Institute for Software Integrated Systems

Vanderbilt University

Service-Oriented Architectures for Networked Embedded Sensor Systems

Xenofon Koutsoukos

Manish Kushwaha, Isaac Amundson, Sandeep Neema, Janos Sztipanovits

Motivation: Chemical Cloud Tracking

Outline

- Challenges of programming wireless sensor networks
- OASiS: A service-oriented architecture
- Programming framework
- Programming model
  - Object-centric
  - Ambient-aware
- Integration of resource-constrained SNs with Web services
- Middleware
- Experimental results
- Research challenges

Programming Challenges

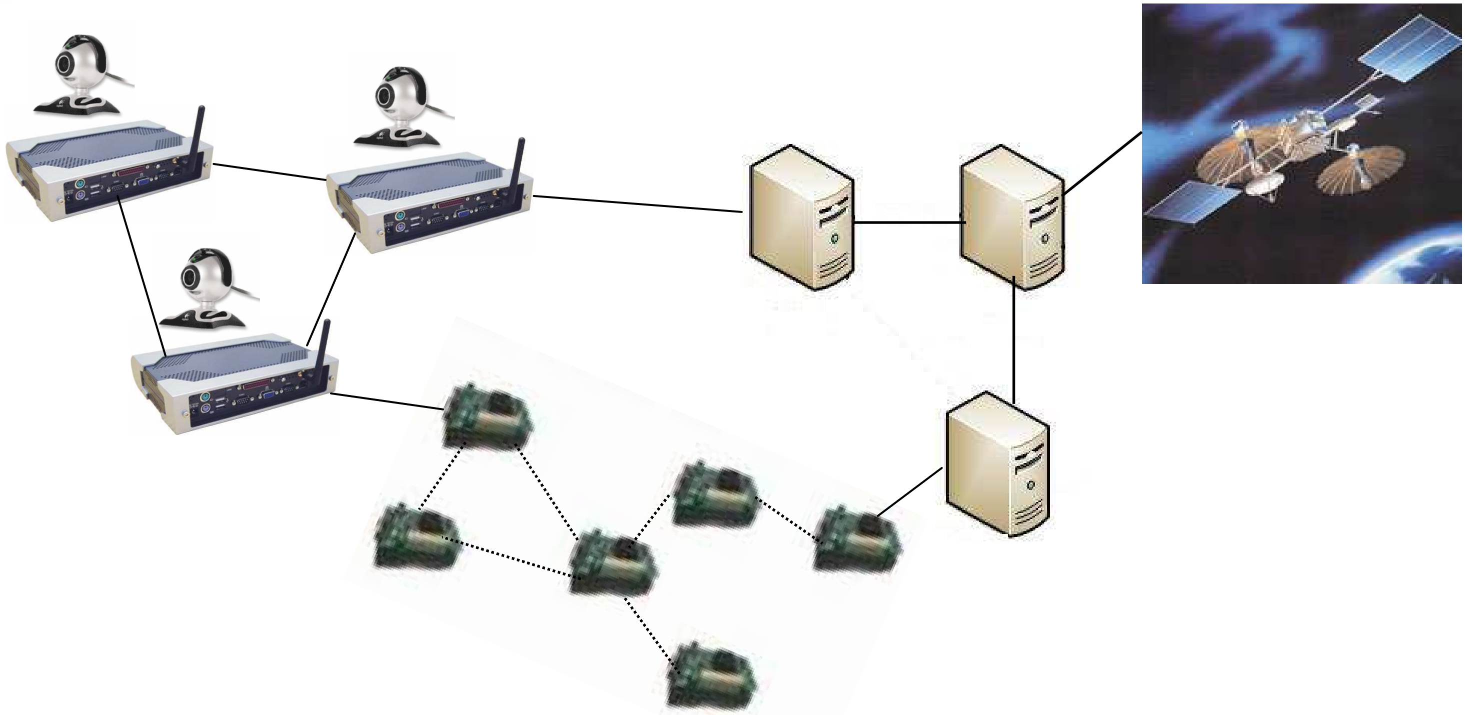
- Limited resources
  - Memory, bandwidth, and power supply
- Dynamic network behavior
  - Communication failure, node dropout, mobility
- Scalability
  - 10s of nodes to 1000s of nodes
- Heterogeneity
  - Sensor nodes, satellite imaging systems, meteorological stations, PDAs, security cameras
  - Rapidly evolving hardware architectures

How do we program a system composed of a large number of heterogeneous, volatile, resource-constrained devices?

Programming Models and Abstractions

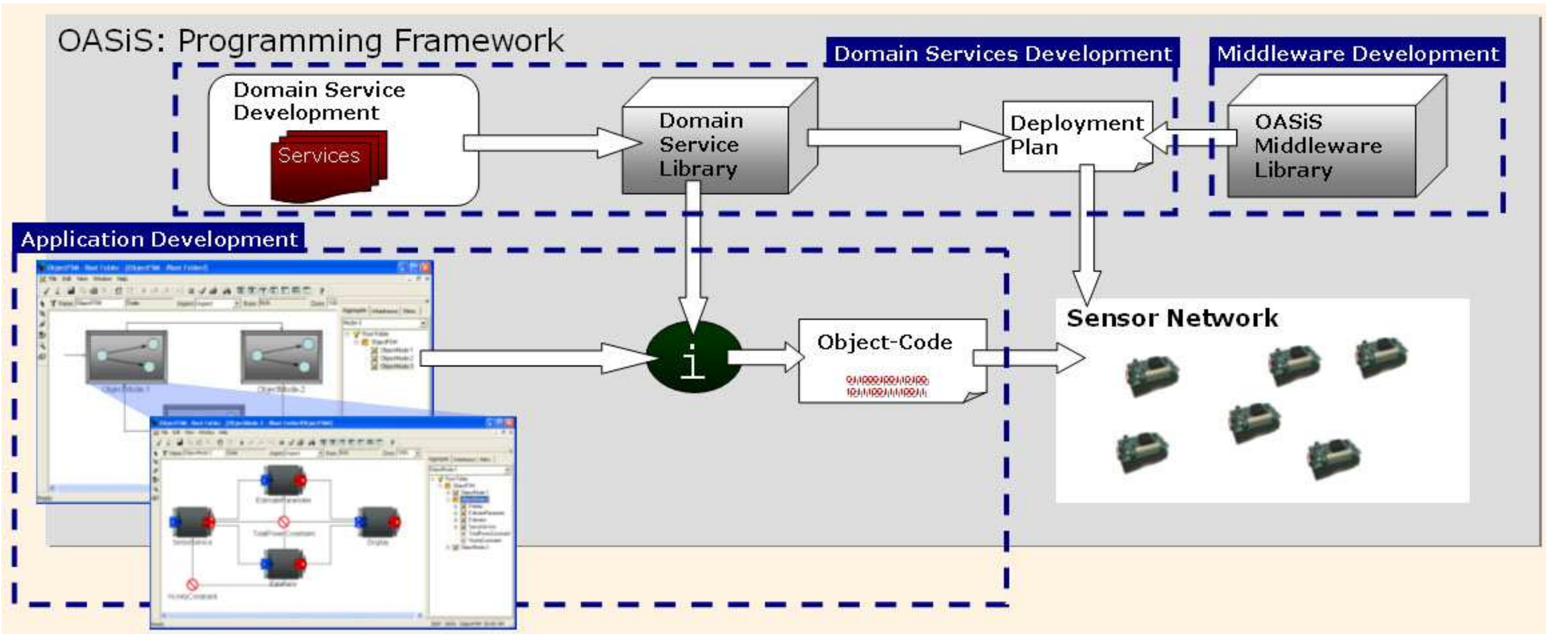
- Virtual machine
  - Mate, Magnet
- Mobile agents
  - Impala, Agilla
- Database
  - TinyDB, Cougar, SINA, DsWare
- Macroprogramming
  - Kairos, Abstract Region, Abstract Task Graph
- Service-oriented
  - SONGS, CodeBlue, Milan
- Object-centric
  - EnviroTrack

