Middleware for Challenged and Dynamic Environments

Mohan Kumar and Jiannong Cao
The University of Texas at Arlington and Hong Kong Polytechnic University

ACM Middleware November 29, 2010

Tutorial Organization

Challenged and Dynamic Environments
Motivation and Justification
Technology changes and application diversity
Role of Middleware in CDEs
Middleware for pervasive, sensor and opportunistic systems
Challenges and Research Issues
Existing Solutions
What are challenged and dynamic systems?

- Mobile nodes
- Changing data
  - Short lived, localized
- Changing context
  - Location, Device used
- Changing connectivity
  - Cellular, WiFi, Bluetooth etc.
- Depleting resources
  - Residual battery energy
  - Connectivity and bandwidth
- Heterogeneity
- User Dynamics
  - Pattern of movement, profile, social activities
- Environmental changes
  - Computing resources, interference/noise,

Middleware

Application Requirements
- Quality of service parameters
- Context

Dynamics
- Mobility
- Changing Data
- Changing Context

Challenges
- Uncertainties
- Lack of communication path
- Lack of resources

Distributed Computing
- Negotiation
- Resource allocation
- Task scheduling
- Load balancing
- Task migration
- Service Composition
- Security, privacy and Trust

Managing Resources
- Services
- Context

11/29/2010
Computing Paradigms

- **Computing – 1940s …**
  - Uniprocessor architectures, limited applications

- **Parallel Computing – 1970s …**
  - Multiprocessor systems, computationally intensive tasks

- **Distributed Computing – 1980s …**
  - Collaboration in networked systems, Resource Sharing, Business applications, the Internet, WWW

- **Mobile Computing – Mid 90s …**
  - Anytime anywhere computing

- **Sensor Systems – 00s…**
  - Window to the physical environment

- **Pervasive Computing – 00s …**
  - User centric, quality of life

- **Cyber Physical Systems and Internet of Things– Mid 00s**
  - Connecting with the physical environment and all kinds of objects, …
Recent Developments

Wireless ad hoc networking
- Novel algorithms and schemes developed
- Cooperation in the absence of infrastructure

Pervasive computing
- Context-aware services to users/applications
- Smart environments

Distributed resources
- Mobile devices possess myriad of resources

Opportunistic Networking and Computing
- Exchange of packets/bundles

Social networks and computing
- Exploit gregarious nature of humans

Current and Future

Cyber Physical Systems
- Window to the physical world
- Measure physical parameters
- Environment

Internet of Things
- RFID based object tracking

Social Networking
- Exploit user social characteristics

Cloud Computing
- Remote services
- Access to resources
### Fading Distinctions

<table>
<thead>
<tr>
<th>Servers and clients</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Distributed systems, P2P systems</td>
<td></td>
</tr>
<tr>
<td>• Cost and time</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Producers and consumers of information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Users are producers of information as well</td>
<td></td>
</tr>
<tr>
<td>• User with a cell phone camera</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service providers and consumers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Resources on user devices can be exploited</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resourceful and resource-poor entities</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Servers, desktops, laptops, mobile phones</td>
<td></td>
</tr>
<tr>
<td>• Grid computing</td>
<td></td>
</tr>
<tr>
<td>• Cyber foraging</td>
<td></td>
</tr>
</tbody>
</table>

**11/29/2010**

<table>
<thead>
<tr>
<th>Resourceful and resource-poor entities</th>
<th>Middleware provides a uniform view</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Servers, desktops, laptops, mobile phones</td>
<td></td>
</tr>
<tr>
<td>• Grid computing</td>
<td></td>
</tr>
<tr>
<td>• Cyber foraging</td>
<td></td>
</tr>
</tbody>
</table>
Why Middleware is different?

- Lack of continuous connections
- Heterogeneous devices and networks
- Limited resources
- Lack of central server
- Temporal and spatial changes

Enable Distributed Computing

Challenged Systems

- Pervasive Computing Environments
- Sensor Systems
  - Cyber Physical Systems
- Opportunistic Systems
  - Participatory Systems

There is no clear distinction between these systems

Overlap with cyber physical systems, Internet of things and Social Systems
Functions

Resource management
- Limited, uneven
- Accessibility

Application interface
- Quality requirements, user dynamics

Services management
- Discovery, composition

Context management
- Processing, Reasoning

Data/information management
- Acquisition, dissemination
- Availability, accessibility

Security/Privacy/Trust/Anonymity

Desirable Features

Lightweight
- Execution and energy consumption

Modular
- Sensors, Cell phones, Laptops

Easy to deploy
- For general users, e.g., iPhone/Android apps

Reusable
- Different applications, spatial, temporal

Adaptable
- Across different platforms

Locally scalable
- Local environments
The scenario uses existing basic component technologies
Laptops, cameras, cell phones, PDAs etc.

What makes these scenarios appear like fiction?
The whole is much greater than the sum of its parts
Intelligent Transport

Wireless Sensor Networks

How to satisfy application constraints? Reliability, Tracking Error, Coverage etc

Example Scenario: Object Tracking Application

Kumar and Cao
Crisis Management

- Sensed Data
  - Video clips
  - Static camera
  - Mobile camera
  - User camera
  - Other sensors

- Video Acquisition, Processing and Synthesis
  - Composite scene creation
  - Virtual observation
  - Virtual tours

- Distributed query processing
- Event detection
- Stream data clustering

Wireless Communications
- Opportunistic networking
- Message exchange
- Routing

Service Management
- Advertisement
- Aggregation
- Composition
- Maintenance

Cyber-Physical Systems

- Build systems that interface between the cyber world and the physical world - ideally, with predictable, if not adaptable behavior.
  - Integrate computation and physical processes.
  - Integrate cyber-concepts with dynamics of physical and engineered systems
  - Cannot easily draw boundaries – need both information and physical feedback - not traditional, post-hoc embedded/real-time systems

Kumar and Cao
Internet of Things

- Extend M2M and P2M communications to Things.
  - IP-connected devices embedded in the environment all around us
  - from everyday household objects to sensors monitoring environments;
- Technologies driving Internet of Things:
  - RFID, sensor technologies, smart things and nanotechnology and miniaturization
- View cyberspace in different way
  - Always-connected, responsive, adaptive, omnipresent in all aspects of our lives

Distributed Service Composition

- Service Graph:
  - S1: Observation at Point T1
  - S2: Observation at Point T2
  - S3: Collaborative Observation
  - S4: Render clips for transmission and display on user’s PDA
  - S5: Display service on user device

User task

11/29/2010 Kumar and Cao
Earthquake Disaster Recovery

- Employ pervasive technologies to opportunistically build social and situational network models in support of context reasoning during temporary relief efforts.

Identify and prioritize critical areas for effective resource assignment

Effectively locate trapped victims

Track rescue supplies to avoid waste and abuse

Resolve missing persons reports

Resolve mail delivery backlogs

Functions

<table>
<thead>
<tr>
<th>Resource management</th>
<th>Pervasive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited, uneven</td>
<td>Wireless</td>
</tr>
<tr>
<td>Accessibility</td>
<td>Mobile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application interface</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality requirements, user dynamics</td>
<td>Opportunistic/social</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services management</th>
<th>Context management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery, composition</td>
<td>Processing, Reasoning</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data/information management</th>
<th>Wireless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisition, dissemination</td>
<td>Mobile</td>
</tr>
<tr>
<td>Availability, accessibility</td>
<td>Ad Hoc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security/Privacy/Trust/Anonymity</th>
<th>Opportunistic</th>
</tr>
</thead>
</table>
Pervasive Computing (PvC)

• Create a **smart environment** with ubiquitous, invisible, interconnected devices that provides **unobtrusive** services to the users.

