REST – Theory vs Practice

Subbu Allamaraju
Mike Amundsen

About the Speakers

- Subbu Allamaraju
  - Architect, Yahoo!
  - Web services standards/practices at Yahoo!
  - Built web portals, and web/WS services (SOAP kind) frameworks (BEA Systems)
- Mike Amundsen
  - Independent Consultant
  - .NET
  - Long-time RESTafarian

RESTful Web Services Cookbook

O'Reilly, March 2010

Objectives

- REST is set of **constraints** and not **rules**
- Knowingly relax constraints
- Work with the plumbing
- Apply sound software engineering

REST as Explained

1. Identify resources
2. Give a URI to every resource
3. Design representations for resources
4. Operate using a uniform interface
5. Use hypermedia as the engine of application state

Address Book 1.0

Build a RESTful address book

```
GET /user/subbu/address/{id}
Host: ex.org

200 OK ...
```

```
PUT /user/subbu/address/{id}
Host: ex.org
If-Match: xyz

200 OK ...
```

```
DELETE /user/subbu/address/{id}
Host: ex.org
If-Match: xyz

200 OK ...
```

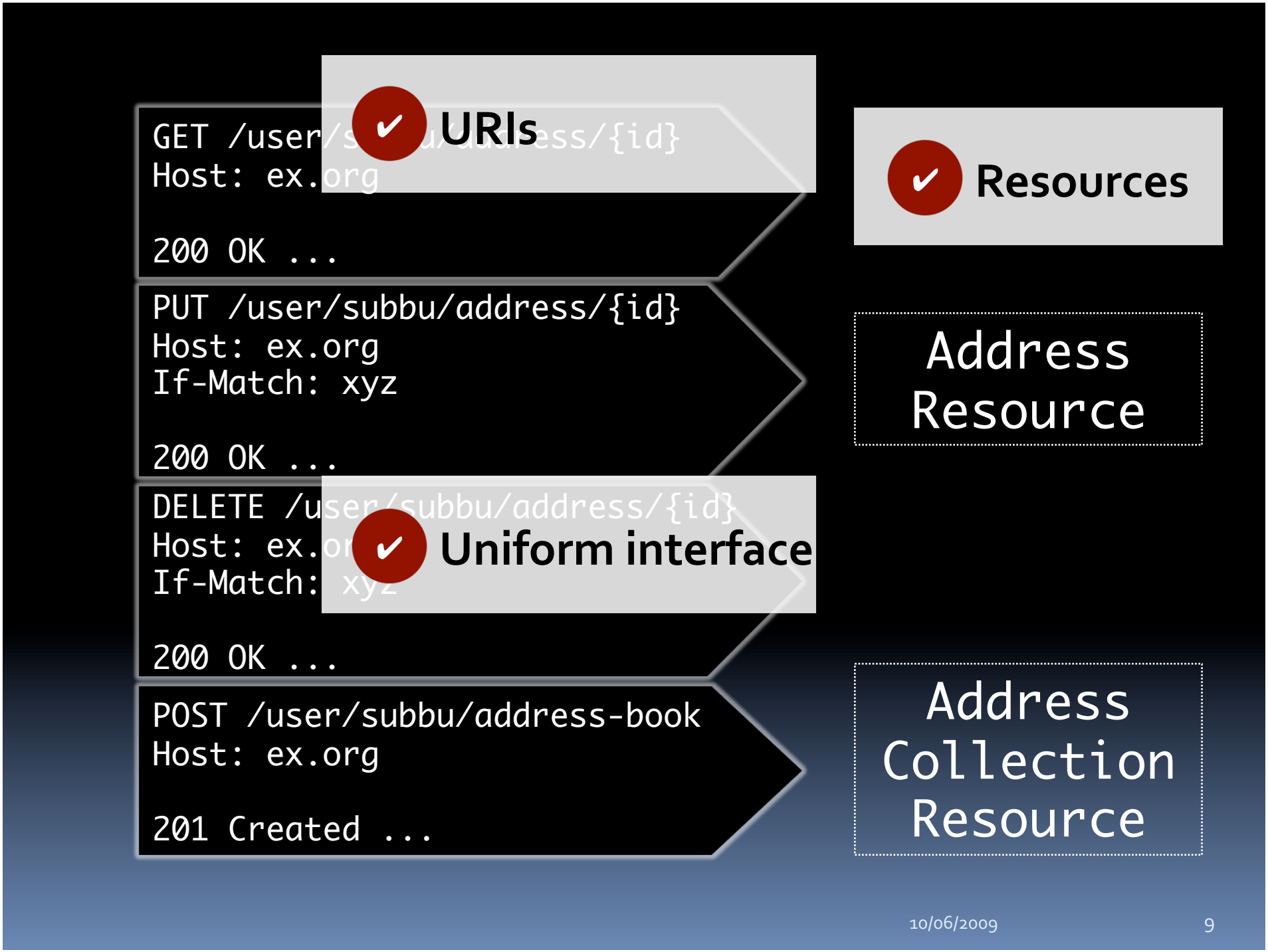
```
POST /user/subbu/address-book
Host: ex.org

201 Created ...
```

Address Resource

Address Collection Resource

```
<address>
  <link rel="http://ex.org/rel/person"
    href="http://ex.org/mike"/>
  <street>...</street>
  <city>...</city>
  ...
</address>
```

```
<address-book>
  <link rel="http://ex.org/rel/owner"
    href="http://ex.org/subbu"/>
  <link rel="next"
    href="http://ex.org/subbu/address-book?p=2/"/>
  <address>...</address>
  <address>...</address>
  ...
</address-book>
```

