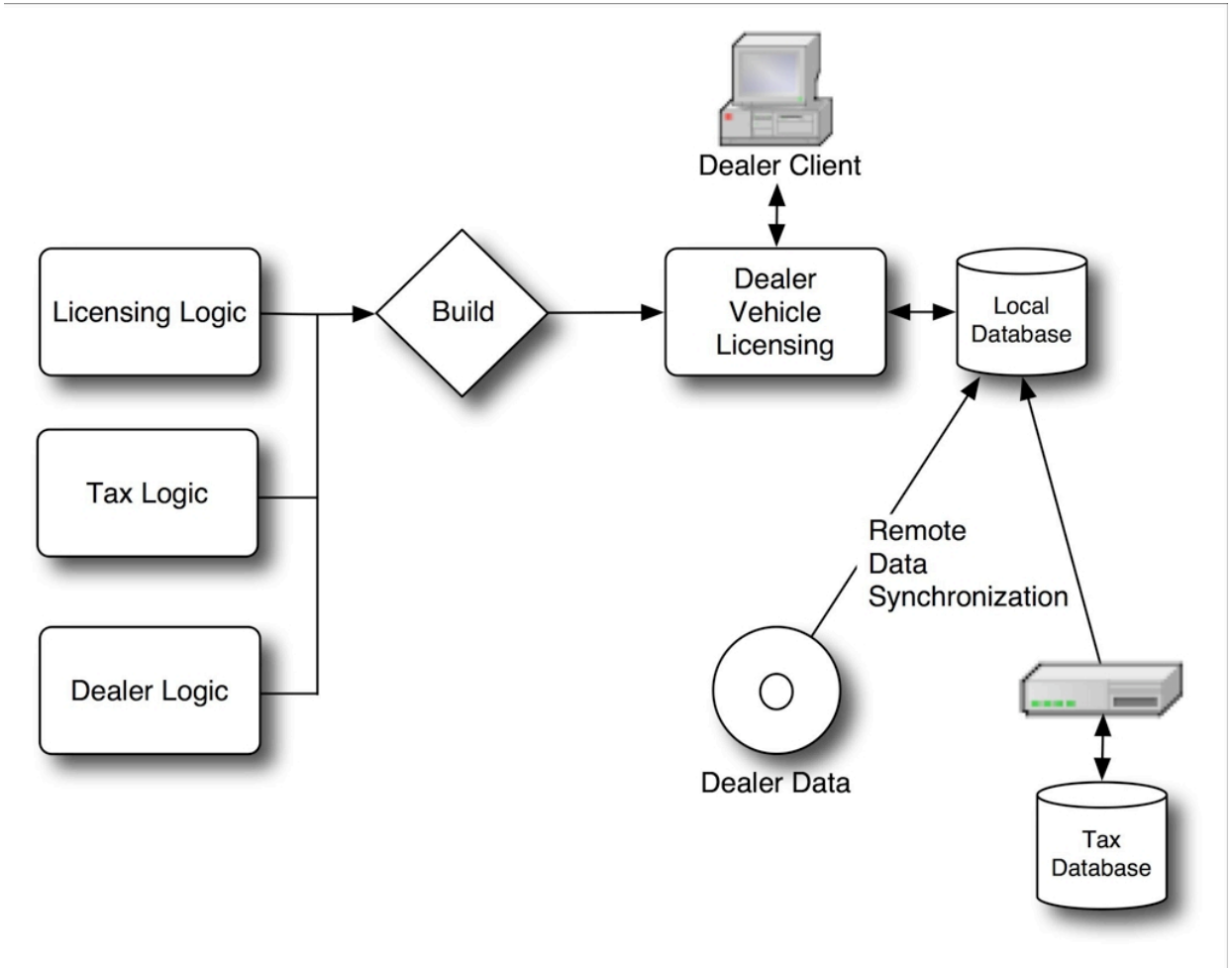
Service Oriented Architectures & Web Services

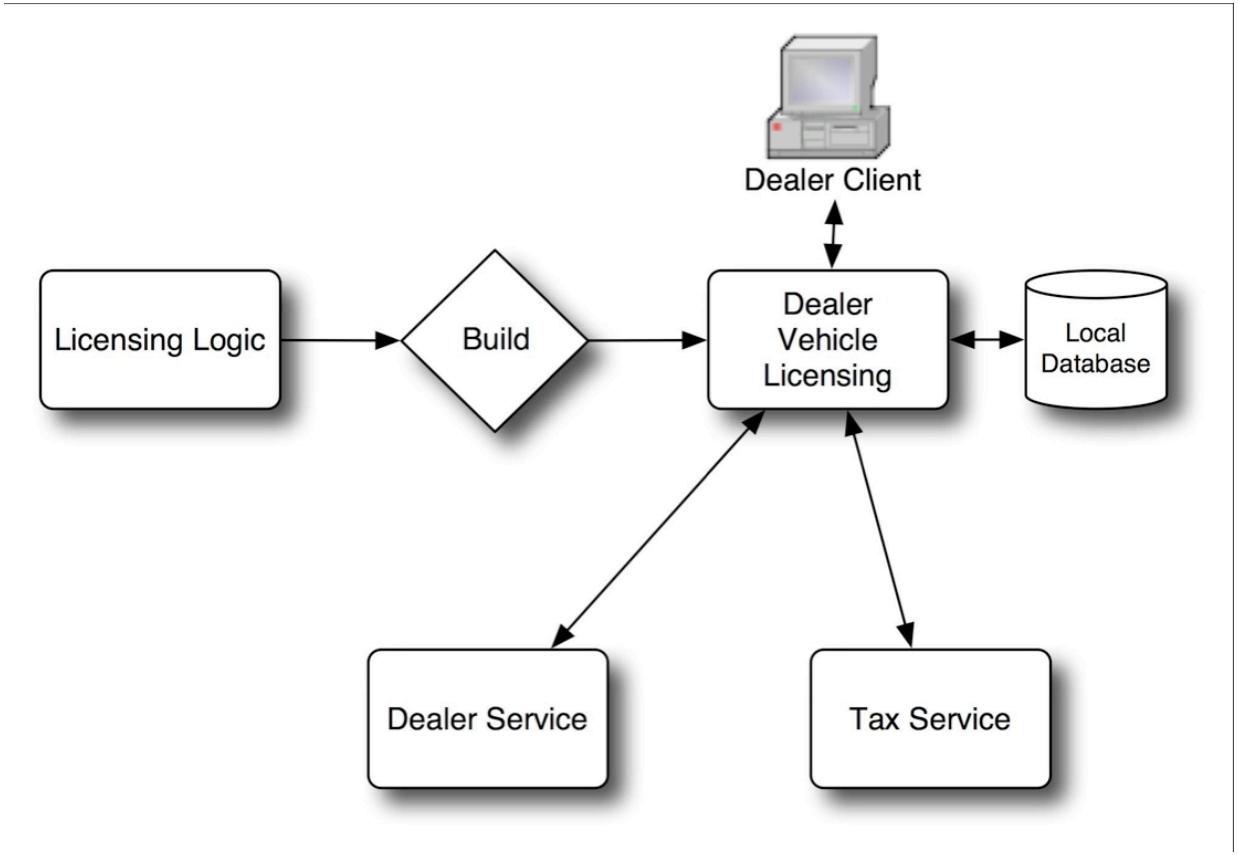
Phillip J. Windley, PhD
Brigham Young University
phil@windley.com
www.windley.com