• Improving users’ experience and quality of life **without requiring them know** the underlying technologies

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

Most cited PvC applications

• Smart spaces
  – home, office, museum, classroom, meeting room, airport, kindergarten

• Assisted Environments for individuals with special needs
  – Healthcare, elderly-care, disadvantaged, etc.

• Intelligent Transportation

• Logistics

• Emergency Services

• Entertainment

Scenario

The scenario uses existing basic component technologies
Laptops, cameras, cell phones, PDAs etc.


What makes these scenarios appear like fiction?
The whole is much greater than the sum of its parts
How can pervasive computing help?

- Desired actions
  - Detect and recognize events
  - Recognize high-level events
  - Discover and deploy services
  - Combine services
  - Match services to resources
  - Address dynamic issues
  - ...
- On a TIMELY, AUTOMATED, TRANSPARENT basis

Pervasive Computing: Challenges

- Proactivity and transparency
  - Delays, resource utilization, unobtrusive services
- Heterogeneity and interoperability
  - Unevenness, incompatibility, h/w, s/w, communication channel
- Location awareness and mobility
  - Handoff: vertical/horizontal, data dissemination/acquisition
- Authentication and security
  - Privacy vs. services, cost, agents, active networks, availability
- Smart environments
  - Deployment, Interference

Middleware For Services in Pervasive Environments

Middleware Services

- Glue heterogeneous entities
  - platform for interaction
- Match services to resources
  - Application specific
  - User profiles
- Combine resources and services
- Respond to user/application needs
- Mask unevenness
- Facilitate context-awareness
- Facilitate cooperation and collaboration
Service Management

Service discovery
- Finding service providers in the environment matching user preferences

Seamless service access
- Enabling seamless service access for mobile users

Service composition
- Combining elementary services to compose a higher level service

Service Management: Service Discovery (SD)

- Service can be any hardware or software functionality of a device that can be needed by other devices for usage
  - Resources, data or computation functions

- Service Discovery finds services on peer devices and determines how to access or utilize the discovered services
Service Discovery: Protocols

<table>
<thead>
<tr>
<th>Properties</th>
<th>Issues</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN 1. Comprises Enterprise environments, home and office settings</td>
<td>Easy-to-manage flexibility in design and reliability in operation</td>
<td>Jini, UPnP, SLP</td>
</tr>
<tr>
<td>2. All devices in a single administrative domain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Generally resource-rich devices with low mobility and high bandwidth network support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAN 1. Large number of devices and services available</td>
<td>Achieving scalability Maintaining consistency of stored information</td>
<td>SSDS, CSP, INS/Twine, Superstring, GloServ</td>
</tr>
<tr>
<td>2. No broadcast or multicast mechanisms available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad hoc Network 1. High mobility, dynamic topology, resource-constraint</td>
<td>Energy efficiency Achieving scalability Achieving reliability</td>
<td>Bluetooth SDP, LANES, GSD, Konark, Allia, Service Rings, DEAPspace</td>
</tr>
<tr>
<td>2. Extensive use of broadcast and multicast mechanisms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Service discovery in PvC:
- distributed, ad hoc, dynamic, fault-prone, constraint in resources.

Service Discovery: Fault Tolerance

- **Service Faults**
  - Causes: Service crash, mobility, and invalidation
  - Issues: how to find a similar service transparently

- **Directory Faults**
  - Causes: Directory crash, directory mobility
  - Issues: how to find and use another directory transparently, without the need of service re-registration at the new directory

- **Seamless Migration**
  - Issues: seamless service provisioning for mobile users across different locations to automatically replace failed or unavailable service providers with new ones
Service Composition

- Combining basic services into possibly complex services
- Typically works on the template matching principle
  - Requirements are specified in the form of a template
  - Runtime environment locates services that fit the placeholders in the template
  - Coordination among identified services is performed by the runtime environment.
- Complex tasks can be broken down into subtasks
  - One template for each subtask
- Invocation of identified services
  - Event based
  - Process based
  - Task dependant

Service Composition Approaches

- Centralized approach - fixed coordinator
  - A set of gateways at home, devices connected to a given gateway.
  - For instance, a home control gateway can coordinate the behaviors of specific devices like shutters or heaters.


Kumar and Cao
Service Composition Approaches

- Centralized approach – dynamic coordinator
  - Infrastructure-less service composition environment. Broker arbitration decides on the Composition Manager (CM) for a certain request. Each composite request may be assigned a different CM.
  - CM on receiving a composite request performs the process of service integration where it discovers the required services and constructs an execution-level service flow.

- HYBRID
  - A hierarchical service overlay mechanism, based on the capabilities of the devices that host the services.
  - Relatively resource-rich devices accommodate their resource poor counterparts, by assisting them in service composition.

More details in following slides
## Service Composition Approaches

### Disadvantages of centralized approaches
- Impractical in large scale systems with dynamic arrivals and departures of service providers
- Requires frequent updates of the central entities - large system overhead
- Relies on central entities to maintain global knowledge
- Inability of the system to serve the request when the central manager is compromised.

### Requirements for new solutions
- Absence of special entity to manage service composition process
- Service providers can communicate only with their local neighbours, not all other service providers
- Lack of global knowledge
- Services provided may be atomic

---

### Service composition in pervasive computing

- Two types of composition mechanisms
  - **Static**
    - Composition orchestrated prior to need
    - Capability to define finer interface dependency details
    - Ideal for stable, managed environments
    - *Insufficient support for dynamism*
  - **Dynamic**
    - Composition formed once the request arises
    - Can consider the current context/service availability
    - Costly in terms of time for composition.

Service composition used as a delivery vehicle for applications. Majority work on *discover + match* style
Service composition in pervasive computing: Discover + Match

- Identify suitable services, mediate interactions, transactions among identified services
  - Spidernet\(^1\), Reactive composition\(^2\), Konark\(^3\)
- Mostly variations of template matching schemes

---


SeSCo Approach

- **Use of PICO middleware**
  - Provides a platform for various devices to cooperate
  - Provides service constructs through software agents

- **Service oriented operation**
  - Manifest device features as services
  - User applications are service aware
    - Look for services rather than device features

- **Support through service composition**
  - Application requirements matched to available services
  - Resource limitations are taken into account

- **Services modeled as directed attributed graphs**
  - Attributes contain service data, service semantics
  - Edges define service interactions

- **Services are aggregated by the PICO middleware**
  - Aggregated services are used for Task Resolution

Middleware Architecture

- Telemedicine
- Manufacturing
- Smart home
- Community
- Delegates
- Devices
- Bluetooth
- 802.11b
- Cellular
- ...
Basic Building Blocks

- **Physical devices** –
  - Computer-enabled devices: small, wearable to large supercomputer
  - Sensing capabilities
  - Computational power
  - Communication capability
  - Actuators

- **Software entities** – Delegents
  - Intelligent SW agents – service provisioning
  - Proxy-capable: exist on the infrastructure
  - Event-driven
  - Execute on host devices
  - Need a host for execution
  - Mobile, capable of communicating

Devices and Delegents

**Devices**
- \( C = \langle C_D, S, F \rangle \)
  - \( C_D \): Device identifier
  - \( S \): System characteristics
  - \( F \): Functionality of device
- For example, \( C = \text{Heart Monitor} \)
  - \( S = \langle \text{operating system; processor type; memory; I/O type; battery; wireless transceiver} \rangle \)
  - \( F = \langle \text{ECG monitoring; processing; communicating} \rangle \)

**Intelligent Delegate**
- Works diligently on behalf of a device, user, application or service

- \( D = \langle \text{Did, Fd} \rangle \)
  - \( \text{Did} \): Delegent ID: \( \langle \text{Id, C, P} \rangle \)
    - \( \text{Id} \): Delegent ID
    - \( \text{C} \): Host Device ID
    - \( \text{P} \): Community
  - \( \text{Fd} \): Functionality of delegent: \( \langle M, R, S \rangle \)
    - \( M \): Program modules
    - \( R \): Delegent rules
    - \( S \): Delegent services

Kumar and Cao
Delegent Example

- A Delegent for ECG monitoring
  - **Functionality:** < Modules, Rules, Goals >
  - **Modules:** Signal processing module, Arrhythmia detector, Software filter, Timer, Communication module.
  - **Rules:** State transitions, Migration rules, Communication rules, Community engagement rules.
  - **Services:** Detect arrhythmia, Upload ECG window, Communicate status.

