Computational REST:
A new model for Decentralized, Internet-Scale Applications

Justin R. Erenkrantz
Final Defense
September 3, 2009

This material is based upon work supported by the National Science Foundation under Grant Numbers 0438996 and 0820222. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.
Runaway success of the Web

Netcraft
239 million sites
30% are “active”
July 2009

The Nielsen Company
Average home user views
1,591 pages/month
May 2009

http://www.worldwidewebsize.com/: 21.8 billion pages
How did we get here?

- In the early-to-mid-90s, the Web faced a crossroads: how could the Web scale?
- Software architecture, in particular, the REpresentation State Transfer (REST) style [Fielding, 2000] guided crucial reformations introduced in HTTP/1.1
- Permitted the superscaling of the Web
Puzzling web apps: Google Maps
Research Question

What happens when dynamism is introduced into the Web?

Dynamism: Phenomena that must be explained as a manifestation of change, whether through interpretation or alteration of the interpreter.
Dramatis personae of the Web

- REST: architectural style
- HTTP/1.1: protocol governed by REST
- URI: naming convention (http://...)
- Apache HTTP Server: origin server
- Squid: gateway and proxy
- Firefox, Safari, Internet Explorer: user agent
REpresentation State Transfer

- Started in mid ‘90s; captured in Fielding’s dissertation (2000) and TOIT (2002)
- Guided the reformations introduced in HTTP/1.1 and URI specifications
- Designed for Internet-scale distributed hypermedia
- Few (if any) clarifications since then...
REST Axioms

1. The key abstraction of information is a resource, named by an URL.

2. The representation of a resource is a sequence of bytes, plus representation metadata to describe those bytes.

3. All interactions are context-free.

4. Only a few primitive operations are available.

5. Idempotent operations and representation metadata are encouraged in support of caching.

6. The presence of intermediaries is promoted.
Approach

- Examine evolution of key infrastructure applications
- Identify the root causes of dissonance in Web applications which are not fully explained by REST
- Introduce a named set of architectural principles (style) that provide more applicable guidance for web applications
- Assess the effectiveness of these principles by:
  - characterizing the dissonance seen in existing systems against new style
  - creating a framework based on the new style that demonstrates novel web applications
Insight: Apache modules & Browser plugins

With little explicit coordination among developers during this period, critical web infrastructure applications evolved rapidly to support dynamism - both architectural and content-focused.
Even for knowledgeable practitioners of REST, REST, in isolation, does not provide enough design guidance for architects to understand why applications fall into architectural dissonance.
Insight: Web Services

Due to implementation deficiencies, SOAP-based Web Services (and its sibling Service Oriented Architectures) are incapable of realizing the promise of fine-grained, composable services without fundamentally violating the REST axioms that permitted the web to scale.
Insight: RESTful Services

Even for hypermedia-related services - such as document management (via WebDAV) - the construction of “RESTful” services has produced inconsistent and incomplete interfaces. Non-content related services have proven even more difficult to create.
In some emerging Web applications, computation has appeared as a first-class concept.
Mobile code

- Mobile agents did not succeed, but code on demand and remote evaluation have found niches [Carzaniga, ICSE 2007]

- Remote evaluation via exchange of expressions: Scheme in Tubes [Halls, 1997]
Due to the improvements in the JavaScript engines, the modern browser is far more powerful and capable today than it was in the mid-’90s. Distributed mobile code systems can be built on top of existing Web infrastructure.
Hypothesis

We can construct a set of axioms that more precisely and effectively guide the architecture of Web applications. These axioms can also further facilitate new and fundamentally different classes of applications to be deployed on the Web than the originally intended distributed hypermedia applications.
Computations and their expressions are explicitly named.

**Services** may be exposed through a variety of URLs which offer perspectives on the same computation; interfaces may offer complementary supervisory functionality such as debugging or management.

Functions may be added to or removed from the binding environment over time or their semantics may change.

Computational loci may be **stateful** (and thus permit indirect interactions between computations), but must also support **stateless** computations.

Potentially autonomous **computations** exchange and maintain state; A rich set of stateful relationships exist among a set of distinct URLs.

The computation is **transparent** and can be inspected, routed, and cached.

The **migration** of the computation to be physically closer to the data store is supported thereby reducing the impact of network **latency**.
CREST Axioms

1. A resource is a locus of computations, named by an URL.

2. The representation of a computation is an expression plus metadata to describe the expression.

3. All computations are context-free.

4. Only a few primitive operations are always available, but additional per-resource and per-computation operations are also encouraged.

5. The presence of intermediaries is promoted.
Computations and their expressions are explicitly named. (CA1, CA2)

**Services** may be exposed through a variety of URLs which offer perspectives on the same computation. (CA1); interfaces may offer complementary supervisory functionality such as debugging or management. (CA4)

Functions may be added to or removed from the binding environment over time or their semantics may change. (CA4)

Computational loci may be **stateful** (and thus permit indirect interactions between computations), but must also support **stateless** computations. (CA3)

Potentially autonomous **computations** exchange and maintain state (CA2, CA3); A rich set of stateful relationships exist among a set of distinct URLs. (CA1)

The computation is **transparent** and can be inspected, routed, and cached. (CA5)

The **migration** of the computation to be physically closer to the data store is supported thereby reducing the impact of network **latency**. (CA2)
Validation

- Characterizing the dissonance seen in existing systems against new style
- Creating a framework based on the new style that demonstrates novel applications
Dissonance redux: AJAX

- Migration and latency: Moving computation from server to client results in visually-rich low-latency applications.
- Mashups: Goggles, AP News feeds, etc.
- CREST also predicts as yet unseen forms of mashups.
New types of mashups

- Derived mashups: Source of a mashup is a mashup itself; combination happens on an intermediary rather than a browser
- Higher-order mashups: a mashup that accepts one or more mashups as input and outputs a mashup itself
Novel applications on the Web

CREST can serve as the foundation for new classes of decentralized, Internet-based applications.
Demo

Thanks to Michael Gorlick, Yongjie Zheng, and Alegria Baquero.
CREST Peers

Weak Peers

Exemplary Peers

Dojo Framework
JavaScript Interpreter
Browser

Sham (HTTP/1.1 server)
CREST (weak) computations
Imposter (HTTP/1.1 client)

CREST (weak) computations
C-1
C-2
C-3
C-4
C-5
C-6
C-7

Apache HC (HTTP components)
SISC Scheme Interpreter
Apache Abdera (ATOM syndication)
Java Virtual Machine
CREST Computations

Weak Peer

Exemplary Peer
Demo (redux)
Demo FAQs

- Isn’t this Google Wave?
  - Wave is just a shared XML document.
- Isn’t this web services?
  - Yes, but far more powerful than SOA.
  - Composability is free.
  - Nano-services can be installed.
Research Question redux

What happens when dynamism is introduced into the Web?

The underlying architecture of the Web shifts, from a focus on the exchange of static content to the exchange of active computations.
Future Work

- Gorlick: Streaming state kinematics
- Recombinant services
- Smart (power) grid, smart cargo, etc.
- Bring framework to Apache
Contributions

- Analysis of the essential architectural decisions of the web, followed by generalization, opens up an entirely new space of decentralized, Internet-based applications

- Recasting the web as a mechanism for computational exchange instead of content exchange

- A new architectural style to support this recasting (CREST)

- Demonstrating how CREST better explains dissonance

- A framework for building applications backed by CREST
Questions?