Integrating Heterogeneous SNs



- Service-oriented architecture
  - Loosely-coupled modular, and autonomous services
  - Well-defined interfaces
  - Published, discovered, and invoked over the network

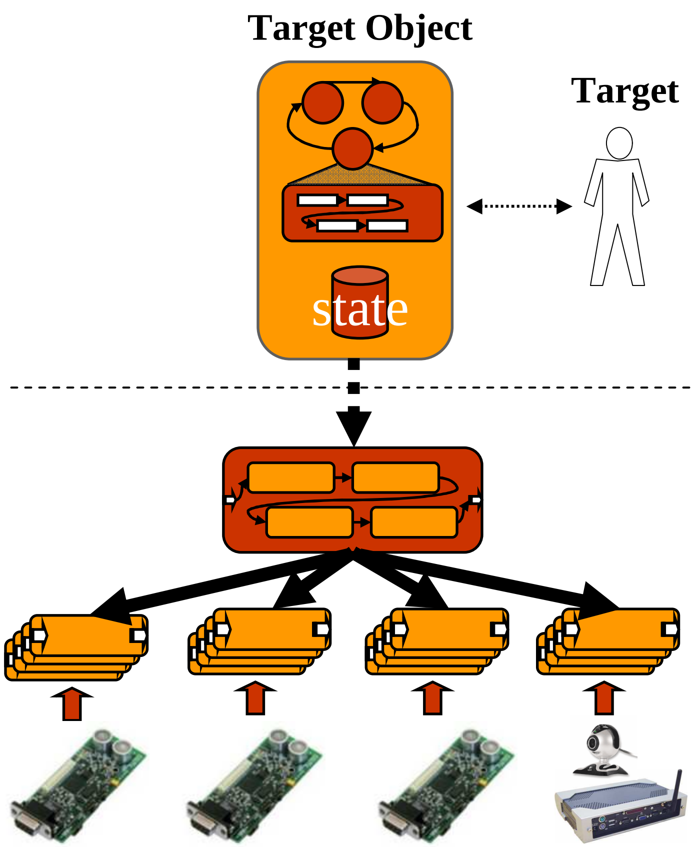
Programming Framework



- Separation of concerns
  - Application domain services
  - Middleware services

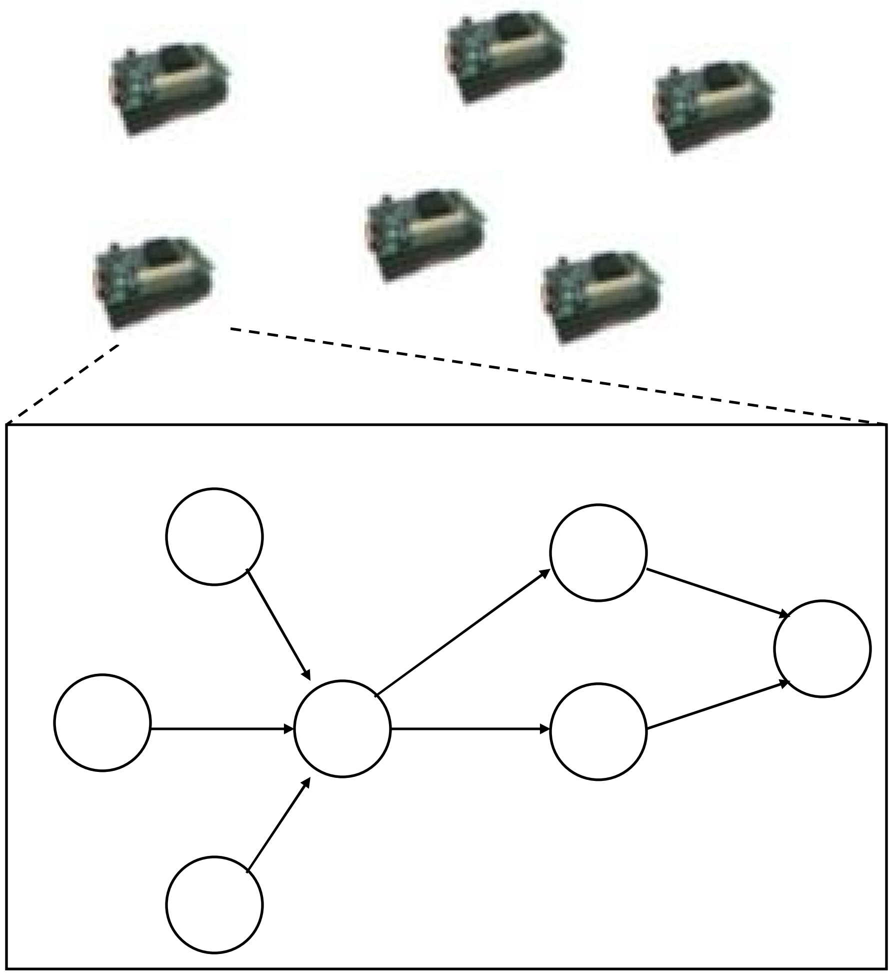
OASiS: A Service-oriented Architecture

- SN applications that can be described by dataflow such as tracking, gesture recognition, etc.
- Each activity is implemented as a service
- Applications are described as service graphs
- Object-centric programming
- Ambient-awareness
- Globally asynchronous locally synchronous (GALS) model of computation