Rules are defined according to goals

- If heart-rate irregular
  - Start saving sensory data
  - Contact cell phone delegent (a new community)
  - Contact ambulance delegent (next level community)

- If heart-rate high/fast
  - Start saving sensory data
  - Collect data for a while
  - Contact physician ...

Service Abstraction

- A Delegent for ECG monitoring
  - **Functionality:** < Modules, Rules, Goals >
  - **Modules:** Signal processing module, Arrhythmia detector, Software filter, Timer, Communication module.
  - **Rules:** State transitions, Migration rules, Communication rules, Community engagement rules.
  - **Services:** Detect arrhythmia, Upload ECG window, Communicate status.

Authenticated

- Yes?
  - Place Call
  - Contact EMS

Authenticated

- No?
  - ST segment > threshold
    - Arrhythmia detected
  - ECG Window
  - Place Call

11/29/2010
Kumar and Cao
Service model

- Each service is treated as a transformational unit, accepting a set of inputs and producing a set of outputs
  - Derived from the state machine representing delegent operation
- Each service faithfully works towards its desired goal
  - Assumption -- there are no malicious services
  - Security and trust schemes need to be employed to enhance the model
- Services are represented using a directed, attributed graph
- Each service has a set of attributes associated
  - Each service is described using both semantic and syntactic attributes

Service model

- Each service is represented by
  \[ G_S = (V_s, E_s, \mu_s, \xi_s) \]
  
  \( V_s \rightarrow \) node (s) representing service (or the state machine for the service)
  
  \( E_s \rightarrow \) Edges to and from the service (I/O)
  
  \( \mu_s \rightarrow \) Vertex attribute function
  - Service Name, location, address, cost, etc.
  
  \( \xi_s \rightarrow \) Edge attribute function
  - Type and size of parameters and messages
- Attributes can also contain semantic descriptions of entities.

Service aggregation at the directory

Semantic aggregation

Syntactic aggregation

Aggregation of services \( G_P \)
Task resolution

Service aggregation process
(Centralized scheme, managed networks)

- Well known directory structure
  - Services register at the directory
  - Directory maintains all available services by aggregating them
- Directory acts as the point of contact for task support
- Service composition performed at the directory
  - Execution of composition left to the client
**Middleware Support**

High level complex services

Manifestation of resources as services

Device Abstractions

Device

---

**Service composition problem is NP-complete**

- Each service represented within the aggregation has a number of attributes associated with them
- Each node within the request comes with a number of parameters to be met
- The process of composition needs to satisfy multiple constraints on each edge during shortest path computation.
- Similar to the Multi-constrained path selection problem, which is NP hard
- Many heuristics exist to achieve polynomial time bounds
  - Average cost method $\rightarrow$ average edge weights
  - Limits on path weights
- We limit the number of services used in each composition
  - Composition length
  - Can be specified within the request
Centralized scheme: *limitations*

- Ideal to build and operate Service Provider communities, managed infrastructures
- Directory structure can be distributed, but all requests need to be resolved at the directory
- Ideal to build SOAs around assets in the infrastructure
  - Managed service definitions
  - Managed resources, therefore managed directory
  - Access to different resources can be controlled
- Directory node becomes the bottleneck
  - No explicit support for heterogeneous resources
  - No explicit support for dynamisms

Seamless service composition (SeSCo)

- **Main Motivation**
  - Resource heterogeneity to be actively exploited
  - Resource poor devices need to be proactively supported by their resource rich counterparts
- **User centric solutions**
  - ‘What, where, and how’ type of services
  - Dynamisms associated with user mobility
  - Quality of composed service
PICO middleware support to device heterogeneity

- Based on the capability of the device
  - Multiple forms of the PICO middleware
  - Each identical in operation, but different in its capabilities
  - Resource poor devices are relieved of a number of tasks such as service discovery, aggregation, composition, coordination, etc.
  - Resource rich devices take care of their resource poor counterparts.
- Capability of the device reflected in its device profile (Device Specification)
- Profile represented as the level of the device (\( \alpha \)).
- Levels 0 .. 3 0- lowest, 3 - highest

Creating the hierarchy

The LATCH protocol
- Devices lower in the hierarchy latch onto those higher in the hierarchy
- Service related information (Service graphs) exchanged during hierarchy creation
  - Includes services available on the local device
  - Plus the services available at all descendant nodes
- Staged aggregation for improved composition support.
- Status changes reflected through periodic updates.
Hierarchical service composition

Addressing challenges in pervasive computing

- Heterogeneity
- Transparency
- Locality
  - Composed services need to be as close to the user as possible
  - Important in user centric solutions.
    - Location based systems
    - Multimedia solutions
  - Hierarchical service composition
    - Utilize services closer to users prior to using those farther away
  - Rendering services need to be located as close to the user as possible.
The middleware model

(Reference implementation)

Delegent Manager
- Delegent Specs (XML Doc)
- Java class modules
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement
- Equipped with XML parser to manage delegent life cycle based on the delegent specifications
- Notify/subscribe style event management, implemented in Java
- To interface with native libraries and Java implement

JAVA IMPLEMENTATION

Event Manager
- Context Manager
- Profile Manager
- Notify/subscribe style event management, implemented in Java

Adaptation Library
- Service layer
- To provide underlying communication transparency
- Responsible for Service discovery, composition and identification of Proxy
- To interface with native libraries and hardware dependent features (sensors, etc.)

DATA MANAGEMENT

11/29/2010

Kumar and Cao
Data Management: Challenges

• Maintain and process a vast amount of heterogeneous, dynamic information
  – Data from everything
  – Data from everyday activities (anytime)

• Multiple sources of same data
  – Inconsistency of contextual data
  – Diverse formats of the same contextual data
  – Redundant or useless contextual data
  – Managing replicas of contextual data (storage, update)
  – Different media forms of contextual data
  – Spatial distribution of contextual data
    • Where to store contextual data (people, space, objects)
  – Different policies regarding contextual data on different sources
    • Privacy and Security

Data Management: Challenges cont.

• Frequent changes of data
  – Updating their relationships with other data items
    • Extremely challenging as data items can share multiple relations with other data items
    • Can we store related / linked data items physically close in order to achieve low update delay?
  – Ensuring consistency among distributed data stores if the data items share relationship
  – The effect of the different updates must be atomic (separately manifested)
  – Contextual data updates may start cascading contextual changes in other smart objects
    • E.g. decrease of temperature may start room heater and close windows and preparation of coffee
Data Management: Challenges cont.

- **Security and privacy**
  - *Random devices interact in random environment*
    - Reliable information
    - Privacy protection
  - *Common approaches like the public-key infrastructure cannot be used due to the resource limitations*
  - *Vast amounts of personal data will be temporarily collected, e.g., location privacy*
  - *Devices will more and more operate in an unknown and potentially hostile environment*

Data Management: Challenges cont.

