Αρχιτεκτονικές και Τρόποι Μετάδοσης Πληροφορίας

Εισαγωγή

1. ΜΟΝΤΕΛΑ ΔΕΔΟΜΕΝΩΝ
   - Μοντέλα
   - Γλώσσες Ερωτήσεων
   - Επεξεργασία Ερωτήσεων
   - Συστήματα (Lore, Strudel)
2. **ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ**
3. **ΤΡΟΠΟΙ ΜΕΤΑΔΟΣΗΣ**

An IEEE ICDE 2000 Tutorial on

Mobile and Wireless Database Access for Pervasive Computing

Panos K. Chrysanthis
University of Pittsburgh & Carnegie Mellon University
Evaggelia Pitoura
University of Ioannina

panos@cs.pitt.edu **pitoura@cs.uoi.gr**

Outline

- **Motivating Example**
- Issues: Mobility, Wireless Communication, Portability
- Adaptability and Mobile Client-Server Models
- Mobile Access to the Web
- Broadcast data dissemination

Party on Friday

- Update Smart Phone's calendar with guests names.
- Make a note to order food from Dinner-on-Wheels.
- Update shopping list based on the guests drinking preferences.

- Don't forget to swipe that last can of beer's UPS label.
- The shopping list is always up-to-date.

Party on Friday

- AutoPC detects a near Supermarket that advertises sales.

- It accesses the shopping list and your calendar on the Smart Phone.
- It informs you the soda and beer are on sale, and reminds you. that your next appointment is in 1 hour.
- There is enough time based on the latest traffic report.

Party on Friday

- TGIF...
- Smart Phone reminds you that you need to order food by noon.
- It downloads the Dinner-on-Wheels menu from the Web on your PC with the guests' preferences marked.
- It sends the shopping list to your CO-OP's PC.
- Everything will be delivered by the time you get home in the evening.

Outline

- Motivating Example
- **Issues: Mobility, Wireless Communication, Portability**
- Adaptability and Mobile Client-Server Models
- Mobile Access to the Web
- Broadcast data dissemination

Mobile Applications

- Expected to create an entire new class of applications
  - new massive markets in conjunction with the Web
  - Mobile Information Appliances - combining *personal computing* and *consumer electronics*
- Applications:
  - *Vertical*: vehicle dispatching, tracking, point of sale
  - *Horizontal*: mail enabled applications, filtered information provision, collaborative computing...

Mobile and Wireless Computing

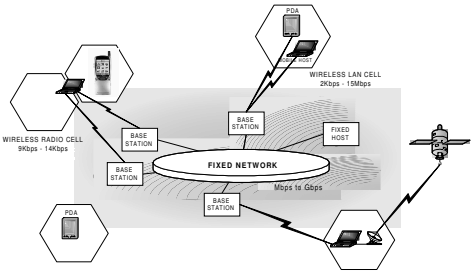
- Goal: Access Information Anywhere, Anytime, and in Any Way.
- Aliases: Mobile, Nomadic, Wireless, Pervasive, Invisible, Ubiquitous Computing.
- Distinction:
  - Fixed wired network: Traditional distributed computing.
  - Fixed wireless network: Wireless computing.
  - Wireless network: Mobile Computing.

➢ Key Issues: Wireless communication, Mobility, Portability.

Wireless Communication

- *Cellular* - GSM (Europe+), TDMA & CDMA (US)
  - FM: 1.2-9.6 Kbps; Digital: 9.6-14.4 Kbps (ISDN-like services)
- *Public Packet Radio* - Proprietary
  - 19.2 Kbps (raw), 9.6 Kbps (effective)
- *Private and Share Mobile Radio*
- *Wireless LAN* - wireless LAN bridge (IEEE 802.11)
  - Radio or Infrared frequencies: 1.2 Kbps-15 Mbps
- *Paging Networks* – typically one-way communication
  - low receiving power consumption
- *Satellites* – wide-area coverage (GEOS, MEOS, LEOS)
  - LEOS: 2.4 Kbps (uplink), 4.8Kbps (downlink)