Railroad and Highways

Traditional Architecture



Service Oriented Architecture



Key Differences

- Creating applications by
  - integrating *network* services
  - that are *very far away*
  - and owned by *strangers*
- Network services not subroutines or objects

Service Oriented Architecture Properties

- Discoverable and dynamic
- Loosely coupled
- Locationally transparent
- Diversely owned
- Interoperable
- Composable
- Network addressable
- Self healing

Loose Coupling

	Tight Coupling	Loose Coupling
Interface	Classes/methods	Fixed verbs
Messaging	Procedure call	Document passing
Typing	Static	Dynamic
Synchronization	Synchronous	Asynchronous
References	Named	Queried
Ontology	By prior agreement	Self describing
Schema	First-order	Higher-order
Communications	Point to point	Pub/Sub
Interaction	Direct	Brokered
Evaluation	Eager	Lazy
Motivation	Correctness & efficiency	Interoperability
Behavior	Planned	Adaptive
Coordination	Centralized	Distributed
Contracts	Implicit	Explicit

Latency

- Absolute limit of system architecture
- One of the few physical limits on computation
- New York is always going to be 30ms from London regardless of Moore's Law

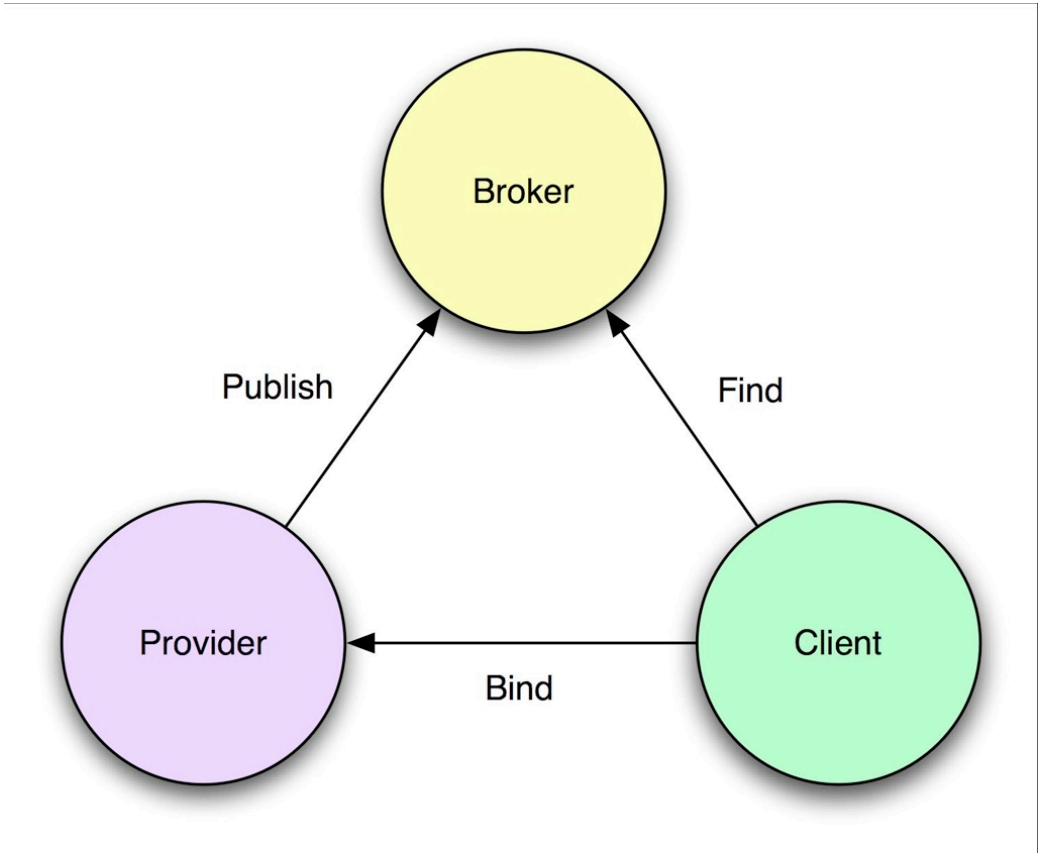
Service Oriented Architecture Benefits

- Code Re-use
- Correctness
- Maintainability
- Productization
- Security
- Focused developer roles
- Better alignment with business goals
- Scalability
- Feature augmentation

Web Services

- Web services are self-contained pieces of code with three distinguishing properties:
  - Communicate in an interoperable XML protocol, such as SOAP.
  - Describe themselves in an interoperable XML meta-format, such as WSDL.
  - Federate globally through XML based registry services, such as UDDI.
- Not defined in terms of SOAP, WSDL, and UDDI.

Roles in a Service Oriented Architecture



Using Web Services to Create SOAs

- SOAP - service binding and functionality
- WSDL - service description
- UDDI - service discovery