- **Dynamic of data sources and consumers**
  - *High mobility*
    - Access point is changing
    - Data sources and consumers come and go
    - Randomness of every device’s neighborhood at any instance of time.
  - *Dynamic data interests*
    - Context is changing
      - User intention is changing
Data Management: Challenges cont.

- Ubiquitous data access
  - Availability (anywhere, anytime, any devices)
  - Invisible user experience (seamlessly connecting)
  - Access the right data at right time
  - Latency, correctness, relevance (Qos guarantee)
  - Understanding user’s intention
  - High distribution of data storage

Data Management: Related Techniques

- Semantic data integration
  - Utilize semantic annotations for smarter (ontology-enabled) data discovery, semantic type checking and conversion
  - Leverage standard semantic web technology, e.g., OWL Web Ontology Language
  - Top-down view (classification)
  - Have to know relationship – predefined (reasoning is dynamic)
Data Management: Related Techniques

• **Semantic Caching (maintenance)**
  – Maintaining a semantic description of cached data, to reduce the response time and communication cost.
  – The semantic description enables the cache to provide partial answers to queries which do not match the cached data exactly.
  – When a query is posed at a cache, it splits into two disjoint pieces:
    • a probe query, which retrieves the portion of the result available in the local cache,
    • a remainder query, which retrieves any missing tuples in the answer from the server. If the remainder query exists then it is sent to the server for processing.

Data Management: Related Techniques cont.

• **Cache replacement**
  – Random replacement
    • Perhaps high miss rate
  – First in, first out (FIFO)
    • Oldest is most likely not needed anymore
  – Least recently used (LRU)
    • The one that has been unused for the longest time is most likely not needed anymore
Prefetching

- Data is preloaded in the client cache for possible later use.
- Given a limited cache size, intelligent prefetching based on the user’s intention can maximize the cache usage.
- For instance, if a user is going to pass a supermarket, the cache manager may prefetch shopping information.

Passive prefetch

- Combining prefetching and caching

Privacy-enhancing technologies

- Access control
  - Limit the number of people that can access your location information
- Encryption
  - Prevent unauthorized access to communications, data
- Security policy
  - Constraints on functions, flow, data
- Anonymity
  - Enable users to communicate anonymously
CONTEXT MANAGEMENT

11/29/2010 77

Kumar and Cao

Context Management

- Context acquisition
  - Collect raw contextual information
  - The collected information tends to be noisy and inconsistent

- Context representation
  - Construct high level abstraction of contextual data and their relationship
  - Facilitate the reasoning about and querying of contexts

- Context derivation
  - Derive high-level contexts from raw contexts
  - Derive implicit contexts from explicit contexts

11/29/2010 78

Kumar and Cao
Context Management: Context Acquisition

- **Context definition**
  - “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”
  - A. K. Dey, 2000

- **Context categories**
  - Physical contexts: noise, light, temperature,…
  - Computing contexts: network condition, CPU, memory,…
  - User contexts: location, activity, status,…

Earthquake Disaster Recovery

- Employ pervasive technologies to opportunistically build social and situational network models in support of context reasoning during temporary relief efforts.

- Identify and prioritize critical areas for effective resource assignment
- Effectively locate trapped victims
- Track rescue supplies to avoid waste and abuse
- Resolve missing persons reports
- Resolve mail delivery backlogs
Earthquake Disaster Recovery
Effectively Selecting Reasoning

1. Effective Context Reasoning
2. Model Modularization & Context Selection
3. Identify context sources/inputs
4. Select suitable reasoning
5. Identify reasoning needed
6. Consider context quality requirements

---

Earthquake Disaster Recovery
Context Analysis

- SocialNetwork
  - Demographic Correlation
  - Victim Reports
  - WordOfMouth
- CriticalSituation
  - TrappedVictim
  - Location Inconsistency
- Observation
  - Strength Distance Analysis
  - Sound Classification
  - RFIDMap
  - GPSMap
  - Profile
  - MicWaveSample
  - MedicalRecords
  - PublicRecords
  - MicWaveSample
  - RFID
  - GPS
  - MicroWaveSample
Earthquake Disaster Recovery

Context Analysis

- Hierarchical
- Modular
- Heterogeneous: uncertainty, temporality, complexity
- Quality Considerations: accuracy, precision, freshness
- Target Platform Restrictions: CPU, RAM, bandwidth, mobility
- Application Requirements: Inference speed, scalability,

Earthquake Disaster Recovery

Effectively Selecting Reasoning

<table>
<thead>
<tr>
<th>Context</th>
<th>Inputs</th>
<th>Potential Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Tracking</td>
<td>GPS, RFID, Profiles, Observation</td>
<td>Dynamic Bayesian Network (DBNs)</td>
</tr>
<tr>
<td>Locate Trapped Victims</td>
<td>Sound Clips from (Microphone), Signal Strength Metrics Syncs of other KBs</td>
<td>Signal Processing (voice, breathing distance), w/ BNs</td>
</tr>
<tr>
<td>Identify and Prioritize Critical Areas</td>
<td>Profiles, Syncs of other KBs</td>
<td>Rules, Logic, BN,DL Inference</td>
</tr>
<tr>
<td>Locate Victims (for missing reports &amp; mail delivery)</td>
<td>Observation, Word Of Mouth, Victim Reports, Social Network Contexts Learned</td>
<td>Rules, Logic, BN, Search Algorithms</td>
</tr>
</tbody>
</table>

In general, DL reasoning is effective for KB
Earthquake Disaster Recovery
Effectively Selecting Reasoning

<table>
<thead>
<tr>
<th>Context</th>
<th>Inputs</th>
<th>Potential Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Tracking</td>
<td>GPS, RFID, Profiles</td>
<td>Dynamic Bayesian</td>
</tr>
<tr>
<td>Locate Trapped Victims</td>
<td>Sound Clips (Microphone), Signal Strength Metrics, Syncs of other KBs</td>
<td></td>
</tr>
<tr>
<td>Identify and Prioritize Critical Areas</td>
<td>Profile, Syncs</td>
<td></td>
</tr>
<tr>
<td>Locate Victims (for missing reports &amp; mail delivery)</td>
<td>Observation, Mouth, Social, Contexts Learned</td>
<td></td>
</tr>
</tbody>
</table>

Context Challenge

- Integrated, customization reasoning approaches are needed to completely & effectively infer contexts in complex context-aware applications

In general, DL reasoning is effective for KB consistency checking, reclassification, subsumption and instance based inference.

