Last Time

- sample services:
  - Bluetooth
  - distributed objects, agents
**This Time**

- sample services, cont'd:
  - grid computing
  - XML
  - RSS
  - RFID

---

**Grid Computing**

- make a collection of distributed resources appear to a user as a single, coherent, computing resource
  - e.g., cpu, storage, visualization & display
  - even if different platforms are involved
- connect resources over a network to make available to user
- may dynamically create such collections
- start with a research-oriented system...
The TeraGrid Project

- NSF stimulated project to create resource for researchers
- consists of:
  - 20 teraflops over 5 distinct locations
  - storage management of 1 Pb (2^{50} bytes)
  - tightly integrated components in each cluster
  - connected to other clusters via 40 Gbs network
- e.g., linux cluster (Itanium® processors): 1 Tf
Grid Computing for Mortals

- same goal: creating computing resource from separate distributed resources
- use standard Internet
- need special software to build grid
  - currently emerging std is Globus Toolkit (4.01)
  - “Legion” from Avaki (Grimshaw at U. Virginia)

Layered Grid Architecture
Grid Software

- need to ‘virtualize’ resources
- Open Grid Services Architecture (OGSA)
  - common ‘view’ of a resource and how to use it
- for service ensembles, OGSA supports:
  - creation
  - maintenance
  - application
- see http://www.globus.org/ogsa

The Emerging Grid

- grid: “...emergence of a new infrastructure upon which first science, and then the whole economy, will be built ” (Larry Smarr)

- do for computing what Internet did for documents

- aiming at worldwide governance & standards body analogous to IETF: http://www.gridforum.org
Evolution of Web Services

- limited to HTML
- improve function via ‘active pages’
  - those that require server to run command parser on requested HTML file and perform computing as needed
- but still limited:
  - to what can be embedded in near–HTML pages
- what if could generate pages containing tags not previously defined?
  - i.e., tags specific to particular application needs?

Working the Web

- current web pages written in html
- browser/client renders html
- html tags:
  - understood by both client to have particular meaning
  - who defines?
  - can you have new ones? whenever you want?
Working the Web

- html tags:
  - understood by both client to have particular meaning
  - who defines?
    - W3C
  - can you have new ones? whenever you want?
    - not really
  - html tags only describe layout of document contents

Beyond HTML

- web purchase example: buy a hard disk
- go to vendor site and
  - navigate through their web pages to find item
  - maybe you don’t like their price
- go to another vendor site
  - navigate through their web pages to find item
  - you like price, but they’re out of stock
- go to another vendor site...
Better Than Before...

- better way?
- software agent on your computer:
  - presents a form where you say what you want
  - it returns a summary of vendors that meet criteria, automagically
- cannot (practically) use html to do this
  - how come?

...but need new language

- html only specifies layout
- to do order form: need tags that apply to content of document
  - e.g., identify it as a vendor query
  - e.g., indicate what part describes sought part
- need something that extends html capabilities:
  - eXtensible Markup Language, xml
xml

- xml, like html, uses tags mixed in with document content
- xml lets you define and create any tag you want in a document, e.g.,
  <partsquery>
    <type>computer hard disk
    <capacity> 60 Gb </capacity>
    <speed> 7200 rpm </speed>
  </type>
  </partsquery>
- but who will understand these tags?
- who can you send this to?
**xslt style sheet**

- xslt to convert xml parts query back to plain text
  - "attribute:value" format

```xml
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xs="http://www.w3.org/1999/XMLSchema"
    version="1.0" >
  <xsl:output method="text" indent="yes" />
  <xsl:template match="*" >
    <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="type">
    <xsl:element name="Part Type">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

**xml and xslt**

- rendering is a transformation operation
  - e.g., from ‘abstract’ xml to plain text
- xslt can be used for any such transformation
  - e.g., from one xml document to a different xml document
- ability to transform depends on having an ‘understanding’ of the source xml document’s structure
  - internal working representation: tree constructed from the xml
xml parsing

- an xml parser builds such a representation
- e.g.,:

