Data Management for Pervasive Computing

Mitch Cherniack Brandeis University
Mike Franklin UC Berkeley
Stan Zdonik Brown University

VLDB, Rome, Italy
September 11, 2001

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

What is Pervasive Computing?

- **" … make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it."**
- **"Ubiquitous (pervasive) computing is roughly the opposite of virtual reality. Where virtual reality puts people inside a computer-generated world, ubiquitous computing forces the computer to live out here in the world with people."**

– Mark Weiser, Xerox PARC

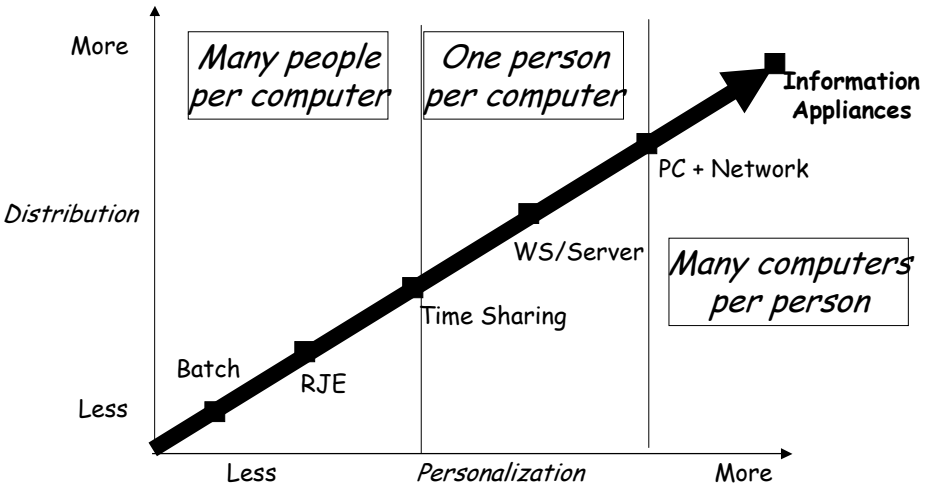
What is Pervasive Computing?

"Pervasive computing is a term for the strongly emerging trend toward:

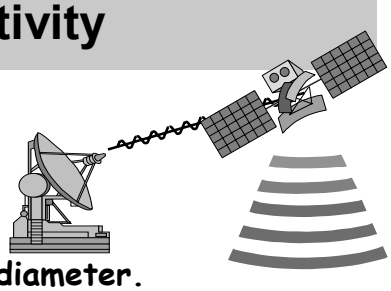
- Numerous, casually accessible, often invisible computing devices
- Frequently mobile or embedded in the environment
- Connected to an increasingly ubiquitous network structure."

– *NIST, Pervasive Computing 2001*

Ubiquitous Computers

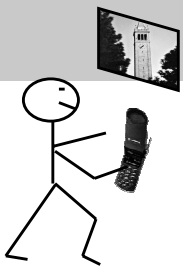


Ubiquitous Connectivity



- **Tremendous improvements in Internet backbone bandwidth and reductions in diameter.**
- **Broadband connectivity to the home and office (i.e. the "last mile") is being solved.**
- **Wireless technologies are enabling anytime-anywhere connectivity.**

Ubiquitous Data Access



- **But, ubiquitous computing and connectivity aren't worth much without *ubiquitous data access*.**
- **"Fundamentally, the ability to access all information from anywhere and have ONE unified and synchronized information repository is critical to making appliances useful." *Hambrecht and Quist, iWord* , 3/99**
- **Ubiquitous data access will put existing data management techniques to the test, in all aspects – searching, location, reliability, consistency, …**

What is Data Management?

- **Intelligent use of** scarce resources **to enhance data access.**
  - Latency, correctness, relevance
- **Exploit:**
  - semantics of application.
  - semantics of data
- **Example techniques:**
  - storage structures (e.g., clustering)
  - indexing
  - cache management
  - replication

Why Does Pervasive Computing Need Data Management?

- **Resource limitations**
  - Bandwidth
  - Memory
- **Scalabilbity**
- **Correctness concerns**
- **Combining many sources**

Optimization + Careful resource sharing

Transactions/imprecision

Data integration

Devices and their characteristics

- **PDA's**
  - Small memory
  - Intermittent connection
  - User data input
  - Access large info. Source
  - Produce small amt. of data
- **MP3 players/ cameras, etc**
  - Dedicated to specific task
  - Specialized processing (jitter elimination)
- **Sensors**
  - Dedicated to simple measurements
  - Push vs. pull
  - Limited computing power
  - Produce streams

Example Application: TRAVELER

- **Setting:**
  - A traveler with a PDA and a wireless connection in a city at dinner time.
- **Problem:**
  - Show restaurants in vicinity that traveler will enjoy and that have less than a 15 minute wait.
- **Issues:**
  - Expressing traveler's eating profile.
  - Getting up to the minute info about occupancy.

Example Application: TRAFFIC

- **Setting:**
  - Cars equipped with GPS and route planning computer.
- **Problem:**
  - Help me get where I'm going most efficiently.
- **Issues:**
  - How does info get to 200,000 cars efficiently?
  - What is the architecture?
    - Cars talk to kiosks
    - Cars talk to other cars
    - Satellite feeds cars

Example Application: Data Recharging

- **Setting:**
  - People with PDA's working on business tasks.
- **Problem:**
  - How to recharge PDA with most relevant data without user intervention.
- **Issues:**
  - Selecting an optimal charge out of a potentially large set of objects with dependant utilities.
  - Picking the most important items first just in case there's a disconnection.

Example Application: Bio-Sensing

- **Setting:**
  - Thousands of soldiers, each with (tens of) sensors in their clothes and on their body.
- **Problem:**
  - Remote triage
- **Issues:**
  - Dealing with unsynchronized reporting intervals.
  - Integrating historical data with "now" data.
  - Controlling the motion of data through mostly wireless networks.

Aspects of Pervasive Environments: (1) Mobility of Sources and Consumers

- Physical connection point to network is always changing.
- There may be times when sources or consumers are disconnected.
- Data interests may change with shifting location.

Aspects of Pervasive Environments: (2) User and Context Awareness

- Need to keep track of and communicate user's state.
- State can be complex – based on position, time, history, workflow.
- Data management decisions can become invalid based on new user context.
  - e.g., caching

Aspects of Pervasive Environments: (3) Lots of Cheap Constrained Devices

- To keep cost and size low, devices will have limited computing power.
- Applications will necessarily run at higher levels in the network where more capacity resides.
- You get what you pay for – cheap devices can be unreliable.

Aspects of Pervasive Environments: (4) Monitoring and Effecting the World

- Large class of applications that have to do with reacting to devices and the model of the world that they suggest.
- Must support large numbers of monitors.
- Monitors must be able to deal efficiently with time-series data.

Aspects of Pervasive Environments: (5) Increasingly Ubiquitous Network

- **The network is everywhere – you will mostly be connected, but characteristics can very a lot.**
- **Bandwidth is <u>not</u> the only limiting factor.**
- **Will require novel data management techniques**
  - e.g., data delivery choice (push vs. pull)
  - e.g., profile-based caching

Requirements: Applications

- **Support for Dynamic Collaborations**
  - relevant devices/information sources can change.
  - trusting collaborators you don't know?
  - finding new collaborators
- **Support for Location-Centric Appl's**
  - Soon every device in the universe will have a GPS.
  - What I care about might change radically depending on where I or my possessions are.
  - Need for specialized operators.
  - Need for Tracking and Monitoring services

Requirements: Adaptivity

- **Adapting to Change & Unpredictability**
  - resource discovery
  - dynamic re-optimization.
  - less than perfect
- **Adapting to User Needs and Roles**
  - Discovering user needs
  - Representing user needs (language)
  - Processing user needs

Requirements: New Semantics

- **Event Stream and Data Stream Processing**
  - Devices can generate data repeatedly (= stream).
  - Streams enable the need for continuous queries.
  - New opportunities for query processing
- **Flexible Sharing Semantics**
  - Many different types of data
  - Different requirements for consistency.
  - ACID transaction semantics not appropriate for most.

Other Requirements

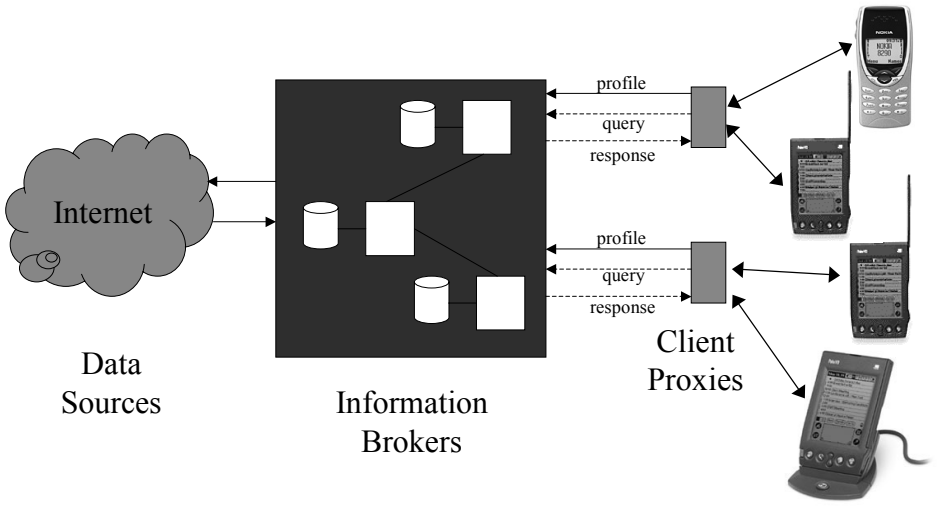
- **Responsiveness**
- **Performance**
- **Scalability**
  - data explosion is outstripping Moore's Law
  - prefetching
  - trade quality
- **Reliability**
  - bounded error
- **Availability**
  - alternative sources
- **Managability**
  - autoadmin