HyCoRE High Level Architecture

HyCoRE High Level Architecture

Context Reasoning
- Context Flow – integration of reusable reasoning for HL inference
- Discovery of other context sources, reasoners and context data models
- Client and server reasoning operation

Context Management Support
- Standalone & distributed operation
- Collaborative context inputs from outside sources
- Synchronization & knowledge mapping with other knowledge stores

Earthquake Disaster Recovery

Limited reasoning

Complex offline situation assessment

Disaster Distribution Center

Limited reasoning

Context sharing

11/29/2010
Related Work

**CoBrA**


**ToolBox**


**Nexus**


---

Overview of High Level Context

- Modular
- Hierarchical
- Multi-sourced

Kumar and Cao
Overview of High Level Context

- Modular
- Hierarchical
- Multi-sourced

Social Context Example

- High Level Contexts
- Location
- Activities
- Availability
- Proximity
Social Context Example

• High Level Contexts
  • Location
  • Activities
  • Availability
  • Proximity

Context Providers
Sensing Devices: GPS, Wi-Fi, Accelerometer, Microphone, Camera
Data Services: Calendar, Web Repositories
Reasoning: FOL, Decision Trees, Belief Networks, Rules
Social Context Example

Context Sources

Sensing Devices: GPS, Wi-Fi, Accelerometer, Microphone, Camera
Data Services: Calendar, Web Repositories
Reasoning: FOL, Decision Trees, Belief Networks, Rules

• High Level Contexts
  • Location
  • Activities
  • Availability
  • Proximity

Social Context Example

Context Providers

Sensing Devices: GPS, Wi-Fi, Accelerometer, Microphone, Camera
Data Services: Calendar, Web Repositories
Reasoning: FOL, Decision Trees, Belief Networks, Rules

Use Quality Measures

Latency, availability, accuracy, fidelity, credibility

• High Level Contexts
  • Location
  • Activities
  • Availability
  • Proximity

Kumar and Cao
Social Context Example

Context Providers
Sensing Devices: GPS, Wi-Fi, Accelerometer, Microphone, Camera
Data Services: Calendar, Web Repositories
Reasoning: FOL, Decision Trees, Belief Networks, Rules

Which Provider?
Use Quality Measures
Which Quality Measures?
Latency, availability, accuracy, fidelity, credibility

• High Level Contexts
• Location
• Activities
• Availability
• Proximity

Kumar and Cao
Social Context Example

What happens to quality if we manipulate source data?

**Context Providers**
- Sensing Devices: GPS, Wi-Fi, Accelerometer, Microphone, Camera
- Data Services: Calendar, Web Repositories

**Reasoning:** FOL, Decision Trees, Belief Networks, Rules

- **Which Provider?**
- **Use Quality Measures**
- **Which Quality Measures?**
- **Application Declared**

**Abstract View Of High Level Context Reasoning (Context Flows)**

B. Beamon and M. Kumar, Adaptive Context Reasoning in Pervasive Systems, 9th Workshop on Adaptive and Reflexive Middleware (ARM 2010), ACM Middleware 2010, November 30-December 2, 2010, Bangalore,
Abstract View Of High Level Context Reasoning (Context Flow)

Disparate Component Quality Indications, Edge Latency

Kumar and Cao

Quality Aggregation

Composite Quality
- Latency: 1000ms
- Accuracy: 92%
- Credibility: 85%

Kumar and Cao
Quality Aggregation

- Challenges:
  - Combining heterogeneous context
  - Coping with missing context or quality indicators
  - Handling redundant or conflicting context

High Level Context

Composite Quality
- Latency: 1000ms
- Accuracy: 92%
- Credibility: 85%

Kumar and Cao

Quality Propagation

- Challenge: Accurately reflecting the reasoning transformation process in resulting inferred context quality indicators

High Level Context

QoC Composite Context Quality

Kumar and Cao
Social Network Example continua…

User Centric High Level Context

- **Location** (home, work, school, restaurant, car, gym)
- **Activities** (inTransit, inMeeting, workingOut, shopping, eating, watchingTelevision, listeningToMusic)
- **Availability** (cell, mobile, email, text, landline, Skype)

High Level Context (QoC)

Reasoners

Adaptable Context Reasoning Plans

Context Middleware

Context Providers

Context Provider Registration (QI)

Context Data Reporting (QI)

GPS, Wi-Fi, Camera, Microphone, Calendar, Accelerometer, Environmental

Adapt Using QoC Measures

Composite Quality Indicators
- freshness, fidelity, completeness, credibility, proximity, cost, latency semantic equivalency, fluidity

Intrinsic Data Quality Indicators
- freshness, fidelity, fluidity

Context Provider Quality Indicators
- availability, error rate, refresh rate, credibility, precision, accuracy, security, sensitivity, (temporal, spatial and informational) resolution

Reasoning Quality Indicators
- inference latency, accuracy, (power, CPU, memory) utilization, error rate, precision, (temporal, spatial and informational) resolution

Kumar and Cao
Middleware Limitation Considerations

- We must balance meeting application requirements with the constraints either defined or intrinsic to our system.

- 1. It is possible that we can only come near to meeting application requirements
- 2. It is also possible providing too much quality. Far exceeding application expectations.

- Context Reasoning adaptation is optimal when application requirements are met while minimizing system cost.

QoCS-Middleware Performance Measures

- Middleware Context Effectiveness
- Middleware Context Cost
- Middleware Operational Efficiency
Related Work

Orchestrator – context processing plans to maximize context reuse – handles node dynamics


MoBe – mobile adapts to user context, loading and uploading applications


Information Quality – quality aggregation using qualities of individual components


“Adaptive Context Reasoning in Pervasive Systems” -

Context Quality


“Adaptive Context Reasoning in Pervasive Systems” -

Adaptive Context Middleware


“Adaptive Context Reasoning in Pervasive Systems” -
WSN Research

• **Building the network**
  - Techniques for deploying and establishing WSN
  - Coverage, connectivity, MAC, routing, clustering, mobility, localization, time synchronization

• **Managing the network**
  - Techniques for monitoring and controlling WSN
  - Energy management, topology control, fault detection and recovery

• **Using the network**
  - Techniques for developing and supporting execution of applications
  - Programming, runtime, data collection and processing (aggregation, fusion, mining), event detection, object tracking

• **Orthogonal issues**
  - Energy efficiency, network lifetime, resource and capability constraints, fault tolerance, scalability, security, accuracy and timeliness
Motivations

• Programming and run-time support of WSN applications remains to be a non-trivial and challenging task
  – Gap between the high-level requirements of WSN applications, and the complex, low-level operations.
    • Interfacing with hardware and network protocols
    • Programming of sensor nodes and their interactions
    • Code deployment and re-deployment
    • Data management (collection, aggregation, fusion)
    • ......

Motivations

• Middleware are software that provide support to the programmer to help bridge the gap
  – *glues together the network hardware, OS, and applications*
  – *hide the complexity and heterogeneity of the underlying hardware and network platforms,*
  – *ease the control of system resources,*
  – *increase the predictability of application executions*
Motivations

• **WSN middleware**
  
  – Support sensing-based applications that make use of the sensor nodes with embedded OS and their collaborations in WSN.
  
  – Provide concepts and abstractions of sensor nodes and sensor data.
  
  – Provide runtime mechanisms, and application-specific services
    
    • facilitate development, maintenance, deployment and execution of sensing-based applications

Motivations

• **WSN poses new challenges to middleware research:**
  
  – Transparent vs. Context aware & adaptive
  
  – Heavy weight (resource rich) vs. Light weight (resource constraint)
  
  – Node centric vs. Data centric (dynamic available)
  
  – Middleware generality vs. Application specificity
  
  – Large scale of deployment, fault-prone nodes, timeliness, security.
WSN Middleware

- **Functionalities:**
  - *High-level programming abstractions:*
    - define the interface of the middleware to the application programmer
    - hide the complexity of individual nodes & their interactions, and provide a holistic view of the network
  - *System services:*
    - provide implementations to achieve the abstractions.
    - develop scalable, adaptive and efficient protocols and algorithms – often encapsulate network-level protocols (e.g., routing) and provide data services with energy saving, robustness, security, etc.
  - *Run-time support:*
    - serves as an extension of the embedded OS to provide a runtime environment to support and coordinate multiple applications.

