TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Chair of Computer Networks

A SURVEY OF APPLICATION DISTRIBUTION IN WIRELESS SENSOR NETWORKS

Presenter: Thomas Gärtner
Supervisor: Dr.-Ing. habil. Waltenegus Dargie

Dresden, 2010/6/23

About the Paper

Paper published in 2004 by Mauri Kuorilehto, Marko Hännikäinen and Timo D. Hämäläinen.
Revised in 2005.

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

Motivation
Overview of WSNs

Wireless Sensor Networks ...

- are probably ad hoc
- usually don’t have pre-defined physical structure
- are data-centric
- have a task
- consist of nodes with limited resources

Motivation
Conclusion

- task of WSN is naturally distributed
- solved by collective effort of all nodes
- even more pressing issue if application is complex

Motivation

A Number of Problems

- How to create a working configuration so application works?
- How to avoid uneven use of resources?
- How to keep working if nodes fail?
- How to let application parts communicate?

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

Essential Distribution Aspects

Service Discovery

- importend in heterogenous networks
- different capabilities of nodes
- different tasks for nodes

Task Allocation

- determine the nodes on which (parts of) the application should be executed

Remote Task Communication

- means of wireless communication between distributed tasks

Task Migration

- methods for transferring a task executable between nodes

Outline

Systems Software for WSNs

Short Explanation

What is it?

- application-independent services
- node resource management

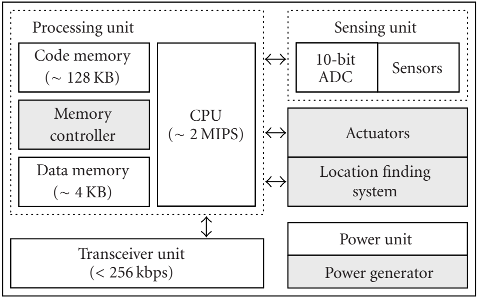
Why do we need it?

- paper focuses on implementation based on systems software
- hides complexity and heterogeneity of platforms

Systems Software for WSNs

Single Node Control

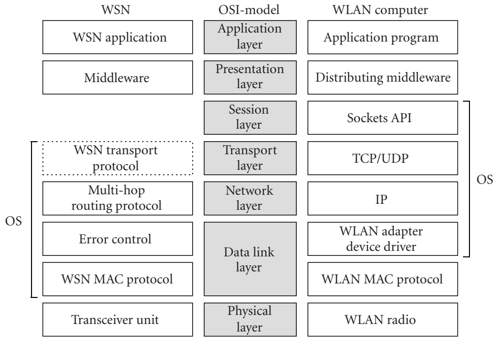
- general-purpose operating system (OS)
- virtual machine (VM)



Systems Software for WSNs

Network-Level Distribution Control

- usually dedicated protocol stack

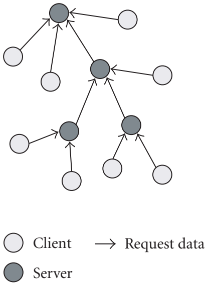


Outline

Architectural Paradigms

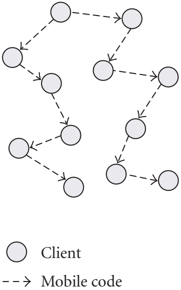
Client-Server

- well known from regular networks
- client accesses services on server
- used for service discovery and remote task communication



Architectural Paradigms
Mobile Code

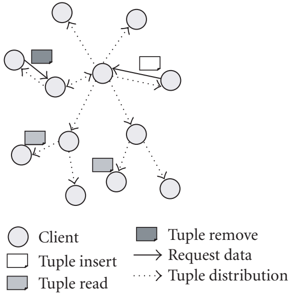
- move code to data origins
- then process locally
- mobile agents
  - carries own state and data
  - can make decision to move
  - usually implemented in VM



Architectural Paradigms

Tuple Space

- tuple = collection of passive data values
- tuple space = pool of shared memory
- insert tuple to announce data/service
- search tuple space for data/service
- use data/service by removing tuple



