Last Time

- Internet service: WWW
- 802.11 wireless
- Bluetooth
- services discovery, agents, ontologies
- pervasive computing
This Time

- pervasive computing (wrap-up)
- xml & web services
- RFID
- some security notions

Pervasive Computing Tenets

- resource discovery
- passive broadcast of data
- make client do the work
- knowledge of physical location
  - server or resource
  - client
- secure data exchanges
- information appliances
- make the technology disappear
Pervasive Pitfalls

- don’t do it just because you can
  - angry fruit salad
  - user-driven: user must see a useful service not a gee-whiz service that looks great but isn’t useful
- pushing the envelope/shooting yourself in the foot
  - pushing tech beyond its practical limits dooms service
  - not just now, but for some time in future

Making It Work

- permanent Internet connectivity
  - whatever value–add non–Internet services provide, users still want the real thing
- more software intervention on user’s behalf
  - less direct intervention by users themselves
- e.g.,
  - stock–watching program; alerts user only under certain pre–arranged conditions
  - synchronous/asynchronous search
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, automatically
- 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?

xml

- xml tags used to describe content of document
  - not really used to describe layout
- so how can you see the content of an xml document?
  - some html browsers have simple xml rendering ability
  - need another document that describes how to render the tags
    - the same content can thus be rendered differently depending on the style sheet that describes how to render
  - what do these style sheet documents look like?
**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

---

**xslt style sheet**

- xslt to convert xml parts query back to plain text
  - “attribute:value” format
    ```xml
    <?xml version="1.0">
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
        <xsl:output method="text" indent="yes" />
        <xsl:template match="*">
            <xsl:apply-templates />
        </xsl:template>
        <xsl:template match="type">
            <xsl:value-of select="." />
        </xsl:template>
    </xsl:stylesheet>
    ```
xml parsing

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

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

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? 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

- who do you send it to?
  - use Universal Description, Discovery & Integration (UDDI)
  - allows for registering of services (and other) information
  - allows for searching such registries so as to discover a site
  - analogous to a Yellow Pages service
  - http://www.uddi.org

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
  - http://www.w3c.org/TR/wsdl
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
  - http://www.w3c.org/TR/soap

Extending Use of Wireless

- pervasive computing “kitchen helper” application:
  - can present recipes (from local or remote db)
  - can show videos of recipes, techniques
    - focused or general
  - can report which local grocery stores have ingredients
    - how many, prices
  - possibly purchase items at your request
Extending Use of Wireless

- pervasive computing “kitchen helper” application:
- an implementation obstacle: how does app know
  - what you have on your shelves, refrigerator
  - what grocery store has on its shelves
- general problem: inventory management and control

Wireless to the Rescue

- solution:
  - db program needs to know
    - count of items
    - location of items
    - allow items to respond to an application query
  - could place small, active transmitter in/on each item
    - continually transmit an identifying string
    - central receiver collects signals and relays to application software
No Bunnies

- even the best battery eventually goes dead
  - so transmitter would stop
- don’t use batteries
- use *passive* circuit that:
  - is normally off
  - gets power from an RF pulse
  - when powers–up, transmits a unique id number
    - factory programmed, typically ≥64–bits
- and RF–generating receiver
  - to generate the RF pulse
  - listen for reply from passive circuit

Radio–Frequency IDentifiers

- the passive circuits often called *tags*
- two frequency ranges:
  - low: 100s of kHz
  - high: 13.56 MHz (ISO 15693)
- work over small distances
  - from < 1m to few hundred metres
- typical usage setting:

```
    tag  receiver  host–side services
```
RFID tags

- the tag (a.k.a. transponder) devices can take many forms:
  - 3.85 X 23mm glass transponder
  - 64 bits memory
  - 134.2 kHz (low frequency), HDX
  - ≤ 60 cm reader distance
  - 70 ms to read value (typical)
  - −40 to +85 °C operating range

- 45 X 45 mm, 0.355 mm thick
- 256 bits memory (8 X 32 bits)
- 13.56 MHz (high frequency), FDX
- −25 to +70 °C operating range
- uplink at 26.7 kBd
- downlink at 6.2 and 9 kBd (CRC)
- "simultaneous identification" feature, i.e., for collisions
The RFID tag