11/29/2010

![Diagram of WSN Middleware](image)
WSN Middleware Paradigms

Conceptual models (level of abstractions)

- Global behavior (macroprogramming)
- Local behavior (data centric, geometric)

Systems models

- Programming wireless sensor networks
- Programming support
  - Virtual machine
  - Database
  - Modular (agents)
  - Application driven
  - Message-oriented middleware

Virtual Machine-based Abstractions

- Similarity to the VM concept in traditional distributed systems
  - providing application semantic transparency from the physical infrastructure
- Provide a set of ISA-like instructions
- Programs written in those instructions can be 'injected' into the network
- Users may define additional instructions
- Possible to interface with high-level languages such as Java
- Maté (ACM ASPLOS’02), MagnetOS (ACM OSReview’02)
VM-based Approaches

Maté (UC Berkley):
- Tied to TinyOS
- Power-centric abstraction
- Simple API to nodes
  - 6 instructions only for sense and send program
- MATE programs are broken up into small instruction capsules (24 byte-long), which form a unit of distribution.
- Communication is synchronous.
- No support for message buffering / large storage.

MagnetOS (Cornell):
- Power-aware, adaptive OS.
- The whole network appears as a single JVM.
- Standard Java programs are re-written as network components.
- Components may then be ‘injected’ into the network using a power-optimized scheme.
VM-based Abstractions

• **Benefit**
  – Suitable for a wide range of applications
  – Extensible – users can define their own instructions
  – Energy efficiency – reduce in code size helps save energy
    for code dissemination

• **Drawback**
  – Programs written in ISA-like instructions are still difficult to
    read / maintain
  – Very simple, fixed network abstraction
  – Assume dump sensor nodes (no intelligence)
  – Impossible to exploit heterogeneity

---

Database-based Abstractions

• **Network is abstracted as a database**
  – represents sensors and sensor data in a database

• **Control of sensors and extracting data occurs**
  through special SQL-like queries

```sql
SELECT Nodeid, Light
FROM Sensors
WHERE Light > 400
EPOCH DURATION 1s
```

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Nodeid</th>
<th>Light</th>
<th>Temp</th>
<th>Accel</th>
<th>Sound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>455</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>399</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>422</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>405</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

• **TinyDB (TODS 05 – UC Berkley),**
  **Cougar (MDM’01 - Cornell),**
Database-based Abstractions

• **TinyDB (UC Berkley)**
  - A *distributed query processor for networks of Mica motes.*
    - On top of TinyOS
  - **Features**
    - Declarative queries
    - Temporal + spatial operations
    - Multi-hop routing
    - In-network storage

Database-based Abstractions

• **SINA (U. Delaware)**
  - *Models the network as massively distributed database.*
    - Network is viewed as a collection of datasheets
    - Each datasheet contains a collection of attributes of each sensor nodes.
    - Each attribute is referred to as a cell.
    - New cells can be created based on the requests.
Database-based Abstractions

- **SINA (U. Delaware)**
  - Allow applications to issue queries and command tasks into, collect results from the network.
  - SQTL (Sensor Query and task Language)
  - Has the capability to interpret simple declarative query statements.
    - `getTemperature`, `turnOn`, `isNeighbor`, `getPosition`, `tell`, `execute`.
  - Based on three functional design:
    - Hierarchical Clustering
    - Attribute-based Naming. e.g `[ type = temperature, location = N-E, temperature > 100 ]`
    - Location Awareness (GPS, optical trackers etc.)

- **Benefit**
  - Easy to use – SQL is familiar to application programmers
  - Very effective in monitoring systems – filters can be easily defined by the where clause

- **Drawback**
  - Not suitable for applications with strong requirements on temporal relationships (e.g. object tracking).
  - Still assumes somewhat dumb nodes, e.g. data is always routed direct to sink, with no scope for in-network processing.
  - Not real time - does not tie in well with monitoring which is largely event-driven.
Agent-based Abstractions

- Modular programming
  - *Modularity of applications facilitate the injection and distribution of code through the network*
  - *Transmitting small modules consumes less energy and supports incomplete updates*
- May support code mobility – mobile agents
- Impala (PPoPP’03), Agilla (IPSN’05), SensorWare (MobiSys’03)

Impala (Princeton):
- *Implementation as part of the ZebraNet project*
- *Focus on efficiency of adaptability and updates to support dynamic applications.*
  - Provides reliability and ease of upgrades for long-running sensor network applications
- *Event-based programming model*
- *Efficient updates are achieved at the expense of heterogeneity (updates are compiled to binary units).*
Agent-based Abstractions

Impala (Princeton)

- Adapter is required to:
  - Increase performance by re-parameterizing the application
  - Improve robustness choosing alternative protocols in case of hardware failures

- Adaptation Finite State Machines are used for parameter-based adaptation
  - \( P_0 \) = Avg number of direct neighbors over the last k cycles
  - \( P_1 \) = battery level

- Device-based adaptation is performed on the basis of Application Device Tables
Agent-based Abstractions

**Impala (Princeton)**
- Updater is designed to handle the following issues:
  - Incomplete updates
  - On-the-fly update of code while executing
  - Propagation protocol
  - Code memory management
- Approach
  - Linking performed on the nodes
  - Use of version numbers
  - Epidemic software transmission

**Agilla (Washington U at St. Louis):**
- Mobile agent-based middleware: provides runtime environments to support efficient code migration

Two variants of each:
1) **Strong** (code + state)
2) **Weak** (code only)
Agent-based Abstractions

Agilla (Washington U at St. Louis):

- Flexible application deployment in dynamic environments.
- Provide support for tuple space operations
- based on Maté and inherits the problems coming with VM-based approaches
Agent-based Abstractions

• Benefit
  – *Multiple users can inject agents into the network*
  – Allows multiple applications to be run in the network at the same time
  – *Efficient approach for object-tracking and robot navigation applications*

• Drawback
  – *May result in high overhead – additional storage and communication cost is introduced to the runtime environment to support code migration*

Event-based Abstractions

• Data acquisition through correctly detecting the events in a timely manner
  – *Application specifies the interest in certain state changes of the data.*
  – *Upon detecting such an event, middleware help send event notification to interested applications.*
  – *Uses data centric storage.*
  – *Can support compound events by combining basic events.*

• Pub/Sub paradigm:
  – *Extends the event-based abstraction by providing additional components including subscriber, publisher and broker.*
  – *Provides better decoupling by allowing subscribers (applications) to subscribe events.*

• DSWare [IPSN’03 – U. Virginia], Mires [MPAC’04], Impala [PPoPP’03]
Event-based Abstractions

Mires [Brazil]

- Pub/Sub allows the communication between middleware services
- responsible for advertises topics
- maintains the list of topics subscribed by the node application
- publishes messages containing data related to the advertised topics

11/29/2010

PublishState interface define the command used by ServiceX to publish their processing results.
Notifier interface defines 3 events
MultiHopRouter - route to the sink
BCast - Broadcast Setup info.

11/29/2010
Event-based Abstractions

Mires [Brazil]

Topic advertisement sequence diagram

Event-based Abstractions

Mires [Brazil]

Topic subscription sequence diagram
Event-based Abstractions

Mires [Brazil]

Data publishing sequence diagram

- Benefit
  - Events are easy to define
  - Event model closely ties to monitoring role
  - Data-centric storage helps to save energy

- Limitation
  - Currently, only simple event combination is considered
System Services

- **Data management**
  - *Data acquisition*
    - Data query – query dissemination, data collection / dissemination
    - Event-based – event definition, subscription, notification
  - *Data processing*
    - Data calibration – require time synchronization
    - Data aggregation, fusion
    - Detection of event region, event boundary
  - *Data storage*
    - F-T storing / caching and look-up of data with associate semantics (e.g., co-related in space)

System Services

- **Query dissemination**

  Semantic Routing Tree in TinyDB (TODS’05)
  - A routing tree which is also an index over an attribute such as location:
  - Queries are forwarded towards the sensor nodes which have the relevant data.
Introduction: WSN Middleware and Resource Management

- Resource Management - essential ingredient of any middleware includes
  - Initial sensor selection and task allocation
  - Runtime adaptation of allocated resources to tasks

- Why study of resource management in WSN is important?
  - Extremely resource constrained
  - High dynamics and uncertainty
  - Large quantity of sensor nodes in un-attended environment
  - Heterogeneous nodes
  - Multiple heterogeneous applications

Example Scenario: Object Tracking Application
Kumar and Cao

How to satisfy application constraints?
Reliability, Tracking Error, Coverage etc

How to optimize resources: Energy, lifetime?