```
  Document Root
    Parts Query
        Type
          Computer Hard
            Disk
                Capacity
                  60 Gb
                Speed
                  7200 rpm
```

xml parsers

- two popular parsers:
  - Domain Object Model (DOM):
    - from W3C
    - passes through entire xml document, builds representation in memory (e.g., tree)
  - Simple API for XML (SAX):
    - from XML-DEV
    - issues callbacks as tags are encountered during parsing
- freely available parsers of both kinds available for many platforms
xml: missing link

- have:
  - arbitrary xml tags describing content of document
  - ability to parse document into structure
  - can use xslt to transform into other structures
- don't have?

xml: missing link

- have:
  - arbitrary xml tags describing content of document
  - ability to parse document into structure
  - can use xslt to transform into other structures
- don't have? anything that knows about
  - data types
  - legal/illegal values for tags we've invented
- what if we name our xml tags the same as someone else's in a different document
  - where they don't mean the same thing?
xml: missing link

- need a guide to describe what our tags mean and what they are/are not allowed to have for values
- an xml schema describes:
  - data types appearing in xml document
  - content, e.g., values that are allowed
  - structure
  - allowed elements
- what about DTDs?

schema vs. DTD

- Document Type Definition (DTD):
  - defines tags appearing in a document
  - any SGML ... including html
  - does not provide info re.
    - values a tag may have, may not have
    - structure (e.g., hierarchy) of data
  - must be global
    - tags described in a DTD have that meaning everywhere
    - can’t have arbitrary xml tags
xml namespace

- schema:
  - provides info about structure and content of data and tags
  - allows for tags to be qualified by namespace
    - so tag ‘product_query’ can appear in two different XML documents and be different in structure and content
    - qualified by, e.g.,
      - acme.com/product_query
      - ozme.com/product_query

Other Web Services Components

- so you have an XML document + schema...
- who do you send it to?
- how do you know if a particular target is able to understand and process your document, and send you a reply?
- how do you get your document there?
Other Web Services Components

- how do you know... destination’s capabilities?
- use Web Services Description Language (WSDL):
  - provides standardized way for a site to make known formats & protocols its service accepts
Other Web Services Components

- how do you get your document there?
- use Simple Object Access Protocol (SOAP):
  - defines 'envelope' for web services communication
  - envelope mappable to http and other transport protocols
  - one-way msg protocol, allows for intermediaries to process or add to msg

Getting The Word Out

- suppose you have a special interest in x
  - e.g., news, sports, technical updates, AV equipment
- how do you stay current?
  - discover/learn which web sites have content
  - visit them often
- sites might prefer to be able to push their wares?
  - member subscription; they send email
  - specialized software 'agent' goes mining web site
Getting The Word Out

A Newer Model

web site D 'knows' a little about content on A, B, and C; you can see summaries and entire articles from A..D just by visiting D
How Build This?

- use specialized content agents
  - like crawlers, but targeted

- content-providing sites provide API for content access
How Build This?

- use specialized content agents
  - like crawlers, but targetted
- content–providing sites provide API for content access
- content–providing sites provide data dumps from their content

A Better Way To Build This

- use standardized ‘blurb’ format embodied in a technology called RSS
  - Rich Site Summary
  - a.k.a. Really Simple Syndication
- like syndicating: publish material to some number of locations
  - e.g., like comics in newspapers, some TV shows
- RSS uses XML
An RSS Item

- RSS provides a set of items within a channel to interested readers
- an item looks like:

