Design, Implementation and Case Study of *WISEMAN*: WIreless Sensors Employing Mobile AgeNts

Sergio González-Valenzuela, Min Chen, and Victor C.M. Leung

Department of Electrical and Computer Engineering
The University of British Columbia
{sergiog,minchen,vleung}@ece.ubc.ca

The 2nd International ICST Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, Berlin, Germany , April 28-29 2009

Presentation Outline

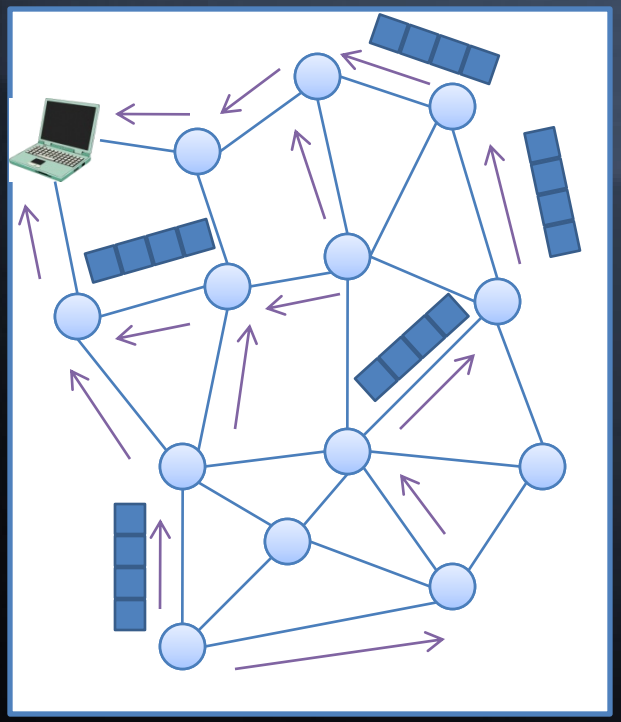
- Motivation: Why is this problem important?
- Rationale: Why use mobile codes in Wireless Sensor Networks?
- Our contributions:
  - A. Earlier system adaptation for WSN re-tasking
  - B. How our system differs from existing approaches
  - C. Best practices when programming with WISEMAN
- Performance evaluations & results
- Discussion & conclusions

What are Wireless Sensor Networks?

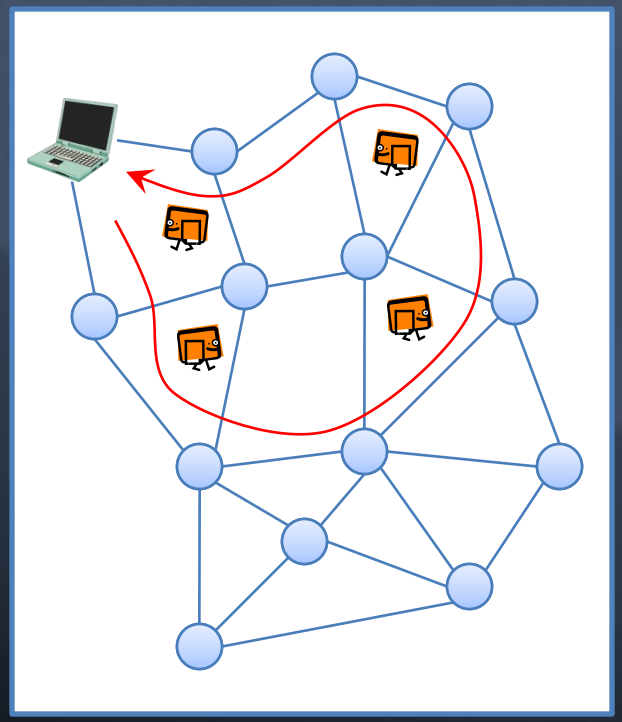
- Low-power devices – battery powered
- Limited memory
- Limited radio range
- Multi-hop paths to forward collected data
- Used for sensing temperature, movement, etc.
- Impractical to replace batteries too often
- As a result:
  - Power-efficient algorithms for data processing
  - Power-efficient protocols for data forwarding

Existing Approaches for collecting WSN data

Client-server, message passing, remote method invocation, etc.



