Emerging Themes of Computational Systems for Distributed Sensor Networks

S.S. Iyengar, Ph.D.

IEEE Fellow, ACM Fellow, AAAS Fellow

Roy Paul Daniels Professor and Chairman

Louisiana State University

Baton Rouge – LA, USA

iyengar@bit.csc.lsu.edu

http://bit.csc.lsu.edu/~iyengar

UBICOMM 2008,

September 29- Oct 4, 2008 Valencia, Spain.

Sponsors and Participates

This research is sponsored by the Defense Advance Research Projects Agency (DARPA), and administered by the Army Research Office under Emergent Surveillance Plexus MURI Award No. DAAD19-01-1-0504, NSF, ONR, DOE-ORNL.

Acknowledgments: Many Ph. D. students and the work which I am going to report is done jointly with Dr. Mengxia Zhu and Dr. R.R. Brooks

Outline

Part –I

- Novel Sensing and Information Gathering (DARPA)
- Distributed Sensor Networks
- Building High Performance Computational Systems

Part – II

- An Overview of Current Research Work

Part –III

- List of open problems and potential solutions

3

Acknowledgments

This presentation is a collection of work by various students and colleagues who worked with Prof. Iyengar and thanks to collaborators all over the world. Please do not reproduce this for the purpose of making money.

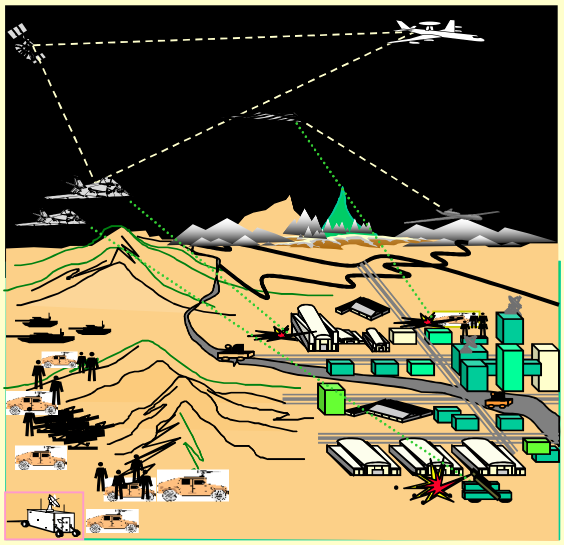
Introduction

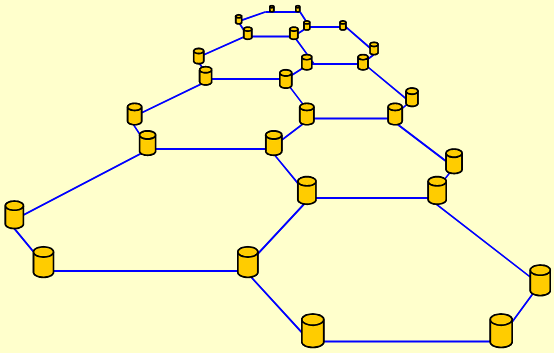
- "*Sensor Technology has rapidly moved beyond still and video cameras as other modalities have become available.*"
- *"Information analysis not only requires analytic methods be developed but for a variety of sensor types, but also relevant sensor data need to be gathered*
    - *requires development of new and novel techniques."*
- *"Good Sensing Modalities can provide critical piece of information for real time applications(Harbor Monitoring, Urban Surveillance)."*
- *"Commercial Sensors that provide a three dimensional information are now available but are not used commonly information gathering."*

5

Novel Sensing and Information Gathering (DARPA)

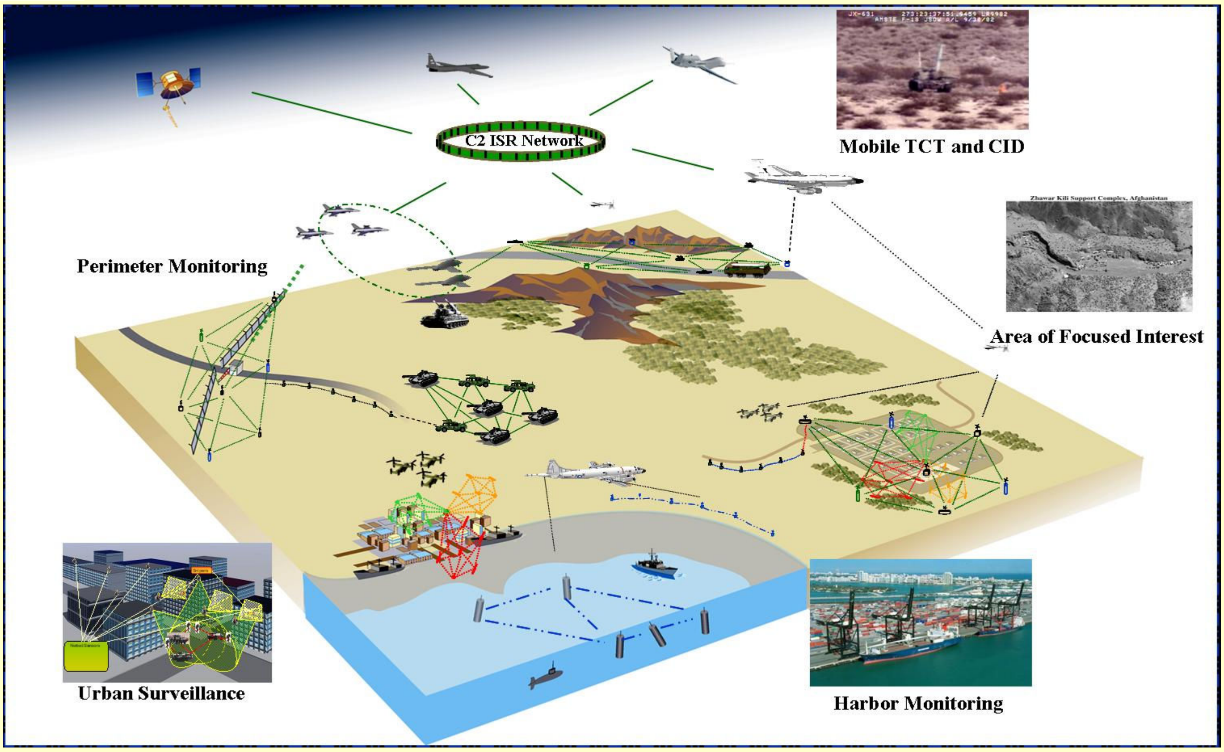
•Development of New and Novel Information Gathering about a complex scene is very critical for any applications.





Iyengar/Mengia,Brooks

Sponsors Applications



7

Courtesy of MITRE

How Can we Provide a Unified View of a Complex Scene

- *"Theory of motes/smart dust (Berkeley/Cross Bow) offers lot of promise for distributed sensing by multiple inexpensive data collectors."*

- *"Some of the most innovative ideas in this area remains concepts in science fiction levels."*

- *Need of the hour – Sensor based computational structure is necessary to meet the growing data intensive scientific needs for Information Gathering/Exploitation.*

- *Data Gathering, Management of data, query processing and visualization tasks (can scale linearly with data volumes).*

8

Sensor Applications

Jan. 17, 2006
“Volcano shoots ash 8 miles high in Alaska”

Habitat Monitoring on Great Duck Island

The College of the Atlantic in Bar Harbor, the University of California at Berkeley and Intel Research Lab

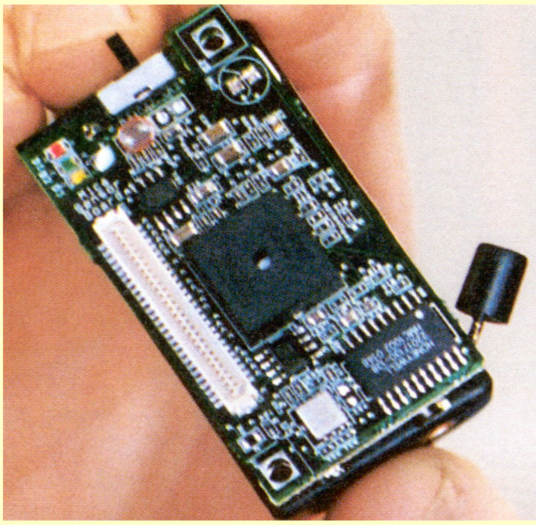
Observations

- *"The computational efforts of most statistical analysis of sensor data mining algorithms increase super linearly."*
- *"For example the pair algorithms on "n" point scale is $O(n^2)$."*
- *"If the data increases a 1000 fold, computational time can grow a factor by million."*
- *"Many sensor data clustering algorithms scale even worse and are infeasible for terabyte scale sensor data sets."*