How to handle change in application state and requirements?
Problem Description: Resource Management in WSN

Given application structure, QoS requirements and current system state, what is the best strategy for allocating tasks to resources so that a given system-wide, application-driven, global parameter can be optimized?


WSN Middleware: Requirements and Design Principles

- High level abstractions
  - Data-centric
  - Publish/subscribe, event based, push/pull
- Efficiency (energy, bandwidth, computational resources)
  - Localized algorithms
  - Data-reduction and other in-network processing techniques utilizing application specific knowledge
- Adaptive
  - Tackle uncertainty and dynamic nature of WSN
  - Pro-active adaptation of state changes and application requirements
- Support multiple applications with their QoS requirements
  - Design of generic framework
  - Task and resource management across multiple applications
Resource Management in WSN: Classification

- **Rule and Predicate logic based** [Liu03,Frank05]
  - requires that all state conditions be known in advance
  - very complex with large number of nodes and high dynamics

- **Constraint-satisfaction based** [Krishnamachari03,Kogekar04,Modi05]
  - difficult to reduce our problem into a linear programming
  - distributed processing is very difficult

- **Agent negotiation/auctions based** [Lesser03]
  - can lead to better resource management
  - high communication and computational resources required.
Resource Management in WSN Classification

- Utility/Market based [Byers00, Mainland05, Sadagopan06]
  - Uses utility functions mapping optimization parameters to real value
  - [Byers00] first to introduce utility functions to WSN
  - [Mainland05] uses market based approach with heuristic reinforcement learning
    - Nodes receive payments for price of goods they produce
    - Reactive system with no state-based learning
    - Doesn't address global behavior

- Common Issues:
  - Uncertainty not addressed
  - Require implementation of algorithms on a case-by-case basis
    - Difficult to create a generic solution
  - Process distribution either very difficult or bandwidth and processing intensive


DReL: Middleware for WSN Management

- **Goal**: Design of concrete middleware that include
  - Easy-to-use abstractions for supporting multiple heterogeneous WSN applications
  - Design of mechanisms for task, data and reward distribution
  - Design of associated data structures

**Problem Description: Resource Management in WSN**

- **Design Principles**
  1. Bottom-up approach
  2. Achieve application’s global optimization goal
  3. Tackle uncertainty/dynamics by continuous autonomous adaptation
  4. Ability to handle change in application’s requirement over time
  5. Generic framework that can facilitate development of wide range of WSN applications
  6. Easy-to-use abstractions for development of WSN applications
  7. All localized interactions that are confined to neighboring nodes
  8. Data-centric task and data dissemination using publish-subscribe mechanism, application specific in-stream data processing/filtering
Directed Diffusion

Pros
- Data centric communication paradigm, publish-subscribe model
- Effective task and data dissemination in WSN using only localized interactions among neighbors

Cons
- Uses of static scheduling of sensor nodes
- Allows optimization of data-dissemination based on only a single metric, i.e. lowest delay path, cannot do application specific optimization goal
- Uses reinforcement but doesn’t give any concrete mechanism and rules

DReL: Middleware for WSN Management

- Attribute-based data-centric naming scheme
- Attributes help associating messages with source, sink and filters via matching
- Operator allows specification of simple constraints on data, e.g. (<, >, equality)

<table>
<thead>
<tr>
<th>Task Description (Interest Packet)</th>
<th>Data Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>/* Fixed values only set by sink*/</td>
<td>/* Fixed values only set by source node*/</td>
</tr>
<tr>
<td>taskid /* Unique task Id */</td>
<td>taskid</td>
</tr>
<tr>
<td>attributes /* Set of &lt;key,type,operator,value&gt; tuples one for each variable sink is interested in */</td>
<td>timestamp /*event generation time */</td>
</tr>
<tr>
<td>timestamp /* task publication time */</td>
<td>attributes /* Set of &lt;key,type,operator,value&gt; tuples that contains actual data */</td>
</tr>
<tr>
<td>qosConstraints /* Set of &lt;key,type,operator,value&gt; tuples representing QoS requirements */</td>
<td>streamid /<em>id of data stream this packet belongs to</em>/</td>
</tr>
<tr>
<td>costParameters /* Set of optimization parameters with their weight factor */</td>
<td>/* Variables that can be updated by any node */</td>
</tr>
<tr>
<td>/* Variables that can be updated by any node */</td>
<td>/* Variables that can be updated by any node */</td>
</tr>
<tr>
<td>payment /* Expected payment/reward for matching data packet (used for setting gradients)</td>
<td>sourced /*id of sending node */</td>
</tr>
<tr>
<td>cost /*running sum of cost associated with this data */</td>
<td>reward /*reward expected by sending node */</td>
</tr>
<tr>
<td>reward /*reward expected by sending node */</td>
<td>/* Variables that can be updated by any node */</td>
</tr>
</tbody>
</table>
Micro-learner: Q-learning [Watkins92]

- **Uses Q-learning, as**
  - *Simple and demands minimal computational resources.*
  - *Doesn't require a model of the environment in order to operate*
  - *Practical for implementation on resource-constrained sensor nodes.*

- **Q-learning uses an utility look-up table** $Q(s,t)$ **across states** $s$ **and tasks** $t$
  $$Q(s,t) = (1-\alpha) Q(s,t) + \alpha ( r + \gamma e(s'))$$
  where $e(s') = \text{Maximum } Q(s',t)$ over all tasks $t$
  - *'\alpha' is a learning-rate parameter between '0' and '1'*
  - *'\gamma' is a discount-factor and also varies from '0' to '1'*

**Importance of '\alpha' and '\gamma'**

Micro-learner: Task selection

At each time step $y$

State of micro-learner constitutes of:
- Success in recent Diffuse task
- Success in recent Source task $T$

Allows micro-learner to learn whether it is acting as a source or router or not participating.

DIRL: Algorithm Performed By DIRL

12/1/2010
Macro-learner: COIN theory

- **Collectives**
  WSN with each sensor node trying to maximize its utility $q(\mathcal{C})$ while goal is to maximize system-wide performance criteria $G(\mathcal{C})$ over world-line $\mathcal{C}$.

- **Subworlds**
  Sets creating exhaustive partition of agents sharing same subworld utility function $g_w(\mathcal{C})$.

- **Constrained-aligned**
  No two agents in separate sub-worlds affect each other.

---


---

Macro-learner: COIN theory

- **Subworld-factored system**
  System where side-effects of subworld $w$ increasing its own utility do not end up decreasing world utility i.e.
  
  $$ g_w(\mathcal{C}) \geq g_w(\mathcal{C}') \iff G(\mathcal{C}) \geq G(\mathcal{C}') $$

- **Learnability**
  Measure of how well RL algorithm can learn to optimize the utility function.

- **Wonderful Life Utility (WLU)**
  $$ g_w(\mathcal{C}) = G(\mathcal{C}) - CL_w(\mathcal{C}) $$

  A constraint-aligned system with WL subworld-utilities is subworld-factored.