OASiS at Run-time

- Sensor network monitors environment for target
- When target is detected:
  - Elect object owner
  - Locate services in network
  - Execute application

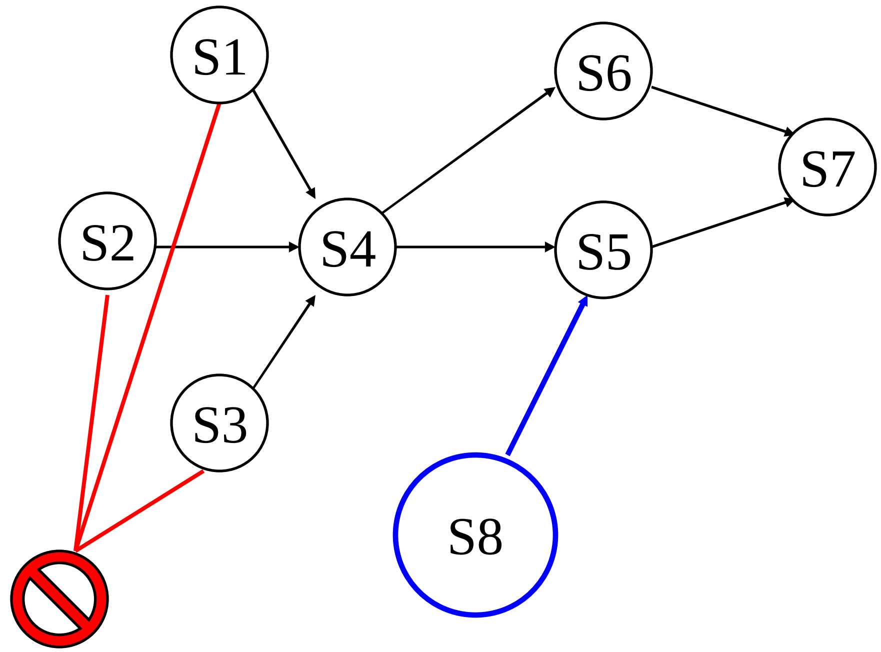


Object-Centric Programming

- Application development from the view point of the phenomenon under observation
- Application is driven by the phenomenon
- Phenomenon is represented as a unique logical object
- Benefits
  - Focus on the object
  - No global network model
  - Scalability
  - Heterogeneity

Service-Oriented Architecture

- Services
  - Modular, autonomous, with well-defined interfaces
  - Published, discovered, and invoked over the network
- Service graph
  - Application functionality described as dataflow
- Service constraints
  - Determine the allocation of services to nodes
- Globally Asynchronous, Locally Synchronous (GALS) model of computation

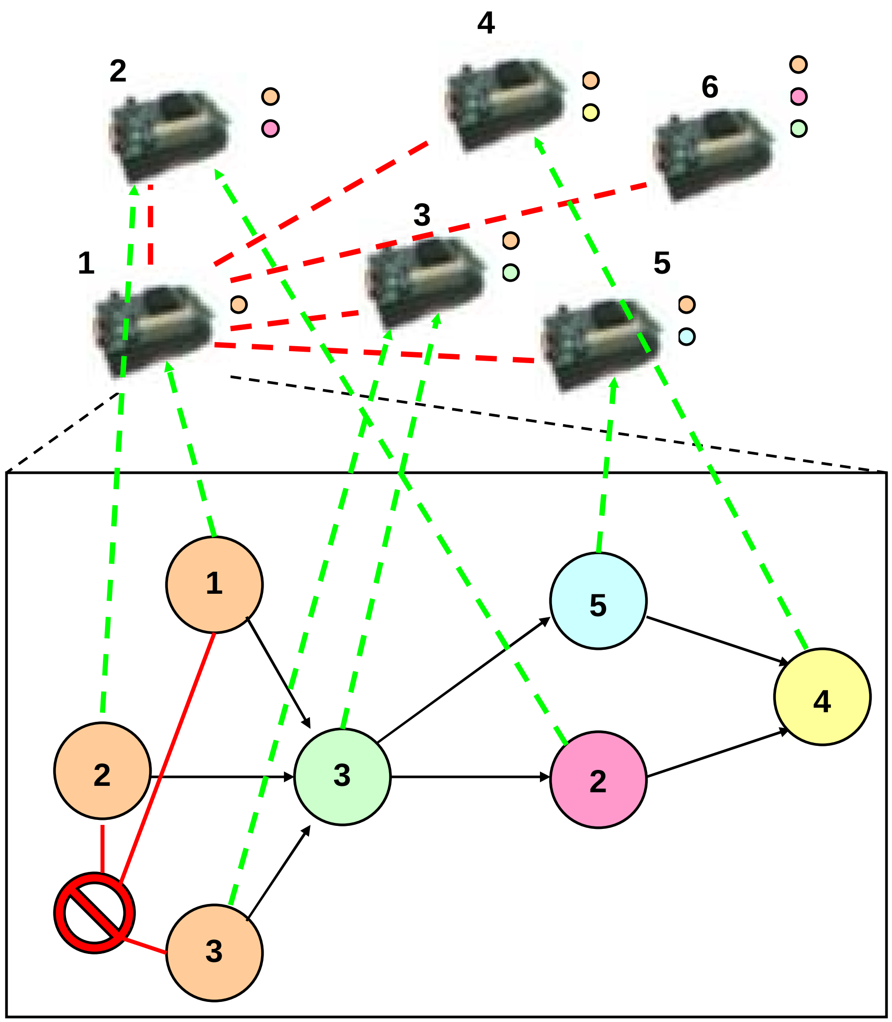


Ambient-Awareness

- Dynamic network behavior
  - Communication failure
  - Node dropout
  - Mobility
- Uncertainty of the physical phenomenon monitored
- Dynamic service discovery
  - New service provider must be selected quickly and efficiently
  - Must satisfy constraints specified in the service graph
- An application capable of adapting to the environment in such a manner is said to be **ambient-aware**