Computer, January 2006, Gordon Bell and Jim Gray,and Alex Szalay , Page 110

Sensor Node Characteristics

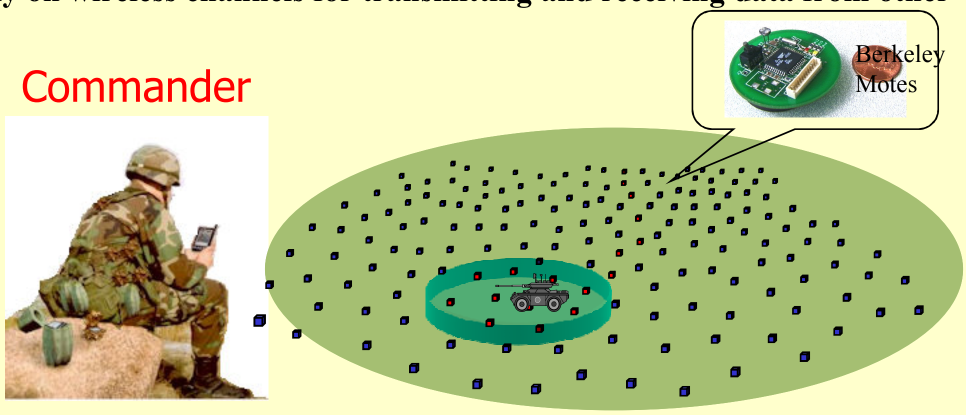
- Tiny
- Easily integrates into the environment
- Negligible cost
- Self contained in terms of energy
- Battery powered, equipped with integrated sensors, data processing capabilities, and short-range radio communications.
- The tiny nodes are equipped with substantial processing capabilities , enabling them to combine and compress their original data

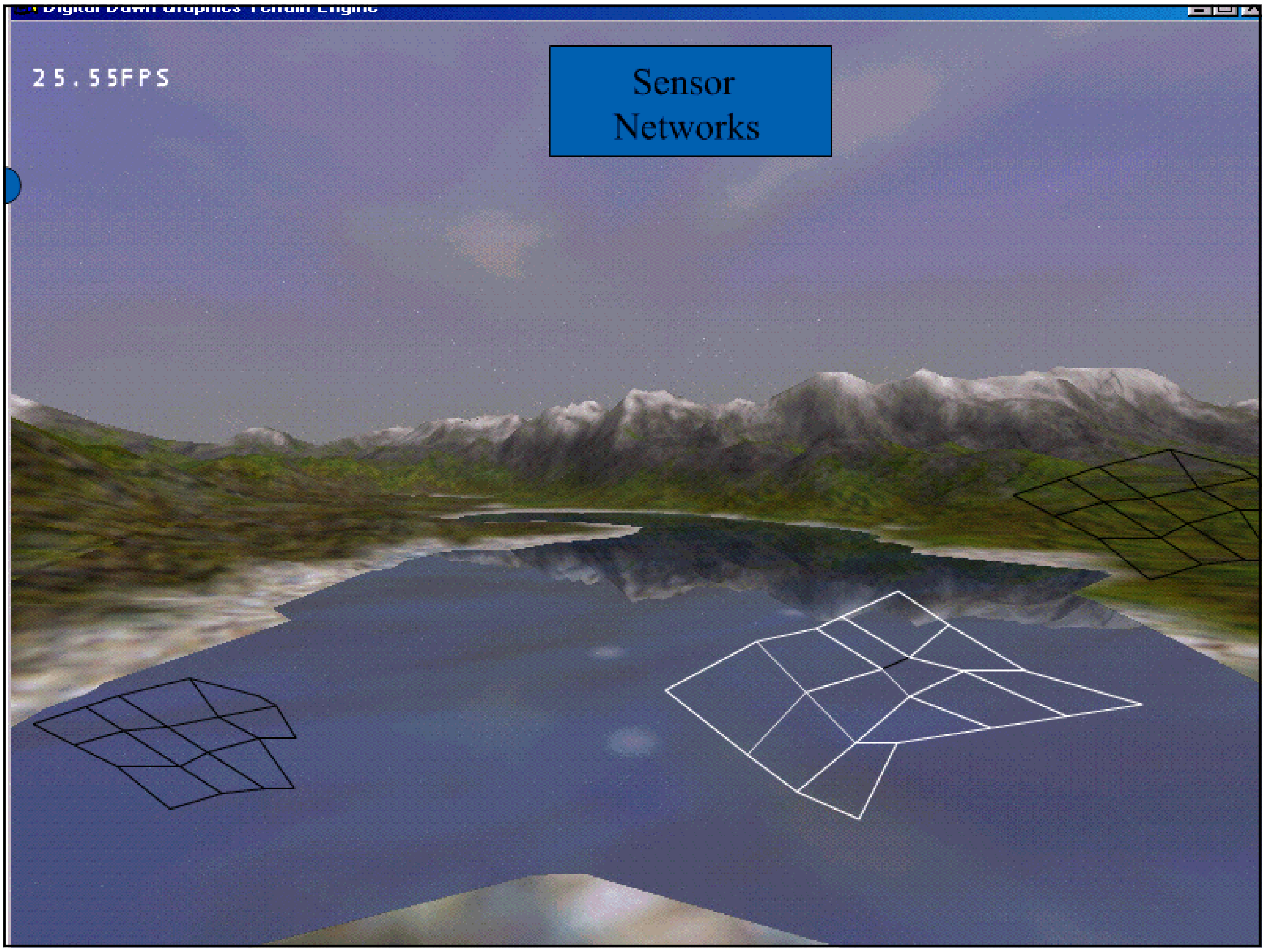


Adopted from MIT – Technology Review

What are Sensor Networks ?

- **Large Number of Sensor Nodes (thousands of small devices)**
- **Sensor nodes can be readily deployed in large number in various types of unstructured environments**
- **They rely on wireless channels for transmitting and receiving data from other nodes**




25.55FPS
Sensor
Networks

Challenges

- *"Realizing the motes are having very low power levels requires a vertical system – <u>level design</u> approach engaging all levels of design abstractions."*

- *"Unfortunately getting to the cost, size and power numbers needs for a truly ubiquitous deployment comes with penalty in reliability."*

- *"A more effective solution is to look at on the unique nature of these networks. i.e Ubiquitous availability of nodes."*

- *"Deployment, Configuration and management of the network is very critical."*

14

An unconventional system

- Principles
  - Distributed Data Aggregation
  - Information Context (looking only for special events)
  - Component Integration (actuators, cameras, TinyDB)
- Symptoms
  - Fragility
    - False alarm rate (false positives, false negatives)
  - Adaptability (context variations for mobile assets)
    - predictable responses leads to vulnerability
  - Scalability
    - Inherent performance limitations (e.g. modeling conflict between attacker, protector and commercialization barriers….)

Performance Sample Metrics

- Higher data quality
- Longer *network* life time
- Scalability
- Reliability of Distributed Sensor Network
  - Unlike star topologies the nodes inevitably have neighbors at different distances (ad-hoc)
  - Traditional shortest path algorithms may not work

16

Computing Paradigms in Sensor Networks

Sensor Networks presents a new class of computation in the following characteristics :

- Geographically distributed large number of tiny nodes (motes) in remote and largely in accessible areas
- These resource constrained tiny devices range from motes to PDA-Class computing systems
- These devices able to sense, compute, communicate and actuate in an unstructured and noisy environment with all the variations
- They involve energy constrained and resource limiting characteristics

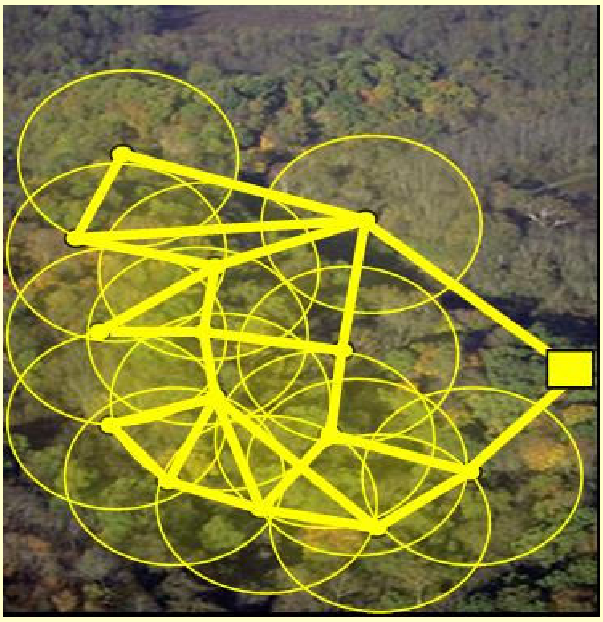
17

Cont’d

- They must be self organizing and self maintaining.
- They must be robust in the context of noise, and failures in these harsh environments.
- In short the emergence of this new computing structure class raises many significant Computational,Design and Networking challenges coupled close to the physical world.

Wireless Sensor Network

- **Infrastructured network**: with fixed and wired base station. A mobile unit communicate with the nearest base station within its transmission range. The base stations act as bridges for the network.
- **Ad Hoc network**: All mobile nodes can be self-organized dynamically in an arbitrarily manner. No fixed router and every node acts as a router. Nodes discover routes and maintain routes to other nodes in the network.
- **Faced Constraints:** Power Conservation, Low bandwidth, High error rate and volatile topology, etc



Adapted from www.dei.unipd.it/ ~schenato/

Factors

- Major Paradigm shifts in the following areas:
  - Application driven network architectures
  - Emerging sensor architectures and technology
  - Resource constrained algorithms
  - Media access control
  - Network algorithms
  - Time synchronization
  - Ranging localization and tracking
  - Query processing and data aggregations

Vision of the Future in this class of Computing

- How do you integrate concepts of ubiquitous and pervasive computing with sensor computing

- How do you design knowledge discovery and information access through the use of ambient intelligence in which human are surrounded by an environment which is sensitive and responsive.