SOAP Example

```
<SOAP-ENV:Envelope>
 <SOAP-ENV:Body>
  <namespl:f2c  xmlns:namespl="urn:temperature">
     <c-gensym3 xsi:type="xsd:float">98.6</c-gensym3>
  </namespl:f2c>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

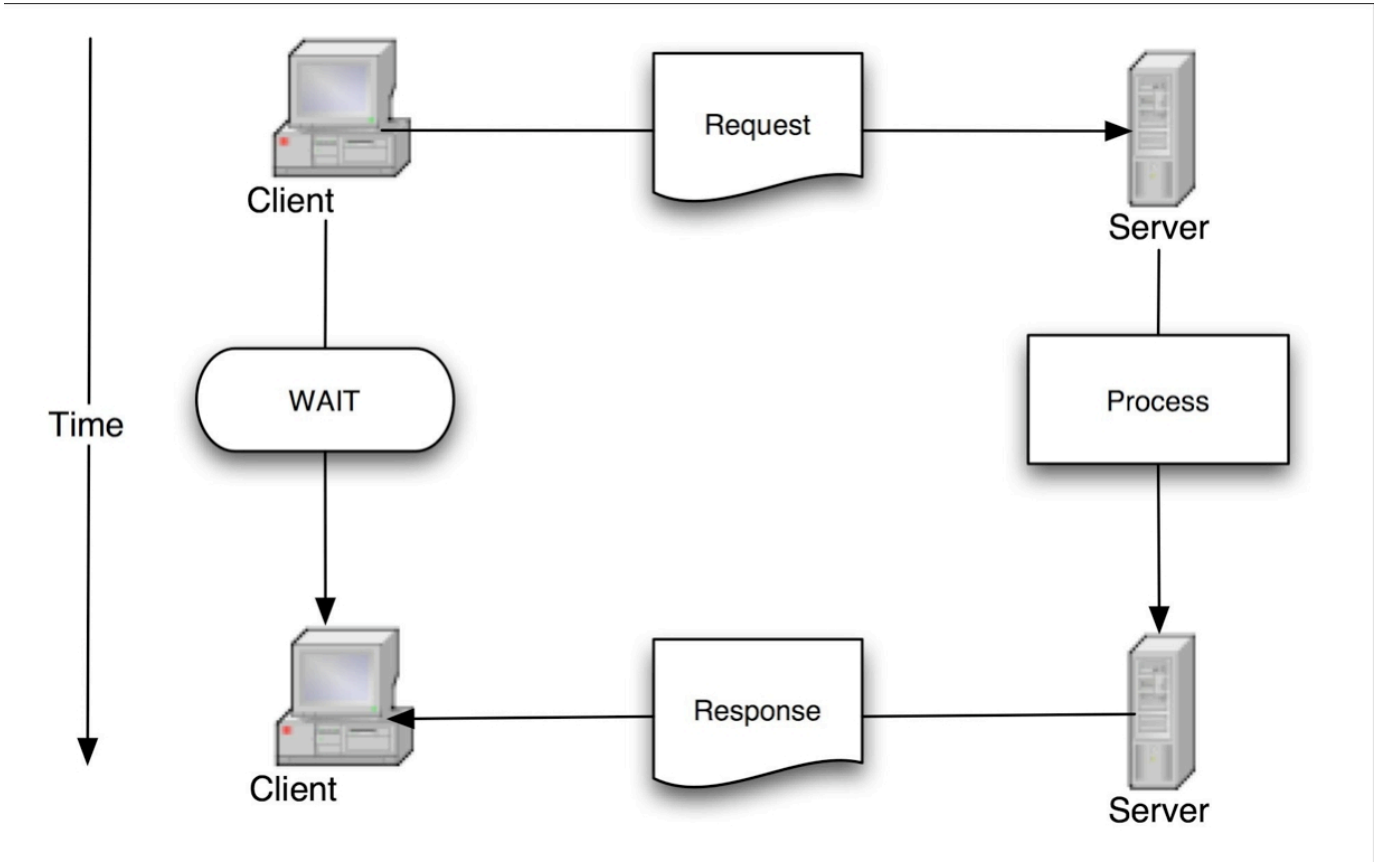
WSDL Example