Mobile agent/codes coordinate the distributed process in the WSN

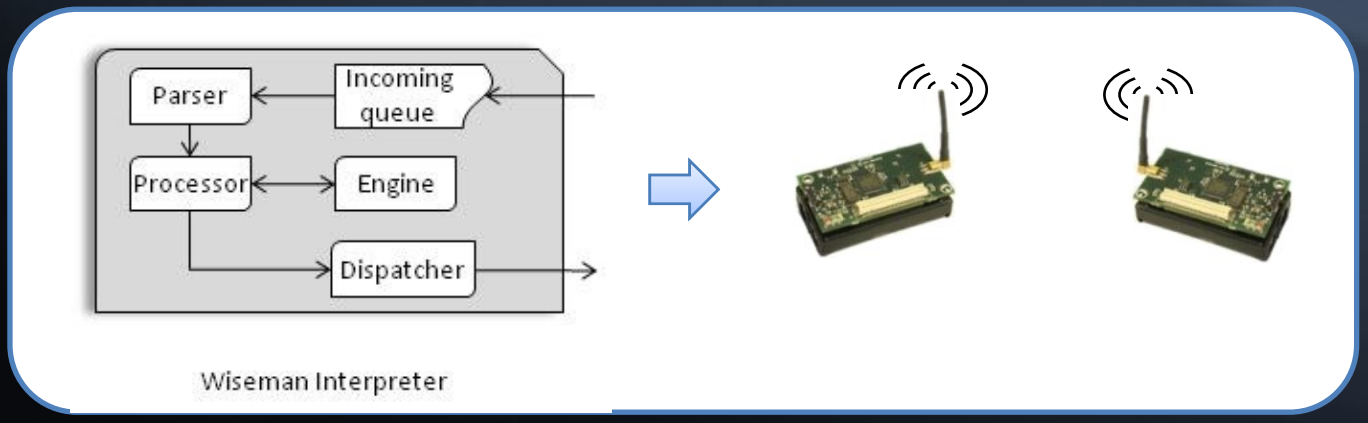


What advantages can we obtain from using agents in WSNs?

- Dynamic programmability of the WSN
- Compact (mobile) codes can save bandwidth
- Agent propagation delay can be shorter
- Reduced power consumption – longer-lived WSN

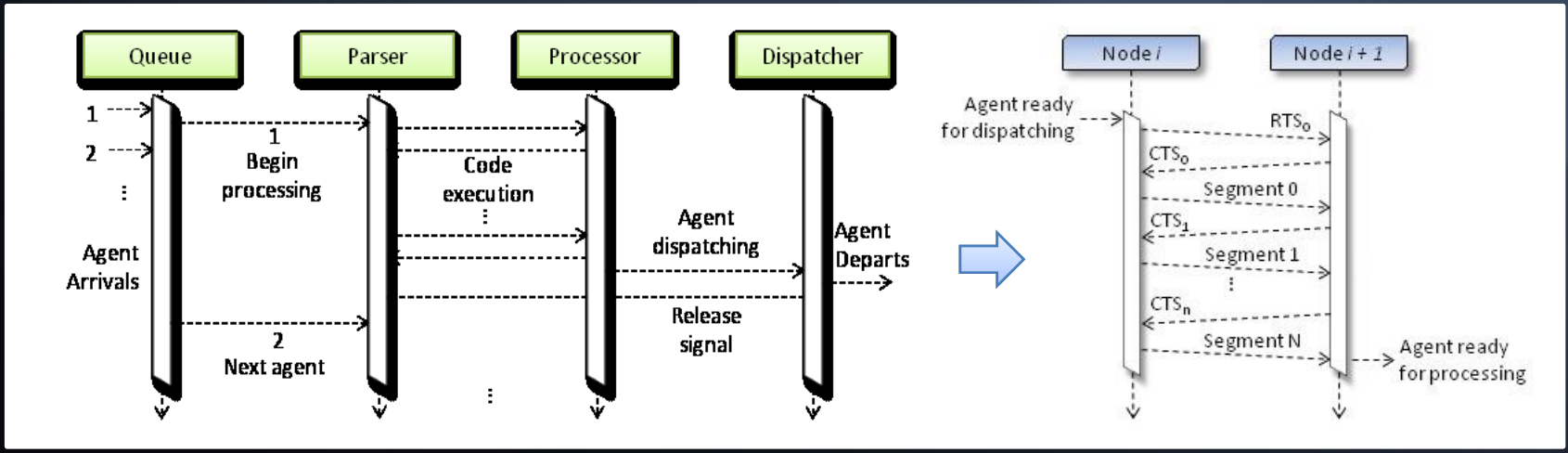
Enabling WSN Programmability with WISEMAN