The Evolution of Sensing Background

- **Sensors on a network (*Connected*)**
  - **Specific mission**
  - **Limited number of homogeneous sensors; often fixed platforms**
  - **Minimal overlapping views {space and time}**
  - **Centralized control, processing, and decision making**

- **Sensor Netting (*Interoperable*)**
  - **Shared but often related missions**
  - **Increased number and mixture of specific sensors and platforms (static)**
  - **Complementary views {space, time, phenomenology}**
  - **Mixture of centralized and node control, processing and decision making**

Courtesy of MITRE

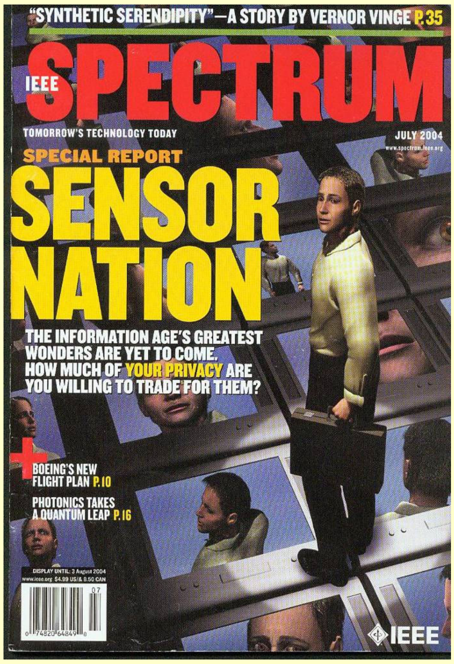
Cont’d

- **Netted Sensing (*Collaborative*)**
  - **Adaptable to support simultaneous missions**
  - **Very large and dynamically changing number of diverse**
  - **heterogeneous sensors and platforms; remote and proximity sensing**
  - **Overlapping views {space, time, & phenomenology**
  - **Distributed collaborative control, processing, and decision making**

Courtesy of MITRE

What's Driving the Commercial World?

- **Collaborative multi-phenomenology remote and proximity sensing is a central tenet within an "Information Age" vision of netted intelligent (smart) objects**
- **Industry is aggressively pursuing the development and commercial applications of netted multi-phenomenology sensing technologies**
- **Commercial marketplace will provide supporting technologies for sponsors' missions**
  - **Leverage and appropriately apply commercial technologies**
  - **Influence future development and augment as necessary**



Courtesy of MITRE

Commercial Applications

- **Manufacturing**
- **Smart Personal Appliances**
- **Smart Buildings**
- **Smart Highways**
- **Personnel Identification**
- **Global Climate Monitoring**
- **Agriculture and Livestock**
- **Entertainment**
- **Aviation**
- **Security**

Intelligent sensor networks for manufacturing control applications

Making wines finer with wireless sensor networks

Livestock breeding- Sensing cow hormones

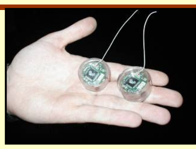
A sensor web pod monitoring conditions at a farm

*"Wireless Sensor Networks, has been identified as One of Ten Emerging Technologies That Will Change the World." **

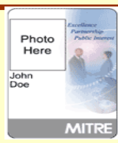
Courtesy of MITRE

Netted Sensors Technologies

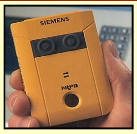
- Proximity Sensors
- Power Sources
- Miniaturization
- System Integration
- Ad hoc & Mobile Wireless Networks
- Collaborative Signal Processing & Fusion
- Distributed Computing
- Information & Resource Management
- Security



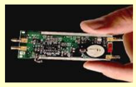
Sophisticated Networking 800 node demo at Intel Developers Forum LLNL



Smart ID Card

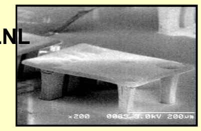


Radiation Sensor



Micro power Impulse Radar

MOTE Netted Sensor Platform

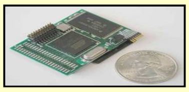


CalTech MEMS Antenna

94 GHz Full Color Wireless Camera

NEC Fuel Cell

Biological Sensor



MITRE Netted Sensor Platform

System on a Chip (SoC)
Wireless-Internet on a chip.
10-50 M transistors.
Embedded µP, FPGA, RF and analog

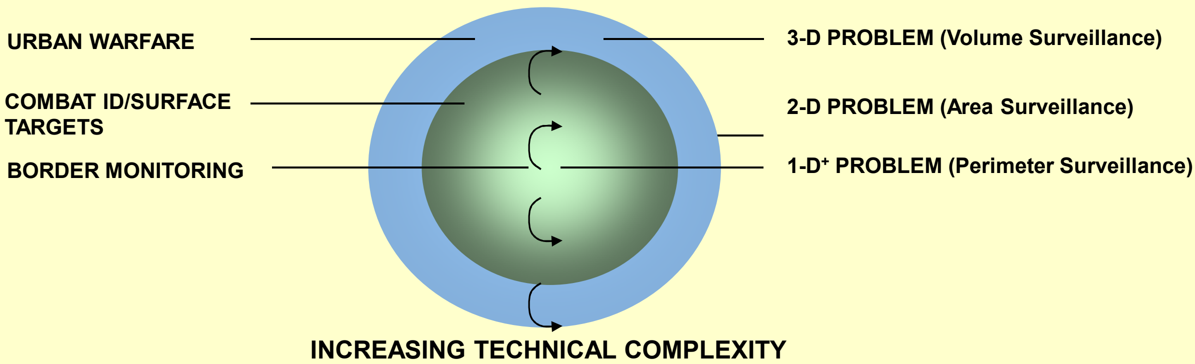
Sarnoff RFID, 250 µm^2; Antenna etched in Si,

26

Courtesy of MITRE

Challenge Problems

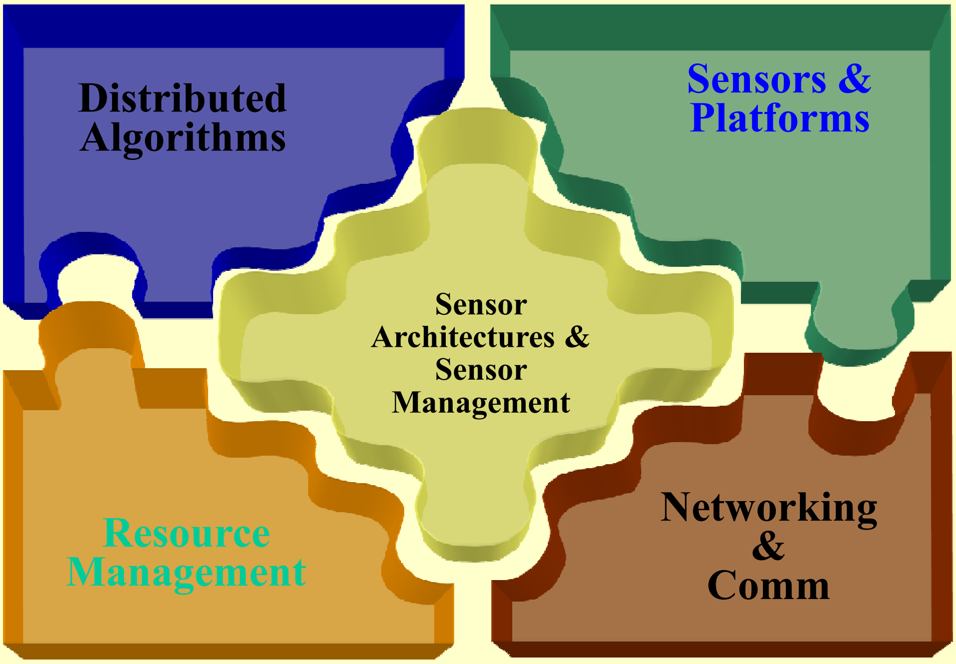
- Challenge problems have key common technology components
- Develop core technology components the first and expand these core capabilities in the following years
- Complexity of the technology components increases as we migrate from the Border Monitoring to the Combat ID and Urban Warfare capability demonstrations



