This Time

- sample services, cont'd:
  - distributed objects
  - Bluetooth

Evolution of a Service

- client asks for information which must be dynamically generated by consulting 2 different databases

Middleware

- intermediate layer of software between clients and servers
- mediates useful communication between clients and servers
- often plays role of broker between client and some set of servers that may be able to provide answer
  - ideally hiding details from client (at least from user)
Distributed Objects

- some requests will be for objects and/or methods that operate on them
- Java has built in support for this capability
- requiring every distributed object to itself be written in Java
- what if we want to use objects in other/different environments?
- like C++, or Ada
- no single common representation for the objects or their methods

CORBA

- a "distributed object" describes services it can provide via Interface Definition Language (IDL)
- a generic language for describing services
- Object Request Broker, ORB, 'connects' clients (requesters) with servers
- interaction is between ORBS: one on server, one on client
- client-side ORB locates a distributed object sought by client, handles server responses
- server-side ORB registers services, handles requests
- http://www.omg.org

Which Broker?

- Common Object Request Broker Architecture (CORBA) a popular standard for these object brokers

Object Brokers

- CORBA is not alone
  - but is supported by Apple, IBM, Sun, and others
- Microsoft has different 'standard': Common Object Model (COM)
  - interoperates with CORBA
- analogous middleware used for software agents to locate and use services
  - e.g., CMU's RETSINA
- hot topic: pervasive computing depends (in part) on locating services and delivering results

Basic Distributed Communication

- mediated by message passing
  - sender
  - receiver
Basic Distributed Communication

- mediated by message passing
  - sender
  - receiver
- delivery class:
  - unreliable
  - reliable

Basic Distributed Communication

- mediated by message passing
  - sender
  - receiver
- delivery class:
  - unreliable
  - reliable
- process action:
  - blocking
  - non-blocking

More Than Just Data

- an application may want to invoke some processing service on a remote system
- use Remote Procedure Call (RPC)
  - should behave to invoker like a local procedure call
  - like a local procedure, has a standard interface with defined type
  - can check type errors during compile
  - can make client and/or server code more portable
  - only re-write ‘new’ stuff; use same RPC

RPC Issues

- parameter passing
  - call by value is ‘straightforward’
  - e.g., sqrt(25.0)

RPC Issues

- parameter passing
  - parameter representation
    - big-endian vs. little-endian
    - ASCII vs EBCDIC vs UNICODE
    - may be handled by underlying network, may not
RPC Issues

- parameter passing
- parameter representation
- persistence of binding
  - connection between caller and callee: binding
  - binding includes state information
  - non-persistent: lives only as long as caller and callee interact
  - persistent: continues after initial call returns
    - good if have frequent calls from same caller

RPC Issues

- parameter passing
- parameter representation
- persistence of binding
- timing action:
  - synchronous
  - caller does not wait for remote procedure to return
  - receives replies whenever they are delivered
  - requires synchronization mechanism

RPC Issues

- parameter passing
- parameter representation
- persistence of binding
- timing action:
  - distributed objects
    - increasingly, "RPC"-like mechanism used to invoke methods on remote, distributed objects
    - all-Java approach: Remote Method Invocation (RMI)
    - otherwise use request broker (e.g., CORBA)

Some Other Names

- MSRPC, MSRPC2
  - Microsoft's remote procedure call standard
  - latter often referred to as ORPC because provides support for objects

Some Other Names

- MSRPC, MSRPC2
  - SOAP: Simple Object Access Protocol
    - supported by w3c
    - designed by Microsoft, IBM, Sun Microsystems...
    - 3 components:
      1. envelope: contents, recipient, status of msg
      2. encoding rules: how to encode for xfer
      3. platform-neutral RPC mechanism for invoking remote procedure
Some Other Names

- MSRPC, MSRPC2
- SOAP, Simple Object Access Protocol
- Jini:
  - from Sun Microsystems
  - Java-based
  - allows a ‘computation’ to be performed by
  - collection of resources distributed over many
  - networked computers
  - allows a process to migrate from one computer to
  - another
  - clients/servers can use any protocol of their
    choosing