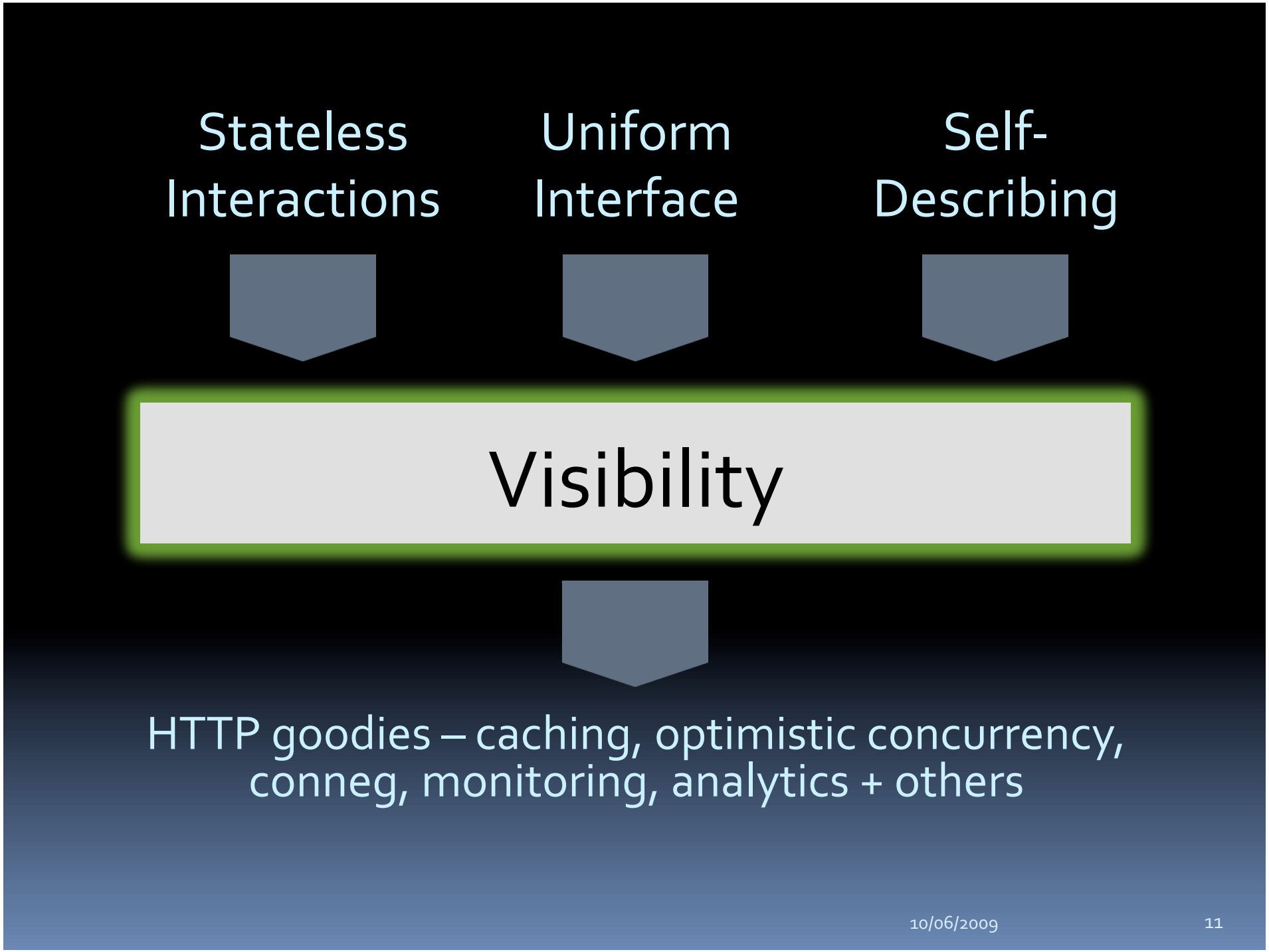

URIs
GET /user/subbu/address/{id}
Host: ex.org
200 OK ...
Resources
PUT /user/subbu/address/{id}
Host: ex.org
If-Match: xyz
200 OK ...
Address Resource
DELETE /user/subbu/address/{id}
Host: ex.org
If-Match: xyz
200 OK ...
Uniform interface
POST /user/subbu/address-book
Host: ex.org
201 Created ...
Address Collection Resource

4 Representations

```
<address>
  <link rel="http://ex.org/rel/person"
    href="http://ex.org/subbu"/>
  <street>...</street>
  <city>...</city>
  ...
</address>
```

5 Application state

```
<address-book>
  <link rel="self"
    href="http://ex.org/subbu/address-book"/>
  <link rel="next"
    href="http://ex.org/subbu/address-book?p=2/"/>
  <address>...</address>
  <address>...</address>
  ...
</address-book>
```


Stateless Interactions
Uniform Interface
Self-Describing
Visibility
HTTP goodies – caching, optimistic concurrency, conneg, monitoring, analytics + others

Address Book 1.1

[UC] Support address book sync for mobile users

GET /user/subbu/address-book

Address Collection Resource

For each address in the collection

Compare local copy

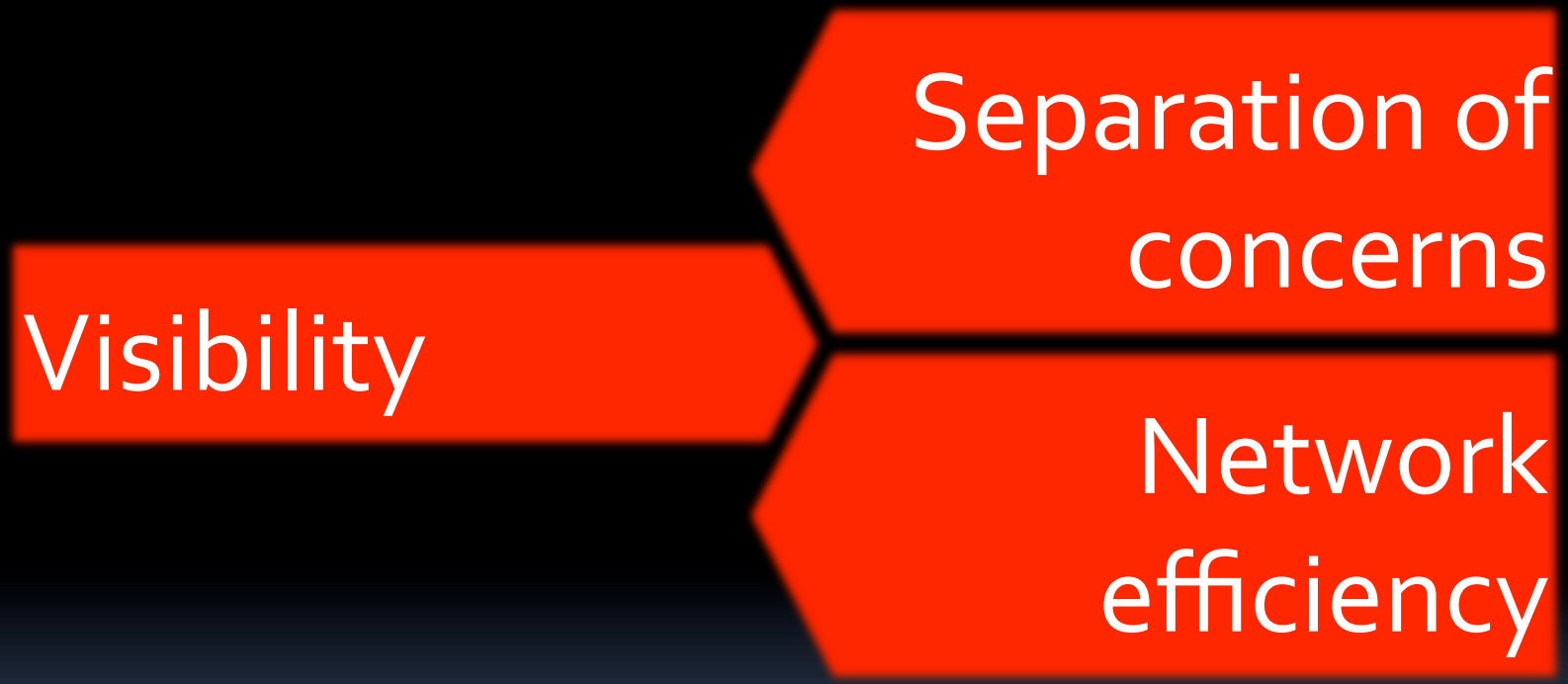
PUT if different

DELETE if missing

Address Resource

POST if new

Address Collection Resource


Visibility
Separation of concerns
Network efficiency

Better Merge

```
POST /user/subbu/address-book/merge
Host: ex.org
Content-Length: xxx
Content-Type: application/xml;charset=UTF-8

<address-book>
  <address>...</address>
  <address>...</address>
  ...
</address-book>
```

```
303 See Other
Location: http://ex.org/user/subbu/address-book
```