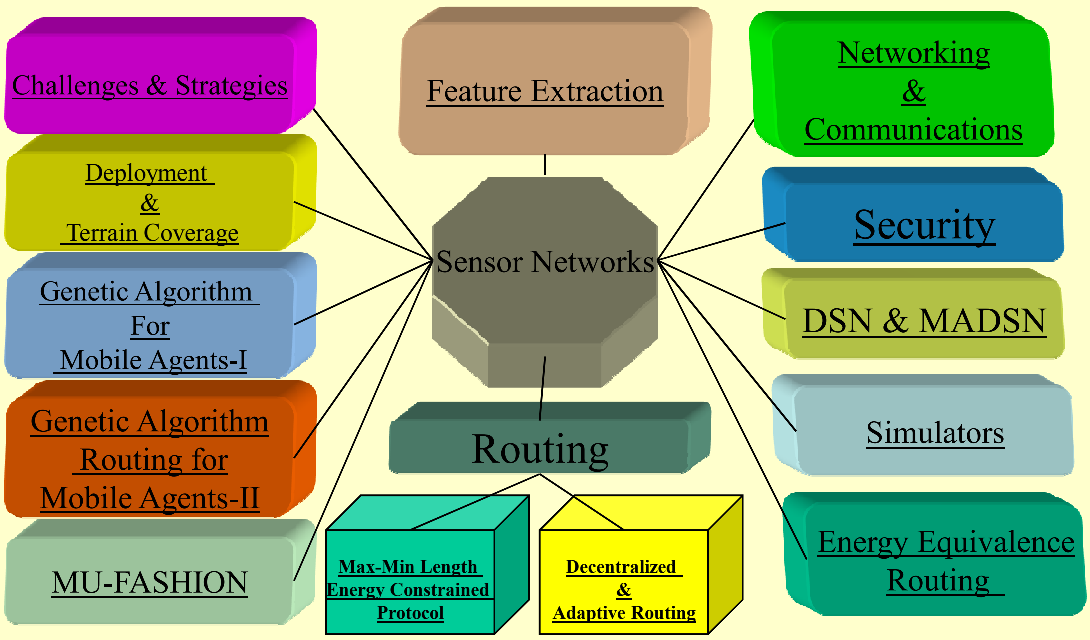
Courtesy of MITRE

Netted Sensing

Interdisciplinary R&D



Research at LSU

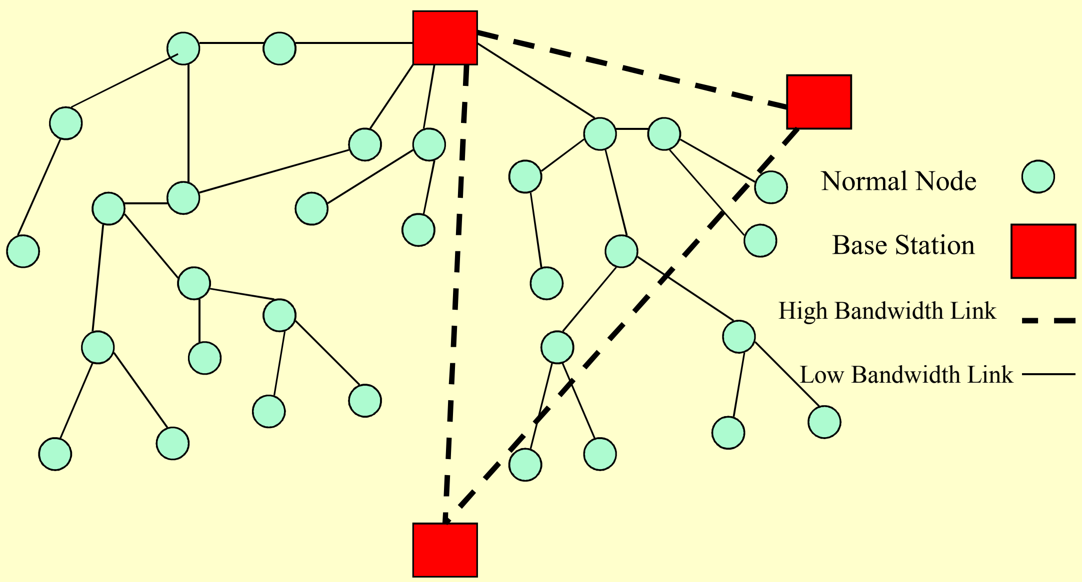


Data Centric Paradigm

- The growing popularity and near ubiquitous deployability of sensor networks [*Saffo, Sohrabi, Brooks, and Iyengar etc*.] is expected to significantly impact the fields of information processing and data management.

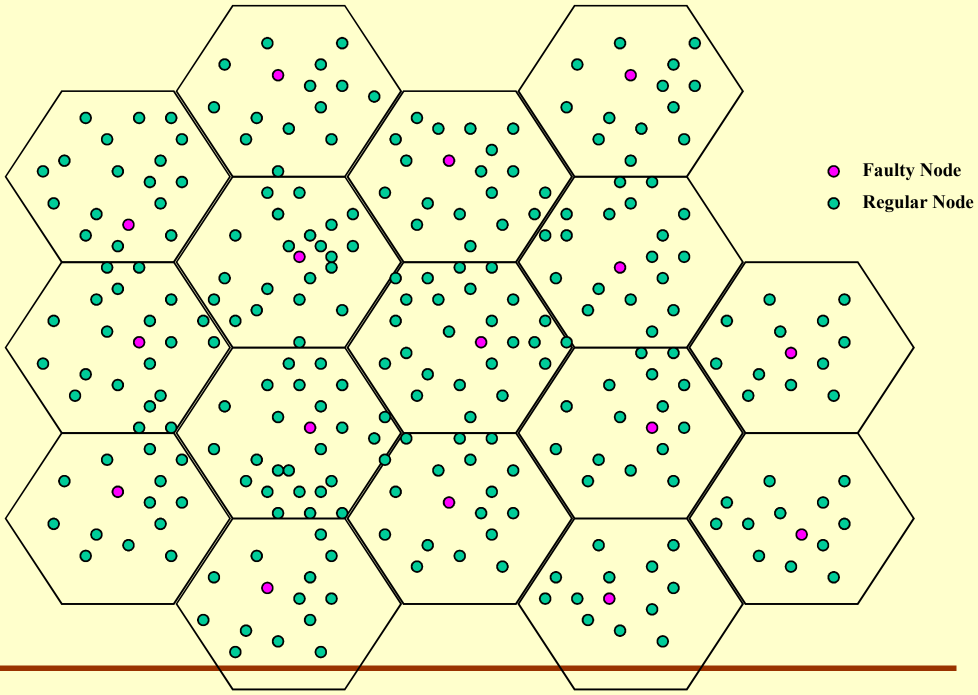
- Network tasks are executed by routing and cooperative processing of sensed data.

Distributed Computing Paradigm in Sensor Networks



Adapted From: Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures, University of California at Berkeley, Chris Karlof, David Wagner, First IEEE International Workshop on Sensor Network Protocols and Applications, 2003

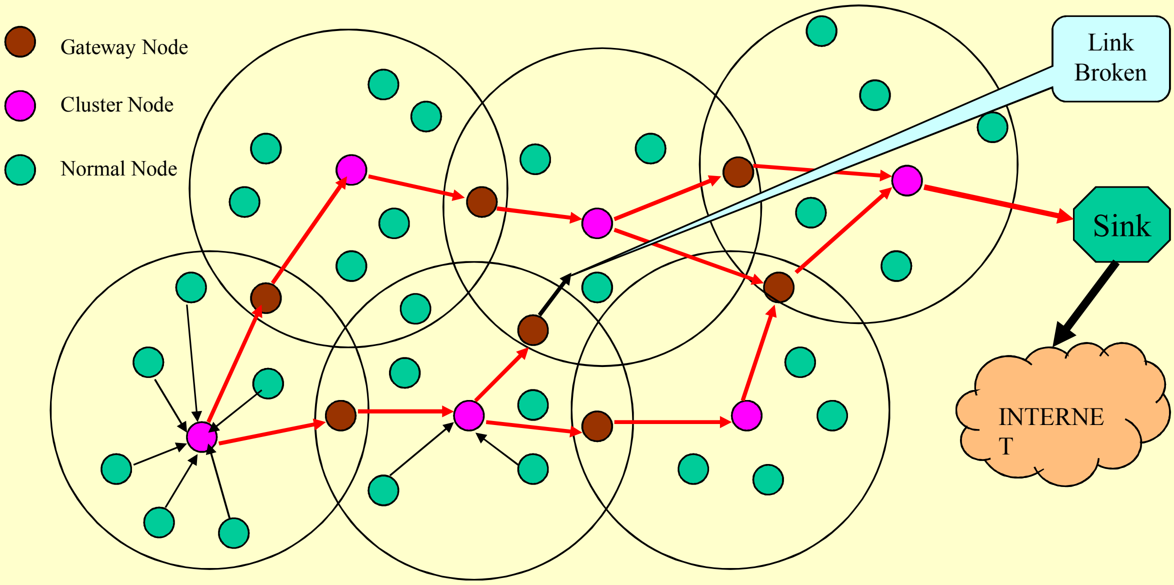
Computation done by clusters of independent processors need not be sensitive to the failure of a small portion of a network



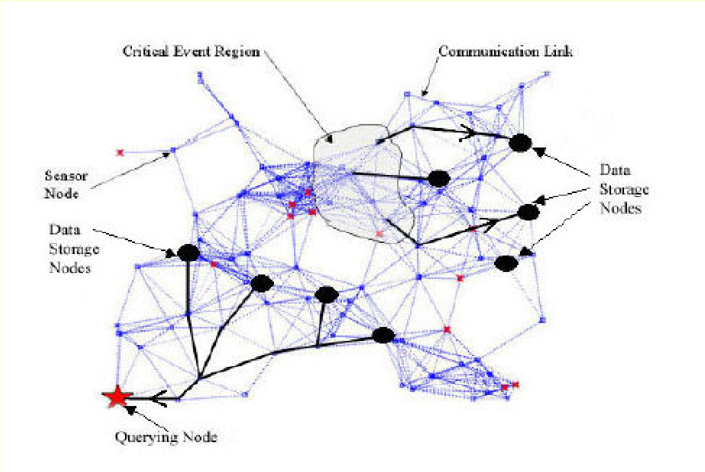
Concept of Data Aggregation

- Data Aggregation
  - Data Aggregation can be defined as the intelligent gathering of data so that redundancy is eliminated and the in network traffic is minimized
  - The idea is to minimize the number of transmissions to save the energy
  - This approach shifts sensor networks from traditional address centric to data centric paradigm

Concept of Data Aggregation



Data Centric Paradigm



Sensed data from an eventful region is being routed to selected storage sensors while others are responding to a data mining query.

Storage sensors, data contents, and query paths are selected optimally by our integrated information management framework.

Data Centric Paradigm

- Sensor networks are uniquely application-specific and data-centric.
- Nodes can collect raw data in various modes such as continuous sensing, event detection, and location-based sensing, depending on the network application.
- Applications such as flood detection , rainfall and water level sensors periodically supply information to a centralized database in a predefined manner.

36

Data Centric Paradigm

- Alternatively, data can be extracted dynamically as replies to specific queries addressed to nodes in the sensor network.
- Users send their interest queries to (subsets of) nodes in the network regarding attributes of sensed phenomena or triggering events.
- Data is aggregated in the network depending on the location of events and the presence of intersecting data reporting paths.
- Projects such as the COUGAR Database consider distributed approaches in interacting with sensor nodes in the network to provide snapshot and long-running queries.