- Based on an earlier system for active networking (Wave)
- Compact mobile scripts – interpreted codes
- Coarse-grained functionality aimed at distributed coordination
- Some fine-grained functionality as needed
- WISEMAN interpreters pre-loaded into nodes



Execution model and forwarding procedure

- Design considers limited resources (processing, memory, bandwidth).
- Only one agent can be executed at a time
- Simple forwarding algorithm between nodes



The WISEMAN instruction set: Operators, rules, variables & delimiters

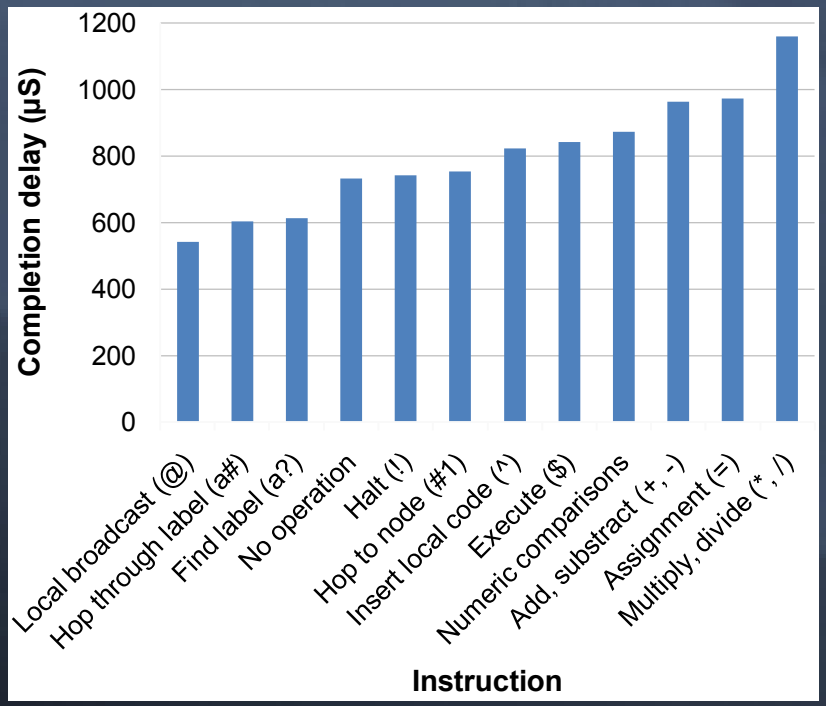
Lexeme	Name	Type	Description
N	Numeric	Variable	Local storage of numeric values
M	Mobile		A numeric value carried by an agent form
C	Character		A character value stored locally
B	Clipboard		Temporary storage of a numerical value
I	Identity	(environmental)	Holds the local ID node value
P	Predecessor	(environmental)	Holds the ID value of the node that an agent arrived from
L	Link	(environmental)	A character value used to label virtual links
O	Or	Rule	Yields true if *any* embraced commands is executed successfully
A	And		Yields true if *all* embraced commands are executed successfully
R	Repeat		Cycles through embraced commands
+ - * / =	Arithmetic	Operators	Used to perform regular arithmetic operations on variables
< <= == => > !=	Comparison		Standard operators to evaluate values and variables
_#_	Hop	Operator	Indicates that the agent will hop to another node or set of nodes
@	Broadcast		Broadcast agent to 1-hop neighbours
_$_	Execute		Performs local operation as indicated by parameters
!_	Halt		Stops execution with success or fail outcome as indicated
_^_	Insert		Inserts locally-stored agent into the
_?	Label query		Tests whether a labelled path exists in local node
{…} […] (…) ;	Semicolon	Delimiters	Used to separate individual expressions and rules

Features of WISEMAN's hardware implementation

- Used Crossbow Technology Micaz motes
- 3V, 2.4 Ghz, Zigbee
- Interpreter written in TinyOS v1.1
- Memory footprint = 19KBytes
- RAM usage ~ 3Kbytes
- Two queues: 3 incoming agents, 5 outgoing, 1 executing
- Max agent size = 170 bytes
- Bandwidth and delay evaluations