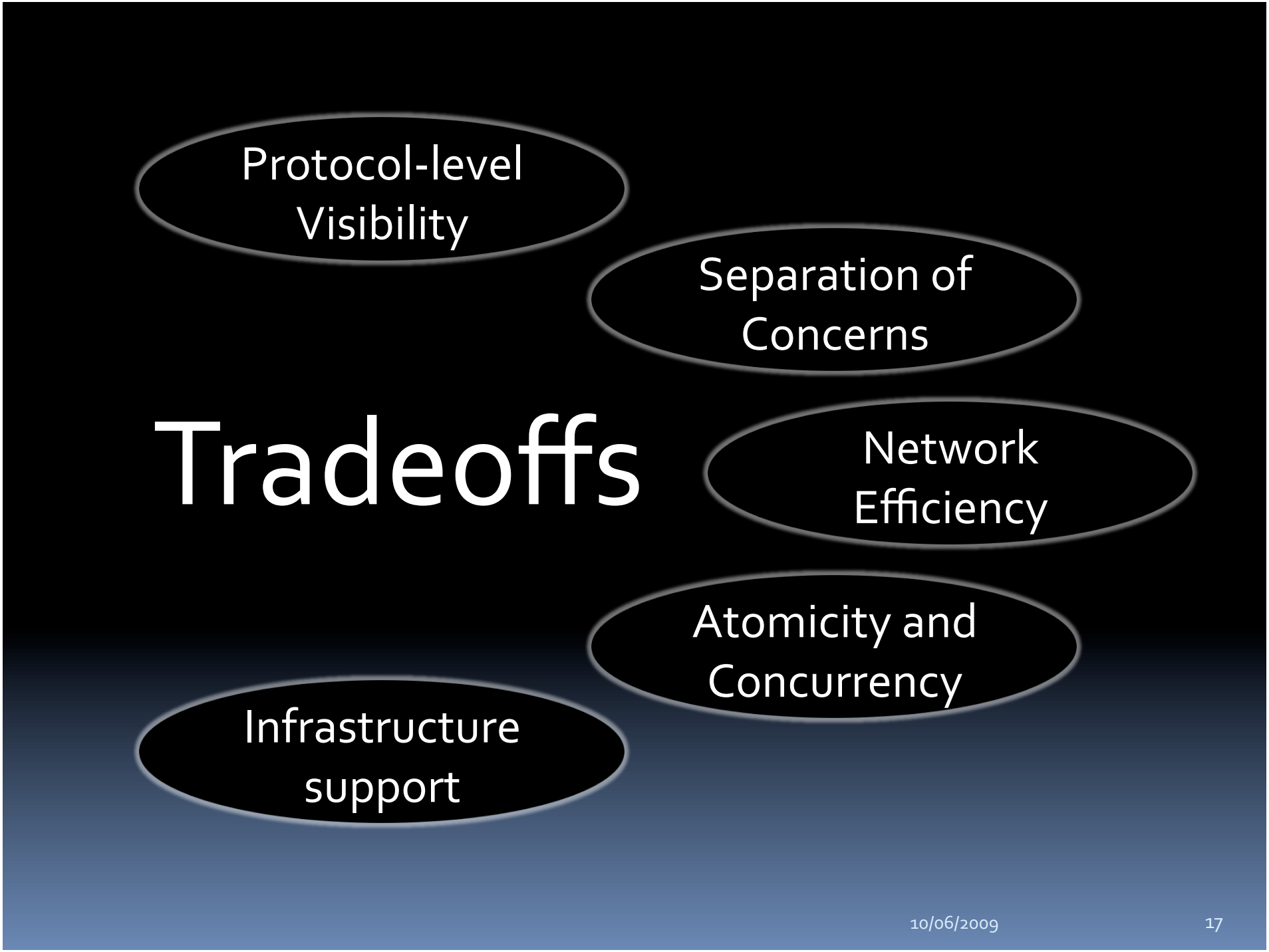
```
POST /user/subbu/address-book/merge
Host: ex.org
Content-Type: application/xml;charset=UTF-8
<address-book>
  <address>...</address>
  <address>...</address>
  ...
</address-book>
```

? **Reduced visibility**

✔ **Better separation of concerns**

✔ **Efficient network use**

```
303 See Other
Location: http://ex.org/user/subbu/address-book
```


Protocol-level Visibility
Separation of Concerns
Tradeoffs
Network Efficiency
Atomicity and Concurrency
Infrastructure support

Take a Step Back

1. Everything at the end of a URI is a resource

Some "things"

"person", "address book"

and some non-"things"

"merge address book", "reserve",
"cancel", "compute distance",
"reimage the virtual machine"

2. Use POST when in doubt

GET	Safe	+ Idempotent
PUT	Unsafe	+ Idempotent
DELETE	Unsafe	+ Idempotent
POST	Unsafe	+ Non-Idempotent

All bets are off with POST

POST limits damage

3. Don't tunnel using POST

```
POST /address-book
```

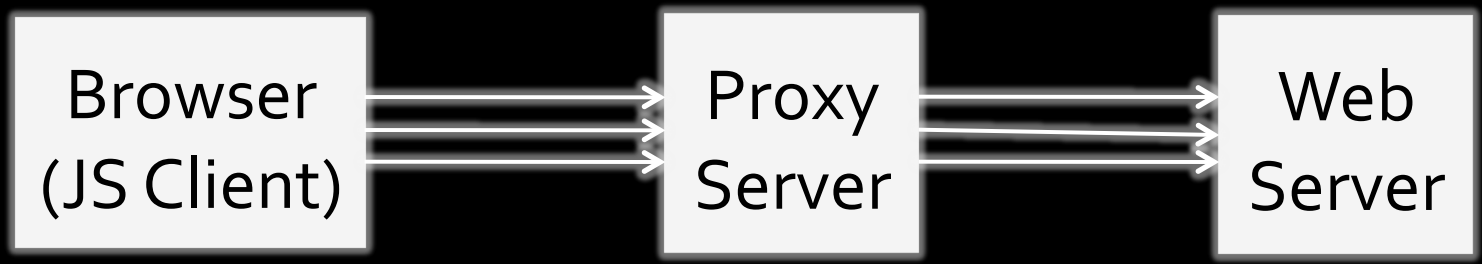
Merge an address book?

Fix duplicates?

Or something else?