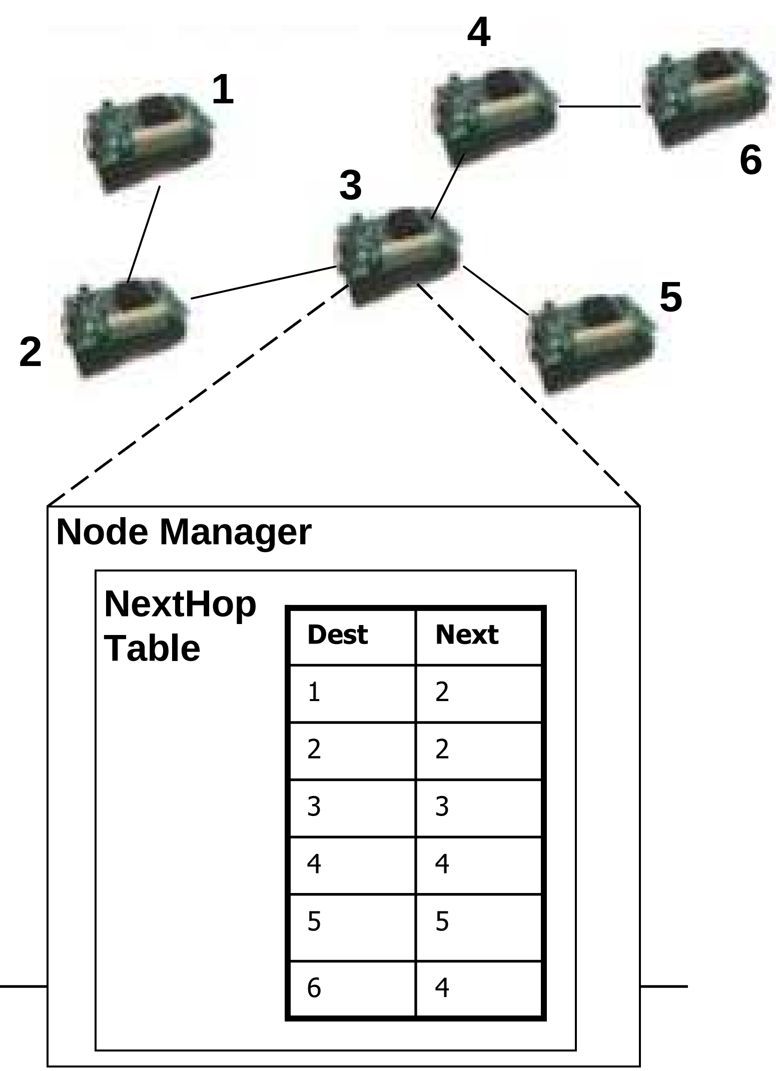
Dynamic Service Configuration

- Service Discovery
  - Service graph cannot be executed until its constituent services are located in the network
  - Passive service discovery
- Composition
  - Constraint satisfaction
  - Binding



Multi-hop Routing

- Similar to Dynamic Source Routing (DSR)
  - Adjusted for the OASiS architecture
- Message types requiring routing information
  - Service discovery reply messages
  - Service binding messages
  - Service access messages
- Each node contains a NextHop table
  - Specifies the next node on the path to final destination
  - Table is filled dynamically by examining the headers of received messages
  - If table does not contain a requested destination
    - A next-hop guess is made
    - Based on which neighbor is closest to the physical location of destination node



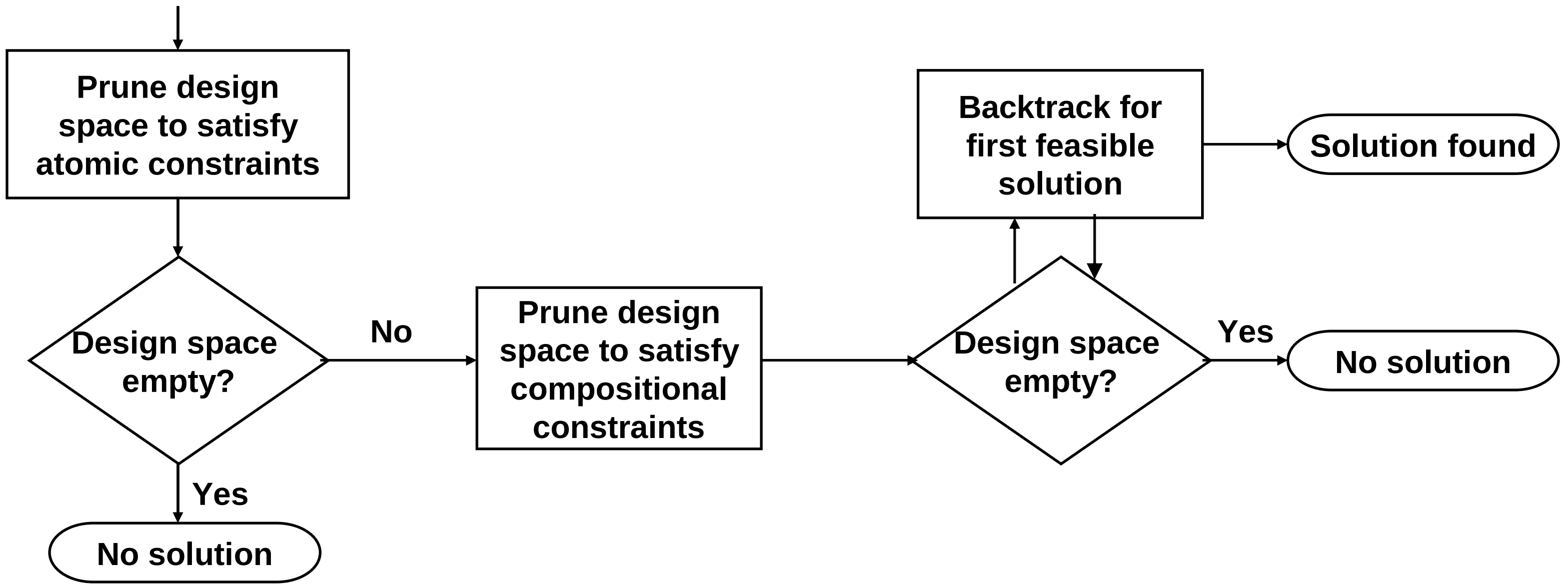
Dest	Next
1	2
2	2
3	3
4	4
5	5
6	4

Multi-hop Service Discovery

- Object Node floods network with a service request message up to MAX_HOP number of hops
- Nodes receiving a request
  - Forward message
  - Start timer
    - Timeout is inversely proportional to distance from Object Node
- When timer expires
  - Create service discovery reply message containing:
    - Provided services on present node
    - Services included in reply messages received from other nodes
  - Send service reply message to the Object Node