Mobile Network Architecture



Future Wireless Communication

Technology	Service	Data Capability	Expected Deployment
GSM	High-speed circuit-switched	28.8 to 56 Kbps service likely	Limited 1999 and 2000
	General Packet Radio Service	IP and X.25 over Kbps	Rollout of service 2001
	Enhanced Data Rates for GSM Evolution	IP to 384 Kbps. Roaming with IS-136 networks	Rollout of service 2002
	Wideband CDMA	Similar to EDGE but adds 2Mbps indoor capability	Initial in 2002 or 2003
IS-136 TDMA	EDGE	IP to 384 Kbps. Roaming with GSM networks	Initial in 2002 or 2003
	WCDMA or Wideband	Similar to EDGE but adds 2Mbps indoor capability	No stated deployment plans
CDMA	IS-95B	IP communications: 64 Kbps	Japanese markets in 2000
	CDMA2000-1XRTT	IP communications: 144 Kbps	Trial in 2001, rollout 2002
	CDMA2000 - 3XRTT	IP: 384 Kbps(out), 2 Mbps(in)	Initial in 2002 or 2003

Source: Rysavy Research, 1999

Wireless characteristics

- Variant Connectivity
  - Low bandwidth and reliability
- Frequent disconnections
  - predictable or sudden
- Asymmetric Communication
  - Broadcast medium
- Monetarily expensive
  - Charges per connection or per message/packet

➢ Connectivity is weak, intermittent and expensive

Portable Information Devices

- PDAs, Personal Communicators
  - Light, small and durable to be easily carried around
  - dumb terminals [InfoPad, ParcTab projects], palmtops, wristwatch PC/Phone, walkstations
- will run on AA+ /Ni-Cd/Li-Ion batteries
- may be diskless
- I/O devices: Mouse is out, Pen is in
- wireless connection to information networks
  - either infrared or cellular phone
- specialized HW (for compression/encryption)

Portability Characteristics

- Battery power restrictions
  - transmit/receive, disk spinning, display, CPUs, memory consume power
- Battery lifetime will see very small increase
  - need energy efficient hardware (CPUs, memory) and system software
  - planned disconnections - *doze mode*

➢ Power consumption vs. resource utilization

Portability Characteristics

- Resource constraints
  - Mobile computers are resource poor
  - Reduce program size – interpret script languages (Mobile Java?)
  - Computation and communication load cannot be distributed equally
- Small screen sizes

➢ Asymmetry between static and mobile computers

Mobility Characteristics

- Location changes
  - location management - cost to locate is added to communication
- Heterogeneity in services
  - bandwidth restrictions and variability
- Dynamic replication of data
  - data and services follow users
- Querying data - location-based responses
- Security and authentication

➢ System configuration is no longer static

What Needs to be Reexamined?

- Operating systems
- File systems
- Data-based systems
- Communication architecture and protocols
- Hardware and architecture
- Real-Time, multimedia, QoS
- Security
- Application requirements and design
- PDA design: Interfaces, Languages

Recurrent Themes

- Handling disconnections (planned failures?)
  - caching strategies
  - managing inconsistencies
- Delayed write-back and prefetch: use network idle times
  - increases memory requirements
- Buffering/batching: allows bulk transfers
- Partitioning and replication
  - triggered by relocation
- Compression: increase effective BW
  - increases battery power requirements
- Receiving needs less power than sending

Outline

- Motivating Example
- Issues: Mobility, Wireless Communication, Portability
- **Adaptability and Mobile Client-Server Models**
- Mobile Access to the Web
- Broadcast data dissemination

Mobility in Db Applications

- Need to *adapt* to constantly changing environment:
  - network connectivity
  - available resources and services
- By varying and (re)negotiating:
  - the partition of duties between the mobile and static elements
  - the quality of data available at the mobile host

  Example: *Fidelity* (degree to which a copy of data matches the reference copy at the server)

Adaptability

Where should support for mobility and adaptability be placed?