- may be read-only
- may be read-write
  - allow user to store other information about the tagged object
  - may be write-once
- may also respond to commands
  - not simply emit id on power-up
  - e.g., “stay quiet”

RFID readers

- perform wireless exchange with RFID tags
- perform exchange with host application
  - relays to app info from tag
  - can send to tag (if its writeable) info from host
  - often wired, often fixed in location
  - serial communication to host
  - 600 – 57600 Baud
  - can store $\leq 909$ read ID values
  - can be synchronized to ‘avoid’ other nearby readers
Application 1: textile manufacturing

- Malden Mills uses RFID in strips of material it manufactures to mark flaws
- mark flaw start and end with pair of tags
- slitting machine automatically adjusts action when encounters first tag, re-adjusts on second
- uses (and re-uses) simple read-only RFID tags

Application 2: airport luggage

- McCarran International Airport, Los Vegas:
  - from check-in desk to pick up at destination: use standard luggage tag with embedded RFID tag
  - uses 900 MHz RFID tags
  - use receivers at critical points along handling path, e.g., explosives scan station
- critical issue: cost
  - how much does use of an RFID tag increase cost of handling?
  - supplier quotes $0.25 per tag
Application 3: supply chain

- complete supply–chain tracking
  - see, e.g., Scientific American Jan 2004 pp. 56 – 65
- spaghetti sauce jars tagged as filled/labelled/lidded
- boxes of *n* jars have their own tags, so too, pallets
- passively scanned as:
  - leave food factory
  - arrive at distribution centre
  - leave distribution centre
  - arrive in grocery store

Application 3: supply chain

- when leave grocery store:
  - shopper carries out past check–out scanner
  - shopper has store card in wallet
  - total ‘shopping cart’ scanned and shopper’s account charged
    - and other shopper records updated
- at home, scanner can tell when jar is removed from pantry: time to order more
- tag on container can be used at recycling plant to separate material
  - or to catch cheaters who put jar in trash?
Not Quite There Yet

- problems remain, largely:
  - volume of responses to RF pulse
  - who already answered?
    - at least wastes cpu time
  - cost of tags needs to be lower
  - acceptance issues:
    - privacy concerns
    - labour worries

Privacy Problem

- Benetton wanted to tag clothes for inventory tracking worldwide (warehouses, stores)
  - Philips claimed tags readable only in-store
- Wal Mart and Gillette wanted to tag products
  - Gillette agreed to tag the packaging, not the product
Proposed Guidelines for Tag Use

- four suggested guidelines:
  1. consumers must be warned that a product has an RFID tag
  2. all RFID tags must be recognizeable and removable by customer
  3. tag dies after leaving checkout counter
  4. tag should be on packaging, not product itself

Security Notions

- confidentiality:
  - information can only be seen by those we wish to see it
  - involves OS, network, as well as “common sense”
Security Notions

- confidentiality
- integrity:
  - resources cannot be modified except by authorized users
  - modification can be detected

Security Notions

- confidentiality
- integrity
- authentication:
  - ensuring communicating parties are really who they say they are
Security Notions

- confidentiality
- integrity
- authentication
- availability:
  - computing resources are available to authorized users
  - different authenticated users may have different access privileges
  - and not available to unauthorized users

Security Notions

- confidentiality
- integrity
- authentication
- availability
- non-repudiation:
  - sender cannot later claim not to have sent msg
Security Notions

- confidentiality
- integrity
- authentication
- availability
- non-repudiation
- key distribution:
  - establish session keys among communicating parties

Network Security Threats

- passive threats:
  - release of message contents
  - analysis of traffic
- active threats:
  - masquerade
  - replay
  - modification of msg contents
  - denial of service
Classes of Threat

- view computer as repository of information
  - data, reports, intelligence, ...
  - programs to process information
- normal use:
  ![Diagram of normal use](image)
  - information source
  - information destination
- user initiates request for info, and receives it

Classes of Threat

- **interruption**: information fails to arrive at destination
  ![Diagram of interruption](image)
  - information source
  - information destination
Classes of Threat

- **interception**: information arrives at its destination and also at another, unintended, destination

![Diagram of interception]

- **modification**: information arrives at its destination but only after being modified by a third party

![Diagram of modification]
Classes of Threat

- **fabrication**: information arrives at its destination but not from presumed source

![Diagram showing information source, "bad guy", and information destination]

Security Threats

- what threat is shown here? how countermanded?