Getting Service

- so far have focused on clients/servers accessed
  over ‘wired’ connections
- requires user (client) to be at ‘fixed’ location
- fastest growing sector of network services is
  targeted at services accessible by portable
  (wireless) devices
- non-voice services for cell phones
- network access at public locations (internet café)
- services for network-capable PDAs

Making Wires Disappear

- the need for cables to interconnect computing devices
  hampers portability and mobile use
- introduced in 1994, Bluetooth is a cable-replacement
  technology using short-range RF wireless
- the Bluetooth Special Interest Group (SIG)
  - formed in 1998 to develop open spec for development
  - and deployment of Bluetooth
  - charter members: Ericsson, IBM, Intel, Nokia and Toshiba
  - now also JiCam, Microsoft, Lucent Technologies and
  Motorolla part of ‘Promoter Group’ within SIG

Bluetooth

- standard supporting wide range of devices
  - computing devices: non-portable, portable, PDAs
  - other Bluetooth devices
  - beyond what might originally have been planned,
    e.g., appliances
- key features:
  - robustness
  - low complexity, ease of use
  - low power
  - low cost
- named after Harald Blåtand, 10th century Danish
  king, united and controlled Denmark & Norway

Networks: Out of the Blue(tooth)

- Bluetooth supports connections both
  - point-to-point
  - point-to-multipoint
- ≥2 units sharing a channel form a piconet
  - one master unit
  - up to 7 slave units
- one unit can participate in ≥ 1 piconet
  - but can only be one of master or slave per piconet
- piconets with overlapping coverage areas form a
  scatternet

Bluetooth Architecture

- Standard Bluetooth architecture model
Bluetooth Radio

- operates in 2.4 GHz ISM band (2400 – 2483.5 MHz)
- range from 10m. to 100m. (i=33 to 328 ft)
- 1 mW to 100 mW transmitting power
- uses frequency-hopping (1600 hops/second)
- TDM for full duplex transmission
- uses GFSK modulation (Gaussian FSK)
- 64 kbps bidirectional on each synchronous voice chan.
- 723.2 kbps in / 57.6 kbps out on asynch data chan. (or, 433.9 kbps symmetrically)
- RF environment can lead to high error rate

Bluetooth Baseband

- manages radio layer, including FH sequences
- performs low-level encryption for secure links
- does packet-over-wireless, including error correction
- synchronize units’ clocks
- discovery of nearby units’ addresses
- establishes connections

Bluetooth Links

- Asynchronous Connectionless (ACL)
  - packet switched connection between master and all active slaves in a piconet
- Synchronous Connection Oriented (SCO)
  - dedicated point-to-point link
  - typically used for voice communication
  - symmetric
  - like circuit-switched connection (i.e., a VC)

Bluetooth: Host Controller

- hardware usually used for lower layers:
  - radio
  - baseband
  - LMP
  - data to LMP conveyed over some bus to host:
    - requires a host controller interface (HCI) on Bluetooth side
    - requires a driver on host computing device side
Bluetooth: L2CAP

- Logical Link Control and Adaptation Protocol
- provides connectionless and connection-oriented services to higher layer protocols
- only supports ACL links
- protocol multiplexing
  - allow multiple apps to use a link between 2 units simultaneously
- segmentation & reassembly
  - packets received from apps segmented as needed
- L2CAP packets are up to 64 Kbytes but baseband’s max payload is 2745 bits

Bluetooth: Applications

- L2CAP directly accessible to apps or via support protocols including:
  - RFCOMM (e.g., emulates serial communication)
  - TCS (e.g., telephony)
  - SDP (service discovery protocol)
- apps may use protocols such as TCP/IP or WAP
- app may use SDP to discover if needed service is available from any accessible remote unit