Laissez-Faire

(-) applications must be re-written which may be very complicated

(-) no focal point of control to resolve potentially incompatible application demands or to enforce limits on resource usage

Application Transparent

(+) existing applications continue to work unchanged

(-) too general, cannot take advantage application semantics

(-) may not be attainable (e.g., during a long disconnection)

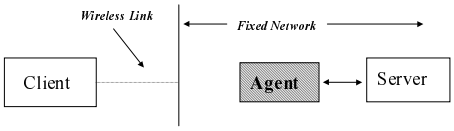
Adaptive Applications

- Need:
  - Measurement of QoS and communication with application
    - A mechanism to **monitor** the level and quality of information and **inform applications** about changes.
  - Programmer Interface for Application-Aware Adaptation
    - Applications must be **agile**: able to reveive events in an asynchronous manner and react appropriately
  - A central point for managing resources and authorizing any application-initiated request.

Application Awareness

- Need for.
  - A mechanism to **monitor** the level and quality of information and **inform applications** about changes.
  - Applications must be **agile**: able to reveive events in an asynchronous manner and react appropriately (triggers)
  - A **central point** for managing resources and authorizing any application-initiated request.

C-SA-C: Server-side Agent



- ***C-SA-C***: The Client/Server-side Agent/Server Model
- Splits the interaction between the mobile client and server: *client-agent* and *agent-server*
  - different protocols for each part of the interaction
  - each part may be executed independently of the other

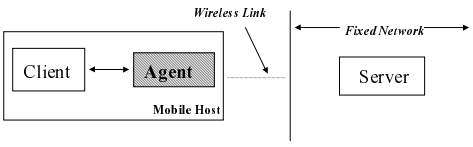
Responsibilities of the Agent

- Messaging and queying
- Manipulate data prior to their transmission to the client:
  - perform data specific compression
  - batch together requests
  - change the transmission order

Role of the Agent

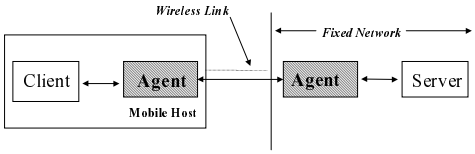
- **Surrogate or *proxy* of the client**
  - Any communication to/from the client goes through the agent
  - Offload functionality from the client to the agent
- **Application (service) specific**
  - provides a mobile-aware layer to specifc services or applications (e.g., web-browsing or database access)
  - handles all requests from mobile clients
- **Filters**
  - provide agents that operate on protocols
  - E.g., an MPEG-agent or a TCP-agent

C-CA-S: Client-side Agent



- *C-SA-S*: The Client/Client-side Agent/Server Model
  - caching
  - background prefetching and hoarding
  - various communication optimizations

C-I-S: Client & Server Agents



- *c-I-S*: Client/Intercept/Server Model
  - Caching, prefetching etc
  - various communication optimizations at both ends
    - E.g., asynchronous queued RPC
  - relocate computation between the agents
  - Client interoperability

Mobile Agents

- Mobile agents are migrating processes associated with an itinerary
  - dynamic code and state deployment
- Implement the agents of the previous architectures as mobile agents, E.g.,
  - server-side agents can relocate during handoff
  - client-side agent dynamically move on and off the client
    - Relocatable dynamic objects (RDO) [Rover]
- Implement the communication using mobile agents:
  - clients submit/receive mobile agents to/from the server
  - E.g., Compacts [Pro-Motion]

Outline

- Motivating Example
- Issues: Mobility, Wireless Communication, Portability
- Adaptability and Mobile Client-Server Models
- **Mobile Access to the Web**
- Broadcast data dissemination