Raw performance evaluations – execution delay

- Reasonably fast
- Slower than Agilla
- Arithmetic operations take the longest to execute
- Hop operations take the the least to execute
- Average ~ 800µS



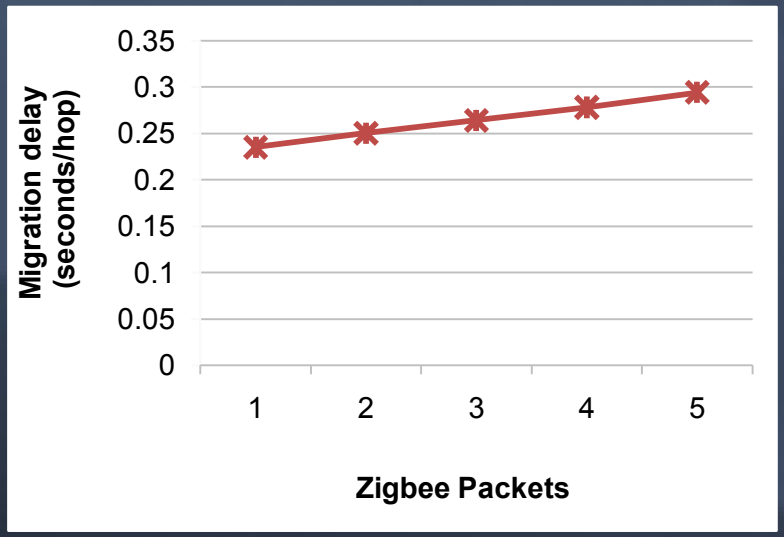
Raw performance evaluations – hopping delay

- (Includes processing delay)
- Different agent sizes
- Results averaged over 1000 hops
- Agent size =

  Session header (5 bytes) +

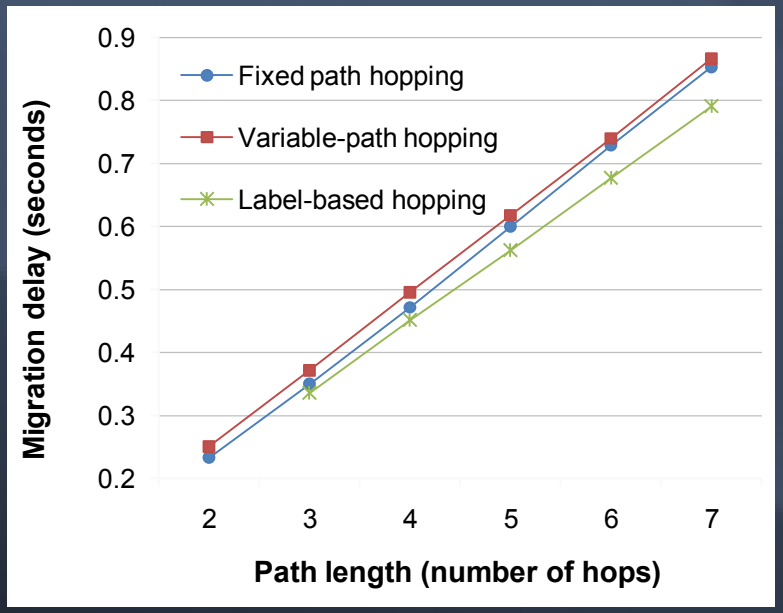
  Agent header (3 bytes min.) +

  Codes (170 bytes max).



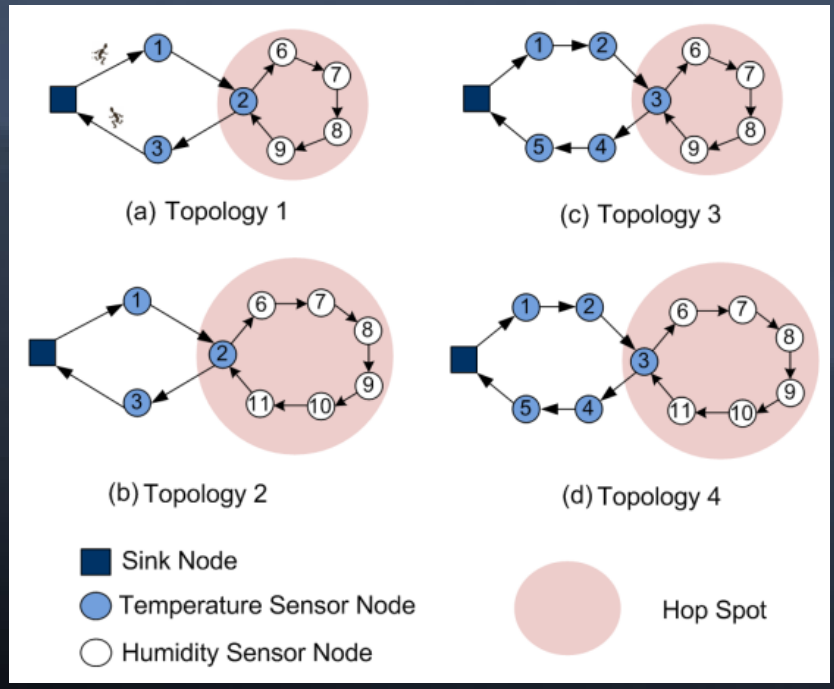
Raw performance evaluations – migration delay