```
<definitions name='com.windley.TempConv' >
  <message name='tempResp'>
    <part name='s-gensym3' type='xsd:float'/>
  </message>
  <message name='tempReq'>
    <part name='c-gensym3' type='xsd:float'/>
  </message>
  <portType name='com.windley.TempConv'>
    <operation name='f2c' parameterOrder='c-gensym3'>
      <input message='tempReq'/>
      <output message='tempResp'/>
    </operation>
  </portType>
  <binding name='com.windley.TempConvBinding' type='com.windley.TempConv'>
    <soap:binding style='rpc'     transport='http://schemas.xmlsoap.org/soap/http'/>
    <operation name='f2c'>
      <soap:operation soapAction='urn:temperature#f2c'/>
      <input>
        <soap:body use='encoded'  namespace='urn:temperature' />
      </input>
      <output>
        <soap:body use='encoded' namespace='urn:f2c' />
      </output>
    </operation>
  </binding>
  <service name='com.windey.TempConvService'>
    <documentation>
      sample temperature conversion service
    </documentation>
    <port name='com.windley.TempConvPort' binding='tns:com.windley.TempConvBinding'>
      <soap:address location='http://www.windley.org/cgi-bin/temper.cgi'/>
    </port>
  </service>
</definitions>
```

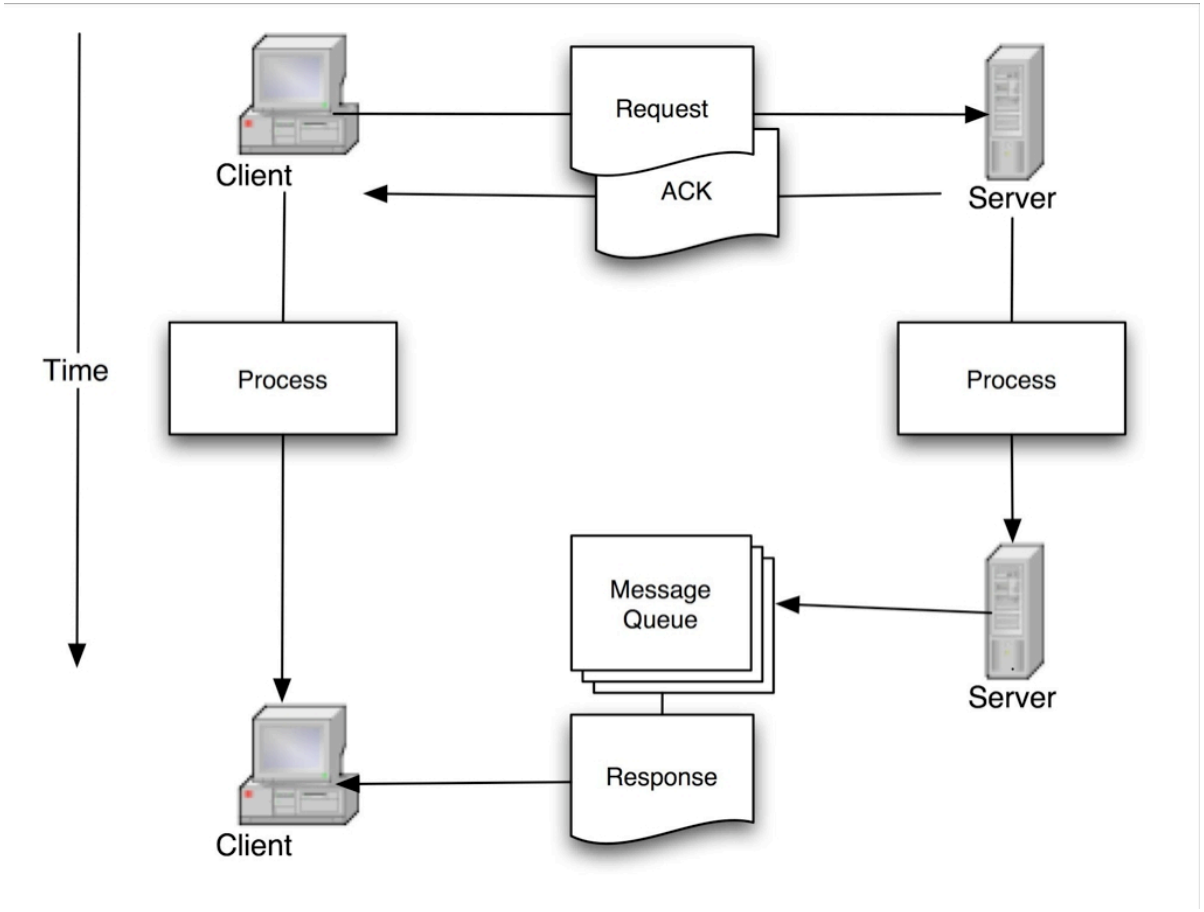
Using Service Oriented Architectures

- Think philosophy, not product
- Build a few pilot projects
- Then do the planning:
  - Enterprise architecture
  - Interoperability framework

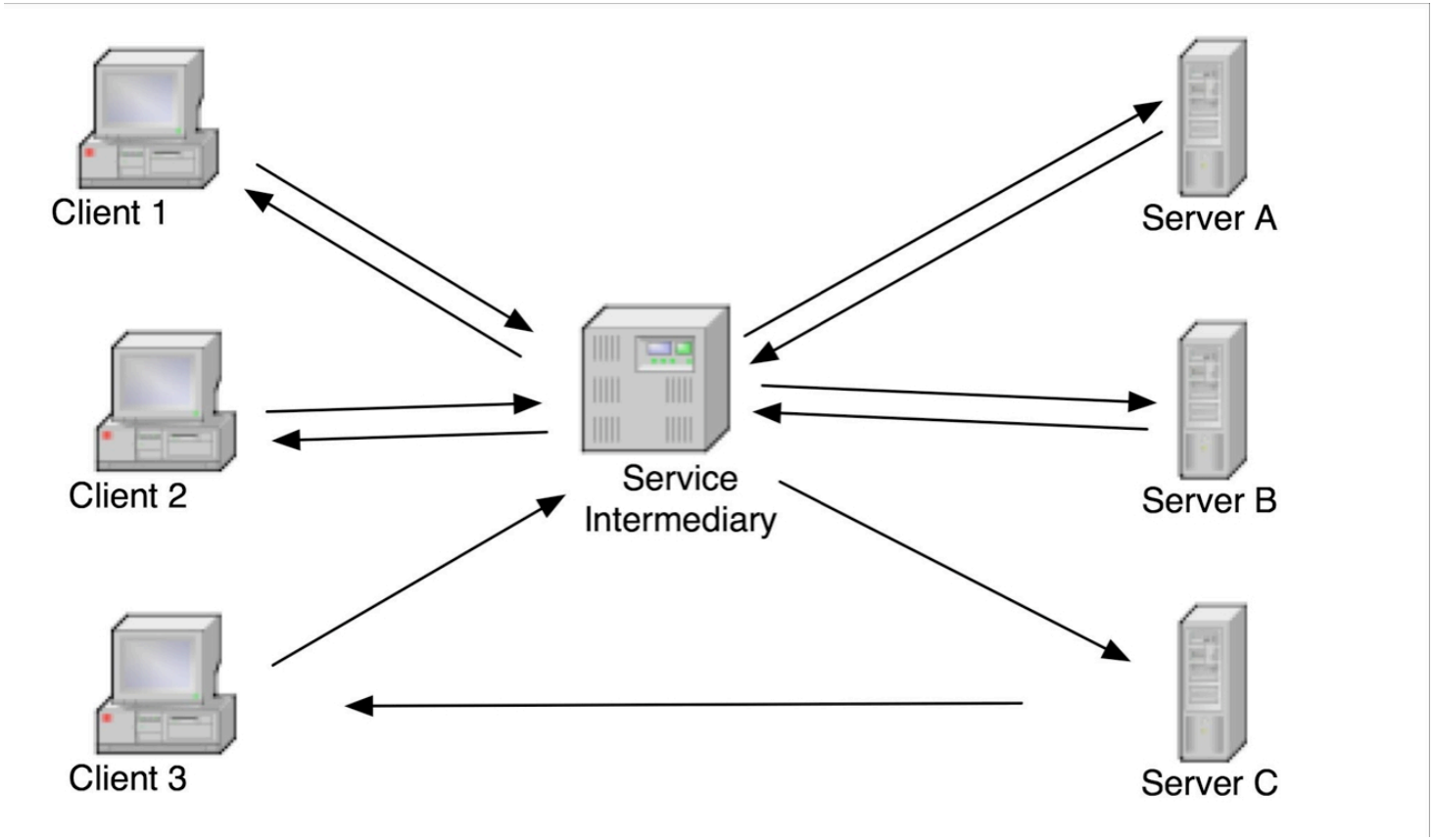
Synchronous Messaging



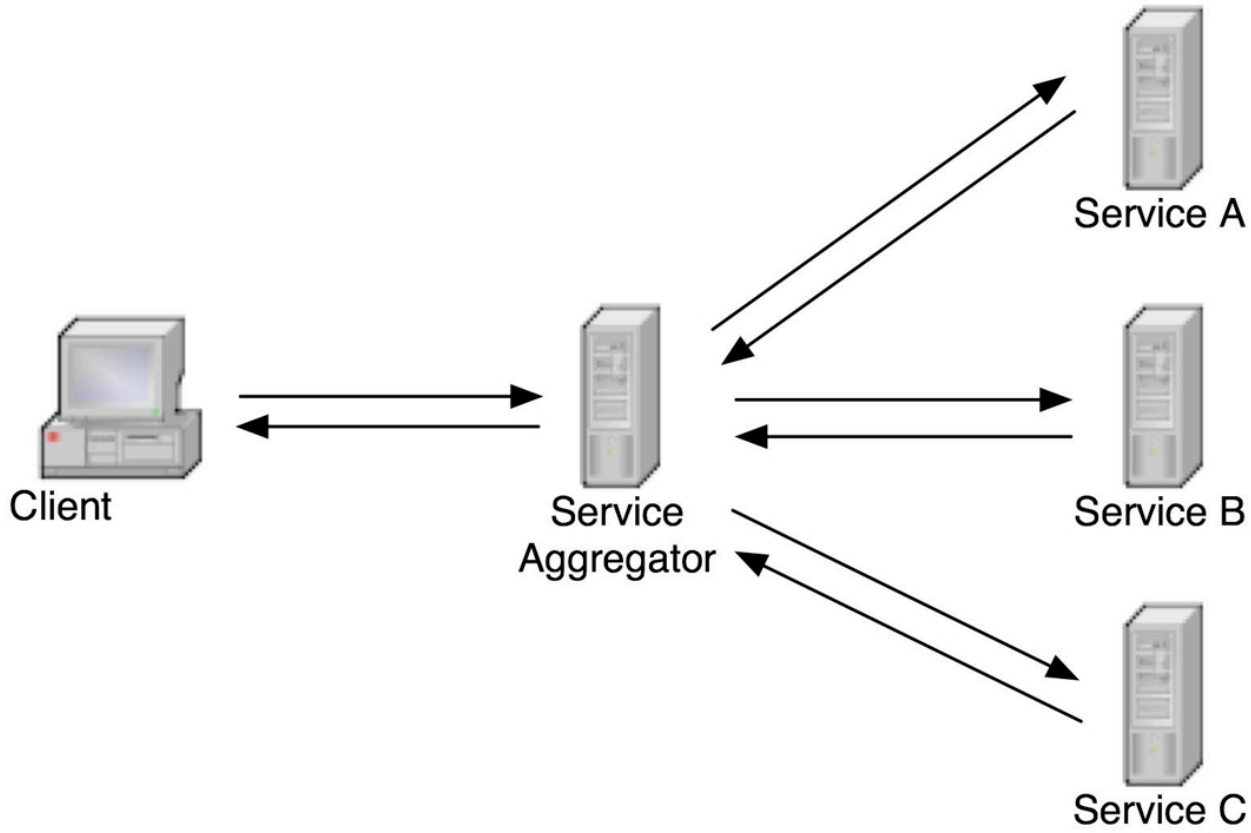
Asynchronous Messaging



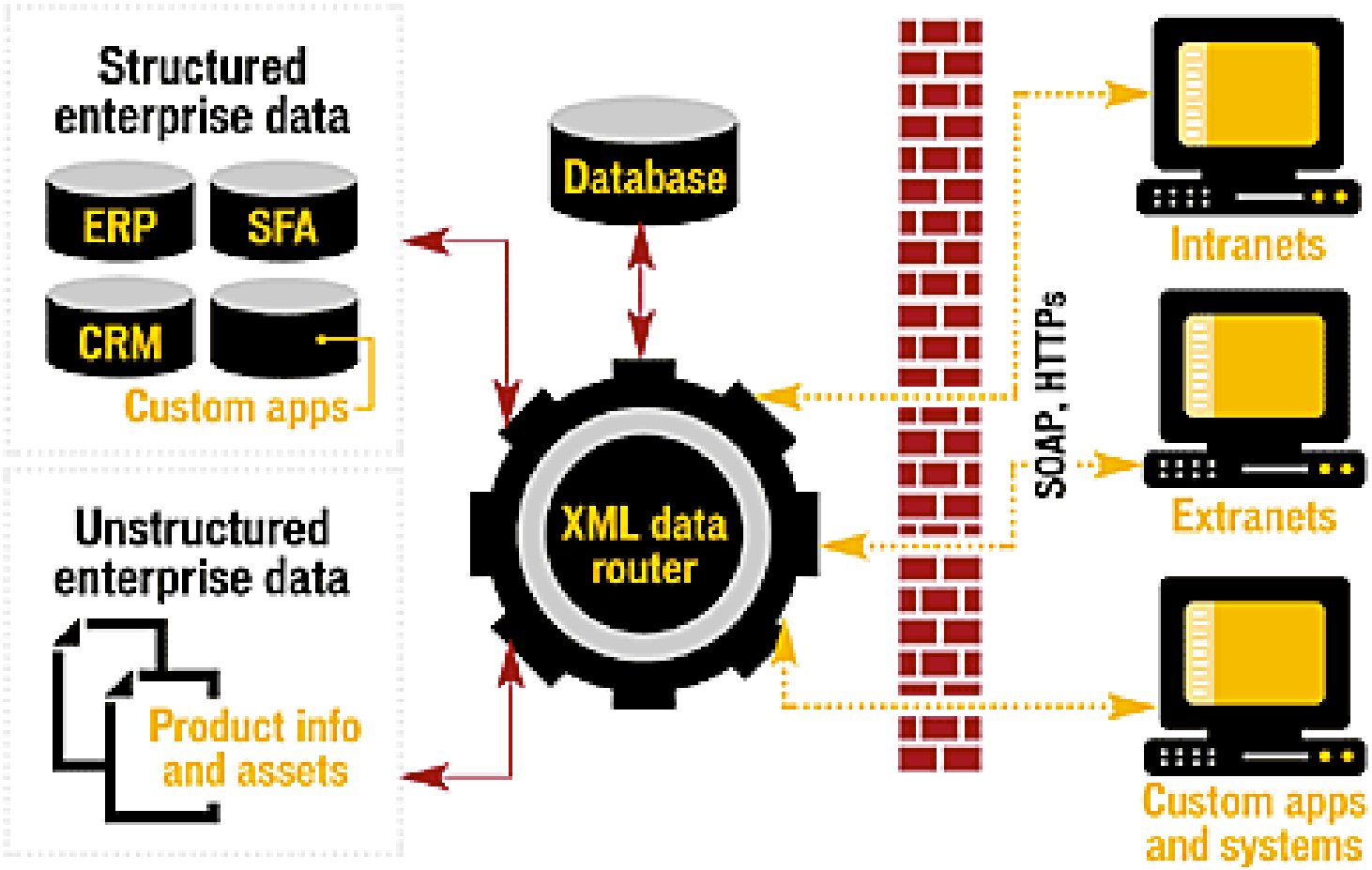
Service Intermediaries



Service Aggregators



XML Firewalls



The Williams Family

Life Event: Moving to Utah

- Change of address
- Register car
- Register to vote
- Enroll child in school
- Bussing
- City services
- Health information
- Child safety
- Check the commute
- Tax information