- **Optimality**
  A collective system which is subworld-factored and has higher learnability, reaches a Nash Equilibrium point which is also pareto optimal point.
DReL: Task Dissemination/Gradient Setup

- Determine if task can be executed locally
- If so, add task to its list

DReL: Data Dissemination/Reinforcement

- Receive/Produce data pkt
- Update cost/reward of pkt
- Choose neighbor with max gradient or send exploration packet
- If no gradient, broadcast
DReL: Cost/Reward functions

Cost of participation in task $T$ by node $i$:

$$cost_{IT} = \max(t_{max} - t_i, 0) \cdot cost_i + weight_i + cost_h + weight_{th}$$

$$t_i = \frac{E_i(1 + \lambda K_i)}{I_i + \lambda K_i}$$

Payment to intermediate node:

$$payment_{IT} = \arg \max_{s \in N} \text{gradient}_{IT} - cost_{IT}$$

Reward for a source node for performing task $T$

$$reward_{IT} = (success_T \cdot sourcePayment_{IT}) - cost_{IT}$$

Global Reward in context of neighborhood

$$R_{IT}(\xi_{IT}) = quality_{AVT} \cdot payable_{IT} - \min_{w \in W} cost_{wT}$$

DReL: Interactions

11/29/2010

11/29/2010

12/1/2010
OPPORTUNISTIC COMPUTING

What is opportunistic computing?

More than exchange of packets/bundles

- Content distribution and information management
- Remote task execution
- Cyber foraging
- Resource sharing
- Service composition
- Trust and authentication
- Enabling pervasive applications
- Anywhere, anyhow, but later
**Delay/disruption tolerant applications**

**Soft-real time applications**
- Vehicle-to-vehicle data dissemination
- Traffic monitoring
- Collaboration among robots

**Sender-receiver disconnection tolerant**
- Document transfers
- Remote task execution

**Non-critical monitoring applications**
- Tagging animals
- Dissemination of events

Email, FTP, message passing

---

**Opportunistic Computing**

Pervasive Applications

Opportunistic Computing
- Trust, security, Collaboration
- Social Computing
- Opportunistic communications

Middleware services

Sensor systems
Pervasive computing

Distributed and Fault-tolerant Computing

Mobile and Ad hoc networking

Algorithms, protocols and schemes

Heterogeneous device and communication technologies

Opportunistic contacts

Cell phones
- 4 billion users worldwide
- Internet Population – 1.3 Billion (2008)
- Global annual growth – 22%
- One in three persons carry a cell phone
- More than 1 billion opportunistic contacts at any given time
- Not counting sensors and RFID Tags

10 billion ARM processors
- In cell phones and other mobile devices

Millions of vehicles on the road
- Many equipped with cameras, computing devices, GPS systems

In a typical downtown (CBD) area
- O(100) street cameras
- O(1000) user cameras
- O(1000) user devices, laptops, PDAs
- O(100) desktops, infoservers

Potential

One Terra opportunistic contacts
- Each processor
  - 100 MIPS
  - 1K distributed tasks per second
- Each contact
  - 200 kb/s (conservative)
  - 5 seconds
- At any time instant
  - 1 Peta distributed tasks
  - 1 Peta bytes of data exchange
Opportunistic path

Comprises multiple opportunistic contacts

Path delay
- number of contacts
- $\Delta$ is the expected delay for each contact

Intermediate nodes
- Store and forward
- Possess adequate buffer space

Opportunistic paths

Challenge: Establishing reliable path for cooperation and collaboration
Routing and Forwarding

Dissemination-based and context-based

- **Dissemination**
  - Message is forwarded everywhere
  - Resource intensive
  - Epidemic routing [Vahdat00]
  - Controlled probabilistic routing [Oikonomou07]
  - PROPHET Coding [Lindgren03]
  - Network Coding [Widmer05]
- **Context**
  - Identify next hop based on context
  - Context-aware routing [Musolesi05]
  - Mobyspace routing [Leguay06]
  - HiBOp [Boldrini07]

*Challenge: Manage information efficiently*

Controlled dissemination – what you want, where you want

Social networking

Social behavior

- Mobility models
- Routing Schemes
- Forwarding decisions

Social structures

- Cooperate and communicate
- Smart pervasive environments

*Socialnets vision*

- **Understand**
  - Human relationship/connectivity
- **Model**
- **Exploit**

*Challenge:*

- Use social models to aid
- Efficient information management, trust and collaboration
Social networking

Inter-group

• *hierarchical*
  • The message packets move from group to group, rather than node to node
  • *Hierarchical data movement*
  • *Worst case – Logarithmic*

Intra-group

• *Constant number of hops*

Delayed

Middleware

- Mask disconnections, delays
- Provide uniform view of the system

Legend: PI- ID, basic user and device information, CI- Content Index, SI – Service Index, RI – Reputation Index
Middleware services

- Route packets
- Perform services
- Disseminate/acquire and find information
- Identify malicious nodes
- Find fastest path
- Find multiple paths
- Respond to queries

Challenge:
- Trust?
- Quality?
- Reliable?

Middleware architecture for opportunistic networks

- Modules are optional
- Indices:
  - PI: Personal Index: Name, ID, address
  - CI: Content Index: Sharable files
  - SI: Service Index
  - DI: Device Index: Communication cost, intercontact time
  - RI: Reputation Index

11/29/2010
Kumar and Cao
Content distribution and management

Lack of distinction between producers, consumers, and forwarders

Content generated anywhere anytime
- Share, transmit
- Time to live and Hops to live limits
- Security, privacy and trust

Limited buffer/cache space
- How to acquire? What to store? Where to store? What to purge?

Effective cache management strategies
- Social group based
- Application based
- Consistency

Query processing and management
- Multiple queries
- Scalability
- Spatial and temporal consistency

Information Caching

Store and forward data
- Default
- Temporary data
- What to purge? and what to store?

Acquisition
- In house applications
- Social group applications
- Priorities

Dissemination
- Generated within node or social group

Caching
- Optimal management of limited cache space
- Data consistency
- Local cache and group cache
Resource sharing

Application on PDA needs a video stream from camera

- No direct link to camera
- Use cell phone as a forwarder
  - Bluetooth connection between camera and cell phone
  - iPaQ PDA receives video stream and transmits processed stream to Blackberry
- Check authentication, process video stream

Query processing

Distributed Information/database
  Video streams
  Data Streams
  Fused data
  Trust/authentication data

Event and Context
Query Processing

Challenge:
User mobility and anonymity
Services and composition

A. Passarella, M. Kumar, M. Conti and E. Borgia, Minimum-Delay Service Provisioning in Opportunistic Networks, Accepted for publication, IEEE Transactions on Parallel and Distributed Systems.

References


B. Tamhane and M. Kumar, ACM Workshop on Mobile and Opportunistic Networks (MobiOpp), Pisa, Italy, February 22-23, 2010.


References


E. Yoneki, P. Hui and J. Crowcroft, Visualizing Community Detection in Opportunistic Networks, CHANTS’97, Sep. 07, Montreal, Canada.

Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, Hao-Hua Chu, "A Hybrid Routing Approach for Opportunistic Networks", SIGCOMM’06 Workshops, September 11-15 2006, Pisa, Italy.


Peng Yang, Mooi Choo Chuah, "Context Aware Multicast Routing Scheme for Disruption Tolerant Networks", PE-WASUN’06, October 6, 2006, Torremolinos, Malaga Spain, pp. 66-73.

A. Passarella, M. Kumar, M. Conti and E. Borgia, Minimum-Delay Service Provisioning in Opportunistic Networks, Accepted for publication, IEEE Transactions on Parallel and Distributed Systems.