Some Challenging Problems

- While much of the recent research has focused on energy efficiency, protocols, and distributed databases, there is much less attention given to sensor data security.
- Security aspects are more important than performance and low energy consumption
- Security is critical in premise surveillance, and in sensors embedded in critical systems such as at airports, in hospitals, etc.
- The data-centric behavior of sensor networks leaves them vulnerable to traffic analysis and identification of event locations and active areas.

Sensor-Centric Paradigm

- Embedded Sensor Networks are massively distributed systems for sensing and *in situ* processing of spatially and temporally dense data
- Each sensor node can contain multiple sensors of different modalities, such as infra-red, chemical, or biological sensors for collecting different types of data
- Node life, transmission cost and energy consumption are all factors in data aggregation

Sensor-Centric Paradigm

- Sensors have more autonomy-
  - they may decide whether to participate in routing or not
  - they may consider maximizing their individual lifetime
- 2 more realistic constraints to data centric paradigm
  - possibility of sensor failure
  - sensors must cooperate to achieve network wide objective while maximizing their individual lifetime
- Uses game theoretical routing model in which *rational* sensors select routing paths by evaluating the trade-offs between reliability and cost of communication.

Sample Performance Metrics

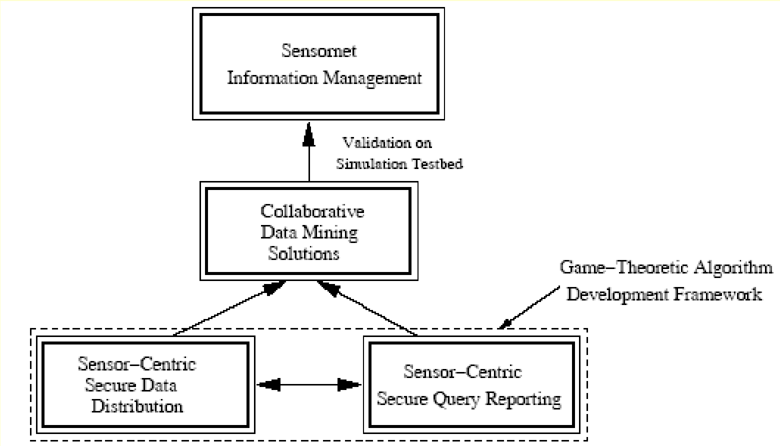
- Uses quality of routing concepts for evaluating data aggregated routed trees, the metrics of which is called path weakness
- Node gain estimations by deviating from the optimal data aggregated tree.
- Given the substantial vulnerabilities of SensorNets to denial of service/data attacks this method guarantees data aggregation while achieving the maximum lifetime of the network.
- Identify the problems of secure data distribution (data storage and content) and data access (in terms of secure query reporting and query collaboration), as two major components of sensor information management.

Sensor-Centric Paradigm

- Iyengar, Kannan et al. formulated a new sensor-centric paradigm of sensor network operation.
- Treats sensors as rational agents cooperating to maximize network wide objectives (such as reporting queries via reliable short paths) without compromising their own survivability (as measured by their energy consumption).
- Developed efficient algorithms for reliable, length and energy-constrained sensor-centric routing and sensor-centric measures of network vulnerability.
- For example, developed a metric for measuring and maximizing the minimal sensor integrity of sensor deployments, developed path weakness metrics to measure the qualitative performance of different routing schemes and provides limits on the approximation of computing paths with bounded weakness.

lnfocom 2003, IEEE JSAC 2005, IPL, JPDC, Sensor Letters,Best-paper at the Baltimore Conference

Components of the proposed information management framework



Embedded Sensor Networks are massively distributed systems for sensing and *in situ* processing of spatially and temporally dense data

Each sensor node can contain multiple sensors of different modalities, such as infra-red, chemical, or biological sensors for collecting different types of data

43

Sample Sensor Area Networks (SANs)

- Mostly Fixed - no or low mobility
- Dense networks of Heterogeneous sensors
  - 1000s of low cost sensors: 6” – 10” apart
  - tens of high end sensors: 10m - 30m
- Easy to deploy - cost effective, work with existing structures
- Low maintenance
- Scalable - progressive deployment over time
- Local processing and filtering of data
- Remote Data Collection, Access and Control

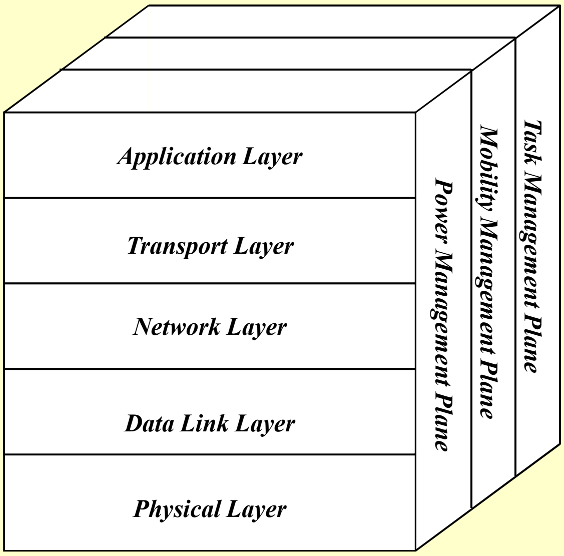
44

Design Challenges

- **Heterogeneity of sensors** - strain, temperature, humidity, video
  - different data collection capabilities
  - different types of data
- **Data transmission needs** – periodic, threshold triggered, query based, time sensitive
- **Prioritization of data streams** based on available network capacity, minimum standard of data quality, reliable response to critical events
- **No Line of sight** – embedded in concrete, in remote spaces, behind beams.
- **Power supply** – grid, battery, self powered
- **Scalability** - progressive growth
- **Robustness** - survive link and node losses
- **Self Healing** - re-configuration of the system due to losses/additions
- **Deployment** - grid based, arbitrary/random, specific angles (cameras)
- **Data collection, aggregation and processing** - multihop, power aware, local data logger, interconnection with Internet for remote sensing/control

45

Design Challenges



Basic sensor network protocol stack

Adapted from I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. Computer Networks, 38(4):393--422, March 2002.

Objectives

- Automatic tracking of characteristics in a geographic region
- Minimize human intervention and management
  - Maximize accuracy
- Hostile environments(ie. Volcanic activity)
- System lifetime
  - Computation/communication becoming cheaper
  - Energy as critical resource

47

What is out there?

- Wired solutions – ethernet, FDDI, token ring, etc.
- Wireless solutions
  - Cellular network
  - IEEE 802.11a/b/g
  - UC Berkeley/Intel Mote project
  - UCLA environmental sensor project
  - LSU Sensor Simulator for Coastal Erosion
  - Bluetooth
  - UltraWide Band technology - new, emerging
    - Low power
    - High bit rates
    - Short distances
    - Good penetration

Available Sensor Technology

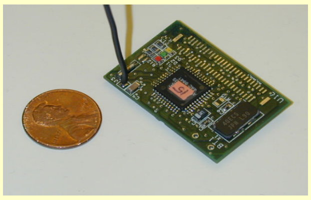
- Today sensors are available to detect:
  - Chemicals, radiation levels, light, seismic activity, motion
  - Audio, video
- Challenges ahead:
  - Miniaturization
  - Untethered communication
  - Extended battery lifetime
  - Self-organization

Network Topology

- One must choose a communication scenario that addresses the unique issues identified for civil infrastructure SANs

Emerging Technology

- MEMS
  - Micro Electro-Mechanical Sensors
  - Potential for tremendous storage capabilities
- Motes
  - 8 bit 4 Mhz Atmel microprocessor, 512 bytes SRAM, 8 K Flash ROM
- Cell computer
  - Full featured PC



51

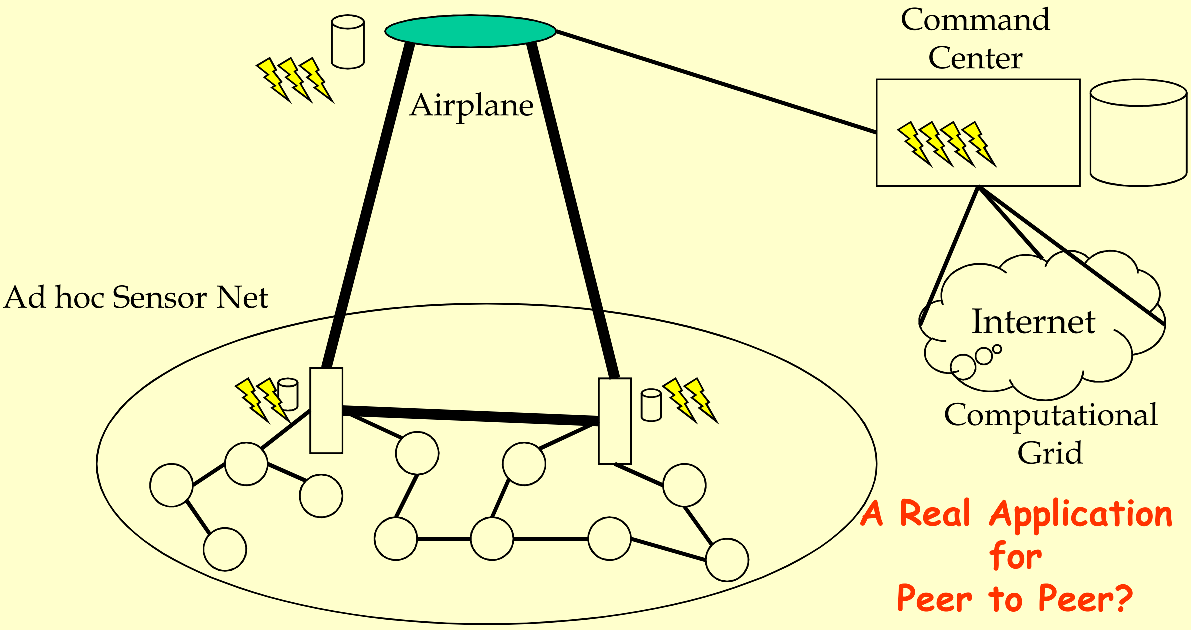
Target Scenarios

- Disaster recovery
  - Aid in search and rescue
- Agriculture
- Industrial tracking
  - Sensors prevalent in automobiles
- Military scenarios
  - Aircraft dropping sensors over a geographic range and circling the area, acting as a base station
- Surveillance?