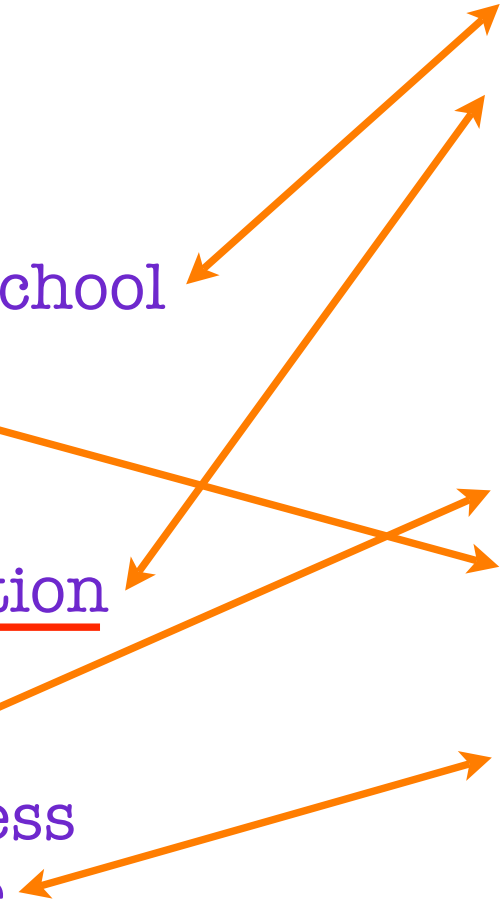
Federating Services

Moving to Utah

Real estate
Taxes
Register car
Register to vote
Enroll child in school
Bussing
City services
Utilities
Health information
Banking
Child safety
Change of address
Check commute

Child in School

Enrollment
Health information
Grades
Tuition and fees
Books
Child safety
Bussing
Federal programs
Check commute



Legacy Data

Web Services for Data

- Start Web services now:
  - Incrementally expose your data
  - Incrementally expose your APIs
- The more data and APIs that you expose the greater the potential interoperability
- Small marginal cost and high return, but... design is important

Design Principles

- Every data element and collection is a resource
- Every resource should have a URI
- Cool URI's don't change
- Preserve the structure of data until the last possible moment (i.e. return XML)
- Make XML Schemas available online
- Data queries on existing resources should be done with a GET
- Use POST to create new resources

Design Principles (cont)

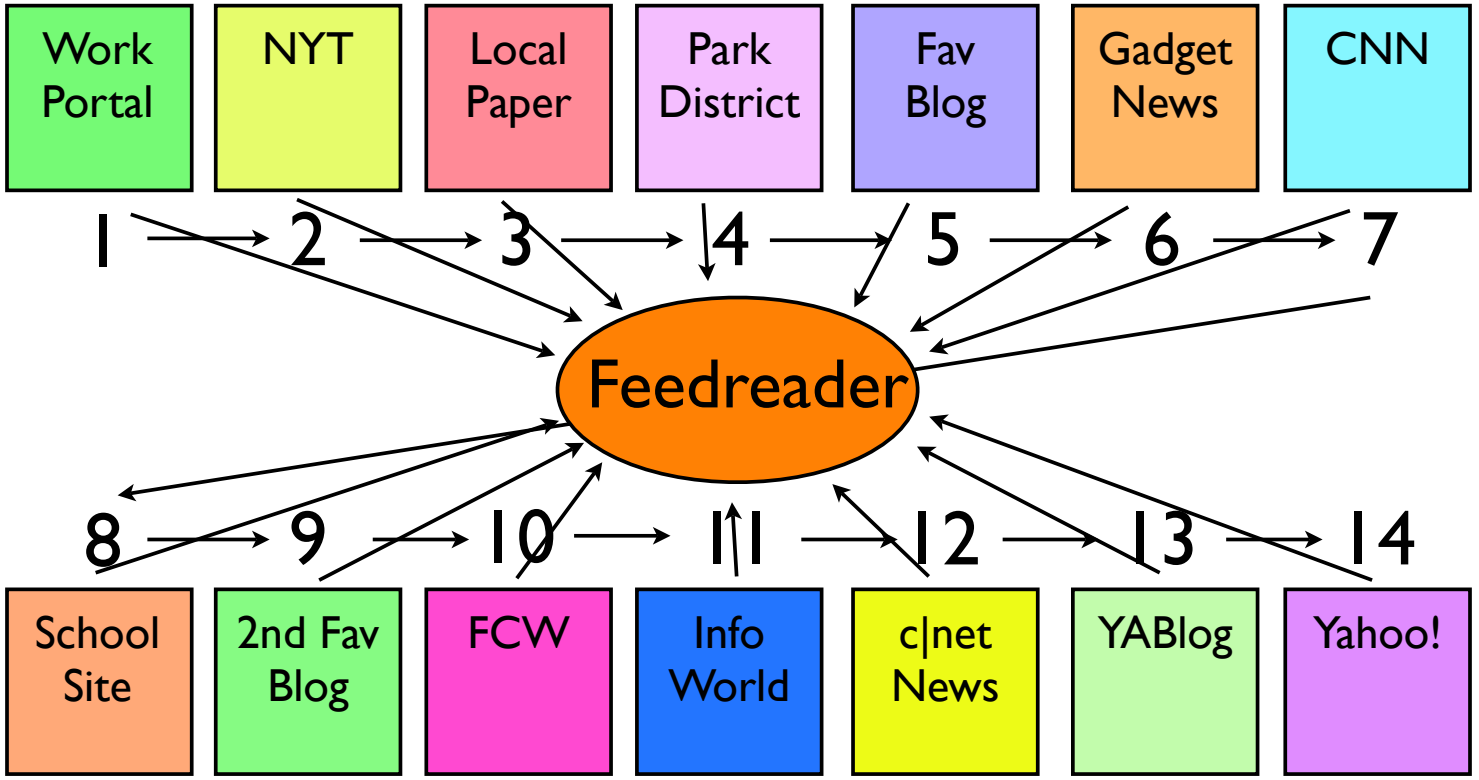
- Document your service API using WSDL, WRDL, or some other standard
- Advertise the presence of the data using WSIL
- Adhere to data standards such as RSS where available
- Use Metadata (RDF) for XML
- Use HTTP authentication as much as possible
- Make data available in multiple flavors (XSLT)

Brainstorming Two Websites

Case Study: RSS

- Rich Site Summary
- XML-based content syndication
- Used by feedreaders as well as Web sites

Why RSS


Work Portal
NYT
Local Paper
Park District
Fav Blog
Gadget News
CNN
1
2
3
4
5
6
7
Feedreader
8
9
10
11
12
13
14
School Site
2nd Fav Blog
FCW
Info World
c|net News
YABlog
Yahoo!

Who Has RSS?


The New York Times
ON THE WEB


TIME
ONLINE EDITION


YAHOO!


utah
dot
gov


DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA


amazon.com.


MOVABLETYPE
Publishing Platform

How Do I Find RSS?


