki.day.com
Search
View
Edit
Attachments
History
Personal tools
uncled's home
Edit your profile
Dev
What's New / Migration Info
View WikiTodo for bugs and stuff
Wiki
Back
Search
Favorites
http://localhost:4302/author/nissan/en/350Z/gallery.html?CFC_cK=1123697415312
Search Web
Welcome Tour
Bookmarks
Mail
Sign Out
communiqué unify
EN ESPAÑOL
NISSAN
SHIFT_originality
VEHICLES
BUYING
OWNERS
Exterior
Interior
Driving Dynamics
Safety
Gallery
Accessories
Edit
Want to see more? The gallery includes large photos, 360° views of the interior and exterior, as well as exclusive video demonstrating some of Z's advanced technologies.
TOOLS
Build Your Z*
Request Quote
Request Brochure
Compare Competitors
Estimate Payment
Locate Dealer
Find a Z* near you
Paragraph System
Edit
Delete
Copy
Photos
New
Communiqué 3.5.5 Unify - Par - 0001 - /content/nissan/en
Text
Image
Image Attributes
Title
Photos
Link for title
Text for Paragraph
Styles
Italic
Bold
Link
Mail to...
External link
Applet Par_0001_Text_APPLET started

# Products


# REST

# Hypertext is the Engine of Application State

# Agenda

Web retrospective

Understanding Architecture

What is REST?

Why REST?

Relaxation

REST at Day

Q & A