Outline

WSN Applications
Overview

- most applications gather, evaluate or aggregate data from different types of sensors
- differences in networking requirements and complexity
- soft borders between application types

WSN Applications

Types

Monitoring

- continually track a parameter value in a given location

Event Detection

- recognizes occurrences of events

Object Classification

- attempts to identify an object or its type

Object Tracking

- traces movements of an object

Outline

Proposal	Target network	Resource requirements (CPU/code memory/ data memory)	Service discovery	Task allocation	Remote task communi-cation	Task migration
			OS-based architectures			
EYES OS [37]	WSN	1 MHz / 60 KB / 2 KB	Resource requests	Not supported	RPC	Not supported
BTnodes [38]	WSN	8 MHz/ 128 KB/ 64 KB	Tuple space	Not supported	Callbacks	Smoblets
TinyOS [39]	WSN	8 MHz/ 128 KB/ 4 KB	Not supported	Not supported	Active messages	Not supported
BerthaOS [40]	WSN	22 MHz/ 32 KB/ 2,25 KB	Not supported	Not supported	BBS	Binary code
MOS [25]	WSN	8 MHz/ > 64 KB/ > 1 KB	Not supported	Not supported	Not supported	Binary code download
QNX [41]	LAN	33 MHz/ 100 KB/ N/A	Network manager	SMP scheduler	Message passing	Not supported
OSE [42]	LAN	N/A/ 100 KB/ N/A	Hunting service	Not supported	Phantom process	Not supported
			VM-based architectures			
Sensorware [17]	WSN	N/A/ 1 MB/ 128 KB	Not supported	Script population specification	Not supported	TCL script migration
MagnetOS [43]	WSN	N/A / N/A / N/A	Not supported	Automatic object placement	DVM [44]	Mobile Java objects
Maté [45]	WSN	8 MHz/ 128 KB/KB	Not supported	Not supported	Not supported	Code capsule update

Proposal	Target network	Service discovery	Task allocation	Remote task communication	Task migration
			Middleware architectures		
MiLAN [20]	WSN	SLP, Bluetooth SDP	Configuration adaptation	Not supported	Not supported
Cluster-based middleware [63]	WSN	Not supported	Resource management	Not supported	Not supported
QoS-aware middleware [64]	MANET	SLP/Jini/SDS	QoSProxy	Not supported	Not supported
SINA [24]	WSN	Not supported	Attribute matching in SEE	Not supported	SQTL scripts
TinyDB [69]	WSN	Not supported	Query optimizer, event-based queries	Not supported	Not supported
Cougar [66]	WSN	Not supported	Query optimizer	Not supported	Not supported
LIME [67]	MANET	Tuple space	Context reaction	Tuple space	Mobile Java objects
MARE [21]	MANET	Tuple space	MARE control	Tuple space	Mobile Java objects
RCSM [27]	MANET	RKS [68]	Adaptive object containers	R-ORB	Not supported
			Stand-alone protocols		
GSD [69]	MANET	Service groups	Not supported	Not supported	Not supported
Bluetooth SDP [31]	Bluetooth	Clients and servers	Not supported	Not supported	Not supported
Task migration in [70]	WSN	Not supported	Not supported	Not supported	Edit scripts

Outline

Testing and Validation of Proposals

Summary

- simulations for middleware
- prototype platforms for OS and VM
- results mainly moderate, poor or none at all
- direct performance comparison not feasible