![Comic strip showing a zebra in an alligator swamp and a message from a concerned zebra]

© 2003 Stephan Pastis/Distributed by UFS, Inc.
Assets

- hardware:
  - can be stolen (entirely or partially)
  - can be damaged
  - hard to automate protection schemes

Assets

- hardware
- software:
  - may be deleted (especially applications)
  - may be modified
    - not to work at all
    - to behave incorrectly
    - to behave differently
  - may be duplicated
Assets

- hardware
- software
- data:
  - may be destroyed accidentally or maliciously
  - may be modified accidentally or maliciously
  - may be unintentionally released

Security Tools

- physical
  - locked doors, access control to physical resources
Security Tools

- physical
- operational:
  - locked doors, access control to physical resources
  - monitoring
  - backups (RAID, off-site storage)
  - disaster testing/readiness exercises

Security Tools

- physical
- operational
- integrity:
  - use of security hashes, msg digests, checksums
Security Tools

- physical
- operational
- integrity
- confidentiality:
  - encryption

Confidentiality Security Tools

- **encryption**:
  - ‘scrambling’ of bits so that message content is unintelligible to third party
  - un-scrambling operation = decryption
- requires use of $\geq 1$ **keys**
  - keys involved in both encryption and decryption
  - key values must not be known to third parties
- **key exchange**: secure way to communicate key values between communicating parties
Encryption Systems

- receive as input plain text, \( P \)
  - i.e., original, “in the clear” message
- using a key, apply a sequence of well-defined operations to plain text to generate ciphertext \( C \)
- 2 major kinds of system:
  - use one key = symmetric key encryption
  - use two keys = public key encryption (PKE)

Encryption Systems: Keys

- key value must remain secret or any third party with key can decrypt message
- how deliver key to recipient?
  - by hand
  - using another encryption scheme
    - e.g., Diffie–Hellman key exchange
Breaking Encryption Schemes

- bad guys listen to encrypted message and try to decrypt it without key
- need to ‘guess’ key
  - keep trying key values until success
- worst case: have to try them all
- expected value: typically need to try half of all key values
- jackpot: be able to determine key for any message ‘easily’
  - break algorithm, not just individual message

Keys: Size Matters

<table>
<thead>
<tr>
<th>Key Size</th>
<th>1/µs</th>
<th>10^6/µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>35.8 min</td>
<td>2.15 ms</td>
</tr>
<tr>
<td>56</td>
<td>1142 yrs</td>
<td>10.01 hrs</td>
</tr>
<tr>
<td>80</td>
<td>19,154 yrs</td>
<td>7 days</td>
</tr>
<tr>
<td>128</td>
<td>5.4 × 10^{24} yrs</td>
<td>5.4 × 10^{18} yrs</td>
</tr>
<tr>
<td>168</td>
<td>5.9 × 10^{36} yrs</td>
<td>5.9 × 10^{30} yrs</td>
</tr>
</tbody>
</table>

- no one really takes under 80 bits very seriously today
- usually want 128 or better
### DES Iterations

- DES is symmetric-key block cipher
  - originally from IBM
  - developed further by NIST
- was broken by EFF in 1968
- now use triple-DES (TDEA, 3DES)
  - effective key length 168 bits (3 × 56)

### Public Key Encryption

- encryption and decryption algorithms are well known
- encryption key is made public but decryption key is kept secret
- PK algorithms based on easy to compute functions whose inverses are very hard to find
  - e.g., base on problem of factoring two large prime numbers (RSA algorithm)
RSA

key pairs

<table>
<thead>
<tr>
<th>PLAIN TEXT</th>
<th>decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIPHER TEXT</td>
</tr>
<tr>
<td>encryption</td>
<td></td>
</tr>
<tr>
<td>private key</td>
<td>public key</td>
</tr>
</tbody>
</table>