Example Scenario

- Plane drops thousands of sensors over target area
  - Heterogeneous capabilities: computation, communication bandwidth/range, battery power
  - Plane circles area acting as base station
    - For subset of nodes with relatively long range (low bw?)
  - Sensors self-organize into hierarchical ad hoc network
  - Communicate findings to plane (with local computation)
  - Plane distills data stream, computes and transmits to command center
  - Feedback from command center, through plane, to sensor net

Example Scenario



A Real Application for Peer to Peer?

54

Necessary Design Features

Needed Features:

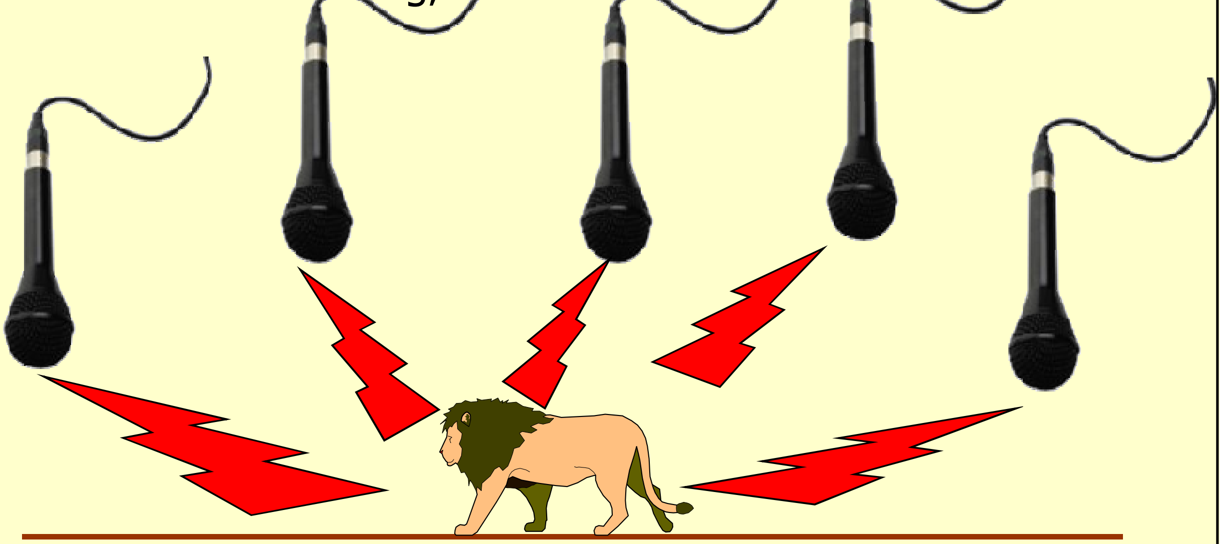
- Short distances between nodes
- High bit rates
- Low power consumption
- 2 way communication
- Node identification
- Ad Hoc network formation
- Scalable
- Robust
- Self Healing
- RF - no LOS

What is not needed:

- High duty cycles
- Long ranges
- Complex protocols

Time synchronization

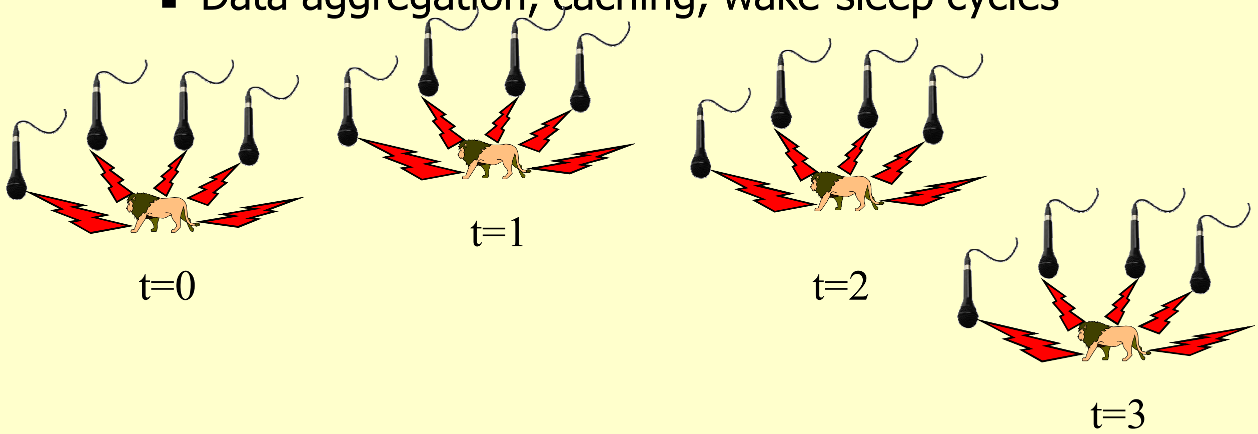
- Time sync is critical at *many* layers
  - Beam-forming, localization, distributed DSP



56

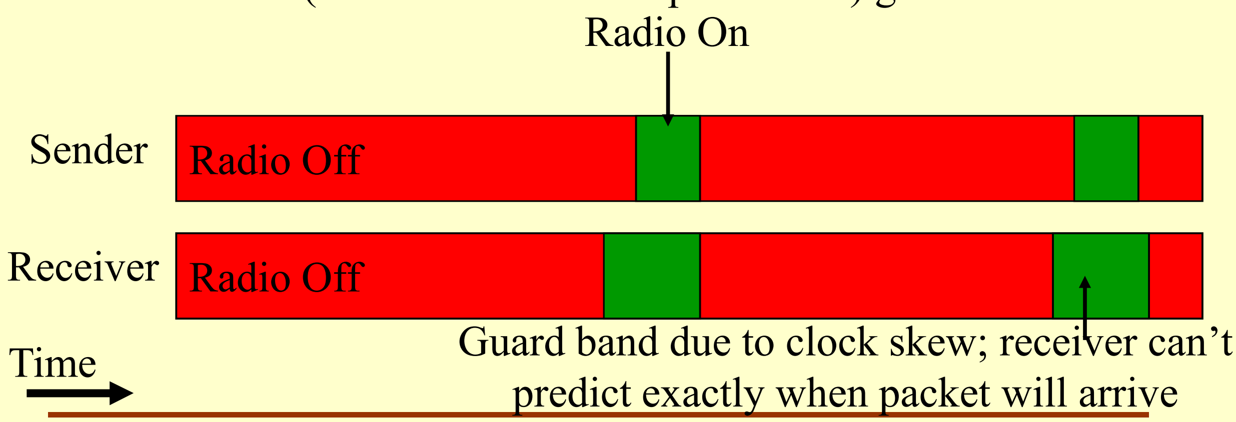
Time synchronization

- Time sync is critical at *many* layers
  - Beam-forming, localization, distributed DSP
  - Data aggregation, caching, wake-sleep cycles



Time synchronization

- Time sync is critical at *many* layers
  - Beam-forming, localization, distributed DSP
  - Data aggregation & caching
  - TDMA(Time Division Multiple Access) guard bands



58

Time synchronization

- Time sync is critical at *many* layers
  - Beam-forming, localization, distributed DSP
  - Data aggregation & caching
  - TDMA guard bands
    - Clock sync for TDMA is more important in sensor nets, compared to traditional nets:
      - Listening is EXPENSIVE
      - Infrequent data means infrequent sync
      - Small data means guard band is relatively big

59

Related work

- Clock sync over computer networks
  - Protocols: NTP, Berkeley, Cristian’s probabilistic alg
- Stable frequency standards
  - Cesium, Rubidium, temperature-controlled…
- National time standards
  - USNO’s time, UTC/TAI
  - Two-way satellite time transfer, GPS
- Virtual clocks (Lamport)

what's wrong with what's there?

- Existing work is a critical building block

BUT...

- Energy
  - e.g., we can't always be listening or using CPU!
- Wide range of requirements within a single app; no method optimal on all axes
- Cost and form factor: can disposable motes have GPS receivers, expensive oscillators? Completely changes the economics…

61

Approach

- Use multiple modes
  - Extend existing sync methods
  - Develop new methods, and compositions of methods
  - Characterize these methods
- Use tiered architectures
  - Not a single hardware platform but a range of hardware
  - Analogy: memory hierarchy
- The set as a whole can (?) be necessary and sufficient, to minimize resource waste
- Don’t spend energy to get better sync than app needs

62

Directed Diffusion

- Example of data centric paradigm
- Sensor network challenges
- One approach: Directed diffusion
  - Basic algorithm
- Other interesting localized algorithms in progress:
  - Aggregation (Kumar)
  - Adaptive fidelty (Xu)
  - Address free architecture, Time synch (Elson)
  - Localization (Bulusu, Girod)
  - Self-configuration using robotic nodes (Bulusu, Cerpa)
  - Instrumentation and debugging (Jerry Zhao)

The Challenge is Dynamics!