Connecting in a Bluetooth World

- inquiry:
  - upon arrival in a new setting, unit initiates inquiry to locate access points it can reach
  - any that answer provide their addresses
  - unit selects one
- paging:
  - unit invokes baseband procedure ‘paging’ which synchronizes unit with access point for clock offset and phase in FH, etc.
- link establishment
  - LMP creates link with access point: ACL or SCO, depending on application (e.g., email uses ACL)

Connecting in a Bluetooth World, cont’d

- service discovery:
  - LMP, via SDP, discovers what services access point provides
- L2CAP channel:
  - create a channel with successful result from SDP
  - may be directly used by app, or another protocol may run over it (e.g., RFCOMM)
- RFCOMM channel:
  - created over L2CAP channel if app doesn’t use channel directly

Connecting in a Bluetooth World, cont’d

- security:
  - access point may request “pairing” if it restricts access to particular set of users or offers secure communication to previously registered users
  - requires user to provide ‘PIN’ to access service
  - if using secure mode, encryption invoked
  - a key generated therefrom is used/xmitted
- link available for use
  - can use PPP over RFCOMM (emulates serial link)
  - then can run, e.g., TCP/IP
Services, Services Everywhere...

- wireless devices may have range of services to choose from
  - how identify a particular sought service?
  - how does a service make known what it can provide?
  - how do clients and servers understand each other?
    - to match client needs with server services

Services Discovery Problem

- suppose you are a server
  - e.g., offering outside air temperature (OAT)
  - how do you make known what you can and cannot do?
- suppose you are a client that wants
  - e.g., OAT
  - how do you find out
    - if anyone serves what you want
    - if someone does, if you can use what they serve

Agents

- software acting on user’s behalf: agent
- usually no restriction on how implemented
  - but may want something that can run anywhere
- agent may:
  - directly itself perform the needed action, or,
  - have to find other agents to perform the needed action
    - works with them or ‘outsources’ to them
- need way for:
  - agent to express what it needs
  - resource to advertise what it offers

How Agents Discover Each Other

- ‘whitepages’ directory of agents
  - ‘yellowpages’ identifies what agents for what service
    - a.k.a. middle agents

How Agents Discover Each Other

- ‘whitepages’ directory of agents
- ‘yellowpages’ identifies what agents for what service
  - a.k.a. middle agents
- Service Location Protocol (SLP) [RFC2608]
  - three kinds of agents:
    - service agent, SA, sought by
    - user agent, UA, who uses the
      - directory agent, DA to find SAs
    - can, if no DA, support multicast–based UA–SA discovery
**How Agents Discover Each Other**

- **Universal Plug N Play (UPnP):**
  - uses XML 'forms' to describe device/resource and services it offers
  - devices register by:
    - get IP address from DHCP server
    - broadcast so control point notices and responds
    - sends URL to control point with device/resource/service characteristics
  - doesn't cover service invocation
  - devices/services have to figure out how to interact

- **Jini**
  - similar to SLP, but absolutely requires a DA
  - DA typically available at 'wired' location
    - i.e., may not be suited to completely mobile application
  - biased in favour of agents written in Java
  - for more info, see:
    - http://www.salutation.org
    - http://www.upmp.org

- **Salutation**
  - another central database-type registry
  - independent of network protocol (uses transport 'filters')
  - units talk to 'their' SLM (salutation manager)
    - may be part of same unit
  - all inter-client device communication via SLMs
  - SLMs discover each other and exchange registration info
  - SLMs use transport specific Transport Modules (TM)s to achieve independence of transport media
  - device features described as <attribute:value> pairs

**What Agents Say to Each Other**

- need common way of expressing
  - what agent wants when it is requesting
  - what resource offers when it replies
- taxonomy
  - informal hierarchical relationship expression
    - e.g., retsina:name_of_service or stock:quote:lbm
  - more formal representation: ontology