Mobile Access to the Web

- Three-tier Architectures: *Client - Web Server - Data Server*
- Web Server can act like a server-side agent
  - Prefetching at its cache can hide some latency
  - Scripts at the Web server can perform user-specified filtering and processing.
- Most solutions use a ***Web proxy*** to avoid any changes to the browsers and servers.
  - **Pythia** [Fox96]
  - **Mobile Browser** (MOWSER) [Joshi 96]
    - ***Distillation***: highly lossy, real-time,datatype specific compression that preserves semantic content
  - **WebExpress** [Housel 97]

WebExpress

- Utilizes the *C*-I-S Model
- Goals: reduce traffic volume and reduce latency
- Intercept any http request and perform four optimizations:
  - *Caching* at both CSA & SSA of graphics and html objects
  - *Differencing*: only changes are communicated
  - *Long-live TCP/IP Connection*: CSA & SSA use a single TCP connection
  - *Header reduction*: SSA includes the required browser capabilities. They are not sent by the CSA.
- While disconnected (off-line mode) uses CSA cache

Outline

- Motivating Example
- Issues: Mobility, Wireless Communication, Portability
- Adaptability and Mobile Client-Server Models
- Mobile Access to the Web
- **Broadcast data dissemination**

Information Dissemination

Goal : Maximize query capacity of servers,
minimize energy per query at the client.

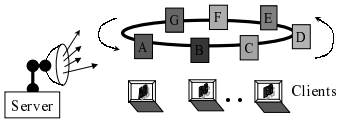
Focus: Read-only transactions (queries).

- Clients send update data to server
- Server resolves update conflicts, commits updates

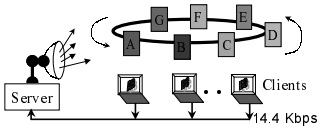
1. **Pull**: PDAs *demand*, servers *respond*.
   - backchannel (uplink) is used to request data and provide feedback.
   - poor match for asymmetric communication.

Information Dissemination

2. **Push**: Network servers *broadcast data*, PDA's *listen*.
   - PDA energy saved by needing receive mode only.
   - scales to any number of clients.
   - data are selected based on profiles and registration in each cell.



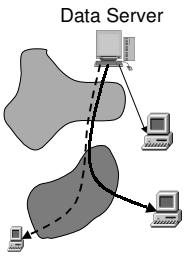
Information Dissemination



3. **Combinations Push and Pull** (Sharing the channel).
   - *Selective Broadcast*: Servers broadcast "hot" information only.
     - "publication group" and "on-demand" group.
   - *On-demand Broadcast*: Servers choose the next item based on requests.
     - FCFS or page with maximum # of pending requests.

Broadcast Data Dissemination

- business data, e.g., Vitria, Tibco
- election coverage data
- stock related data
- traffic information
- sportscasts, e.g., Praja
  - Datatacycle [Herman]
  - Broadcast disks



Organization of Broadcast data

- **Flat**: broadcast the union of the requested data cyclic.

A	B	C

- **Skewed** (Random):
  - broadcast different items with different frequencies.
  - goal is that the inter-arrival time between two instances of the same item matches the clients' needs.

A	A	B	C

Broadcast Disks

- **Multi-Disks Organization** [Acharya et. al, SIGMOD95]
  - The frequency of broadcasting each item depends on its access probability.
  - Data broadcast with the same frequency are viewed as belonging to the same disk.
  - Multiple disks of different sizes and speeds are superimposed on the broadcast medium.
  - No variant in the inter-arrival time of each item.

Selective Tuning

- Basic broadcast access is *sequential*
- Want to minimize client's ***access time*** and ***tuning time***.
  - active mode power is 250mW, in doze mode 50μW
- What about using database access methods?
- *Hashing*: broadcast hashing parameters h(K)
- *Indexing*: index needs to be broadcast too
  - "self-addressable cache on the air"
    (+) "listening/tuning time" decreases
    (-) "access time" increases

Access protocol

- Index buckets hold the directory, data buckets hold data.
- User tunes in to find out when a needed index bucket is broadcasted.
- Synchronize by accessing a pointer that tells the user when to tune in for the data.
- After you synchronize you must access the data in the same broadcast.
- Tune in to the data at the right time.