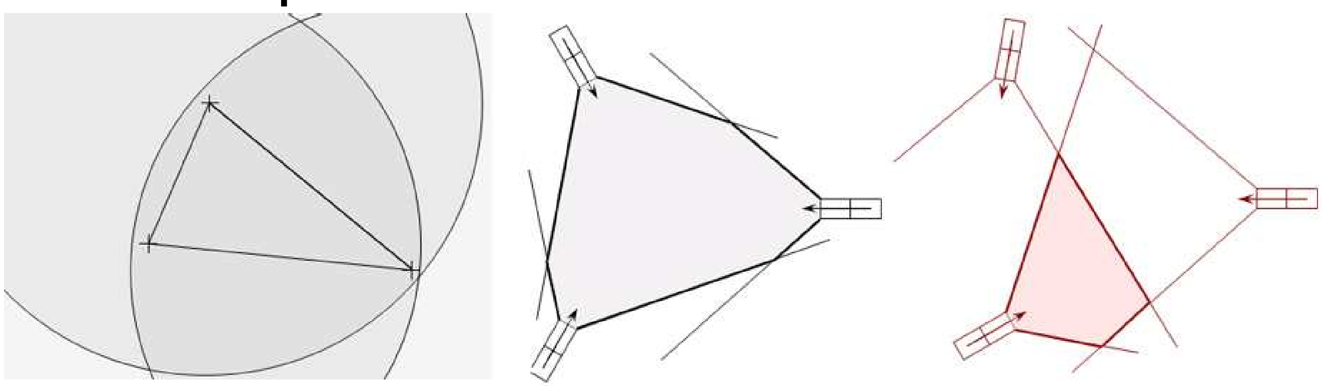
Service Graph Constraints

Constraint	Atomic	Compositional
Property	a.provider.z > 1	**average**(provider.power) > 85 **over** {a,b,c}
Resource	a.provider.id ≠ 143	**different**(provider.id) **over** {a,b}



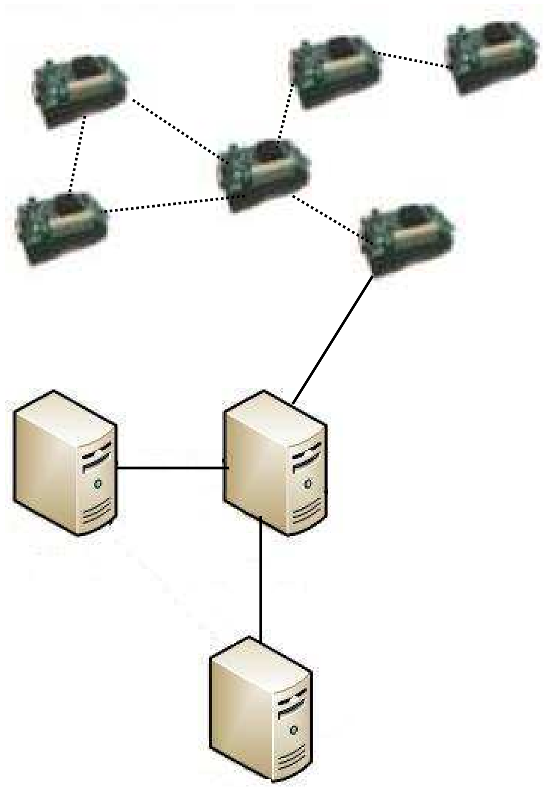
ENCLOSE Constraint

- Specifies spatial configuration of services
- ENCLOSE(L) over {a,b,c}
  - Location L must be surrounded by nodes providing service instances a, b, and c
- L is surrounded by {a,b,c} if there is no line in the plane that separates L from all of {a,b,c}
- Definition depends on domain



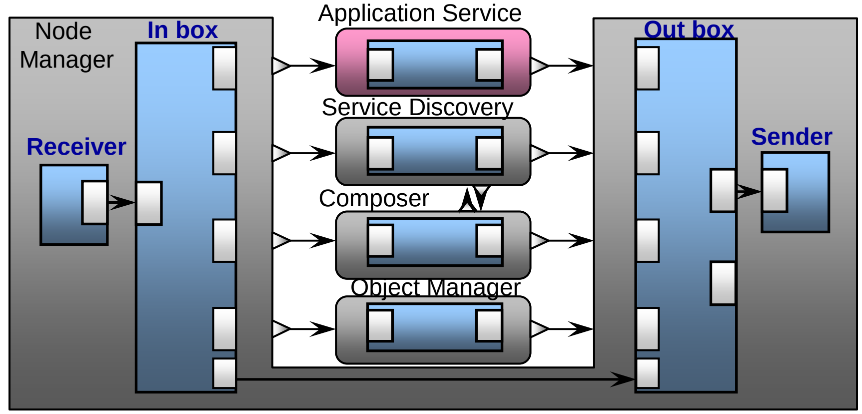
WWW Gateway

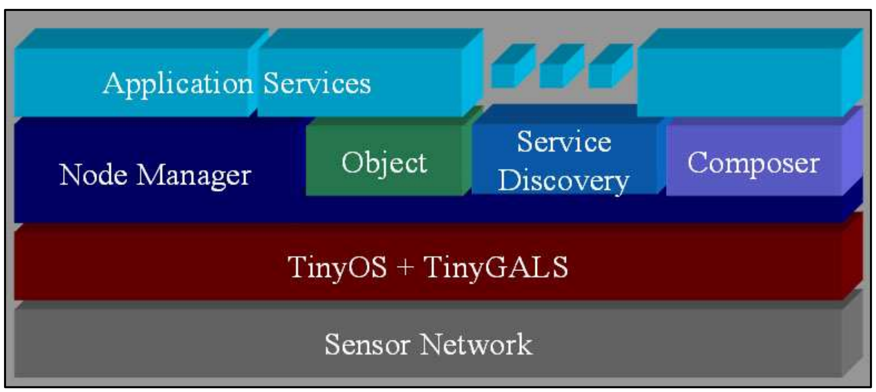
- Bridge between SN and Internet
- Translation of node-based byte sequence messages to XML-based messages
  - Web service discovery
    - UDDI
  - Service access and return messages
    - SOAP Envelope
- Transparent!
  - Similar to any other node in the network



Middleware Implementation

- Target platform
  - Mica2 motes
- Implementation in galsC
  - GALS extension of nesC
- WWW Gateway
  - Java
  - Apache WS tools