Outline

Comparison of Technologies

Service Discovery

Technology	Communication	Scalability	Fault tolerance	Requirements	Benefits (pros)	Problems (cons)
Resource requests	Requests to neighbors	Restricted to neighbors	Broadcasted to all neighbors	Resource declaration	One-hop communication	Scalability
Tuple space	Tuple operations	Balancing between memory and scale	Redundant information	Memory pool in each node	Source and target independency	Communication/ memory load
Network manager	Name resolution requests to manager	Local manager area, but extensible	Possibly redundant network managers	Resource managers, register to manager	Scalability due to naming	Name resolution, communication load
Hunting service	Broadcast hunt service requests	Not restricted	Lost services can be rehunted	Remote service identification	Lightweight after initiation	First hunt latency and communication load
Bluetooth SDP	Peer-to-peer link	Only nearby nodes one at a time	Service information only in the host	Bluetooth protocol stack	Querying for available services	Scalability, no broadcast
RKS	Advertises for potential clients	Only to nearby clients	Advertisements when context and clients applicable	Context definitions for services	Advertisements	Scalability
GSD service groups	Service and group advertisements	n-hop diameter, but groups span wider	Redundant information	Service registration	Request routing based on group advertisements	Communication load (both advertisements and requests used)

Comparison of Technologies
Service Discovery

- most proposals based on client-server architecture
- don't scale well
- tuple space and GSD are communication intensive
- improve by announcing over more hops

Comparison of Technologies
Task Allocation

Technology	Approach	Scalability	Fault tolerance	Requirements	Benefits (pros)	Problems (cons)
SMP scheduler	Scheduling of tasks to free resources	Not restricted	Redundant	High-speed bus, shared memory	Efficiency and transparency	Inapplicable requirements
Script population specification	Specification in migrating scripts	Not restricted	Multiple copies in network	Control in application scripts	No control required	Expressivity of specification
Automatic object placement	Activating and moving objects near to source	Not restricted	Multiple agents available	Object placement algorithms	Reduced data communication	Complexity
Configuration adaptation	Mapping tasks to available resources	Not restricted	Changes active nodes adaptively	Feasibility analysis, state updates	Application QoS consideration	Control communication
Resource management	Heuristic algorithm balancing load [75]	Restricted to a cluster	Continuous allocation	Control messages	Network lifetime maximizing	Algorithm complexity
QoSProxy	Component and service adaptation for resources and application QoS	Network-wide in small networks	Adaptation according to conditions	Application QoS specification	QoS adaptation dynamically to available resources	Server required, complexity and communication
Attribute matching in SEE	Matching script attributes to node parameters locally	Not restricted	Multiple copies in network	Accurate attribute specifications	Local late binding	Restricted expressivity
Query optimizer	Optimizing query routing to network	Optimization in gateway node	Redundancy in queries	Disseminated query plans	Only required set of nodes activated	Networking load of query plans
Event-based queries	Initiate query on occurrence of event	Not restricted	Possibly several event detectors	Event identification capability	In-network reaction	Loading of event source node
Context reaction	Reactions on tuples and executed on matching context	Reaction restricted to a location	Redundancy in tuple space	Location identifying	Task executed only when its context is applicable	Scalability
MARE control	Nearby agents form an execution environment	Restricted to nearby agents	Possible redundancy	Agent managers controlling agents	Agent cooperation in complex tasks	Scalability
Adaptive object containers	ADC activates tasks in correct context	Not restricted	Possible redundancy	Context interface specifications	Only applicable tasks activated	Complex context specifications

Comparison of Technologies
Task Allocation

- more varied approches
- mostly middleware layer implementations
- application-QoS-based approach looks promising
- application needs to take care of allocation control
- better adaptability