```
<link rel="alternate" type="application/rss+xml" title="RSS" href="http://www.windley.com/rss.xml" />
```

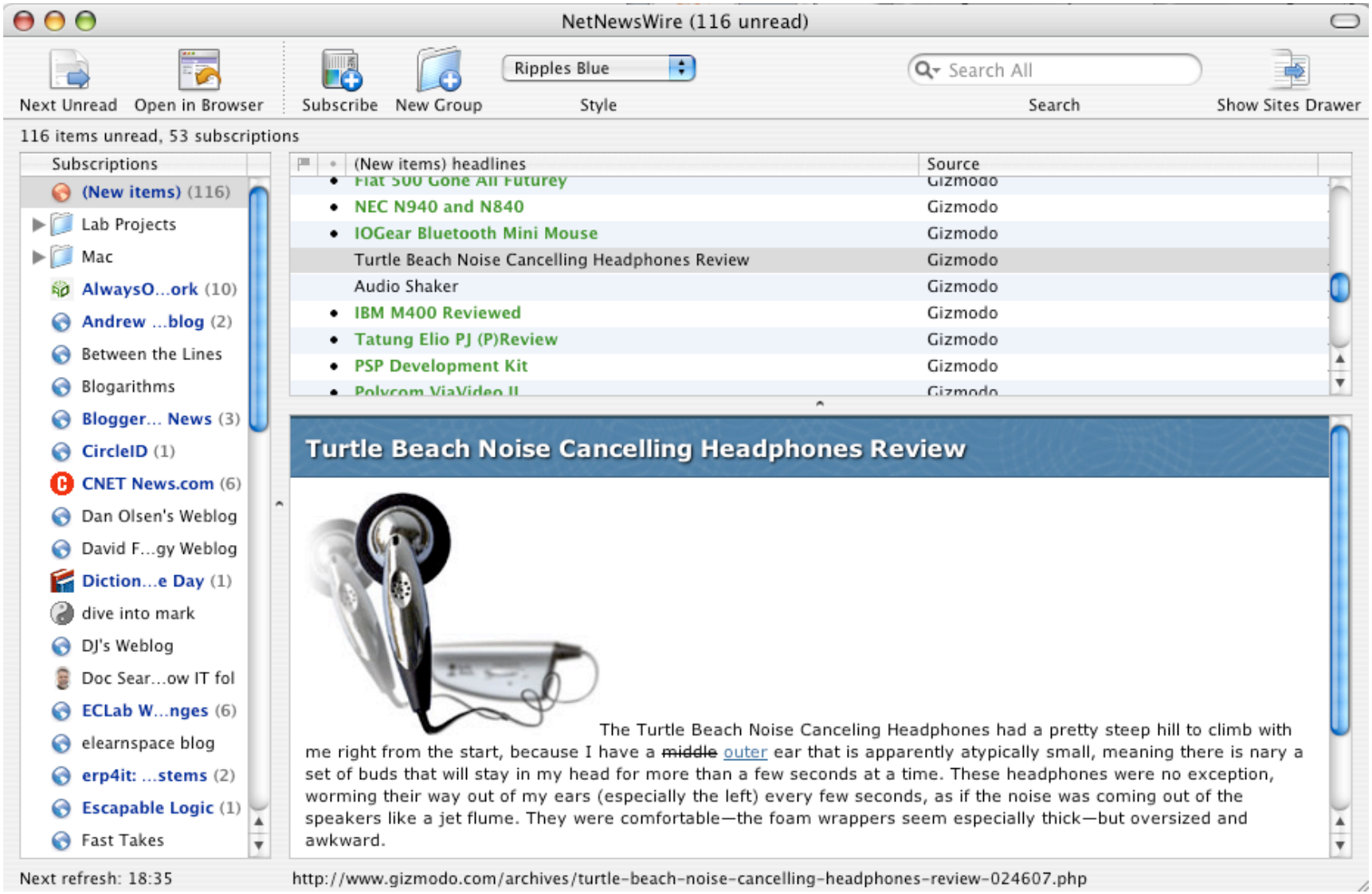
www.windley.com

What's It Look Like?

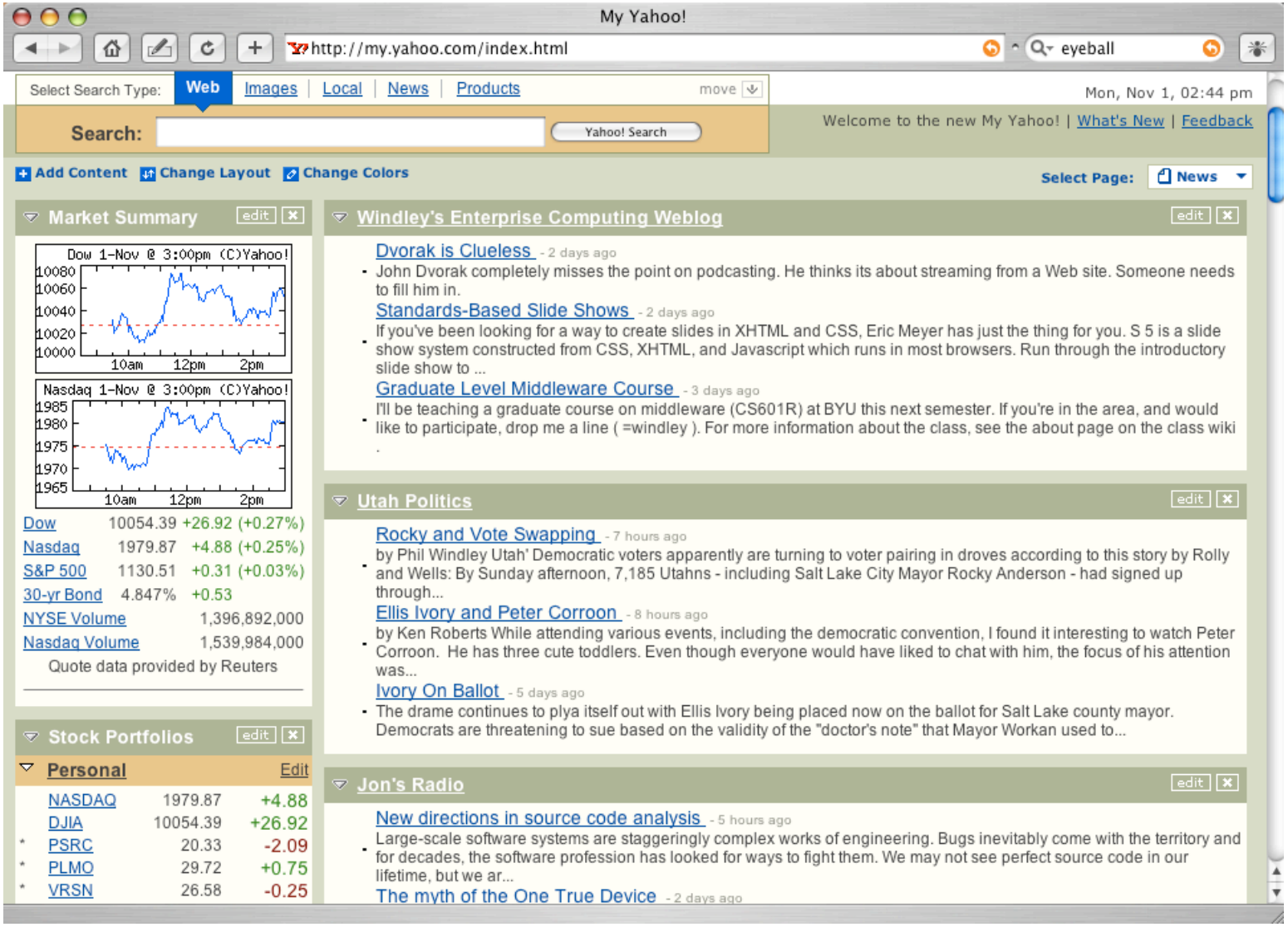
Source of http://www.windley.com/rss.xml