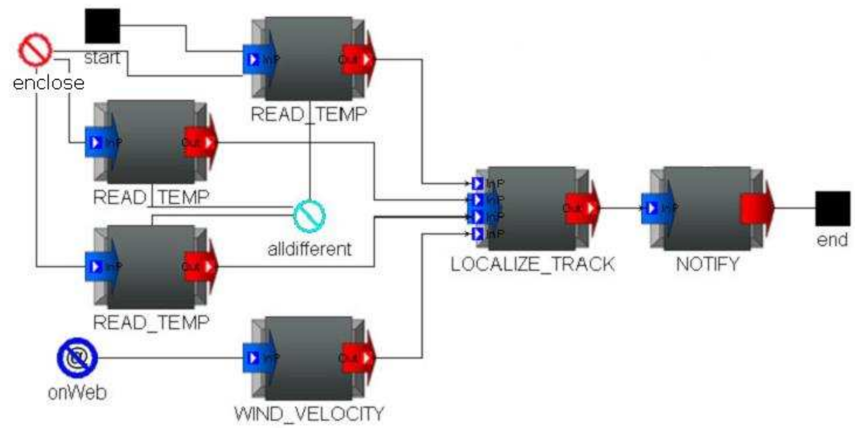




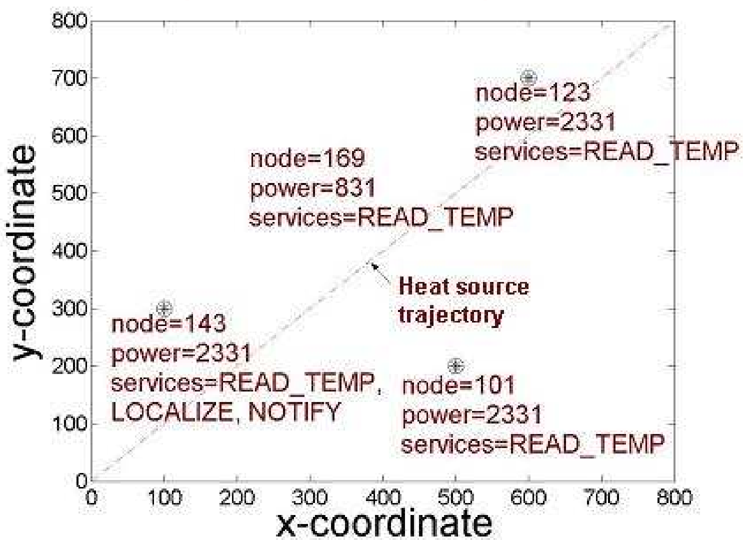
Service	Program Memory (bytes)	Required RAM (bytes)
Node Manager	8500	367
Service Discovery	3858	313
Composer	8036	509
Object	3560	151
GALS queues and ports	702	1013
Total	40248	2820
Available	64000	4000

Case Study

Motivation: Chemical cloud tracking



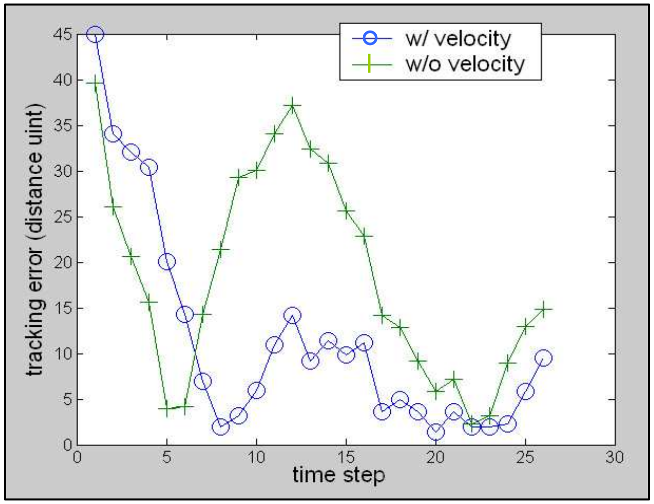
Simplified indoor experiment

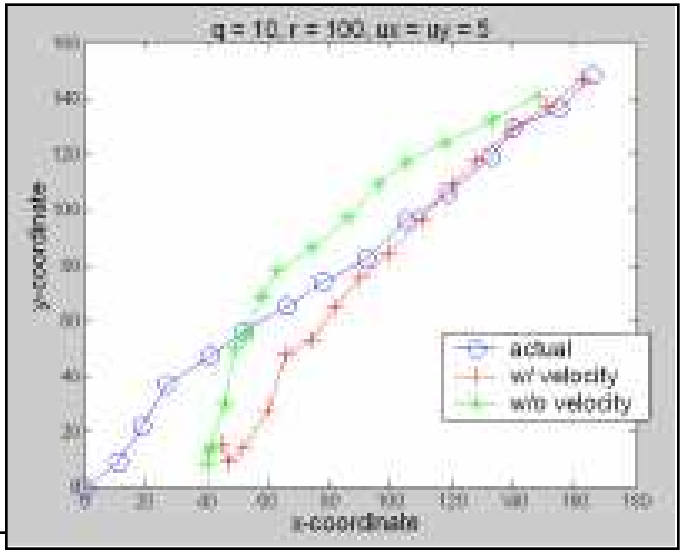


- Tracking of moving heat source
  - Temperature sensors
  - Hypothetical wind velocity (Web service)
  - Kalman filter