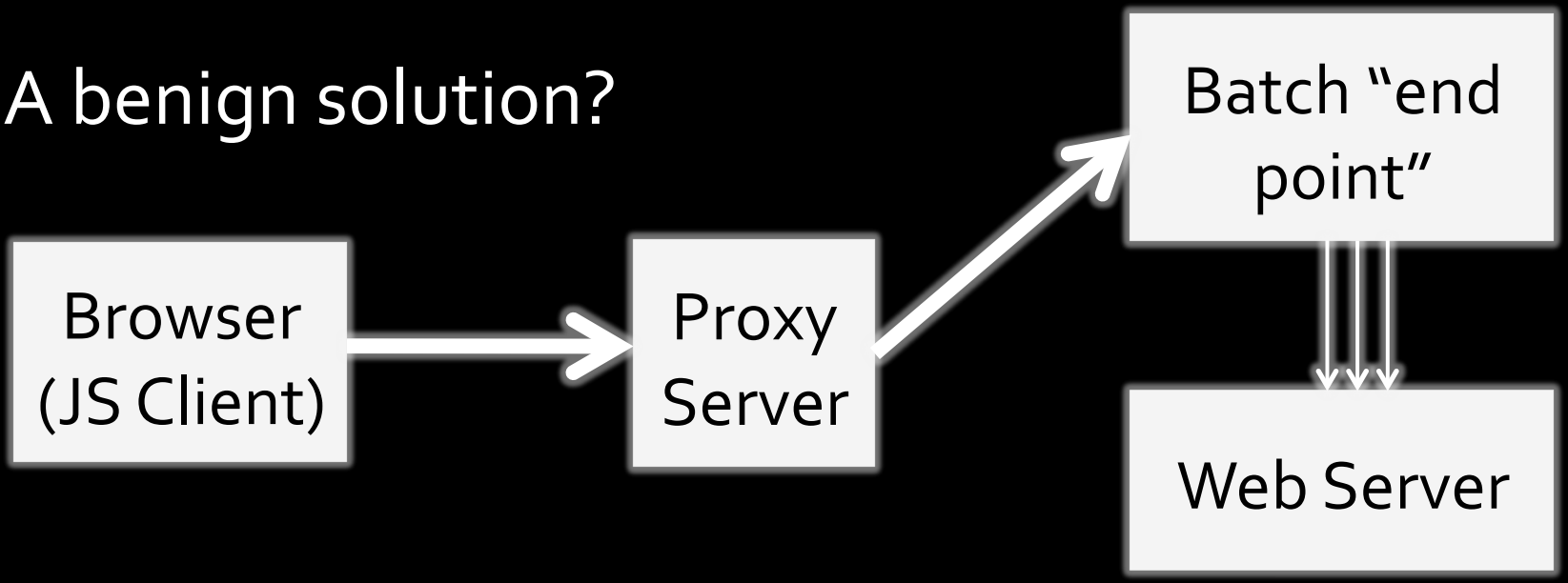
Tunneling = Back to dark ages

This is not a “pedantic” point of view

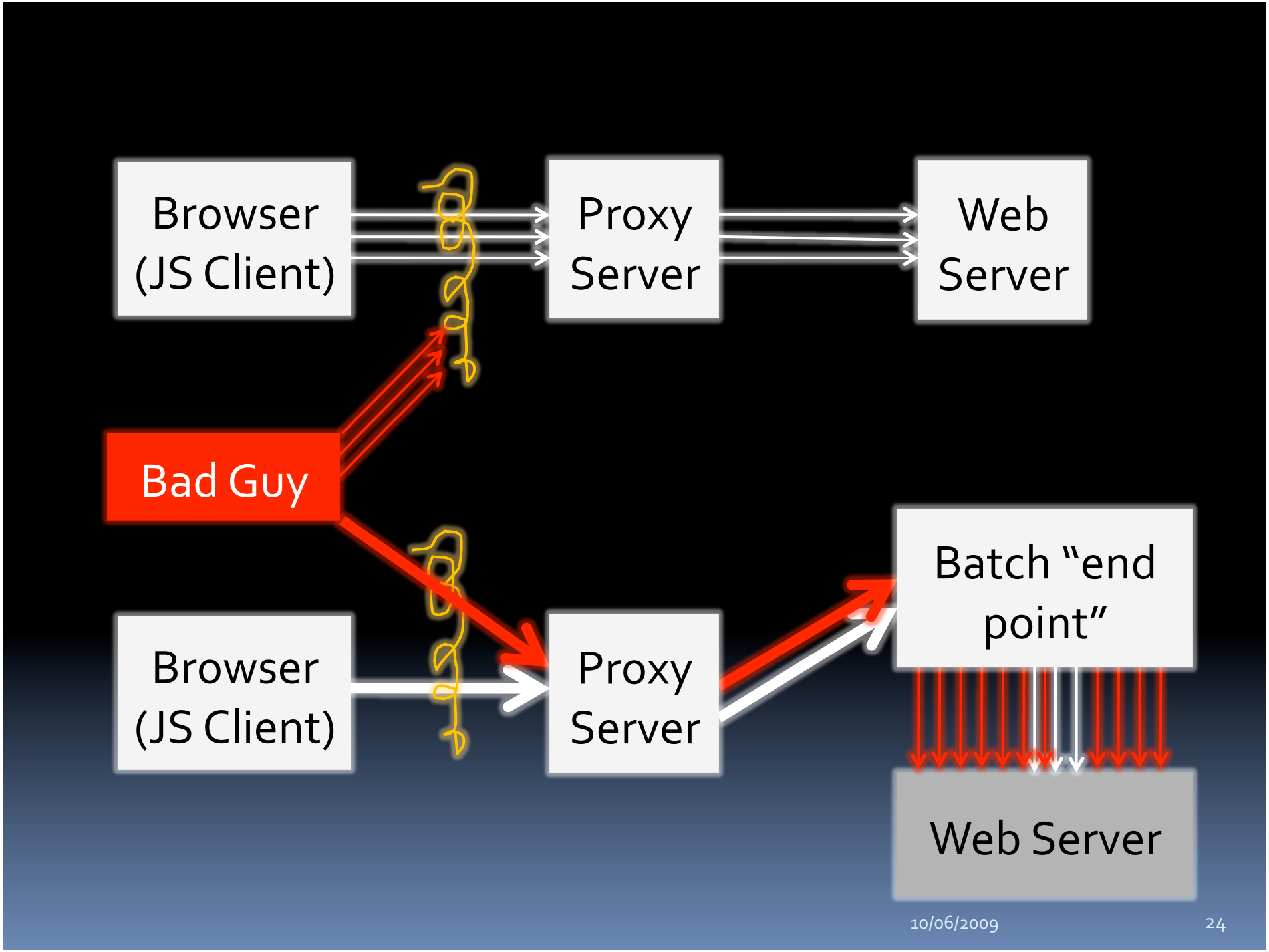


Dev: “We should find a way to make REST 'faster' and/or provide batching support”

A benign solution?




```
POST /batch
Host: ex.org
Content-Length: xxx
Content-Type: application/xml;charset=UTF-8

<batch>
  <request method="PUT" uri="/addr">...</request>
  <request method="GET" uri="/poi">...</request>
  <request method="GET" uri="/deals">...</request>
</batch>
```