- The physical world is dynamic
  - Dynamic operating conditions
  - Dynamic availability of resources
    - … *particularly energy!*
  - Dynamic tasks
- Devices must adapt automatically to the environment
  - Too many devices for manual configuration
  - Environmental conditions are unpredictable
- Unattended and un-tethered operation is key to many applications

64

Approach

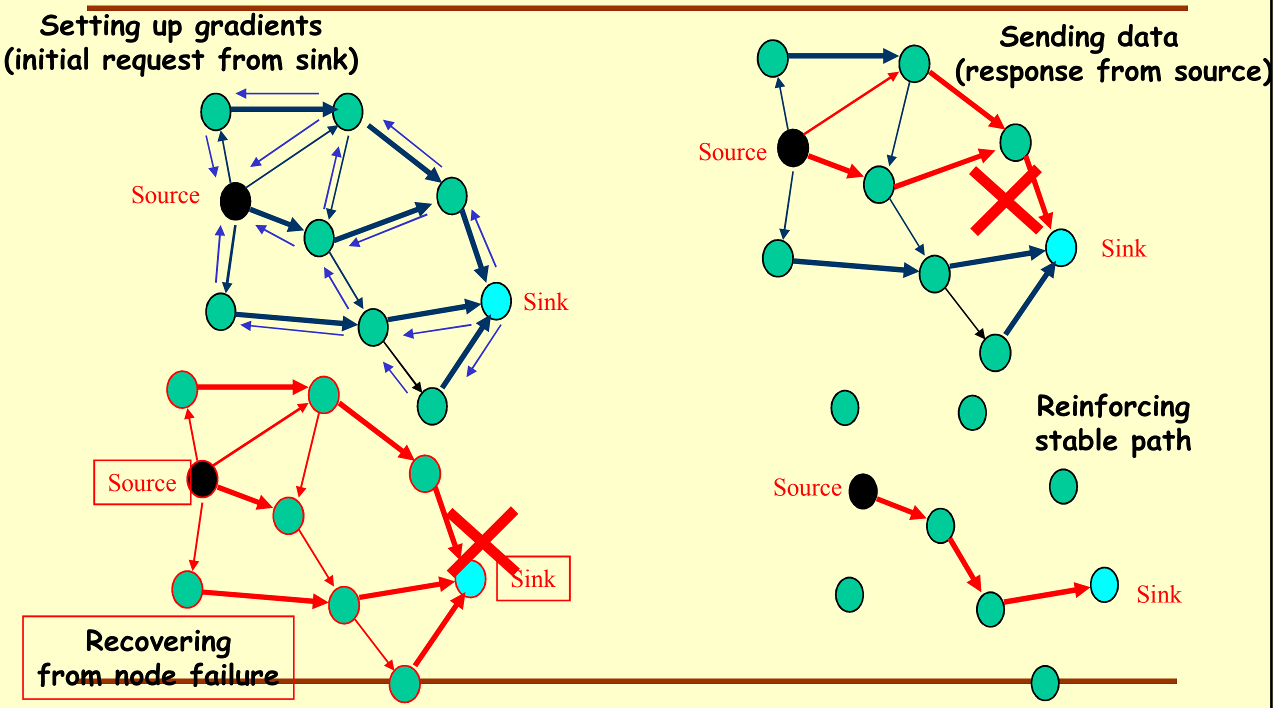
- Energy is a bottleneck resource
  - Energy is consumed in sensing, computing, and communication
  - communication is a major consumer--avoid communication over long distances
- Pre-configuration and global knowledge are not applicable
  - Achieve desired global behavior through localized interactions
  - Empirically adapt to observed environment
- Leverage points
  - Small-form-factor nodes, densely distributed to achieve Physical locality to sensed phenomena
  - Application-specific, data-centric networks
  - Data processing/aggregation inside the network

65

Directed Diffusion Concepts

- Application-aware communication primitives
  - expressed in terms of named data *(not in terms of the nodes generating or requesting data)*
- Consumer of data initiates interest in data with certain attributes
- Nodes diffuse the interest towards producers via a sequence of local interactions
- This process sets up gradients in the network which channel the delivery of data
- Reinforcement and negative reinforcement used to converge to efficient distribution
- Intermediate nodes opportunistically fuse interests, aggregate, correlate or cache data

Illustrating Directed Diffusion



Local Behavior Choices

1. For propagating interests
   - In our example, flood
   - More sophisticated behaviors possible: e.g. based on cached information, GPS
2. For setting up gradients
   - Highest gradient towards neighbor from whom we first heard interest
   - Others possible: towards neighbor with highest energy
3. For data transmission
   - Different local rules can result in single path delivery, striped multi-path delivery, single source to multiple sinks and so on.
4. For reinforcement
   - reinforce one path, or part thereof, based on observed losses, delay variances etc.
   - other variants: inhibit certain paths because resource levels are low

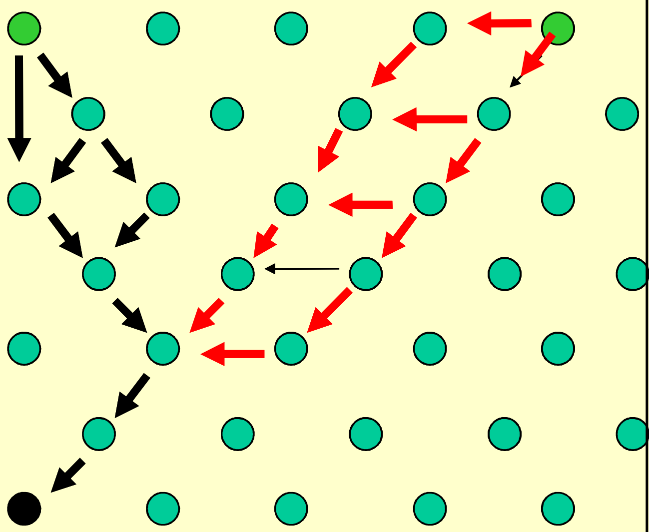
68

Simulation studies

- IDLE time dominates energy consumption…need low duty cycle MAC, driven by application.
  - With 802.11ish contention protocols you might as well just FLOOD
- Easy to get lost in detailed simulations but in the wrong region of operation …
  - Node density, traffic load, stream length, source and sink placement, mobility, etc.

Diffusion based Aggregation

- Scaling requires processing of data INSIDE the net
- Clustering approach:
  - Elect cluster head (various promotion criteria)
  - Aggregation or Hashing (indirection) to map from query to cluster head
- Opportunistic aggregation:
  - Reinforce (request gradient) proportional to aggregatability of incoming data (Amit Kumar)



70

Adaptive Fidelity

- In densely deployed sensor nets, reduce duty cycle: engage more nodes when there is activity of interest to get higher fidelity
- Adjust node's sleeping time according to the number of its neighbors.
- Initial simulations applied to ad hoc routing
- Performance Metric: Percentage of survived nodes over time.
  - The more nodes survive, the longer network lifetime

Conclusion

- Sensor technology continuing to improve
- Key is to integrate computation, communication, and sensing on single miniature platform
- Power management
- Redundancy to maximize fault tolerance
- Dynamically react to changing network conditions
- Minimize human interaction to maintain system
- Empower people to make proper high level decisions

Part –III
Open Research Problems

- When is co-operation worthwhile? (collaborative communication, distributed signal processing).
- Does spatial proximity necessarily lead to data correlation? Give examples for and against this.
- What is a nice way to make clusters and elect cluster heads? Use Voronoi diagrams and some form of Vector Quantization (C-MAC, Akyildiz of GA-Tech)? Any other way?
- How to form coalition, make negotiation etc?
- Is there any such thing as optimal clustering? What criteria limits a cluster size? – Application dependent.

Cont’d

- How does a node know whom to correlate with?
- How to avoid overhearing? Use “trip-wire” nodes?
- How to form logical clusters? What are logical clusters (nodes providing similar data)?
- How to deal with interference while not providing time-slots to nodes?
- Identification of nodes? RETRI – Random Ephemeral Transaction Identifiers.
- How best to use “precision and recall” (data retrieval)?
- Do we have the ability to tell which sensor should do what? How can that be achieved?

Cont’d

- How to minimize the data (the number of times it is sent as well as the length of the data packet) being sent while maintaining the quality of the data? (Fusion).
- How to achieve good adaptive duty-cycling?
- How to utilize Game Theory effectively in sensor networks? All nodes are selfish (they don’t want to spend energy) and equally smart. At the same time, all of them should collectively work to provide the event information to the user. Read Stephen Wicker (Cornell) and Allen MacKenzie papers.
- How to use Information Theory in Sensor Networks? We can use channel coding and error correction mechanisms during transmissions. Also, we can use Vector Quantization techniques to elect a node from a Voronoi region.

Cont’d

- How best to handle dynamic topologies? If a node joins or leaves a cluster, how to react? How to detect it fast (should we do it fast??), should we re-run topology management algorithm immediately, calculate the effect (multiple aspects) of its exclusion or inclusion?
- Monitoring and Maintenance of sensor nodes – For what kind of apps is it necessary and how can it be realized? Read “eScan” and “digest” (papers by D. Estrin and R. Govindan).
- Use of Computational Geometry to solve unique issues in sensor networks. Find problems in this area –Xiang Yang Li (Illinois Institute of Tech).