Architectural Context

A key concept is that of an *Overlay Network*

1. "application-level" network built on top of Internet protocols; interacts with the "regular" internet.
2. May use both public and private communication links.
3. Exploits *"Data Centers"* deployed around the world.
4. Content Routing can be done at the application level so can be based on application and data semantics.
5. Caching, Prefetching, Staging, etc. can be done transparently.
6. E.g., CDNs such as Akami, FastForward Networks

Network Components



Background: DBIS Framework

Outgrowth of "Broadcast Disks" project. [SIGMOD 95]

Framework proposed in OOPSLA 97 (Franklin & Zdonik)

Toolkit Developed and Demonstrated at SIGMOD 99

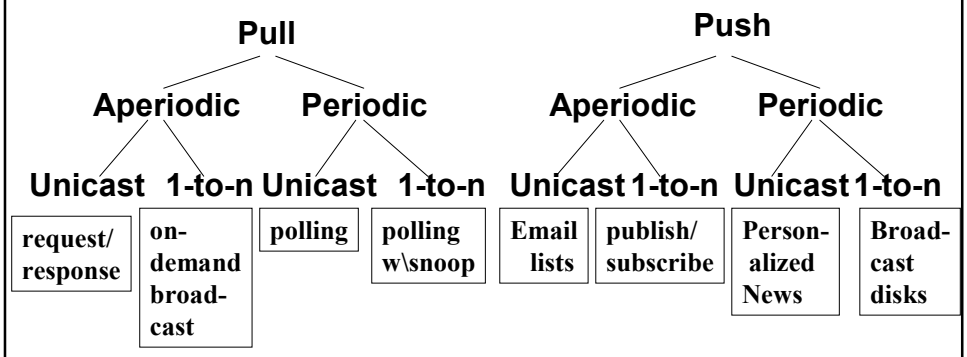
The DBIS Framework is based on three fundamental principles:

1) No one data delivery mechanism is best for all situations (e.g., apps, workloads, topologies).

2) Network Transparency: Must allow different mechanisms for data delivery to be applied at different points in the system.

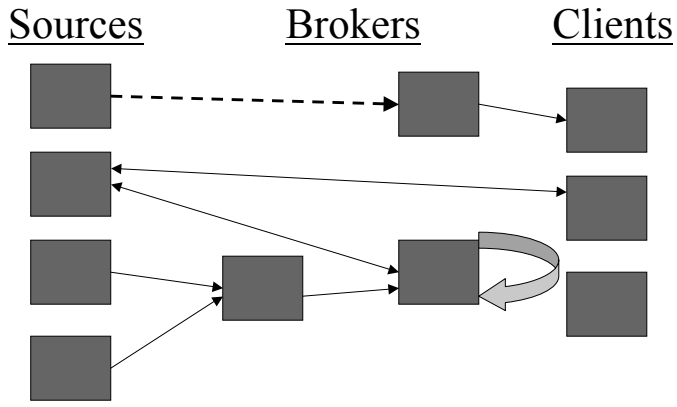
3) Topology, routing, and delivery mechanism should vary adaptively in response to system changes.

Data Delivery Mechanisms



Dimensions are largely orthogonal – all combinations are potentially useful.

Network Transparency



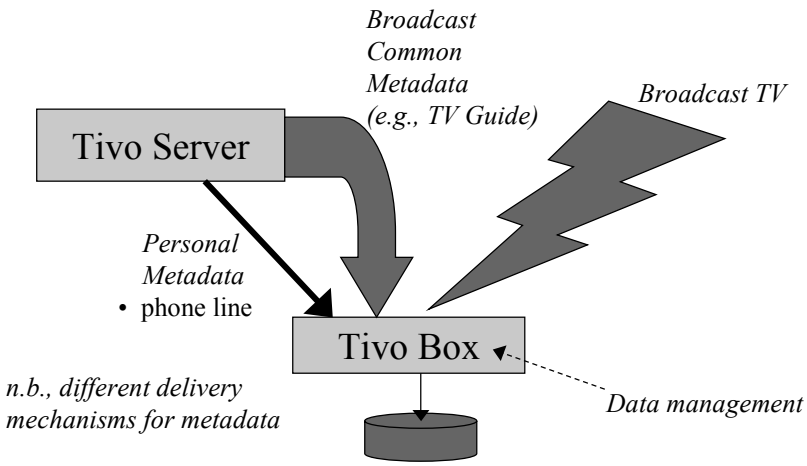
A fundamental principle for systems design:

Type of a link matters only to nodes on each end.

More on Brokers

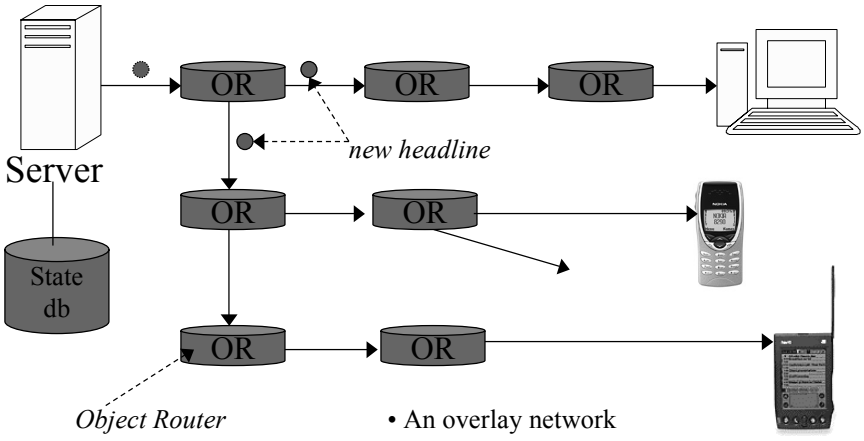
- **Brokers are middleware components that can act as both clients *and* servers.**
- **Must support data caching**
  - Needed to convert pushed-data to pulled-data
  - Also allows implementation of hierarchical caching
- **Profile Management**
  - Allow informed data management: push, prefetch, staging, etc.
- **Profile Matching**
  - No profile language sufficient for all applications.
  - Need an API for adding app-specific profiling

Example: Tivo



Example: Bang Networks

<H1id=Bang$MyLiveTag>Headline<\H1>



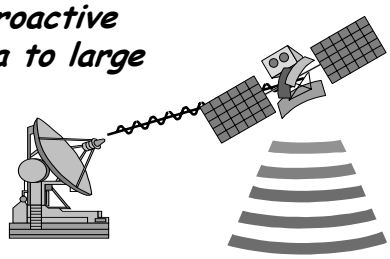
- An overlay network
- OR's implement multicast

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Data Dissemination

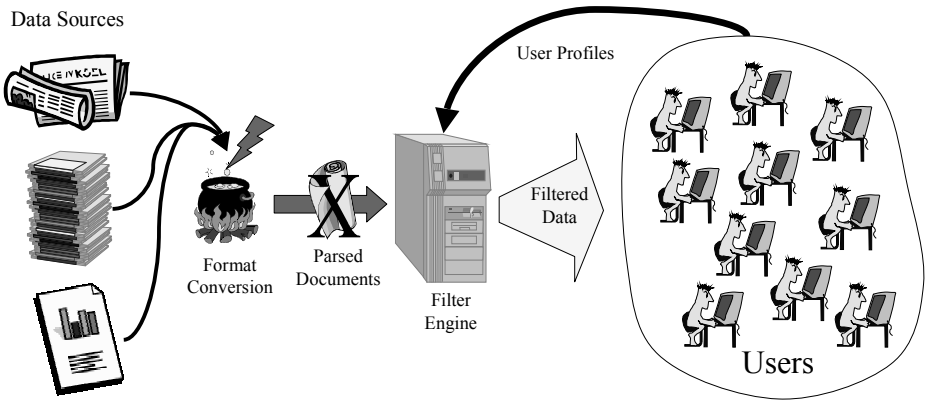
- **Data Dissemination is the *proactive distribution of relevant data to large numbers of users.***
  - Stock and sport tickers
  - Personalized news delivery
  - Traffic information systems
  - Software distribution
  - "real-time" business processes



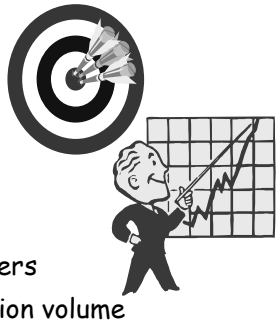
- **Main Issues**
  1. How to represent user interests?
  2. How to match data/events to interests?
  3. How to distribute the data to users?