Browser
(JS Client)
Proxy
Server
Web
Server
Bad Guy
Browser
(JS Client)
Proxy
Server
Batch "end
point"
Web Server

```
POST /batch
Host: ex.org
Content-Length: xxx
Content-Type: application/xml;charset=UTF-8

<batch>...</batch>
```

Create application specific resources with distinct URIs

```
POST /updateAddressGetPoiDeals
Host: ex.org
Content-Length: xxx
Content-Type: application/xml;charset=UTF-8

<address>
  ...
</address>
```

4. Be creative with URIs

Fixed and known URIs (Cool URIs)

```
http://ex.org/user/subbu
```

Resources with many URIs

```
http://ex.org/user/1234/profile;t=3231231dasd
http://ex.org/user/1234/profile;t=3da8432stgs
```

Ephemeral URIs (Uncool URIs)

```
http://ex.org/act/4567/status;t=rfdsf3adsd23das
http://ex.org/act/transfer?
  f=12&t=32&sig=a359d72d424cbd913686435bc6e7e372
```

5. IDs are not bad, but URIs are better

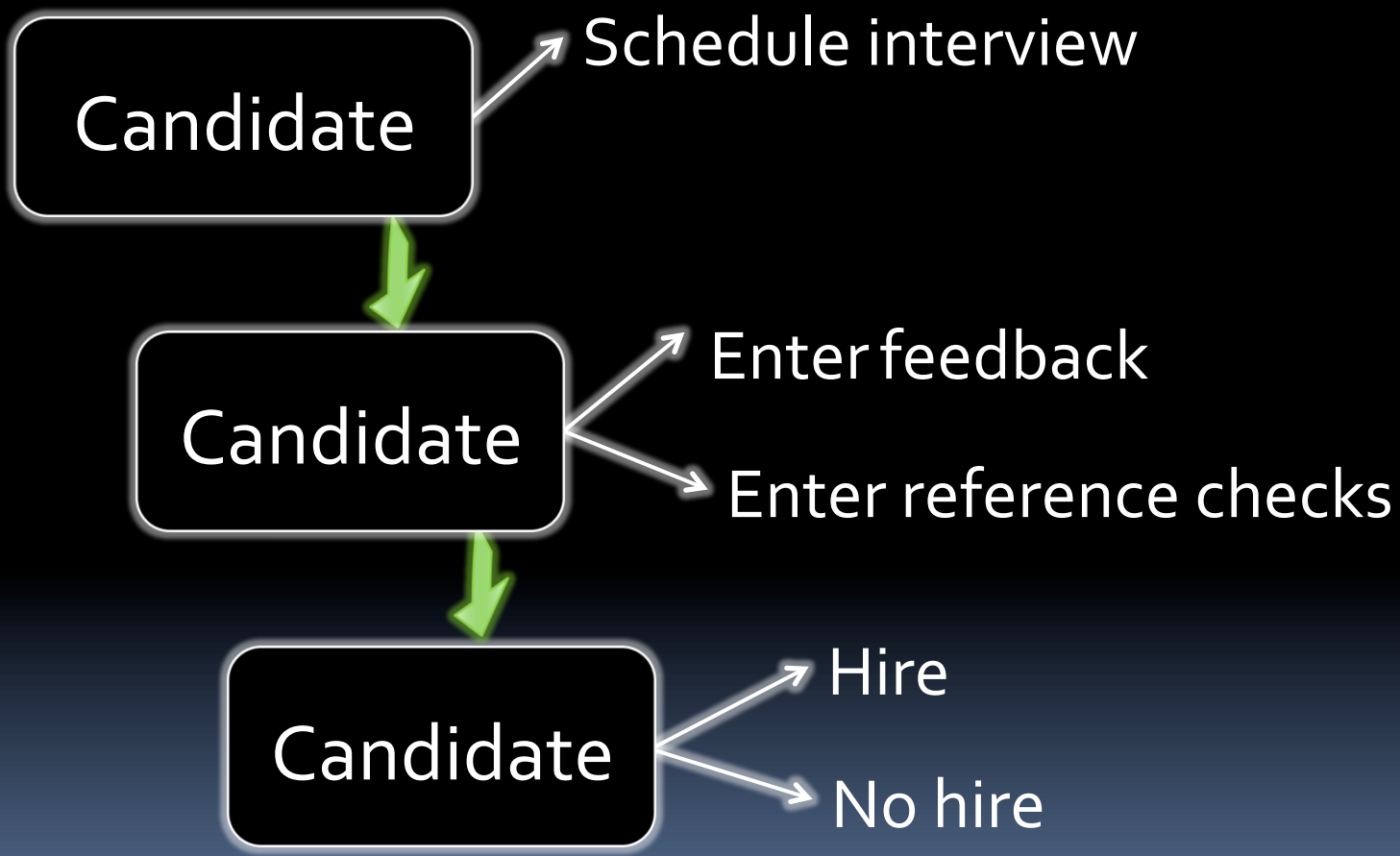
How much “hyper”media?
Should you care?

```
<album>
  <photo id="1234">...</photo>
  <photo id="5678">…</photo>
</album>
```

Bad?

```
<album xml:base="http://ex.org">
  <photo>
    <link href="/photo/1234"/>...
  </photo>
  <photo>
    <link href="/photo/5678"/>...
  </photo>
</album>
```

Good?


Candidate
Schedule interview
Candidate
Enter feedback
Enter reference checks
Candidate
Hire
No hire

```
GET /transfer/token?from=1234&to=5678
Host: ex.org
```

```
200 OK
Content-Type: application/xml;charset=UTF-8

<token>
  <link rel="http://ex.org/rels/transfer"
    href="http://ex.org/transfer;9ihrdsadas"/>
  <from>
   <balance>...</balance>
  </from>
  <to>
    <balance>...</balance>
  </to>
 </token>
```

URI decoupling	****
Application flow	*****
Opaque application state	*****

6. Managing concurrency

```
GET /subbu/address/1
```

```
200 OK
Date: Mon, 28 Sep 2009 14:30:53 GMT
Etag: “8cf498a1ca3ceb67fe50d401d4759e34”
Last-Modified: Mon, 28 Sep 2009 01:30:53 GMT
Cache-Control: public,max-age=3600

<address>...</address>
```

```
PUT /subbu/address/1
If-Unmodified-Since: Mon, 28 Sep 2009 14:30:53 GMT
If-Match: “8cf498a1ca3ceb67fe50d401d4759e34”
```

```
412 Precondition Failed
```

```
GET /acct/1234
```

```
200 OK
ETag: "f091aae21b44c71:6b9"
Content-Type: application/xml;charset=UTF-8

<account>
  ...
</address>
```

```
GET /acct/5678
```

```
 ...
```

```
POST /transfer
Host: ex.org
Content-Type: application/x-www-form-urlencoded

amount=1000&from=1234&to=5678...
```

```
GET /acct/1234
```

```
200 OK
ETag: "f091aae21b44c71:6b9"
Content-Type: application/xml;charset=UTF-8

<account>
  ...
</address>
```

```
GET /acct/5678
```