```
<?xml version="1.0"?>
<!-- RSS generated by Radio UserLand v8.0.8 on Sat, 30 Oct 2004 00:45:24 GMT -->
<rss version="2.0">
    <channel>
        <title>Windley&apos;s Enterprise Computing Weblog</title>
        <link>http://www.windley.com/</link>
        <description>Organizations Get the IT They Deserve&lt;sup class=&quot;realsmall&quot;&gt;SM&lt;/sup&gt;</description>
        <language>en</language>
        <copyright>Copyright 2004 Phil Windley</copyright>
        <lastBuildDate>Sat, 30 Oct 2004 00:45:24 GMT</lastBuildDate>
        <docs>http://backend.userland.com/rss</docs>
        <generator>Radio UserLand v8.0.8</generator>
        <managingEditor>phil@windley.com</managingEditor>
        <webMaster>phil@windley.com</webMaster>
        <category domain="http://www.weblogs.com/rssUpdates/changes.xml">rssUpdates</category>
        <skipHours>
            <hour>0</hour>
            <hour>1</hour>
            </skipHours>
        <ttl>60</ttl>
        <item>
            <title>Dvorak is Clueless</title>
            <link>http://www.windley.com/2004/10/29.html#a1499</link>
            <description>&lt;p&gt;
John Dvorak &lt;a href=&quot;http://www.pcmag.com/article2/0,1759,1682993,00.asp&quot;&gt;completely misses the point&lt;/a&gt; on podcasting.
He thinks its about streaming from a Web site.  Someone needs to fill him in.
&lt;/p&gt;</description>
            <guid>http://www.windley.com/2004/10/29.html#a1499</guid>
            <pubDate>Sat, 30 Oct 2004 00:45:19 GMT</pubDate>
            <category>Newsletter</category>
            </item>
        <item>
            <title>Standards-Based Slide Shows</title>
            <link>http://www.windley.com/2004/10/29.html#a1498</link>
            <description>&lt;p&gt;
If you&apos;ve been looking for a way to create slides in XHTML and CSS, Eric Meyer has just the thing for you.  &lt;a href=&quot;http://
www.meyerweb.com/eric/tools/s5/&quot;&gt;S&lt;sup&gt;5&lt;/sup&gt; is a slide show system&lt;/a&gt; constructed from CSS, XHTML, and Javascript
which runs in most browsers.  Run through the &lt;a href=&quot;http://www.meyerweb.com/eric/tools/s5/s5-intro.html&quot;&gt;introductory slide
show&lt;/a&gt; to get a feel for its features (including slide advance when the space bar is pressed and a disappearing navigation menu).  All
the slide content is in a single file, so once its loaded it&apos;s loaded.  What&apos;s more, since its based on CSS, printable slides are
constructed from the same source file automatically.  I don&apos;t think I&apos;m ready to give up Keynote and Powerpoint just yet, but for
some things, this might come in real handy.
&lt;/p&gt;</description>
            <guid>http://www.windley.com/2004/10/29.html#a1498</guid>
```

Feedreaders



My Yahoo!



www.windley.com

Using RSS

- Aggregate feeds onto Web sites
- Portals
- Press releases
- Calendars
- Del.icio.us
- Enclosures and podcasting

Creating RSS

- Edit by hand
- Blog software
- Calendaring tools
- Custom applications

Why RSS, Part II

- Simple, HTTP-based Web service
- Easy to create, easy to use
- RSS is an email replacement for many applications
- No Spam, pull instead of push
- Works for infrequent updates (and frequent ones)

Serendipitous Applications

- Small, scripted aggregations lead to serendipitous applications
- Example: Udell's Library Lookup

Example Web Services

- Common Payment Gateway
  - Web services creates easy interfaces
  - No need for multiple SDKs

SOAP Request