```xml
<item>
  <title>RSS Resources</title>
  <link>http://www.webreference.com/authoring/languages/xml/rss/</link>
  <description>
    Defined in XML, the Rich Site Summary (RSS) format has quietly become a dominant format for distributing headlines on the Web. Our list of links gives you the tools, tips and tutorials you need to get started using RSS.
  </description>
</item>
```

RSS Channel

- channel provides set of items in some way related
  - e.g., most recent, same topic
  - up to 15 items per channel
- RSS element may contain at most 1 channel
- each channel must contain tags:
  - title channel's title
  - description brief text description of channel
  - link an HTML URL to channel's web site
  - language language encoding for channel (e.g., en-us)
  - item from 1 to 15 items
RSS Channel

- each channel may contain additional tags, including:
  - copyright designates content as copyrighted, names holder
  - pubDate date this channel was published
  - lastBuildDate time of last update to channel
  - image some graphic for channel image
- seems like a good idea, in general

Too Good...

- RSS developed by Netscape for a service it wanted to provide: version 0.90
- then UserLand Software did work to simplify original: version 0.91
- further UserLand refinements: 0.92, 0.93, 0.94
- RSS–DEV adopted 0.90 and evolved it into version 1.0
  - based on RDF
- UserLand most recently produced 2.0
- most versions mutually incompatible
Using RSS: source-side

- to provide RSS feed from your site to others:
  - need xml file defining your RSS channel
  - need your httpd server to know about this file
    - so can be served on demand
- need to keep RSS file up-to-date as content changes on your site
  - can do by hand
  - many tools to semi- or fully automate

Using RSS: client-side

- RSS-clients
  - receive and render XML of the feed
  - maintain local ‘tracking’ info so can know which feeds user subscribes to
    - and (perhaps) also status of feeds
  - may integrate into browser
    - many current browsers will render the xml file as plain text; ugly, but readable, sort of
Sample RSS ‘page’

Better RSS XML Handling

- RSS ‘viewer’
- top picks as recommended by blogsphere.com:
  - for Macintosh: NetNewsWire
  - for Windows: SharpReader
  - for Linux: Straw
  - for web: Bloglines
- common to use Python as part of receiving/rendering process
  - a scripting language similar in some ways to Perl and Tcl
Aggregation

- RSS on server does publishing or syndication
- on client-side, collect feeds from multiple sources
  - “...aggregators collect news, weblog and other feeds over the web and aggregate them so the news items are readable from a single place, regardless of their source.”
  
  -- http://www.nongnu.org/straw
- a site performing aggregation may issue the aggregate as its own RSS feed

Some RSS Feeds...

- tech news: http://slashdot.org/index.rss
- very widespread use with blogs
- fastest growing need type of feed?
Some RSS Feeds...

- tech news: http://slashdot.org/index.rss
- very widespread use with blogs
- fastest growing need type of feed?
  - the podcast
  - simple RSS–pushed feed
  - uses RSS 2.0 “enclosures”
    - can be downloaded independently of main msg

RSS Standard?

- current RSS users should provide support for 1.0 and 2.0
- what about a single standard?
  - e.g., what’s the IETF say?
RSS Standard?

- what about a single standard?
  - e.g., what’s the IETF say?
- IETF:
  - has “atompub” working group
  - no RFCs yet
  - but has Internet-drafts for proposed atom standard
    - The Atom Syndication Format
    - The Atom Publishing Protocol
    - Atom Feed Autodiscovery

Atom

- from “draft-ietf-atompub-format-03:"

```xml
<feed version="draft-ietf-atompub-format-03: do not deploy" xmlns="http://purl.org/atom/ns#draft-ietf-atompub-format-03">
  <title>Example Feed</title>
  <link href="http://example.org/">
    <updated>2003-12-13T18:30:02Z</updated>
  </link>
  <author>
    <name>John Doe</name>
  </author>
</feed>

<title>Atom-Powered Robots Run Amok</title>
<link href="http://example.org/2003/12/13/atom03"/>
<id>venmi://example.org/2003/32397</id>
<updated>2003-12-13T18:30:02Z</updated>
</entry>
</feed>
```