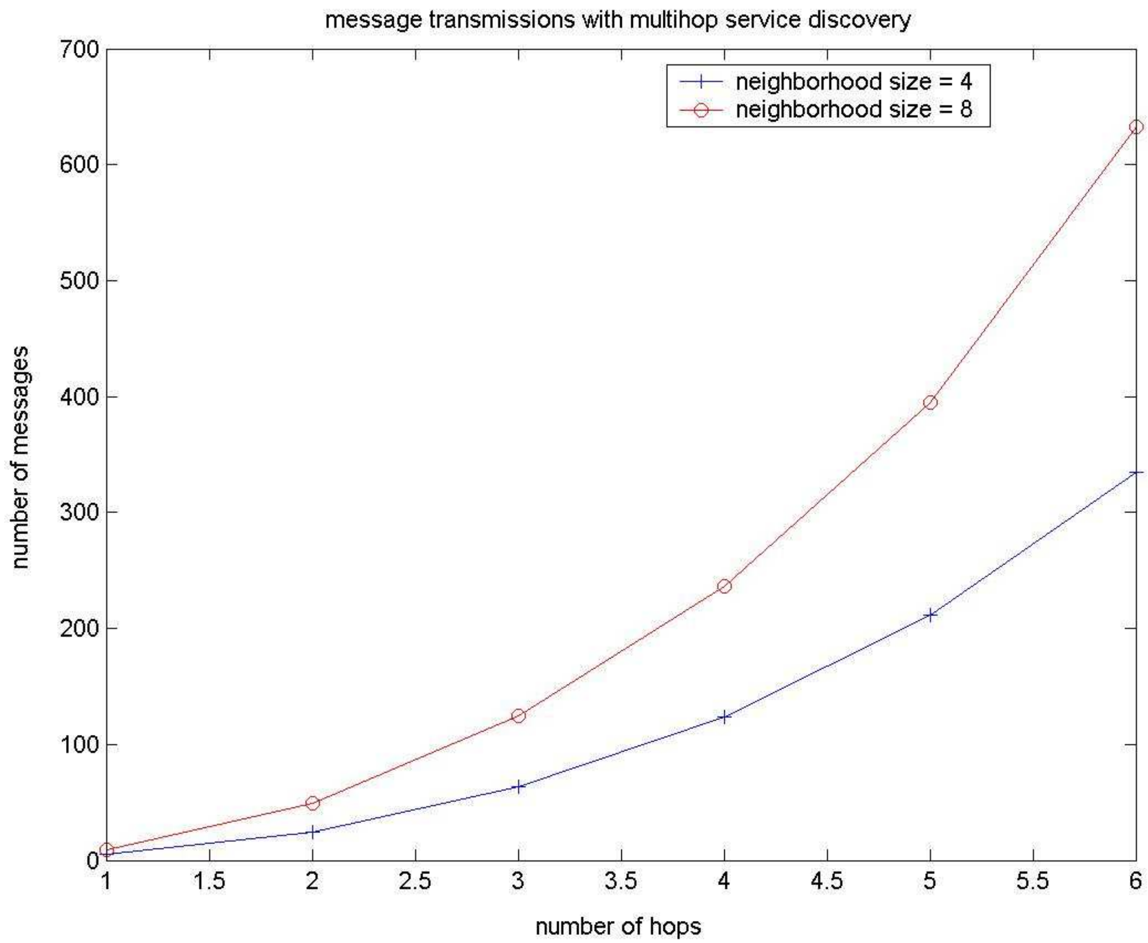
Case Study: Results

Operation	Response Time (ms)	Number of messages (worst case)
Service Discovery (with Web service)	4092	Service graph size X Neighborhood size
Service Discovery (no Web service)	1400	Service graph size X Neighborhood size
Constraint Satisfaction	15	0
Service graph execution (no Web service)	81	Service graph size
Web service access	502	2
Localization Service	11	5