Comparison of Technologies

Remote Task Communication

Technology	Communication implementation	Scalability	Fault tolerance	Requirements	Benefits (pros)	Problems (cons)
Active messages	Remote handler, data encapsulation	Not restricted	N/A	Awareness of remote handler	Mapping to TinyOS event model	Handler name in ASCII
BBS	Message posting to neighbor BBS	Restricted to neighbor nodes	Message posted to all neighbors	Neighbor posting enabled by sender	One-hop communication	Scalability, memory load
EYES OS RPC	N/A	Restricted to neighbors	N/A	N/A	One-hop communication	Scalability
Callbacks	Callback registered to a tuple	Restricted to nodes sharing tuple space	Callback registered only in one node	Shared tuple space between nodes	Callback fired only on an event	Fault tolerance
Message passing	Custom networking (QNet) operations	Not restricted	Possibility for redundant messages	Name resolution	Mapping to local IPC	Network naming overhead
Phantom process	Messages sent by link handler	Not restricted	Possible secure channels	Created channel for communication	Mapping to local IPC	Required handshaking, communication load
DVM	Invocation redirection	Not restricted	N/A	Compile time script modification	Seamless IPC between objects	Communication and processing load
Tuple space	Tuple operations	Not restricted	Redundant	Shared tuple space between nodes	Distributed in space and time	Communication/ memory load
R-ORB	Message-oriented communication	Requires nearby recipient	Activated when link available	Context sensing	Activated only in applicable context	Scalability

Comparison of Technologies

Remote Task Communication

- mostly RPC or RMI
- problem: client needs to know the server
- callbacks and tuple space have known drawbacks

Comparison of Technologies

Task Migration

Technology	Communication	Scalability	Fault tolerance	Requirements	Benefits (pros)	Problems (cons)
Binary code	Binary code after negotiation	Only to one neighbor at a time	Simple checksum	Initiated by the binary code itself	Runtime initiation	Scalability, bit errors, binary size
Binary code download	Binary code from workstation	No in-network initiation	No protection	User initiates downloads	Possibility to update OS components	Errors, binary size, user interaction
Smoblets	Java applet modules	Execution only in laptops/PDAs	Java interpreter protection	Efficient platforms	Complex processing outsourcing	Executed only in efficient nodes
TCL script migration	TCL scripts	The scale specified in scripts	TCL interpreter protection	Injected to network by a user	Dynamic migration, small size of scripts	Complex population specifications
Mobile Java objects	Objects on top of JVM	Not restricted	Interpreter protection	Event initiating mobilization	Scalability	Communication and processing load
Code capsule updates	Small capsules in one active message	Script populated to all nodes in network	Maté interpreter protection	Injected to network by a user	Small size of scripts	No controlled migration
SQTL scripts	Custom query scripts	The scale specified in scripts	SEE interpreter protection	Injected to network by a user	Small size of scripts	Communication cost in broadcast
Edit scripts	Scripts containing changes to old code	No in-network initiation	Erroneous/missing scripts requested from neighbors	Generation of edit scripts in workstation	Small size of scripts	Complexity, no in-network operation

Comparison of Technologies

Task Migration

- mobile agents most favored
- small and fault tolerant but have overhead for interpretation
- binary code more efficiant
- needs extra security measures

Outline

Recommendations and Conclusions

Recommendations

- some proposals are pretty good at some aspects
- a more complete solution requires integration of OS and middleware
- increase of mutual awareness and efficiency
- use task migration only if absolutely necessary
- group nodes to virtual clusters for decrease of memory and communication load

Recommendations and Conclusions
Conclusions

- no common benchmarks
- no detailed large-scaled experiments
- research focused on
  - single node problems
  - distinct aspects
- a more integrated solution is needed

Sources

1. Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. 2005. A survey of application distribution in wireless sensor networks. EURASIP J. Wirel. Commun. Netw. 2005, 5 (Oct. 2005), 774-788. DOI= `http://dx.doi.org/10.1155/WCN.2005.774`
2. Wikipedia on Tuple Space for clarification `http://en.wikipedia.org/wiki/Tuple_space`