✗ No concurrency control

```
...
```

```
POST /transfer
Host: ex.org
Content-Type: application/x-www-form-urlencoded

amount=1000&from=1234&to=5678...
```

```
GET /transfer/token?from=1234&to=5678
Host: ex.org
```

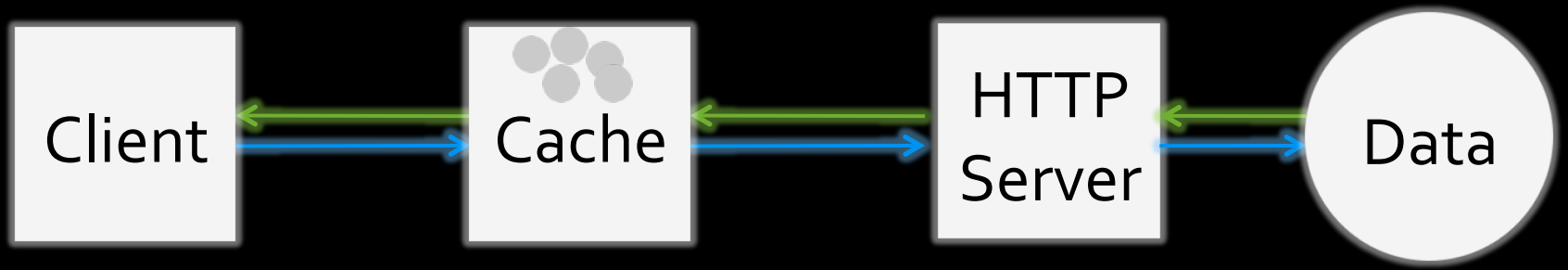
```
200 OK
Content-Type: application/xml;charset=UTF-8

<token>
  <link rel="http://ex.org/rels/transfer"
    href="http://ex.org/transfer;9ihrdsadas"/>
  <from>
   <balance>...</balance>
  </from>
  <to>
    <balance>...</balance>
  </to>
 </token>
```

```
POST /transfer/token;9ihrdsadas
Host: ex.org
Content-Type: application/x-www-form-urlencoded

amount=1000
```

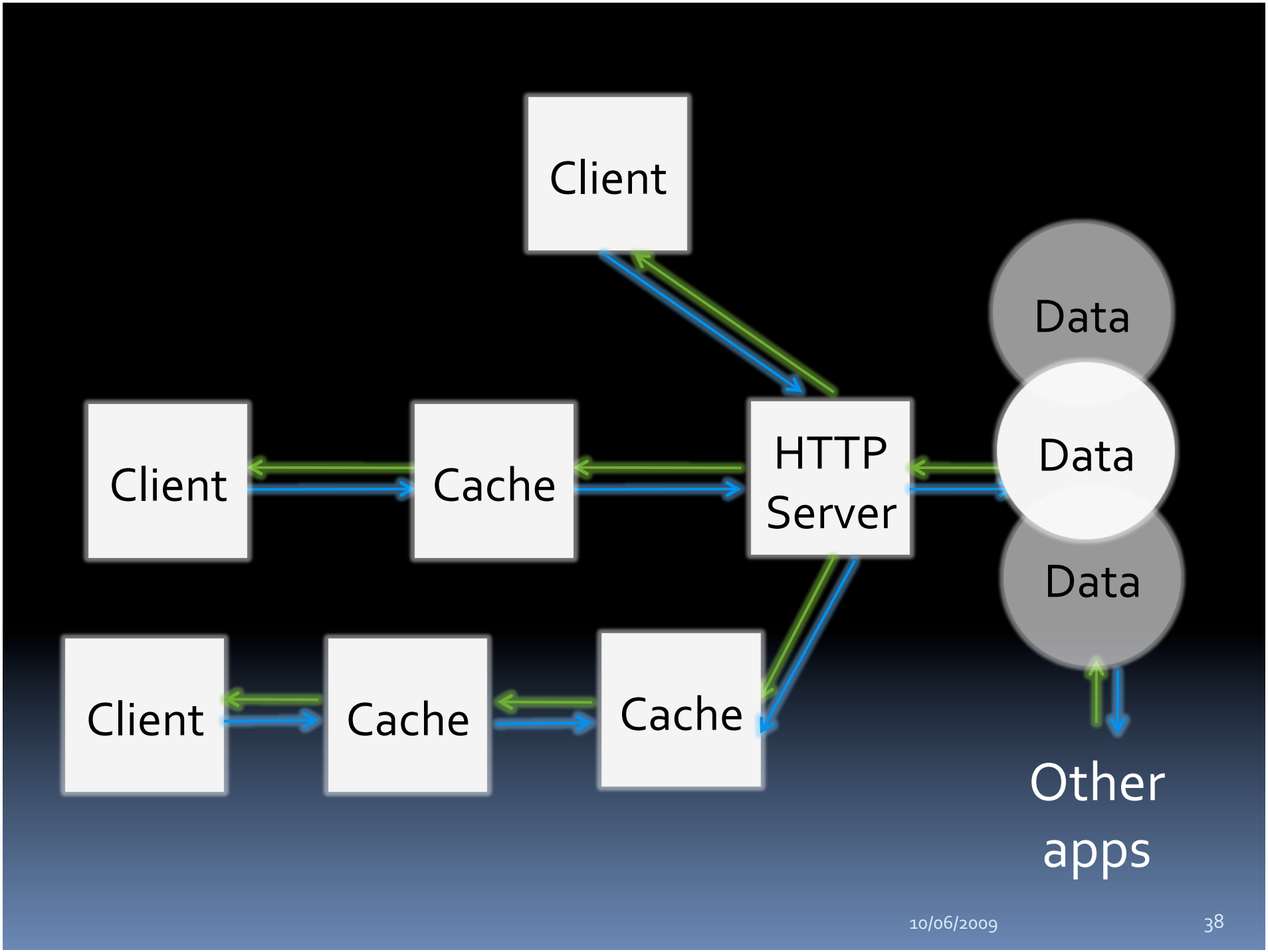
```
201 Created
Location: http://ex.org/transfer/1234
Content-Type: application/xml;charset=UTF-8

<transfer>
  <created>2009-09-30T15:00:00Z</created>
  <from>
   <balance>...</balance>
  </from>
  <to>
    <balance>...</balance>
  </to>
 </transfer>
```