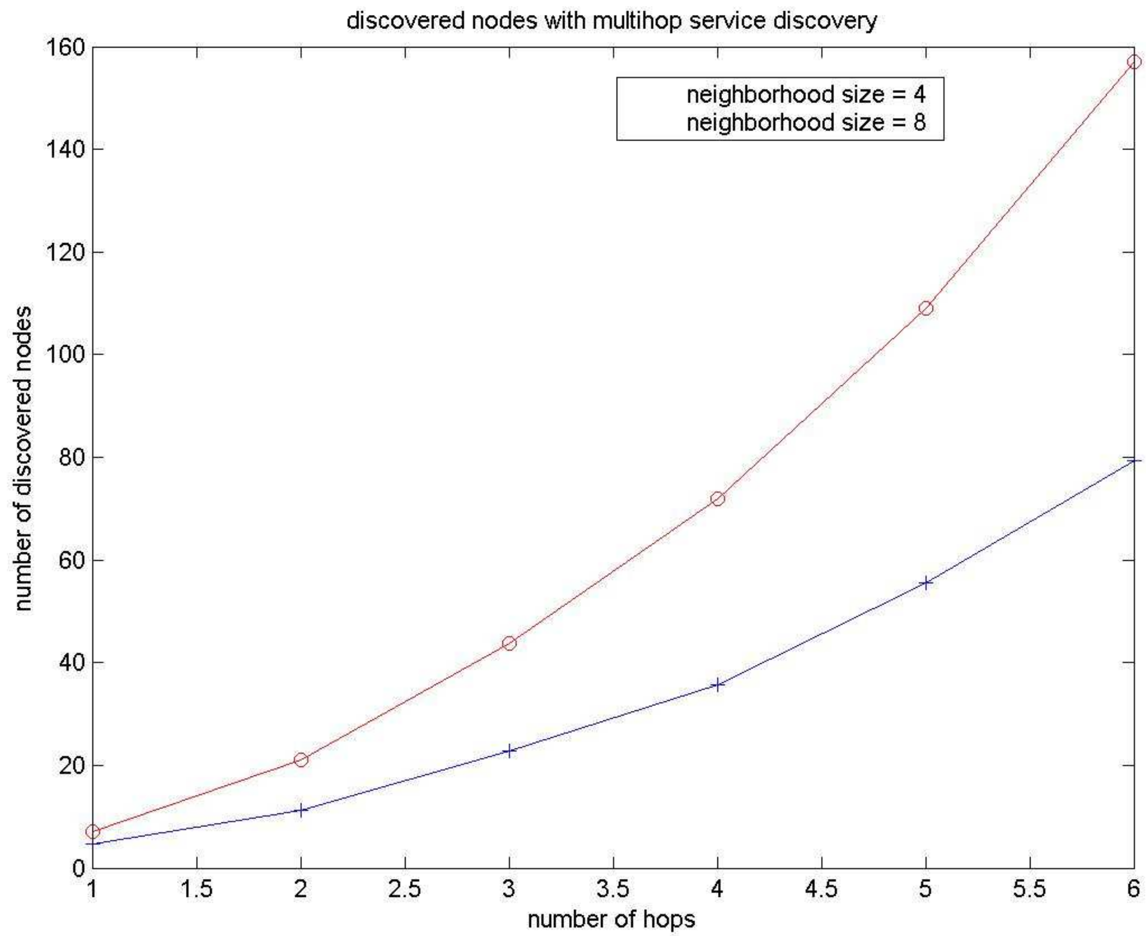




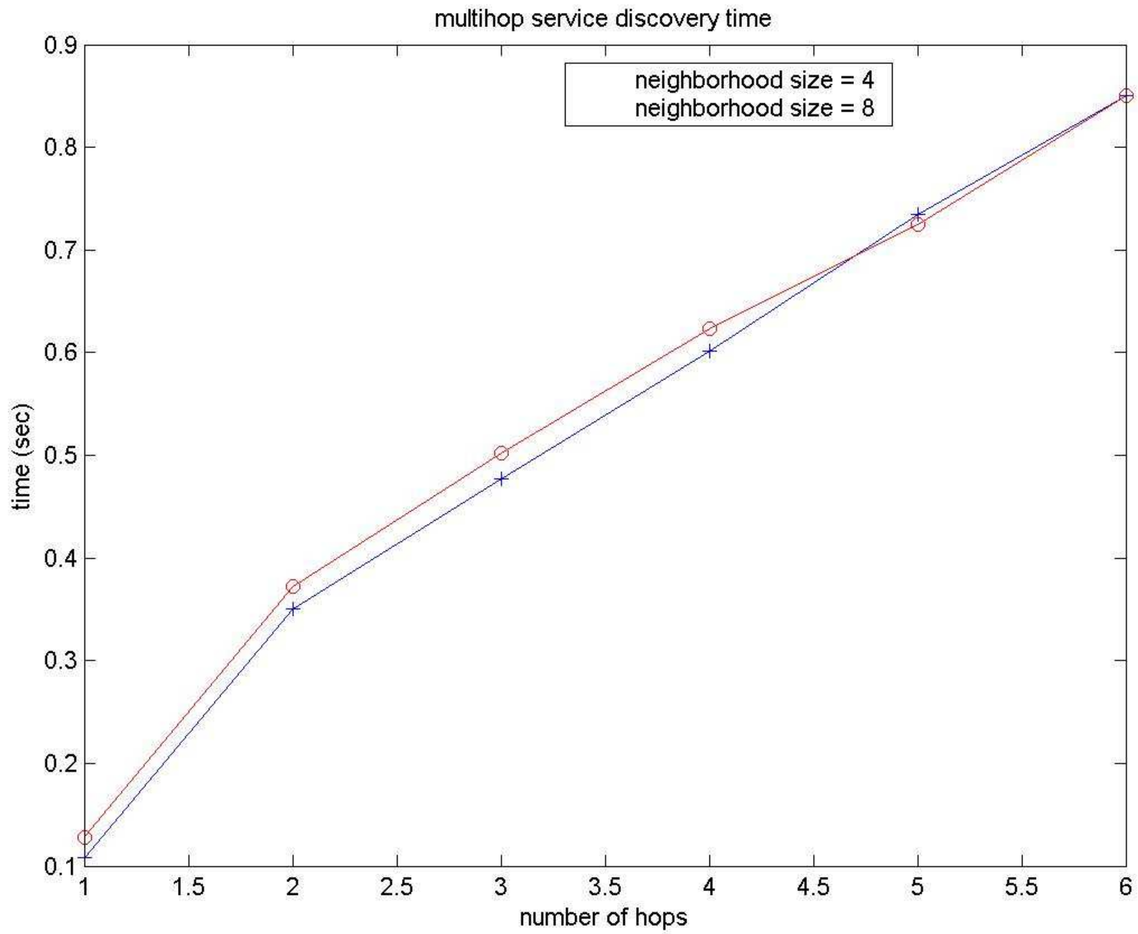
Scalability (1)



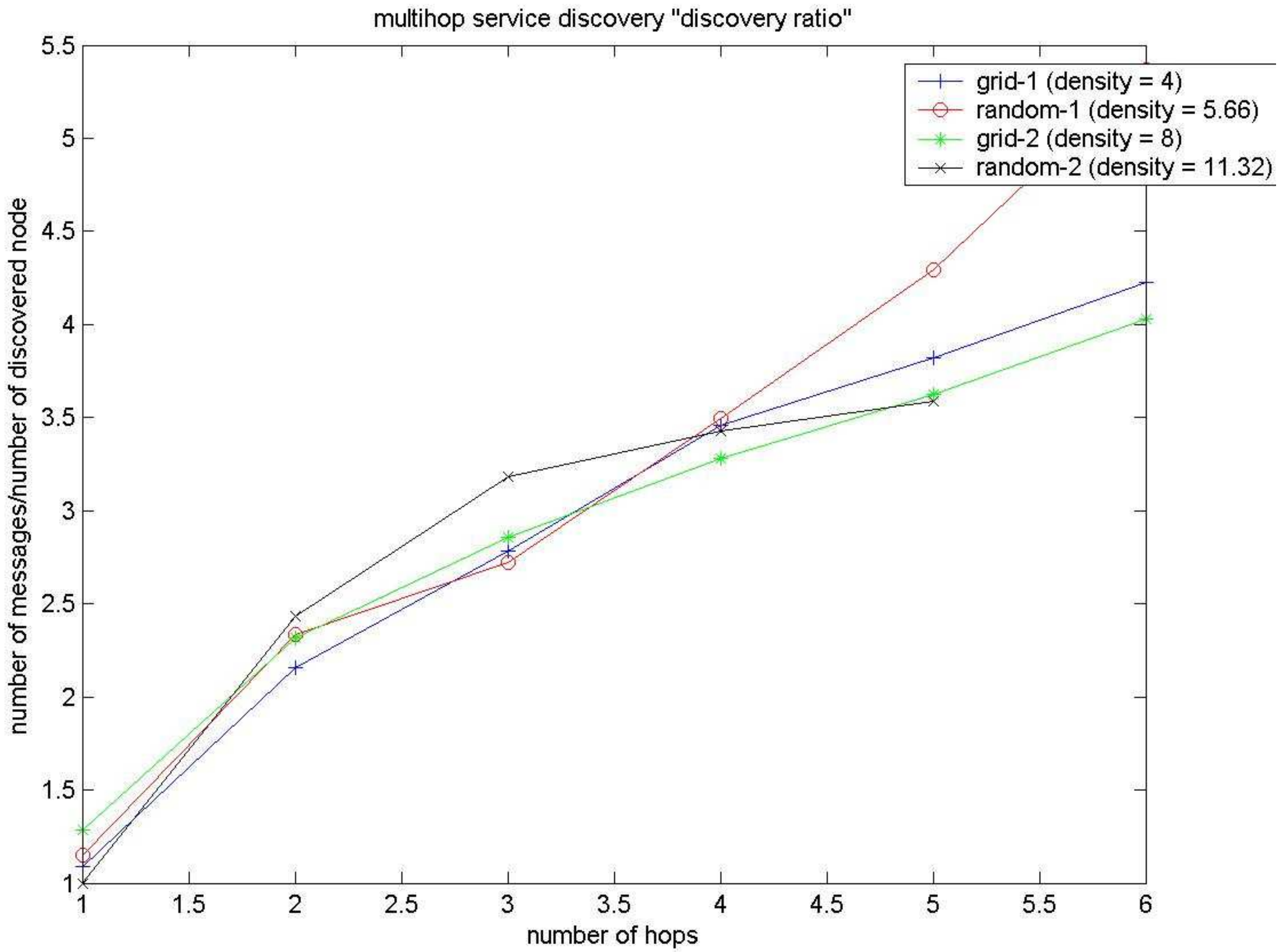
Scalability (2)



Scalability (3)



Scalability (4)



Concluding Remarks

- Service-oriented architectures are feasible in resource-constrained SNs
- Integration of SN applications with Web services can enrich the available functionality
- Ambient-awareness based on dynamic service discovery
- Real-world integration by considering spatio-temporal constraints
- Scalability by focusing on a network neighborhood of the object node

Current & Future Work

- Failure detection/recovery
- Multiple objects
- Multiple applications
- Mobility

OASiS Web Site

- www.isis.vanderbilt.edu/Projects/OASiS
  - Papers
  - Presentations
  - Downloads
- Acknowledgements
  - Microsoft External Research
  - NSF