Access Protocols

- Two important factors affect access time:
  - Size of the broadcast
  - Directory miss factor - you tune in before your data but after your directory to the data!

Trade-Off: ⇓ Size means ⇑ Miss factor
Trade-Off: ⇑ Size means ⇓ Miss factor

(1,M) Indexing

- We broadcast the index M times during one version of the data.
- After broadcasting 1/M of the file we broadcast the index.
- ~ All buckets have the offset to the beginning of the next index segment.
- Data access protocol for record with key Q
- Listen to current bucket.
- Read the offset.
- Go to sleep till the next index segment
- From the index determine when Q arrives.
- (May have to follow several levels of indexes.)
- Tune in again to pick up record.

Distributed Indexing

- Goal: Let's cut down on the replication of index material
- Solution: It is sufficient to broadcast only the portion of the index which describes the data segment that follows. No replication.
- Problem with the no replication approach.
  - User can't determine when relevant indexing is coming!
  - Number of probes grows linearly with number of levels in tree, and
  - the number of pointers in an index bucket. Very bad.
- New Solution: Distributed indexing. Divide the index into:
  - replicated top l levels
  - non-replicated bottom 4-l levels
  - Result: * Index overhead vs. + tuning time

Flexible Indexing

- broadcast divided into p data segments.
- data items are assumed sorted
- a control index is associated with each segment
  - binary control index to determine the data segment
  - local index to locate the specific item within the segment

Indexing

- (1,M) Indexing:
  - We broadcast the index M times during one version of the data.
  - All buckets have the offset to the beginning of the next *index segment.*
- Distributed Indexing
  - Cuts down on the replication of index material
  - Divides the index into:
    - replicated top $\mathcal{L}$ levels, non-replicated bottom 4-$\mathcal{L}$ levels
- Flexible Indexing
  - Broadcast divided into p data segments with sorted data.
  - A *binary control index* is used to determine the data segment
  - A *local index* to locate the specific item within the segment

Caching in Broadcasting

- Data are cache to improve access time
- Lessen the dependency on the server's choice of broadcast priority
- Traditionally, clients cache their "hottest" data to improve hit ratio
- Cache data based on ***PIX***:
  Probability of access (P)/Broadcast frequency (X).
- Cost-based data replacement is not practical:
  - requires perfect knowledge of access probabilities
  - comparison of *PIX* values with all resident pages
- Alternative: ***LIX***, LRU with broadcast frequency
  - pages are placed on lists based on their frequency (X)
  - lists are ordered based on L, the running avg. of interaccess times
  - page with lowest ***LIX*** = L/X is replaced

Prefetching in Broadcasting

- Client prefetch page in anticipation of future accesses
- No additional load to the server and network
- Prefetching instead of waiting for page miss can reduce the cost of a miss
- PT prefetching heuristic [Acharya et al. 96]
  - ***pt***: Access Probability (P) * period (T) before page appears next
  - A broadcast page b replaces the cached page c with lowest ***pt*** value
- Team tag - Teletext approach [Ammar 87]
  - Each page is associated with a set of pages most likely to be requested next
  - When p is requested, D (D:cache size) associated pages are prefetched
  - Prefetching stops when client submit a new request

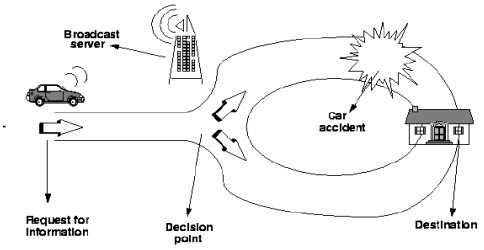
Cache Invalidation Techniques

- **When?**
  - Synchronous: send invalidation reports periodically
  - Asynchronous: send invalidation information for an item as soon as its value changes; E.g., *Bit Sequences* [Jing 95]
- **To whom?**
  - Stateful server: to affected clients
  - Stateless server: broadcast to everyone
- **What?**
  - invalidation: only which items were updated
  - propagation: the values of updated items are sent
  - aggregated information/ materialized views