✔ Concurrency control

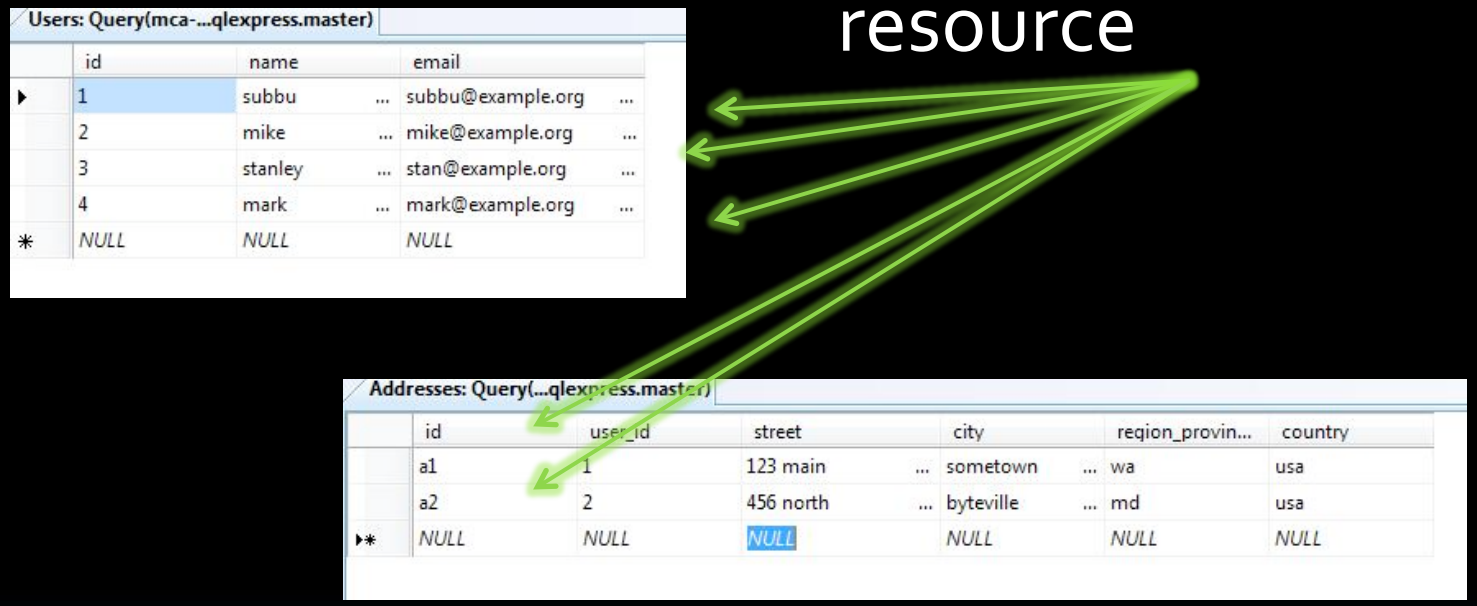
7. Caching is not perfect



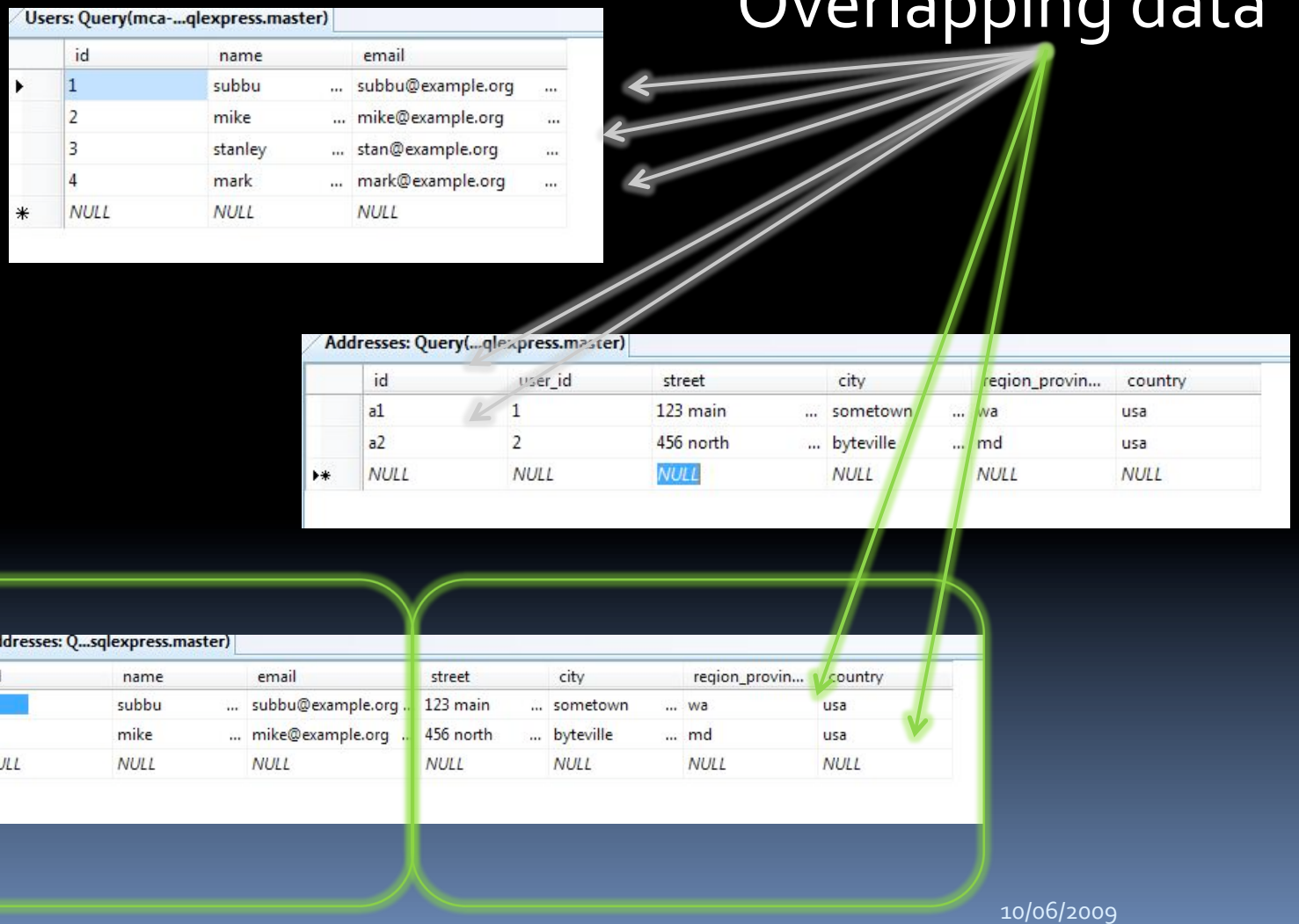
Ideal – perfectly visible

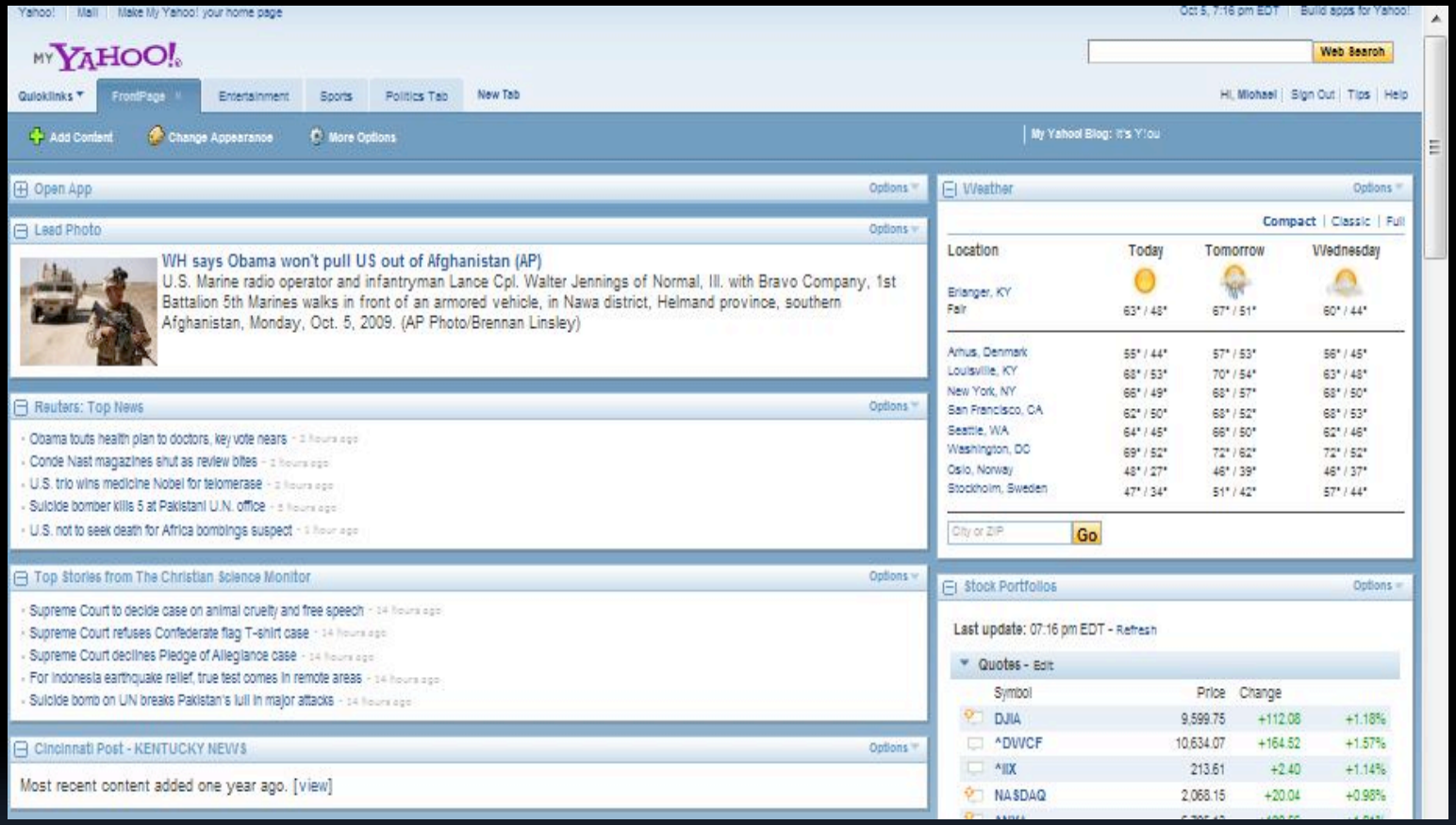

Client
Data
Client
Cache
HTTP Server
Data
Data
Client
Cache
Cache
Other apps

Every row is a resource