**Ontology**

- structured way to represent concepts of a domain
  - classes
  - properties describe features/attributes of concepts
    - properties or slots
    - facets (restrictions on properties)
  - knowledge base = ontology + instances of classes
  - some 'standard' ontologies now publically available
  - still largely manually created
  - reference for newcomers to ontologies:
    - "Ontology Development 101: A Guide to Creating Your First Ontology" Noy and McGuinness, Stanford
**Automating Resource Discovery (More)**

- hand-toolled methods capture
  - structure between concepts (horizontal, vertical)
  - semantic quality of concepts
- lots of useful information is already in web pages
  - often that what an agent is looking for
  - agents can’t read (current) web pages very well
- meta-info in current html pages only provides layout information
- introduce additional meta-information to describe page’s semantic content for agents to read ⇒
  semantic web is born

**Who Does This Stuff?**

- a partial list...
  - U. Maryland at Baltimore
    - http://www.ability.org
  - CMU
  - MIT
  - U. Washington
  - Stanford
  - IBM
  - HP (e.g., CoolTown project)
      - rch/pal
    - http://www.cooltown.hp.com
  - Sun Microsystems

**Some Scenarios**

- look at a few scenarios for new services using wireless
- think about what it would take to actually build and implement services like these:

**Pervasive Applications**

- the ‘smart airport’
  - your handheld has your e-ticket number in it
  - on arrival at airport, it gets from airport services:
    - gate for your flight
    - departure time
    - any advisories or other notices related to flight or destination
    - gives you updated summary (user configurable)
    - can guide you through airport (directions, map) to where you need to go
  - need not involve much specific query interaction with airport services ⇒ make client do the work

**Pervasive Applications:**

- smart buildings:
  - your handheld has reserving for hotel room
  - on arrival at hotel, exchange with hotel-server to:
    - confirm arrival
    - get room assignment
    - confirm payment method
    - get key to room (in handheld) if locks use IR or Bluetooth
    - directions to room from hotel entrance
    - notices, advisories, etc., from hotel (e.g., mspa)
  - analogously for arriving at, e.g.,
    - conference center
    - office building

**Pervasive Applications**

- users of smart buildings:
  - visitors
  - employees
    - scheduled maintenance work
  - building notes, e.g., new office or restaurant
  - operators
    - access to maintenance functions (get, set)
  - electric, lights, elevators, HVAC, access points
    - like SNMP, but for building
Pervasive Applications

- help in the kitchen:
  - a wall-mounted display with touch-screen
  - can display recipes
  - can display videos of a recipe being made
  - can check with local grocery store for an ingredient you are missing
  - can’t cook and clean up (yet)
- pervasive computing need not be portable computing
- dedicated device for specific role: kitchen help
- seamless use of Internet when needed

Pervasive Applications

- and two application areas that are already well developed:

Pervasive Applications

- your car:
  - engine, powertrain, brakes all part of on-board network
  - standards being developed
    - in the US as J1851 and ODB-II
    - in Europe as Controller Area Network, CAN
  - CAN is
    - bus architecture network, twisted pair bus
    - three classes:
      - A < 10 kbps for ‘convenience’ features
      - B ≤ 125 kbps for body electronics + diagnostics
      - C ≥ 1 Mbps for mission-critical dynamic systems

CAN can

- deliver data up to 1 Mbps
  - but with variable latency
- alternative approach: Time Triggered Protocols (TTP):
  - each node assigned explicit time slot
  - latency is constant
  - but fixed: time slot assignments cannot be dynamically re-assigned
- LIN: Local Interconnect Network
  - Class A serial, single-wire bus
  - for low-cost, low-traffic nodes
  - spec covers phy layer + xmit protocols + APIs

CAN and LIN

Pervasive Applications

- elder-care residence: identify residents needing care when/where they need it
  - locator badge
  - embedded weight sensors
  - computer display with touch-screen
  - data obtained can be used to check against
    - sudden weight change
    - sleeplessness
  - databases per resident
    - resident vitals (e.g., BP, weight)
    - sudden changes in values flagged for medical atten
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/async search

© Asim Media 2003 11:44