Anatomy of a Dissemination System

"the right data to the right people at the right time"



Dissemination Challenges



- Accuracy
  - Profile language expressiveness
  - Matching accuracy
  - Ease of maintenance
- Scale
  - Would like to support millions of users
  - Need to handle increasing information volume
    - matching efficiency
    - delivery mechanisms
- Reliable Delivery esp. w/movement & disconnection
- Quality of service/real time guarantees

Dissemination Technologies

- **Information Retrieval**
  - Selective Dissemination of Information (SDI)
  - Information Filtering
  - Document Routing
- **Database Systems**
  - Continuous Queries
  - Active Databases and Event Processing
  - Stream/time-series Query Processing
  - Cache consistency maintenance
- **Networking and Systems**
  - Publish/Subscribe
  - Notification Systems
  - Multicast
  - PDA Synchronization (e.g. AvantGo)

User Interest Specification

- **Information Retrieval**
  - Bag of Words – Boolean Model
  - Bag of Words – Vector-space Model (similarity)
  - Structured Docs – SGML, XML
- **Database Systems**
  - Extensions to SQL
    - Temporal
    - Continuous
  - Xpath, Xquery
- **Networking and Systems**
  - Simple Category("channel")/Keyword approaches, e.g. ("business.stock.quotes(IBM)")

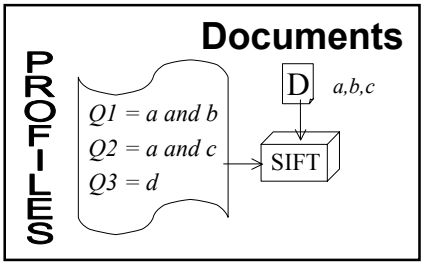
Interest Specification - Issues

- Expressiveness of query language dictates:
  - Accuracy of data delivery (i.e., avoid spamming)
  - Efficiency of matching (scalability).
- Structure? (e.g., Xpath vs. bag of words)
- How to represent priorities and requirements?
- How much "memory" is needed?
  1. no history
  2. windowed operators (i.e., bounded history)
  3. changes and trends in historical data

SDI Profile Matching - SIFT

- Stanford Information Filtering Tool
  overview: [Yan & Garcia-Molina, TODS 2000]
- An early implementation of SDI for disseminating netnews articles.
- Explored both *Boolean* and *Similarity-based* matching models.
- Focus on efficient matching of queries and documents with centralized and distributed filters.
- Pioneered approach of:
  "Index the queries, not the data"
  - Traditional query processing turned on its head.

Profile Matching



Strategy:

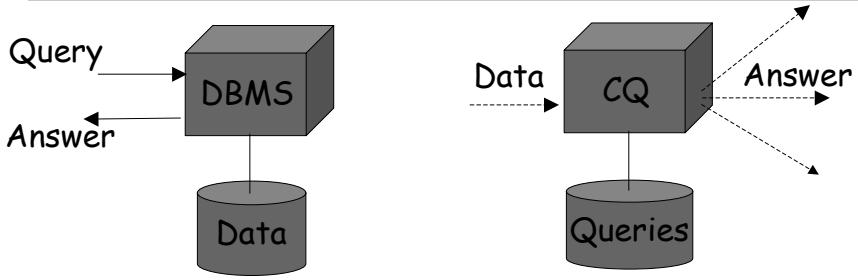
- build inverted index on queries

a	(Q1,Q2)
b	(Q1)
c	(Q2)
d	(Q3)

- merge lists for words in D
  => (Q1, Q2)
- filter list against D.
- Prefiltering can also be used.

- Further scalability obtained by common sub-expression grouping and client filtering.
- For XML, must also handle structure queries.

Continuous Queries



- Conventional approach: a query executes over the *current state* of the database and *terminates.*
- Continuous queries are *always running*, and produce new answers incrementally as the database changes.
  - Often defined over "append-only" databases.

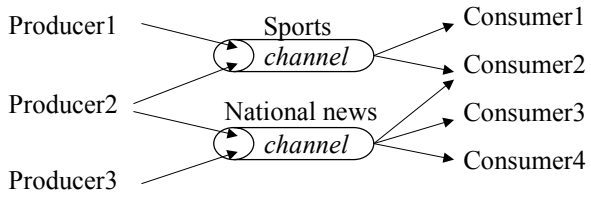
Continuous Queries Continued

- Queries are expressed in a slightly modified SQL.
  - e.g., addition of a "Continuous" directive.
- Typically require a schema.
- Employ query processing and multiple-query grouping (common-subexpression).
- Focus is on scalability of simple queries.
- Examples:
  - Xerox Parc [Terry et al., SIGMOD 92]
  - OpenCQ (OGI/Georgia Tech) [Liu et al., TKDE 99]
  - NiagraCQ (Wisconsin) [Chen et al., SIGMOD 00]

Active Databases and Triggers

- Not focused on filtering and routing *per se*, so more general, complex and less scalable.
  - Triggers can also update the database.
- Examples:
  - [Widom & Finklestein, SIGMOD 90]
  - [Stonebraker et al., SIGMOD 90]
  - ASSERT [IBM Almaden]
- More recent work on triggers has focused on scalability [Hanson et al., ICDE 99]
- Change Detection in Semi-structured data [Chawathe et al, ICDE 98]

Publish/Subscribe



- Information published on logical "channels"
  - a form of *semantic multicast*
- *Tightly* or *loosely* coupled (i.e., transactional or not)
- Examples
  - Tibco "Information Bus" [Oki et al. SOSP 93]
  - Vitria, Bang Networks, IBM Gryphon project

XFilter: XML Document Filtering

- XML is a key technology for Internet data exchange.
- Needed: efficient filtering (routing) of XML docs against many structure and content-based profiles.
- XFilter:
  - Represents XPath queries as Finite State Machines (FSMs)
  - Indexes and processes FSMs
  - Accepts any XML document (no DTDs needed)
- Originally developed by Mehmet Altinel (now at IBM Almaden) [Altinel & Franklin, VLDB 00].

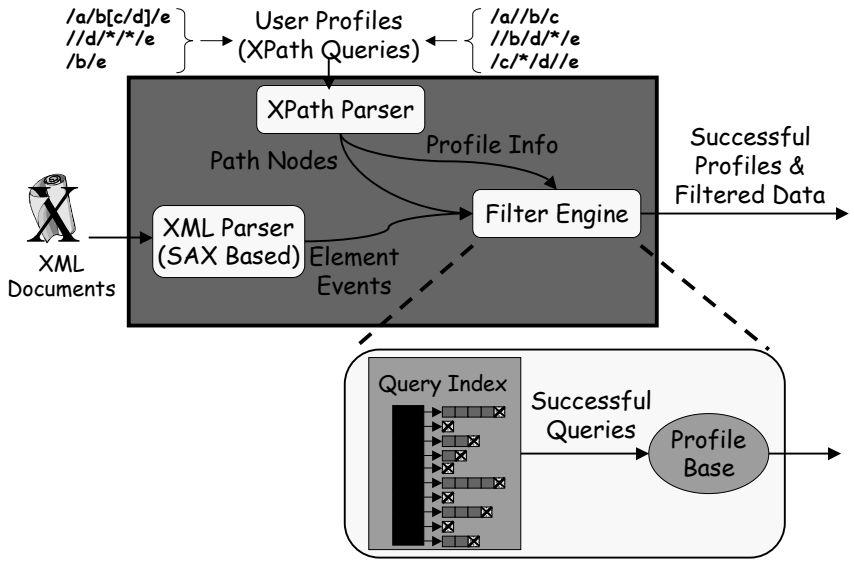
Relevant XPath Features

- **Parent/Child ('/') and Ancestor/Descendant ('//'):**
  /catalog/product//msrp
- **Wildcards (match any single element):**
  /catalog/*/msrp
- **Element Node Filters to further refine the nodes:**
  - Filters can contain nested path expressions

//product[price/msrp < 300]/name

Filter applied to ***product*** element node

XFilter Architecture



XML Parsing and Filtering

- Event-based XML Parsing using SAX API
- XML documents are converted to a linear sequence of events that drive the execution of the filter
- Callback functions are implemented to deal with the different events
  - Start Element
  - Element Data
  - End Element

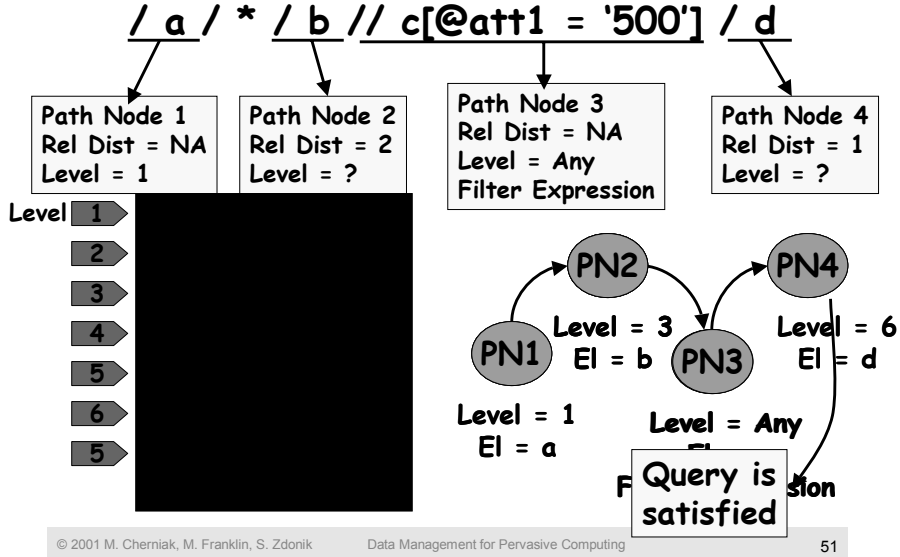
Filter Engine

- **Tricky aspects of the XPath language:**
  - Checking the order of elements in the queries
  - Handling wildcards and descendent operators
  - Evaluating filters that are applied to element nodes (Nested path expressions)
- **Solution:**
  - Convert each XPath query into an FSM
  - A profile is satisfied when its final state is reached
  - Index the ***states*** of FSMs

FSM Representation

- **Each element node is a state, represented as a** *Path Node.*
- To evaluate a state:
  - Compare the level of element name in input document with the level value of the path node,
  - Evaluate element node filter if there is one.
  - Locate next path nodes for the state transition.
  - Calculate the expected level values of next states using relative distance values.