```
<SOAP-ENV:Envelope
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema"
   xmlns:
    SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <namesp1:authorize
         xmlns:namesp1="http://www.windley.com/Demo">
      <c-gensym3 xsi:type="xsd:string">
        ccn4111111111111111</c-gensym3>
      <c-gensym5 xsi:type="xsd:string">
        y2003</c-gensym5>
      <c-gensym7 xsi:type="xsd:string">m10</c-gensym7>
      <c-gensym9 xsi:type="xsd:int">12</c-gensym9>
      <c-gensym11 xsi:type="xsd:int">0</c-gensym11>
    </namesp1:authorize>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
 <soapenv:Body>
  <ns1:authorizeResponse
   soapenv:
   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="http://www.windley.com/Demo">
   <ns1:authorizeReturn xsi:type="ns2:string"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:ns2="http://www.w3.org/2001/XMLSchema">
     Return code: 100
     Auth code: testauth
     Comment: Authorized $12.00
    </ns1:authorizeReturn>
  </ns1:authorizeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

A Word of Warning

- Good architects should do everything they can to avoid data serialization.
  - Web services is nothing but serialization.
- When serialization cannot be avoided, it can be mitigated through caching in some cases.
  - SOAP over HTTP makes caching difficult (uses POST).

Summary

- First steps:
  - Don't let the hype scare you
  - Don't try to figure it all out first
  - Adopt simple principals
  - Jump in and do something
- The keys are
  - XML
  - Incrementally exposing data and APIs

Contact Information

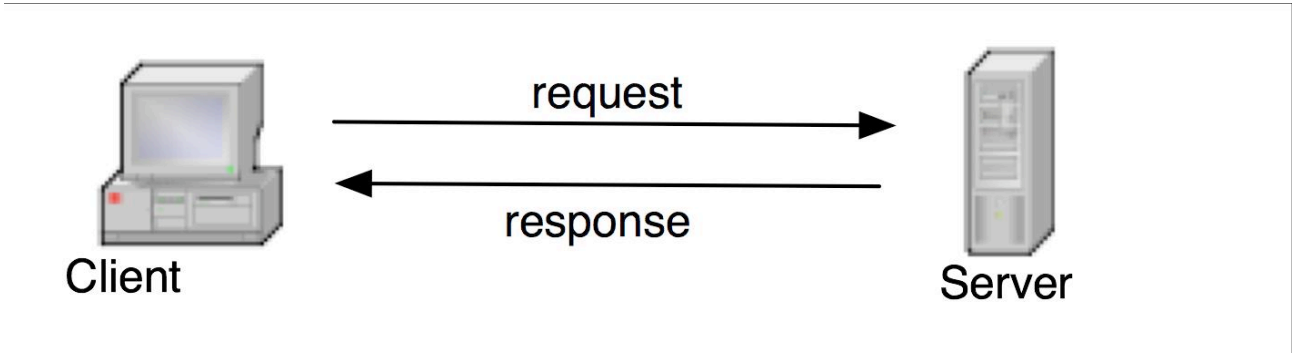
Contact me

- phil@windley.com
- www.windley.com

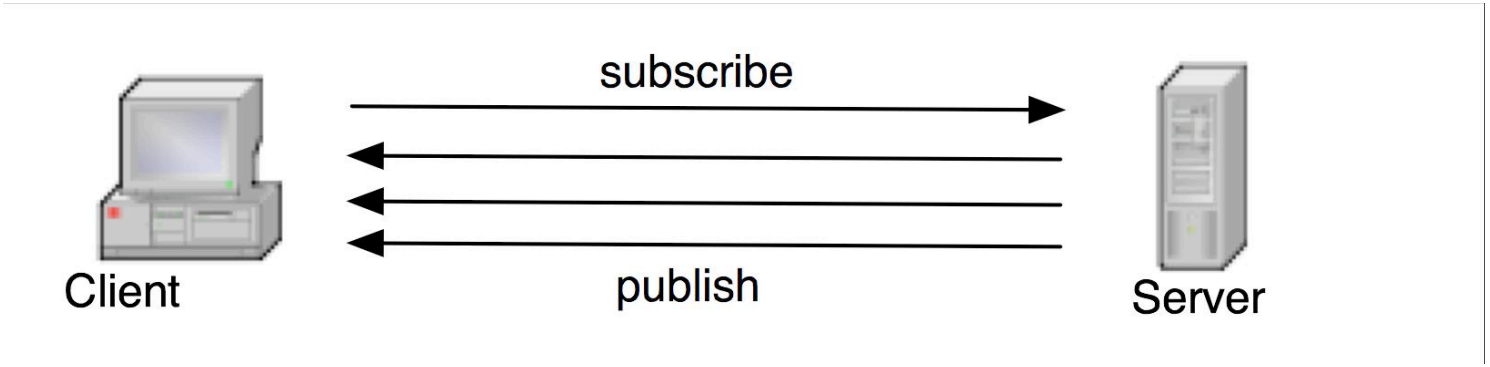
Questions?

Appendix

Messaging Pattern: Request-Response



Messaging Pattern: Publish and Subscribe



Messaging Pattern: Broadcast and Multicast