- Evaluated three methods
- Fixed-path hopping better when hopping through short paths
- Variable-path hopping eventually beats fixed-path hopping
- Label-based always better, but depends on label maintenance overhead
- Combinations are possible



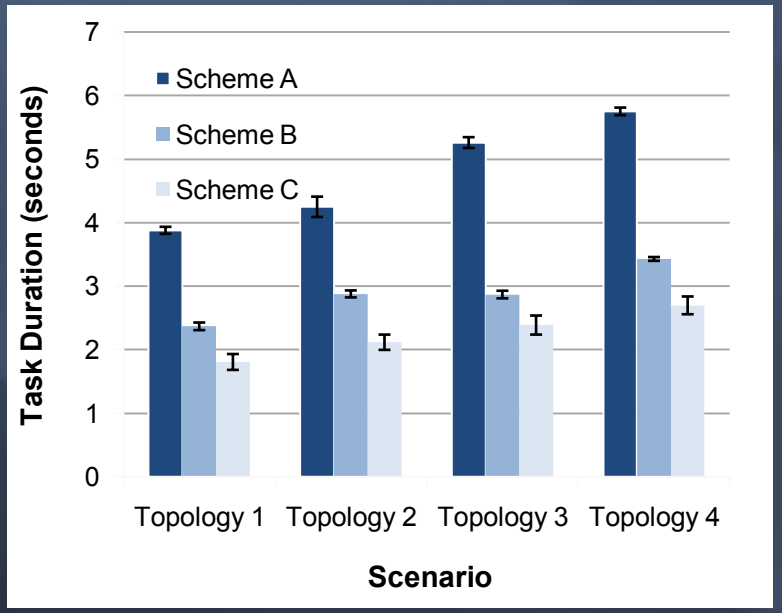
Case study: Fire-hazard monitoring in forests

- Evaluate a practical example of a WSN in a forest setting
- Good example to evaluate migration delay & bandwidth
- Evaluate three migration methods
- Metrics: delay & BW
- Four topologies
- Results averaged over multiple runs



Case study: outcome – delay

- Scheme A: Fixed-path hopping
- Scheme B: Variable-path hopping
- Scheme C: Label-based hopping



- Scheme A always takes the longest to complete
- Best performance obtained when sub-task agents are cached at the nodes (less BW)
- If not possible, variable-path is still a good second choice
- Best solution is application dependent

Case study: outcome – bandwidth

- If labelled paths requires little maintenance, then this is the best approach.
- Efficiency of the variable-target approach depends on the process that precedes the hopping decision
- Fixed path hopping is always the last resort

Scheme	Topology 1	Topology 2	Topology 3	Topology 4
A	161 [44(A1)+117(A2)]	187 [44(A1)+143(A2)]	209 [66(A1)+143(A2)]	235 [66(A1)+169(A2)]
B	135	165	165	195
C	99 [36(A1)+63(A2)]	117 [36(A1)+81(A2)]	126 [54(A1)+72(A2)]	144 [54(A1)+90(A2)]

Discussion & conclusions

- WSN programmability can be advantageous
- Best suited for WSNs deployed in settings where underlying environment's behaviour is unpredictable
- WISEMAN provided the necessary functionalities, but it can be expanded
- Advantages: text-based scripts – more flexible than Agilla
- Disadvantage: longer agent processing time than Agilla – processing text-based scripts is the culprit
- Porting to TinyOS v2 is planned