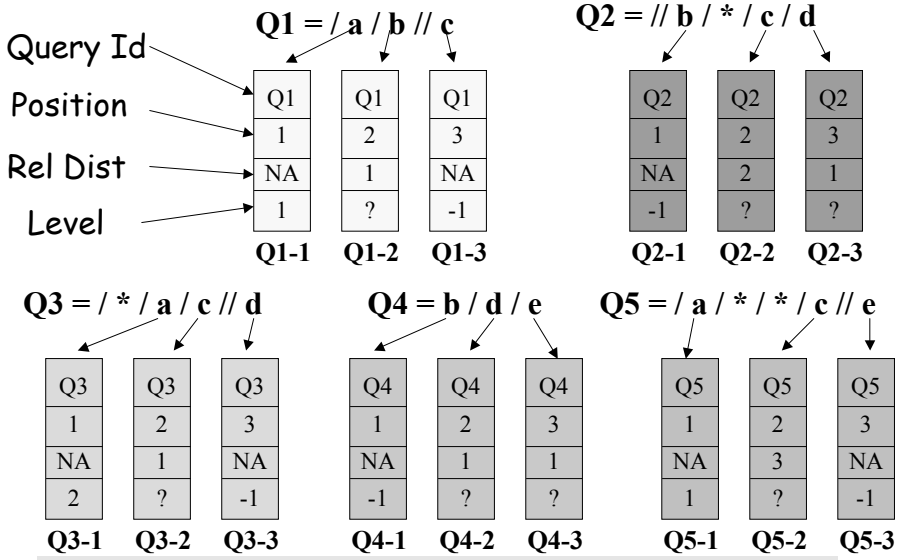
Path Node Decomposition



Handling Multiple Queries

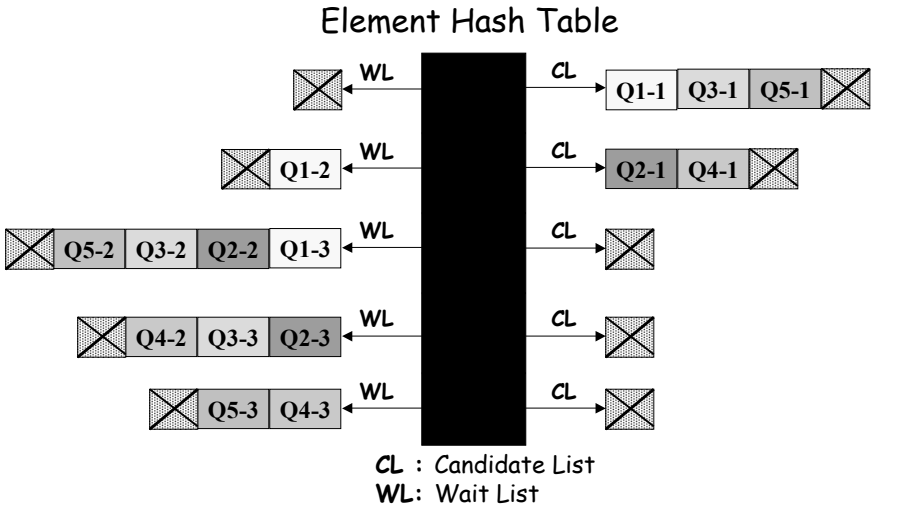
- Key insight for scalable SDI:
  - Index the queries instead of the data
- Hash table based on the element names in the queries
- Each node contains two lists of path nodes:
  - ***Candidate List***: Stores the path nodes that represent current state of each query
  - ***Wait List***: Stores the path nodes that represent the future states
- State transition is represented by promoting a path node from the Wait List to the Candidate List

Examples



 Data Management for Pervasive Computing 53

Query Index Construction



 Data Management for Pervasive Computing 54

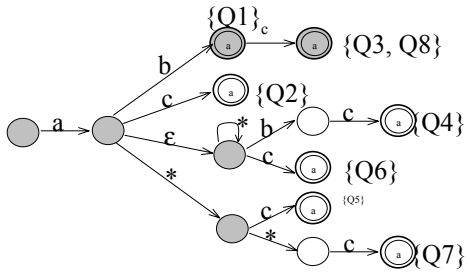
Other Details

- Currently, entire documents are returned
- Boolean combinations are handled in post-processing
- Nested Queries
  - Treated as separate queries, assumed "true" until proven otherwise.
- Basic approach is subject to "query skew"
  - Techniques to handle this:
    - List Balance
    - Prefiltering (using SIFT key-based algorithm).

Sharing Prefixes

- Initial Xfilter work didn't handle overlap.
- Alternative is an NFA-based approach

Q1=/a/b
Q2=/a/c
Q3=/a/b/c
Q4=/a//b/c
Q5=/a/*/c
Q6=/a//c
Q7=/a/*/*/c
Q8=/a/b/c

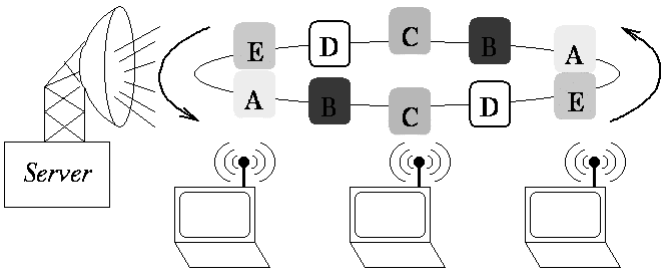


- Trick is to make the bookkeeping efficient.

Data Distribution

- Recall the three main issues in Data Dissemination:
  1. How to represent user interests?
  2. How to match data/events to interests?
  3. How to distribute the data to users?
- Not just a *networking* issue - there are many *data management* aspects as well.
- Examples:
  - Broadcast scheduling
  - Energy efficient querying
  - Combining push and pull

Broadcast Disks



- Repetition creates a revolving disk.
  - Good for intermittent connection, limited memory, high turn-over, or huge client population.
  - [Archaya et al. SIGMOD 95, *IEEE Pers. Comm.* 95]
- Teletext [Ammar&Wong, *Perf. Eval* 85]
- Datacycle [Herman et al. SIGMOD 87]

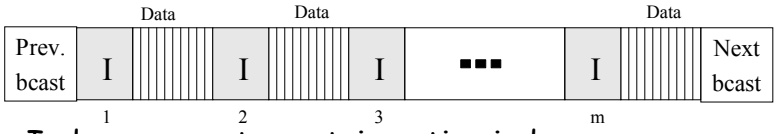
Key Data Management Issues

- Model can be generalized to a storage *hierarchy*
- How does server construct broadcast program?
  - Access probabilities for client population is given.
  - Must balance needs of multiple clients.
- How does client manage its local cache?
  - Broadcast program is given.
  - Must choose best cache replacement policy.
- How are updates handled?
  - Sleepers & Workaholics [Barbara & Imielinski SIGMOD 94]
  - Integrated Scheduling [Acharya et al. VLDB 96]

Energy Efficient Indexing

- Dataman Project [Imielinski et al SIGMOD 94]
- Primary design imperative: conserve battery life.
- CPU must be active to listen.
- Secret of (battery) life: *Sleep as much as possible*.
- Two metrics: *Access time* and *Tuning time*
  - Tuning costs battery
  - Tradeoff by varying amount and placement of index information.
- Example: (1,m) Indexing

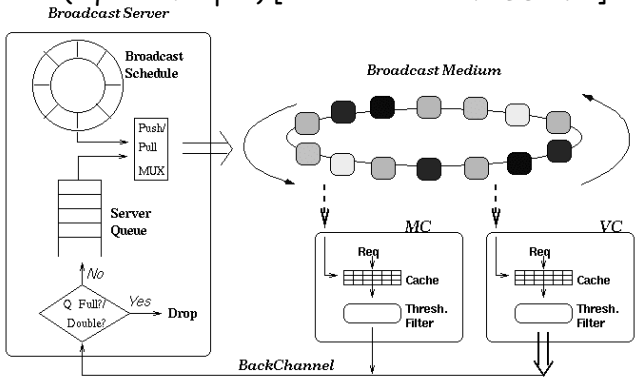
(1,m) Indexing



- Index segments contain entire index.
- All buckets have offset to beginning of next index segment.
- Access Proceedure:
  - Tune in current bucket.
  - Read offset of nearest index segment.
  - Sleep then tune in index.
  - Successive index probes to key *k*.
  - Sleep then tune in record with key *k*.

Integrating Push and Pull

- Push is scalable; Pull is more responsive.
  - So, push hot stuff, pull colder.
- Interleaved Push and Pull (*IPP*) [Acharya *et al.* SIGMOD 97]
- Adaptive (liquid vs. vapor) [Stathatos *et al.* VLDB 97]



Data Dissemination (Summary)

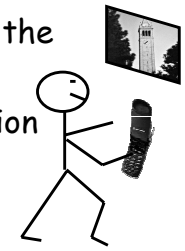
- Related technologies have been developed in the IR, DB, and Networking/Systems communities.
- Data dissemination and event-management are related
  - Key to many pervasive applications: computing infrastructure takes an *active* role in data management, delivery, and notification.
  - Also important for "real-time" business processes
- Main issues addressed
  1. How to represent user interests?
  2. How to match data/events to interests?
  3. How to distribute the data to users?

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Synchronization and Transactions

Pervasive computing exposes key limitations of the traditional ACID Transaction model.