Backup Slides

Applications

Application	Type	Requirements	Data amount and frequency	Scale and density
Great Duck Island [14]	Environmental monitoring	Data archiving, Internet access, long lifetime	Minimal, every 5–10 min, 2–4 h per day	32 nodes in 1 km^2
PODS in Hawaii [15]	Environmental monitoring	Digital images, energy-efficiency	Large data amounts, infrequently	30–50 nodes in 5 hectares
CORIE (Columbia River) [16]	Environmental monitoring	Base stations, lifetime	Moderate data amounts, infrequently	18 nodes in Columbia River
Peek value evaluation [17]	Environmental monitoring	Collaborative processing, minimal network traffic	Moderate data amounts, periodically	Case dependent
Flood detection [18]	Environmental monitoring	Current condition evaluation	50 bytes every 30 s	200 nodes 50 m apart
SSIM (artificial retina) [19]	Health	Image identification, realtime, complex processing	Large data amounts, frequently every 200 ms	100 sensors per retina
Human monitoring [20]	Health	Quality of data, security, alerts	Moderate data amounts, depend on the human stress level	Several nodes per human
Mountain rescue [21]	Health	Communication intensive	Large data amounts in high frequency	One per rescuer in mountain area

Application	Type	Requirements	Data amount and frequency	Scale and density
WINS for military [22]	Military	Target identification, realtime, security, quality of data	Large data amounts, infrequently	Several distant nodes
Object tracking [23]	Military	Collaborative processing, realtime, location-awareness	Large data amounts with high frequency near an object	7 (prototype) nodes in proximity
Vehicle tracking [24]	Military	Identification and coordination, realtime	Large data amounts every 8 s near an object	1024 nodes in 40 km^2
Intelligent input/output [25]	Home entertainment	Communication intensive	Large data amounts with high frequency	One node per input device
WINS condition monitoring [22]	Machinery monitoring	Data aggregation, machinery lifetime projection	Depend on machinery complexity and its current status	Few nodes per machinery
Smart kindergarten [26]	Education	Video streaming, identification, location-awareness	Large data amounts in variable frequencies	Tens of sensors, indoor
Smart classrooms [27]	Education	Context-sensing, data exchange	Large data amounts in random frequency	Several nodes in classroom

Backup Slides

Testing and Validation

Proposal	Test environment	Simulation and testing tools	Prototype platform	Result accuracy	Published results
			OS-based architectures		
TinyOS [39]	Prototype	TOSSIM [72]	Motes	Accurate	Component sizes, OS routine delays, computation costs
BerthaOS [40]	Prototype	None	Pushpin	None	Functionality mentioned
EYES OS [37]	None	None	None	None	None
MOS [25]	Prototype	PC emulator XMOS [25]	Nymph	Moderate	Memory and power consumption, test application performance results
BTnodes [38]	Prototype	None	Micro-size BTnodes	Moderate	Component sizes, energy consumption
			VM-based architectures		
Sensorware [17]	Prototype	SensorSim [73]	Linux IPAQ	Accurate	Framework size, execution delays, energy consumption
MagnetOS [43]	Windows/Linux JVM	Custom packet-level simulator	PC	None	Internal algorithm comparison in simulator
Maté [45]	Prototype	TOSSIM [72]	TinyOS mote	Accurate	Bytecode overhead, installation costs, code infection performance

Proposal	Test environment	Simulation and testing tools	Prototype platform	Result accuracy	Published results
			Middleware architectures		
MiLAN [20]	None	None	None	None	None
Cluster-based middleware in [63]	Algorithm simulation	Custom simulator	None	None	Heuristic resource allocation, algorithm performance
Qos-aware middleware in [64]	None	None	None	None	None
SINA [24]	Simulations	GloMoSim [74]	None	Poor	SINA networking overhead, application performance
TinyDB [65]	Simulations, prototype	Custom environment	TinyOS mote	Accurate	Query routing performance in simulations, sample accuracy and sampling frequency in prototypes
Cougar [66]	None	None	None	None	None
LIME [67]	JVM	None	PC	Poor	Approximations about Java code size
MARE [21]	JVM	None	PDA	Poor	Service discovery performance
RCSM [27]	Prototype	None	PDA with custom hardware	RCSM poor, RKS accurate	RCSM memory consumption, RKS size, communication, energy consumption
			Stand-alone protocols		
GSD [69]	Simulations	GloMoSim [74]	None	Poor	Influence of internal parameters on service discoverability
Task migration in [70]	PC	None	Tested in EYES nodes	Accurate	Algorithm performance, influence of internal parameters