**Who Can You Trust?**

- Suppose Bob encrypts a session key with his private key then Alice’s public key
  - to be sent to Alice
- How does Bob *know* that the key he got as Alice’s public key really is Alice’s public key?
  - because she says it is?
Signing Authorities

- Alice can register her public key with a trusted signing authority:
  - she sends her public key
  - they (somehow) ensure the key is really hers
  - they 'sign' her key attesting its genuineness
  - generate MD5 hash of this message
  - encrypt result with their private key
  - result is a certificate

- now anyone can verify key using signing authority’s public key

Certificates

- X.509 (v3) standard certificate:

<table>
<thead>
<tr>
<th>version</th>
<th>serial number</th>
<th>signature</th>
<th>issuer</th>
<th>validity</th>
<th>subject</th>
<th>subject public key info</th>
<th>issuer unique ID</th>
<th>subject unique ID</th>
<th>extensions</th>
</tr>
</thead>
</table>

- some other types of certificates:
  - PGP: Pretty Good Privacy
    - see, e.g., RFC1991, RFC2440
  - SPKI: Simple Public Key Infrastructure
    - see http://www.ietf.org/html.charters/spki-charter.html
Implementations

- Secure Sockets Layer (SSL):
  - provides certificate-based authentication
    - server → client
    - client → server
  - provides key-exchange
  - provides encryption
  - open-source version available (for all platforms): OpenSSL http://www.openssl.org

Implementations

- Secure Tunnel (stunnel):
  - provide authenticated, SSL-encrypted tunnel
  - transparent to applications communicating
  - works by re-mapping ports, described in a start-up config file, e.g.

  [59900]
  accept = 127.0.0.1:59900
  connect = 129.174.65.2:59900
Stunnel Session Log

59900 started
59900 connected from 127.0.0.1:2011
59900 connecting 129.174.65.2:59900
waitforsocket: ok
SSL state (connect): before/connect initialization
SSL state (connect): SSLv3 write client hello A
SSL state (connect): SSLv3 read server hello A
VERIFY OK: depth=0,
/C=us/ST=virginia/L=manassas/O=gmu/OU=netlab/CN=Charles M Snow
SSL state (connect): SSLv3 read server certificate A
SSL state (connect): SSLv3 read server key exchange A
SSL state (connect): SSLv3 read server certificate request A
SSL state (connect): SSLv3 read server done A
SSL state (connect): SSLv3 write client certificate A
SSL state (connect): SSLv3 write client key exchange A
SSL state (connect): SSLv3 write certificate verify A
SSL state (connect): SSLv3 write change cipher spec A
SSL state (connect): SSLv3 write finished A
SSL state (connect): SSLv3 flush data
SSL state (connect): SSLv3 read finished A
Negotiated ciphers: EDH-DSS-DES-CBC3-SHA
Au=DSS Enc=3DES(168) Mac=SHA1
SSL alert (read): warning: close notify
SSL closed on SSL_read
SSL write shutdown (output buffer empty)
Socket write shutdown (output buffer empty)
SSL alert (write): warning: close notify
SSL_shutdown successfully sent close_notify
Connection closed: 6 bytes sent to SSL, 305 bytes sent to socket
59900 finished (0 left)
NEW System Design

NEWv4 Privacy Design
**NEWv4 Privacy Design**

**on client**: use *stunnel*
- intercepts outbound traffic directed to ports on local loopback
- encrypts it, forwards encrypted traffic to Internet
- receives encrypted traffic, decrypts it and forwards to port on local loopback
- no changes to actual client code itself

**NEW Privacy Design**

**on server** use **Java Secure Sockets Extension (JSSE)**
modified code of basic server to use JSSE for certificate-based authentication and encryption of data
Escrowed Keys

- in 1990s, law enforcement drove idea of:
  - require (by law) everyone to use an NSA developed encryption algorithm (SkipJack)
  - 80-bit encryption
- but has ability to deliver ‘master–key’ to let any msg so encrypted be decrypted by law enforcement
- needed pair of keys to make up ‘master’
  - one held by NIST
  - other held by Dept. of the Treasury
- not put into force; algorithm now openly avbl