1. Weak Connectivity/Frequent Disconnection
   - State of wireless comm. & cost issues
2. Large-scale Replication
   - Device-local caching required due to #1
3. Close User Interaction/Feedback
   - Allows negotiation, partial & preliminary results
4. Long-running tasks
   - Always a problem for ACID systems
5. Real-time Constraints

Replication Taxonomy [Gray *et al.*, SIGMOD 96]

- Propagation
  - Eager: All replicas updated in a single transaction
  - Lazy: Updates propagated asynchronously
- Ownership
  - Group: Any replica can be updated
  - Master: Only primary copy can be updated

- Disconnection and large number of replicas cause problems for all of these.
- Result is "system delusion": database is inconsistent and there is no obvious way to repair it.

Solution Approaches

- Multiple tiers of hosts:
  - Inner ring – high connectivity, resource-rich
  - Clients – weakly connected, mobile, expendable
- Two classes of copies:
  - Servers retain "copies of record"
  - Clients cache secondary ("soft-state") copies
- Reads see weaker-consistency (snapshot isolation)
- Updates happen *without* two-phase commit.
- Synchronization *metadata* kept at clients & servers.
- Synchronization process attempts to make these mutually consistent.
- Run *conflict resolution* when a problem arises.

Example: Palm HotSync

- Supports 2-way synchronization
  - updates can be made at devices and/or at servers
- Data on device is stored as *records* in *PalmDBs*.
  - Each PalmDB is associated with an application
  - Each record has a set of *status bits.*
    - Indicate if record has been created, modified, or deleted *since last synchronization.*
- Desktop maintains it's own copies of the palmDBs, including it's own versions of the *status bits.*
  - Also maintains a *snapshot* of each palmDB taken immediately after most recent synchronization.

HotSync Protocol

- Device initiates synchronization protocol:
- Was device last synced with *this desktop*?
- **Yes → Fast Sync**
  - Device sends data and status only for those records whose status bits are set.
  - Conduit can do efficient comparison of bits, update its copy of palmDB and send updates to the device.
- **No → Slow Sync**
  - Can't compare bits – device sends *entire* palmDB to the conduit, which does a *field by field* comparison to figure out what changed.

HotSync (continued)

- By comparing status bits (and possibly palmDB snapshots) the *synchronization logic* determines what actions to perform.

Status on Device	Status on Desktop	Action
Created	NA	Send to Desktop
NA	Created	Send to Device
Deleted	No Change	Delete from Desktop
Deleted	Updated	Send to Device
No Change	Updated	Send to Device
Updated	Updated	Invoke Conflict Resolution

SyncML Standard

- Industry Consortium with most major players:
  - Ericsson, Nokia, Motorola, Palm, Psion, IBM, …
- Goal is to enable cross-format, cross-system synchronization.
- Simple architecture:
  - Client: PDA, Phone or PC; *intermittently connected.*
  - Server: typically PC or Server; *continuously available.*
- A standard set of message types, represented as XML.
- Supports different interaction models including "request/response" and "blind push"

SyncML Sync Types

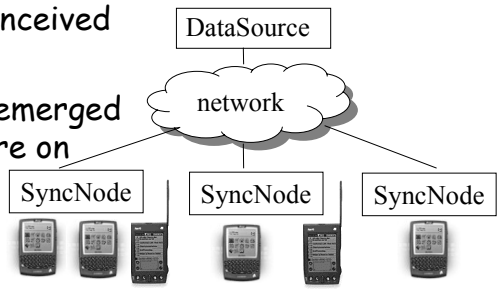
1. Two-way – "normal (fast) sync", client sends first.
2. Slow-sync – client sends all data
3. One-way, client only – client sends only modified records to server; server does not send to client
4. Refresh, client only – client sends entire DB to server
5. One-way, server only
6. Refresh, server only
7. Server Alerted – Sync initiated by server (push?)

SyncML (continued)

- Standard requires servers to maintain mappings between its own record IDs and the IDs of records as kept by the client.
- Conflict Resolution logic is (of course) dealt with abstractly by the standard. It provides standard status codes that can be used to implement typical policies.
- Contains support for authentication of clients and servers.
- www.syncml.org

Synchronization Services

- PDA Sync was originally conceived as a standalone process.
- Web-based services have emerged to allow sync from anywhere on the Internet (FusionOne,MyPalm,…)
- Sync Node is an access point on network: caches metadata, runs sync logic
- Data Source is a data repository and metadata log
  - Can use off the shelf ORDBMS technology (e.g., EDISON project [Denny & Franklin 01])
- *A variant of the dissemination architecture discussed earlier!*

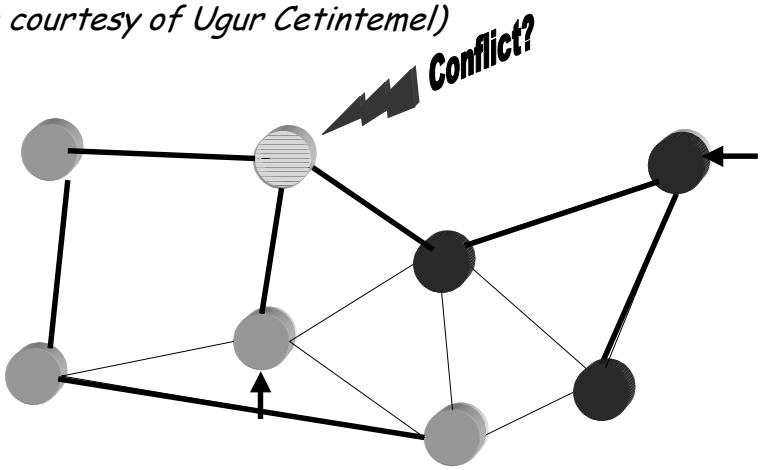


Peer-to-Peer Approaches

- Synchronization in Peer-to-peer environments is more complicated than in the asymmetric PDA world
- Centralized algos require connectivity at *specific* times.
- Alternative: *Epidemic Algorithms*
- Conflict detection via timestamps, version vectors,…
- Conflict Handling (update commitment):
  - Optimistic (resolution) – often manual
  - Pessimistic (avoidance) - primary copy, write-all or voting.
- Early work: Bayou, Ficus

Epidemic Protocol Illustration

(Picture courtesy of Ugur Cetintemel)



Deno – Epidemic Voting

Pessimistic, Asynchronous (epidemic), voting-based

"Bounded" weighted-voting:

- Each replica is assigned a currency c_i s.t. $0 \leq c_i \leq 1.0$
- Total currency in the system is bounded, i.e., $\Sigma c_i = 1.0$
- Currency can be re-distributed for optimization or planned disconnection.

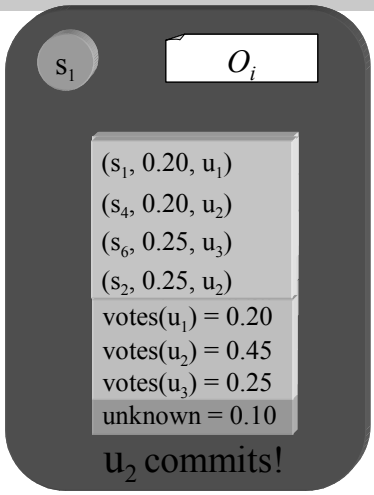
An update's life:

- Sites issue *tentative* updates
- Updates and votes are propagated in a pair-wise fashion
- Updates gather votes as they pass through sites
- An update *commits* when it gathers *plurality* of votes

- **[Cetintemel, Keleher, Franklin, ICDCS 01]**

Decentralized Commitment

- A site *s* maintains its view of:
  - the sum of votes *u* gained so far
  - the sum of votes unknown to *s* (i.e., 1.0 – Σ *votes(u)*, for ∀*u*)
- *u* commits *iff* for all conflicting updates *u'* <> *u* :
  votes(u) > votes(u') + unknown
- Each site can make its decision independently *and* correctly.
  - *Even if no more than 2 sites can communicate at any given time.*
- *Issues: time to commit; abort rates*



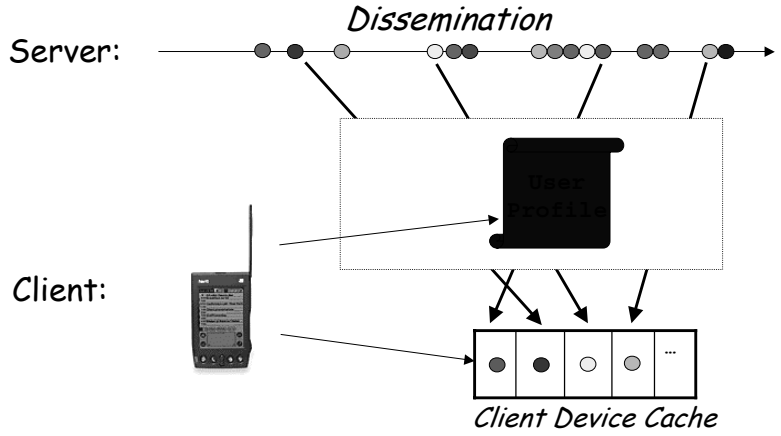
Mobile Transactions

- Mobility adds further complications:
  - Disconnect/Handoff
  - Lots of potential failure modes
- Some approaches:
  - Kangaroo Transactions [Dunham et al. 1997]
    - Subtransactions ("Joeys") are created as device moves between base stations.
    - Control shifts among base stations as device moves.
  - Semantics-based approaches classify transactions such as escrow, read-only, etc. and exploit semantics. (e.g., [Walborn & Chrysanthis 95])
  - Cluster-based approaches divide world (or database) into units that are loosely coupled together (e.g., [Pitoura & Bhargava 96]). Inconsistency between clusters is allowed.