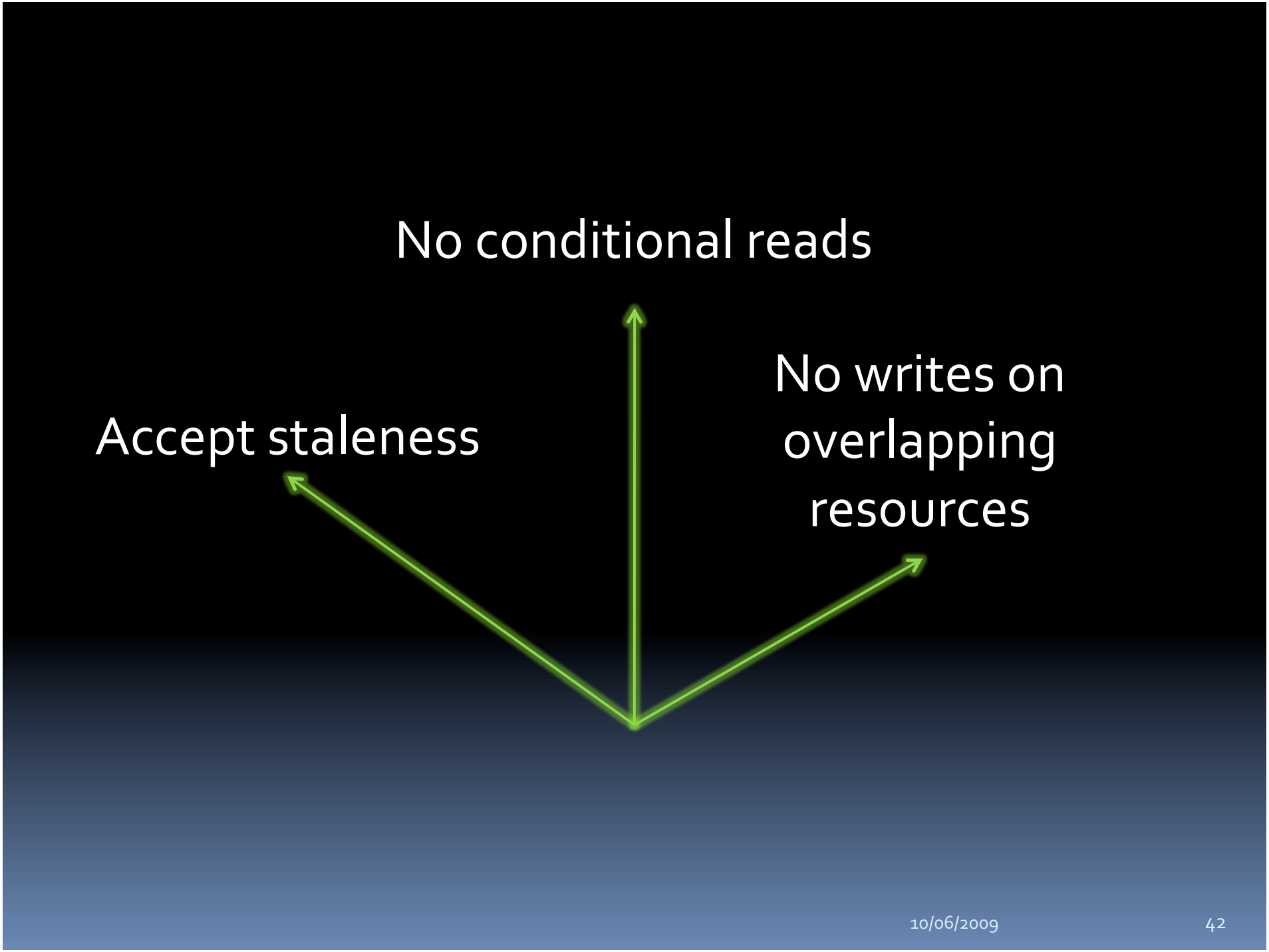


Overlapping data





Some resources are like home pages


No conditional reads
Accept staleness
No writes on overlapping resources

Conclusion

- Focus on tradeoffs
- Relax constraints judiciously, but not accidentally or by ignorance
- Put HTTP and its plumbing to good use