Synchronous Invalidation

- Stateless servers are assumed.
- Types of client: Workalcholic and sleepers [Barbara Sigmod 94]
- Strategies:
  - **Amnestic Terminals**: broadcast only the identifiers of the items that changed since the last invalidation report
    abort T, if x ϵ RS(T) appears in the invalidation report
  - **Timestamp Strategy**: broadcast the timestamps of the latest updates for items that have occurred in the last w seconds.
    abort T, if ts(x) > tso(T)
  - **Signature Strategy**: broadcast signatures.
    A signature is a compressed checksum similar to the one used for file comparison.

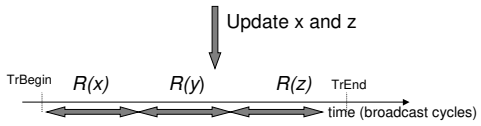
Time Constrained Broadcasts



Client-Server Communication Model

- ***Asymmetric communication environments***
  - **Server broadcasts data to clients using high bandwidth broadcast links**
  - **Clients listen to the broadcast to fetch data**
  - **Clients communicate with the server using low bandwidth upstream links**
- ***Update handling***
  - **Clients send update data to server**
  - **Server resolves update conflicts, commits updates, and broadcasts updates to clients**

Semantic Consistency



Are *x*, *y*, and *z* mutually consistent?

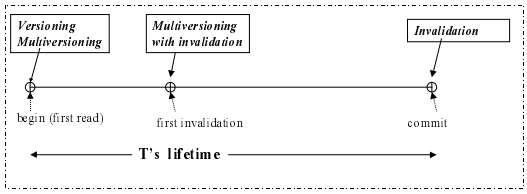
Consistency and Currency

- Only committed data are included in the broadcast
- Does a client read current and consistent data?
- Currency interval is the fraction of bcycle that updates are reflected
- Span(T) is the # of currency intervals from which T read data
- if Span(T) = 1, the T is correct (read consistent data)
  else ?
  … several consistency models

Consistency Criteria

- Latest value: clients read the most recent value of a data item [Garcia-Molina TODS82, Acharya VLDB96]
- Serializability: Certification reports [Barbara ICDCS97]
- Update consistency: clients commit of their reads are not invalidated – read mutually consistent data
  - F-Matrix method [Shanmugasundaram SIGMOD99]
- 2-level serializability: Each client is serializable with respect to the server
  - SGT method [Pitoura&Chrysanthis ICDS99]
  - Multiversion [Pitoura&Chrysanthis VLDB99]

Currency in Multiversion Schemes



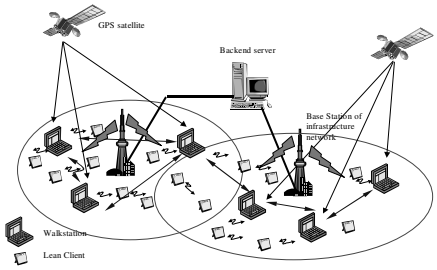
Adaptive Hybrid Broadcast

- Cycle-based, bidirectional hybrid broadcast server
- Issues:
  - Dynamic computation of bandwidth allocated to each broadcast mode
  - Dynamic classification of data items (periodic vs. on-demand)
  - Scheduling periodic and on-demand broadcasts

An Approach

- After each broadcast cycle, items classified as *periodic* or *on-demand*, depending on bandwidth savings expected
- Periodic broadcast occupies up to ***BWThreshold***
- Periodic broadcast program is computed to satisfy all deadlines of periodic data
- On-demand broadcast uses on-line EDF (Earliest Deadline First) algorithm + batching

Broadcast Miss

- Broadcast as an ***air-cache*** for storing frequently requested data
- Continoulsy adjust the broadcast content to match the database hot-spot
- How? By observing the ***broadcast misses*** - requests for data not on the broadcast

Field Computing