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Data Recharging
Data Dissemination Meets Synchronization



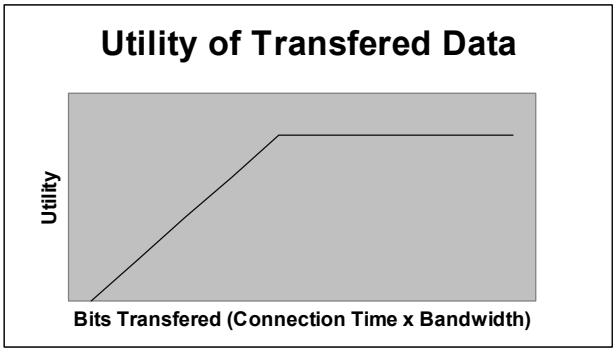
Data Recharging
The Metaphor

Idea: Synchronization Made Like Battery Recharging

1. *Location-Independent*
   i.e., plug device into any network jack to recharge
2. *Incremental*
   i.e., more bits transferred ⇒ "better data",
   graceful disconnections

"Better Data": Determined by user profiles

Data Recharging
User Profiles: Not Just Queries



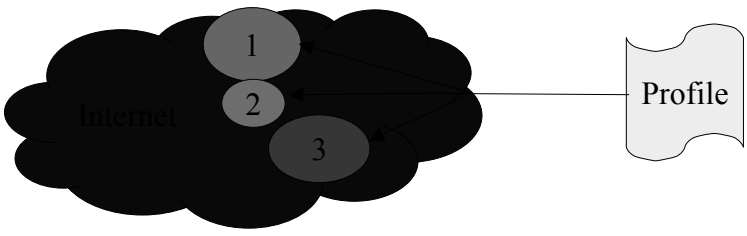
Should return *partial* results in case of:

- *Device cache < Size of Desired Data*
- *Premature Disconnection…*

Data Recharging
User Profiles: Not Just Queries

The Role of Profiles:

1. *Identify Data User Cares About (Domain)*
2. *Specify Relative Worth of Data (Utility)*



Data Recharging
Desiderata for Profiles

Declarative

"Hi-Tech Stock Quotes from US Exchanges" > "www.nasdaq.com"

Like queries: permits more flexibility in processing

Expressive

Thresholds:	*3 Biotech Quotes Suffice*
Conditions:	*IBM Quote if a competitor included*
Resolution:	*DJI can substitute for 30 stocks*
Context:	*Quotes for companies situated where I am*

Data Recharging
An Example Profile

```
PROFILE Traveler
 DOMAIN
  R = related: www.hertz.com
  S = +Logan +Boston +Shuttle
  D = +Logan +Boston +Directions
 UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
END
```

Traveler: Traveler to Boston wants to get downtown

Domain: *What data interests me?*
Utility: *What is its relative worth?*

Data Recharging
An Example Profile

```
PROFILE Traveler
 DOMAIN
  R = related: www.hertz.com
  S = +Logan +Boston +Shuttle
  D = +Logan +Boston +Directions
 UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
END
```

Domain Clause:

R = *web pages for rental car companies*
S = *shuttle schedules to Downtown Boston*
D = *directions to Downtown Boston*

Data Recharging
An Example Profile

```
PROFILE Traveler
 DOMAIN
  R = related: www.hertz.com
  S = +Logan +Boston +Shuttle
  D = +Logan +Boston +Directions
 UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
END
```

Utility Clause:

Thresholds: *1st shuttle schedule worth 2, others worth 0*

Data Recharging
An Example Profile

```
PROFILE Traveler
 DOMAIN
  R = related: www.hertz.com
  S = +Logan +Boston +Shuttle
  D = +Logan +Boston +Directions
 UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
 END
```

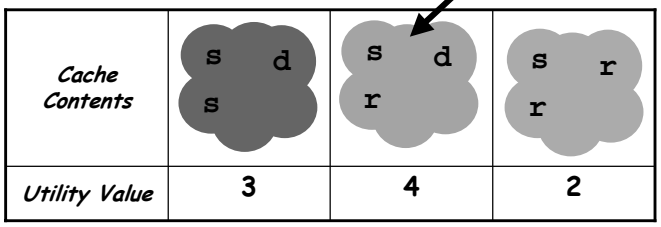
Utility Clause:

Conditionals: *Rental car web page worth 1 if directions included*

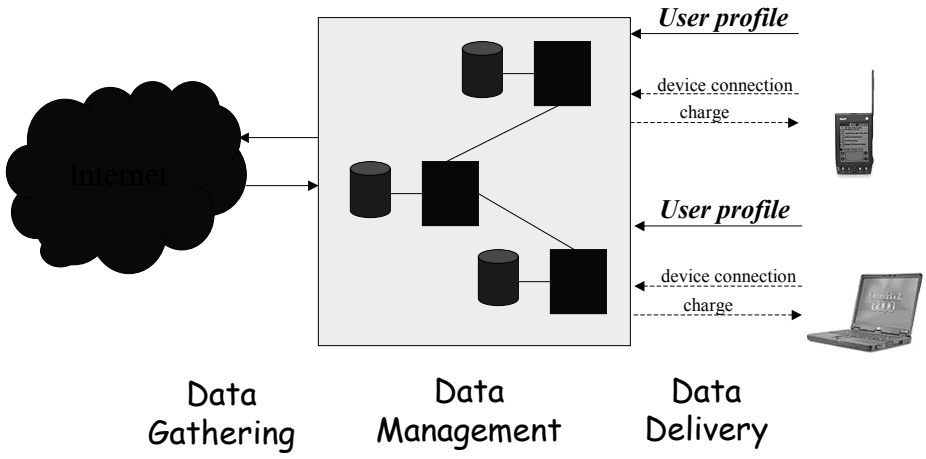
Data Recharging
Interpreting the Traveler Profile

```
PROFILE Traveler
 …
 UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
END
```

Best "charge" of 3 objects!



Data Recharging
An Architecture



Data Recharging
Issues

1. **Profile Processing**

   How are profiles used to recharge devices?

2. **Profile Formulation**

   Where Do Profiles Come From?

3. **Profile Expressivity**

   What Else Should be Expressible in a Profile?

Data Recharging
Profile Processing Issues

- *Locating objects*
- *Copying vs. referencing*
- *Replication and distribution policies*
- *Update policies (for volatile objects)*
- *Physical organization*
- *Algorithms for Determining Charge*
- *Moving Objects to Connection Location*
- *Graceful Disconnections*

Data Recharging
Profile Formulation Issues

Authored By Users?

Augment "Personalization Profiles"
With help of GUI (as with SQL)
Libraries of canned profiles

Learned By Data Mining?

Analysis of Clickstreams

Likely Some Combination of Above

Data Recharging
Profile Expressivity Issues

1. Resolution

a. Domain

Type	*Low-res*	*Med-res*	*Hi-res*
Images	*100x200*	*200x400*	*400x800*
MP3's	*128 Kbps*	*192 Kbps*	*256 Kbps*
Documents	*Abstract, TOC, Concl'ns*	*All But Figures*	*Entire Document*

b. Utility

E.g., Documents have value of 1 (lo), 2 (med) or 3 (hi)
E.g., Give me only 1 version of an object
E.g., Give me hi-res images only with hi-res documents

Data Recharging
Profile Expressivity Issues

2. Context

a. Geography

Give me restaurant reviews for restaurants within walking distance

b. Time

Walking distance in daylight is 1 mile; after dark is 2 blocks

c. Workflow

If I am within 24 hours of a deadline, restaurants described should offer fast service

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Profile-Driven Data Management

Data Recharging and PDDM

Data Recharging: One Application of PDDM

Basic Idea:

- *Limited, Shared Resources (bandwidth, servers, cache, …)*
- *Data Requirements Specified with Profiles*
- *Profile Processing ⇒ Data Management Policy*

Potential Applications:

- *Data Freshening [CGM00]*
- *Automatic Indexing [HC75, CN98]*
- *Web Cache Prefetching [FS00]*

Profile-Driven Data Management
Pervasive Computing and PDDM

Why is PDDM Suitable for Pervasive Environments?

Pervasive Environments:

- *thousands of data sources (e.g., sensors)*
- *thousands of users*
- *dynamic environment (users and sources come and go)*

Data Management Must Be:

- *Automatic*
- *Adaptive (constant reconfiguration)*

Profiles Replace DBA ↔ User Interactions

Profile-Driven Data Management
Related Work: Profiles and Ranking

Personalization

- *Portals: my x (x ∈ {yahoo, cnn, nbci, ucla, …})*
- *Avant Go*
- *Publish/Subscribe Systems [OPS+93, YGM99]*

Ranking (Managed Resource: User Attention Span)

- *User Preferences [AW00]*
- *PREFER [HKP01]*
- *Search Engines (Link Popularity, Direct Hits, …)*

Profile-Driven Data Management

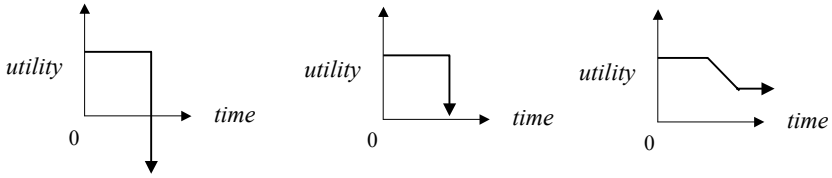
Related Work: Real-Time Systems [St92]

Applications issue resource requests w/ deadlines

- *resulting resource workload intensified*

Time-based Utility Functions Implicit

- *serve to classify deadlines*



"Hard Deadline" *"Firm Deadline"* *"Soft Deadline"*

Profile-Driven Data Management

Related Work: Quality-of-Service

Multimedia Object (Stream) Delivery Systems:

Real-time + large objects ⇒ intense bandwidth contention

Natural notions of resolution ⇒ adaptive responses

Utility/Benefit Functions

$QUASAR_{OGI}$ [St96, WKL+99], $QUASAR_{SRI}$ [CSS+97, CDS97]

Specify functions over dimensions of "quality loss":
horizontal/vertical resolution, frame rate, color depth, …

Influences choice of approximate stream sent

Profile-Driven Data Management

Related Work: Digital Video Recording

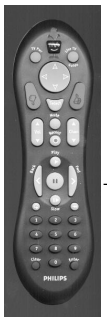
Digital Video Recording:

- *Examples: Tivo, Replay TV, Ultimate TV*
- *Distributed Disk Management (10-60 Gb per disk)*
- *Users specify desired recorder content (profiles)*
  - *declarative: independent of broadcast times, channels*
  - *e.g., record 3 most recent World Cup qualifying matches*

Profile-Driven Data Management

Related Work: Digital Video Recording

The Tivo© Profile Generator



	Profiles	Tivo©
Domain	*Declarative* *Explicit* *Refinement*	*Wishlist* *TV Guide* *Thumbs-up/down*
Utility	*Priority* *Resolution* *Thresholds*	*Recording Guarantees* *Video Quality* *# Episodes to Keep*

Profile-Driven Data Management

Related Work: Self-Tuning Databases

Automatic Index Creation

- *Hammer and Chan: [HC75]*
- *DB2: [SV99]*
- *SQL Server (AutoAdmin) [CN98], [CCG+99], [ACN00]*

Memory Management

- *Weikum et. al. [WCKS99]*
- *SQL Server (AutoAdmin) [CCG+99]*

Cost Models

- *SQL Server (AutoAdmin) [AC99]*

Profile-Driven Data Management

An Example: Investor

```
PROFILE Investor
 DOMAIN
  cnn = www.cnn.com
  ny  = www.nyse.com
  na  = www.nasdaq.com
 UTILITY
  U (na [age < 30 min]) = 3;
  U (ny [age < 20 min]) = 2;
  U (cnn [age < 10 min]) = 1;
 END
```

Conditional Object Values:

Object values (could also be based on size, expiry time …)

Profile-Driven Data Management

An Application: Data Freshening [CGM00]

Idea: Proactively Refresh Cache of Volatile Data

Keep the Most Important Volatile Data Up-To-Date

E.g., Assuming One Update per 10 min:

Time	*Cache*	*Choose*	*Cache Value*
2:00	$\{na_{20}, ny_{10}, cnn_{30}\}$	na	5
2:10	$\{na_{10}, ny_{20}, cnn_{40}\}$	ny	5
2:20	$\{na_{20}, ny_{10}, cnn_{50}\}$	na	5
2:30	…		

Profile-Driven Data Management

An Example: Academic

```
PROFILE Academic
 DOMAIN
  S =   SELECT School
        FROM www.acm.org/sigs/sigmod/dbjobs.db
        WHERE Area = "Databases"

  P =   SELECT Title, Author, Affil
        FROM www.informatik.uni-trier.de/~le/db/papers.db
        WHERE year > 1997
 UTILITY
  …
END
```

Domain Clause:

structured data ⇒ expressible with queries

Profile-Driven Data Management

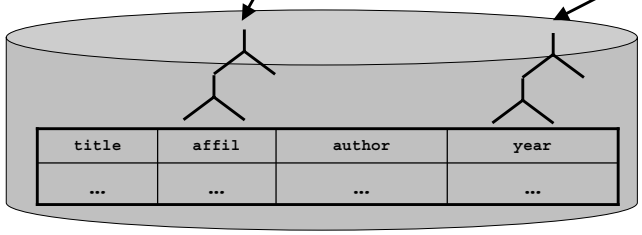
An Application: Automatic Indexing [HC75, CN98]

Idea: Index Domains With Common Data

Example:

```
PROFILE Academic1
  P = {<title, author, affil>}
```

```
PROFILE Academic2
  P1 = {<title, year>}
```



Profile-Driven Data Management

Profile Processing: CAP

Informally:

How to "best" fill a cache given:

O: A Finite Set of Candidate Objects

S: O → Int (object sizes)

P: A Profile

where "best" is determined by utility values and where the cache is not filled beyond its capacity, C

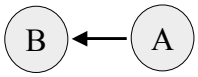
Applications:

Recharging, Freshening, Cache Prefetching

Profile-Driven Data Management

Profile Processing : CAP + PCKP

Idea: Knapsack problem + "precedence constraints"

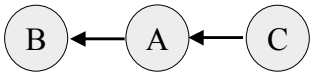


A allowed in cache only if B in cache

CAP ≠ PCKP :

```
PROFILE Problem …
  U (A [#B > 0]) = 1;
  U (B) = 1;
  U (C [#A > 0]) = 100
```

PCKP →



Misses best solution for cache of 2 objects: {a, c}

Profile-Driven Data Management

Profile Processing: CAP + Greedy Algorithm

```
PROFILE Traveler
DOMAIN R, S, D
UTILITY
  U (S) = UPTO (1, 2, 0);
  U (D) = UPTO (1, 1, 0);
  U (R [#D > 0]) = 1
END
```

# Objects Sent	Data In Cache	Total Utility
1	**S**	2
2	S, **D**	3
3	S, D, **R**	4
4	S, D, R, **R**	5

Profile-Driven Data Management

Profile Processing: Combining Profiles

Informally:

How to reduce n profiles to 1 ***representative*** *profile*

Applications: Generalizing any processing algorithm to n profiles

One Approach: Combine Profiles

```
PROFILE P1 …
  U (R) = 1;
  U (S) = UPTO (1, 2, 1);
```

+

```
PROFILE P2 …
  U (R) = 2;
  U (S) = UPTO (2, 3, 1);
```

=

```
PROFILE P1+P2 …
  U (R) = 3
  U (S) = UPTO (1, 5, UPTO (1, 4, 2));
```


Profile-Driven Data Management

Profile Processing: Combining Profiles

1. Does Combination Lessen Profile Complexity?

Could append profiles: result longer but narrower

2. How Do We Recognize "Equal Domains"?

Easy for URL's, Undecidable for Queries

Don't Need a Complete Solution

Profile-Driven Data Management
Summary

Pervasive Computing Environments

- *Many data sources*
- *Many users*
- *Highly volatile environment*

Data Management

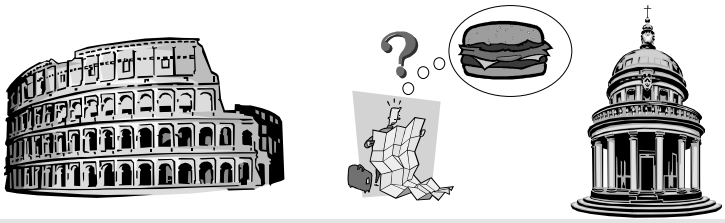
- *Must be automated*
- *Must be adaptive*
- *Must be Profile-Driven*

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Location-Aware Applications

- **Many potential "killer-apps" for pervasive computing are location-aware:**
  - Tracking people, trucks, taxis, bunnies
  - Find the nearest: restaurant, hospital, …
  - Personal tour guides
- **Enabled by advances in GPS and GIS technology**



Location-Aware Data Management

- **Many data management problems:**
  - Location data is continually changing
    - Indexing and Updating issues
    - Represent movement as a function. Only update database when function changes.
  - Spatiotemporal query languages and processing
  - Uncertainty and Imprecision
  - Data presentation and user interaction
    - esp for vehicle-based systems, small devices, etc.
  - Ad hoc and self-organizing networks

DOMINO
[Wolfson *et al.* SIGMOD 99]

- **Data model extended with *dynamic attributes***
  - *Idea is to be able to interpolate changing values:*
    value at A.updatetime + t_o = A.value + A.function(t_o)
  - Locations can be modeled similarly
- **Spatiotemporal query language including bounded temporal operators:**
  - Eventually_within_c (g): *g* is true within *c* time units.
  - Always_for_c (g): *g* is continually true for at least *c* time units.
- **Semantics for uncertainty: "May" vs. "Must"**
  - take uncertainty bounds into consideration

Indexing Moving Objects

- **Goal is to provide quick access to objects current and projected locations.**
  - e.g., find all airplanes in the path of my plane
- **Basic approach: map trajectories into appropriate dimensions and use Spatial Indexing techniques:**
  - Quadtrees [Tayeb et al. PODS 98]
  - Time-parameterized R-tree [Saltenis et al. SIGMOD 00]
  - Hashing over regions [Song & Roussopoulos MDM01]
  - SS-Trees [Chon et al. MDM 01]
- **Also, a hot topic of theory research :**
  - Kinetic Data Structures [Basch et al. SODA 97]

Outline

1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Service Discovery

Idea:

Automatic detection of devices, services

- *Devices: printers, fax machines, …*
- *Services: mail servers, …*

Applications:

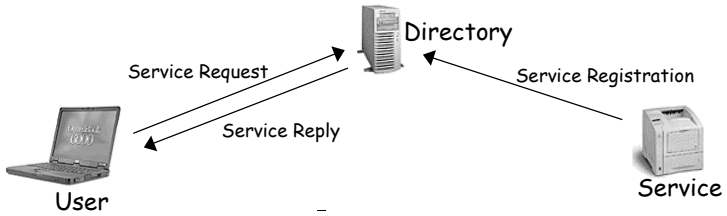
- *0-configuration networks*
- *Mobile computing environments*

Alternative Protocols

SLP [GPV+99], Jini [Sun99], UPnP [Mic00], Salutation [Sal99], …

Service Discovery

Protocols: General Approach



Service Registration: Service advertises existence w/ directory

Service Request: User queries directory for available services

Directory Discovery (for User or Service):

1. *Static: Address obtained via DHCP*
2. *Active: Service requests sent to multicast group address*
3. *Passive: Directories periodically multicast ads for service*

Service Discovery

Protocols: A Comparison (Courtesy of [BR01])

Feature	SLP	Jini	Salutation	UPnP
Developer	IETF	Sun	Consortium	Microsoft
Network Transport	TCP/IP	Independent	Independent	TCP/IP
Programming Language	Independent	Java	Independent	Independent
OS and platform	Dependent	Independent	Dependent	Dependent
Operation w/o Directory	Yes	No	Yes	No directory

Data Discovery

- Problem: How to find data that is distributed across millions of devices?
  - Use a hierarchical directory, like DNS.
- VIA uses an overlay network to dynamically form hierarchical "clusters" of sites that are physically "close". [Castro et al. MOBICOM 01]
- DataSpace proposes a global network of geographically nested "data cubes" [Imielinski & Goel MOBIDE 01]
- Peer-to-peer file systems such as Gnutella provide early implementations and testbeds.

Outline

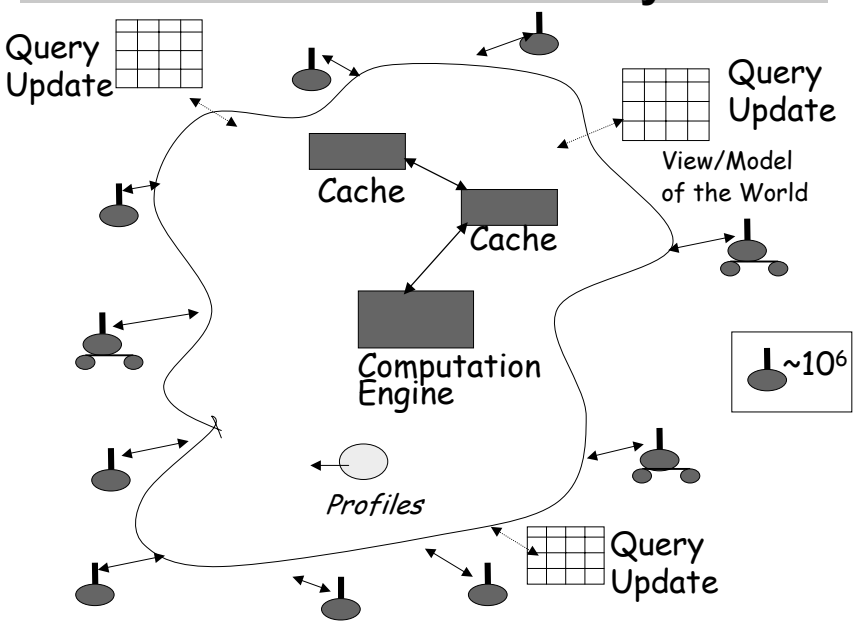
1. **Pervasive Computing - Applications and Requirements.**
2. **Architectural Concepts**
3. **Data Dissemination**
4. **Data Synchronization**
5. **Data Recharging**
6. **Profile-Driven Data Management**
7. **Other Topics**
   a. Location aware and moving objects
   b. Service discovery
   c. Sensors

Sensors: The Brave New World

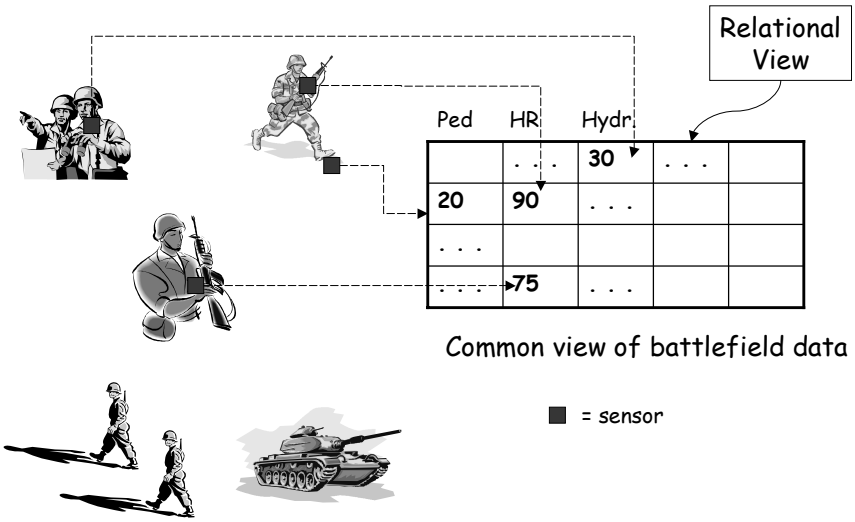
- **Tiny devices "measure" the environment.**
- **Communicate streams of values upstream.**
- **Goal: Construct a model of said environment**.

- **Need support for:**
  - querying
  - monitoring
  - imprecision
- **Must handle:**
  - "now" data
  - historical data
  - a combination

Data-intensive Pervasive System



Application:
Battlefield Monitoring



Sensor Types

- Pull-based
- Push-based with *fixed period*
- Push-based with *settable period*
- Push-based with *events*

What you have will effect query processing.

Streams

- **Generated by many types of sensors.**
- **Characteristics:**
  - List of tuples
  - Ordered by time
  - Potentially infinite
  - Reporting intervals can vary
  - Live data
- Suggests a algebra for streams
  - e.g., SEQ [SLR94]

Consistent View of Reality

- **Sensors report data at different rates**
- **Sensor reports experience different latencies**

⇒ **Current view might never have existed.**

❖ **Analogous to observation in astronomy.**

Things you <u>see</u> are not coincident in time.

=>cannot be used <u>directly</u> for prediction/planning.

Synchronization

- **Clock synchronization**
  - Post-facto synchronization [EE01]
    - Sources of error:
      - Receiver clock skew
      - Variable delays in receivers
      - Propagation delay of synchronization pulse
- **Stale or missing values**
  - Predictive techniques
    - Interpolation/extrapolation

InfoSphere [PSW01]

- **Problem: How to connect up information flows (plumbing).**
- **InfoPipes**
  - TypeSpec (schema + QoS)
  - 1-1, 1-M, N-1, N-M (1-M + N-M)
  - Translates from input to output TypeSpec.
- **Composable InfoPipes**
  - IP A's input conforms to IP B's output if
    - 1. Ops(TS(A)) contained in Ops(TS(B))
    - 2. Schema(TS(A)) compatible with Schema(TS(B))
    - 3. Properties(TS(A)) falls within range of Properties(TS(B))

Cougar [BGS01]

- ***A Sensor Database System***
  - Sensors: ADT interface
  - Data: time-stamped values
  - Queries: conditions on time and space
- **Query:** *Every minute, return the temperature measured by all sensors on the third floor.*

```
R (loc point, floor int, s sensorNode)
    Select R.s.GetTemp( )
    From R
    Where R.floor = 3 and $every(60)
```


Telegraph – Fjords [MF01]

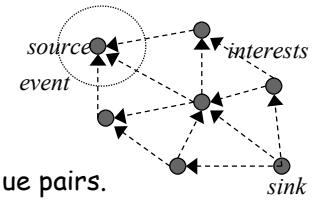
Key Insight: Stream-based systems must operate on traditional (pull-based) sources too!

- E.g.: traffic streams + accident reports

- **"Fjord": Query-plan structure for combining streaming (push) and traditional (pull) data sources.**
  - Operators assume non-blocking queue interface.
  - Queues implement push vs. pull

- **Supports parallelism between operators via queues, state machines, and OS in an operator transparent way.**

Controlling Information Flow

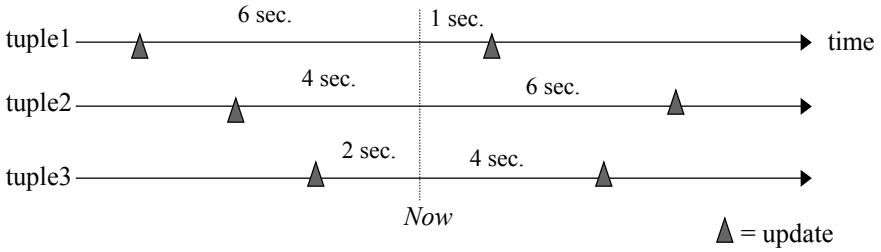


- **Get data to points of interest.**
- **Directed Diffusion [IGE00]**
  - Data is named with a set of name-value pairs.
  - Sensing tasks disseminated as interests for named data.
  - Events are drawn towards interests by gradients.
- **Example interest:**

```
Type = four-legged animal
Interval = 20ms  //send back events every 20ms
Duration = 10 sec  // . . . For next 10 sec.
Rect = [-100, 100, 200, 400]
```

- Broadcast interest to neighbors
- Neighbors check interest cache.

Query Processing Complexities



Strategies:

Use current values

Wait 2 sec. / 4 sec. / 6 sec. ?

Depends on utility of quick response vs. more accurate measures.

Summary

- Data Management plays a crucial role in pervasive computing.
- Decades of experience with query processing, transactions, replication, caching etc. provide a solid base of technology on which to build.
- But, Pervasive Computing brings challenges in all aspects of data management