Cont'd

- QoS in a cluster – reliable data within a latency threshold by using power below certain threshold? Define QoS.
- What MAC model would be nice to reduce latency? Remember – the more the number of nodes send to their common parent faster, the better. TDMA is inherently bad (scalability issues). CSMA is OK (use of RTS-CTS mechanism). Goal is for multiple access of channel at the same time. The individual source nodes can send their event data by choosing mutually orthogonal frequencies or by using some codes (like CDMA).

Implementing a Sensor Network.

S. Srivathsan
Ravilochan G Shamanna
Prof. Iyengar
Prof. Kannan

Dept of Computer Science, LSU.

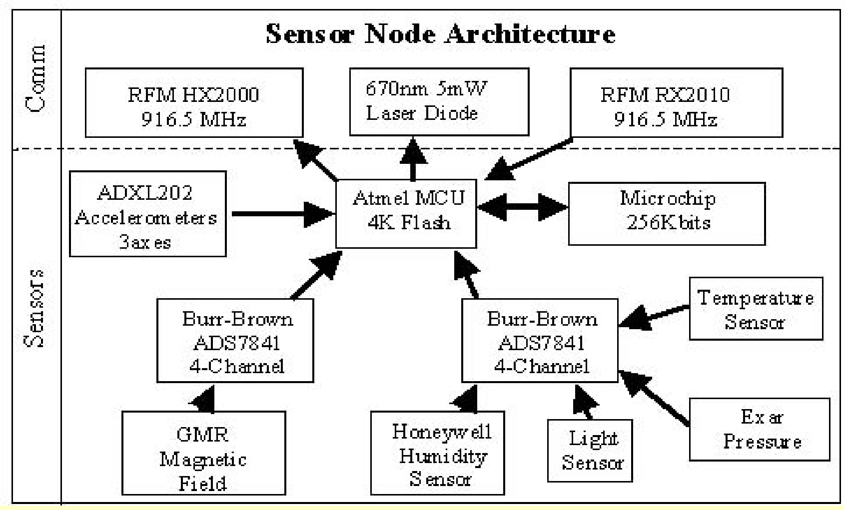
References used for this presentation.

1. System architecture directions for network sensors, Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, ASPLOS 2000, Cambridge, November 2000.
2. A Transmission Control Scheme for Media Access in Sensor Networks, Alec Woo, David Culler, Mobicom 2001,Rome, July 2001.
3. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks, Sam Madden, Michael Franklin, Joe Hellerstein and Wei Hong, OSDI 2002, Boston, December 2002.
4. The nesC Language: A Holistic Approach to Networked Embedded Systems, David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler, Proceedings of Programming Language Design and Implementation (PLDI), San Diego, June 2003.
5. The Design of an Acqusitional Query Processor for Sensor Networks, Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, SIGMOD, San Diego, June 2003.
6. TinyOS Tutorial. http://www.tinyos.net/tinyos-1.x/doc/tutorial/
7. The Emergence of Networking Abstractions and Techniques in TinyOS, Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Alec Woo, Eric Brewer and David Culler, Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), San Francisco, March 2004.
8. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, Philip Levis, Nelson Lee, Matt Welsh, and David Culler, Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, November 2003.

References. *continued*

9. Software Design Patterns for TinyOS,
David Gay, Philip Levis, and David Culler, Proceedings of the ACM SIGPLAN/SIGBED 2005 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05),
Chicago, June 2005.
10. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks,
Alec Woo, Ternence Tony, David Culler, ACM SenSys 2003,
Los Angeles, November 2003.
11. CrossBow Training Materials and slides.
12. Mobisys Training Materials and slides.

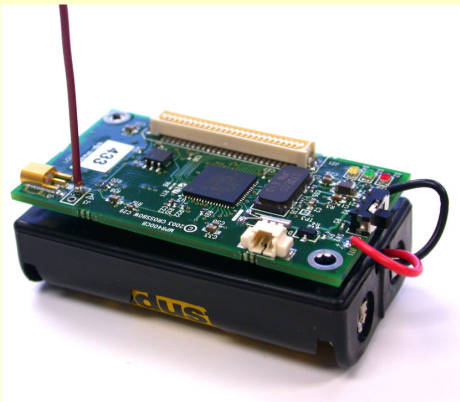
Typical Sensor Mote Architecture



81

Mica2 and Mica2Dot

- ATMega 128L 8-bit, 8MHz, 4KB EEPROM, 4KB RAM, 128KB flash
- Chipcon CC100 multichannel radio (Manchester encoding, FSK). Up to 500-1000ft.



[6]

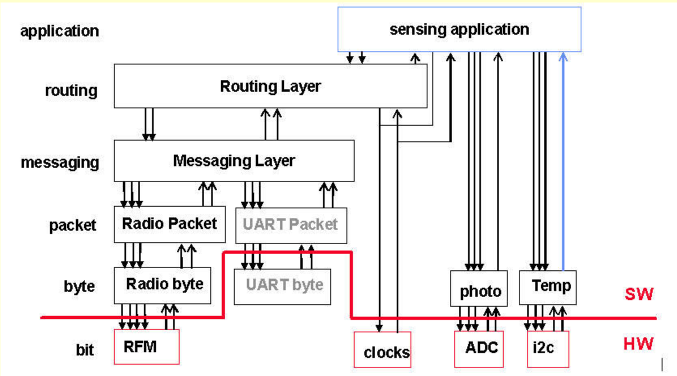
Operating System Design for Sensor Networks - TinyOS

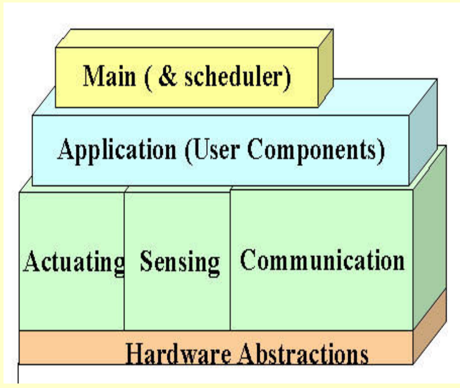
What are all the possible issues while designing an operating system for sensor nodes?

- Application specific nature of the devices.
- Concurrency and Events
- Small Memory Footprint
- Power efficient
- Efficient modularity

TinyOS

- application = scheduler + graph of components
- event-driven architecture
- single shared stack
- NO process/memory management, virtual memory

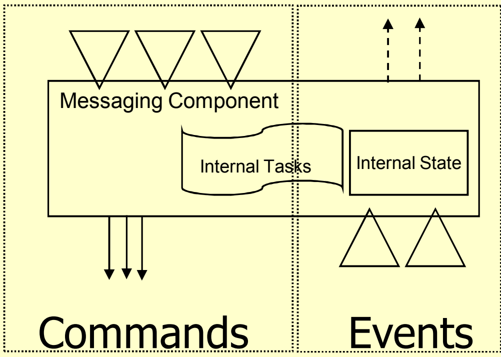




84

TinyOS component model

- Component has:
  - Frame (storage)
  - Tasks: computation
  - Interface:
    - Command
    - Event

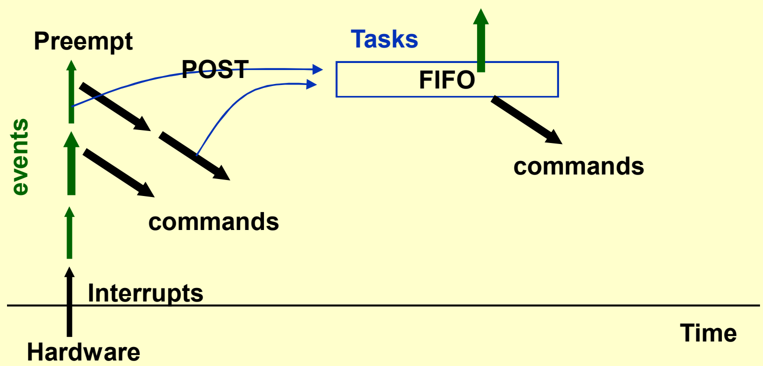


- Frame: static storage model - compile time memory allocation (efficiency)
- Command and events are function calls (efficiency)

85

TinyOS Two-level Scheduling

- **Tasks do computations**
  - Unpreemptable FIFO scheduling
  - Bounded number of pending tasks
- **Events** handle concurrent dataflows
  - Interrupts trigger lowest level events
  - Events prempt tasks, tasks do not
  - Events can signal events, call commands, or post tasks



Networking in TinyOS

- Socket/TCP/IP?
  - Too much memory for buffering and threads
    - Data are buffered in network stack until application threads read it
    - Application threads blocked until data is available
  - Transmit too many bits (sequence #, ack, re-transmission)
  - Tied with multi-threaded architecture

- TinyOS solution: **active messages**

Active Message

- Every message contains the name of an event handler
- Sender
  - Declaring buffer storage in a frame
  - Naming a handler
  - Requesting Transmission
  - Done completion signal
- Receiver
  - The event handler is fired automatically in a target node

✓ No blocked or waiting threads on the receiver
✓ Behaves like any other events
✓ Single buffering

Networking layer support in TinyOS

- Physical Layer – Radio Transmission Layer
- MAC Layer – CSMA, S-MAC
- Routing Layer – Multi-hop routing
- Transport Layer – Is it an overhead??

89

Networking Abstractions in TinyOS

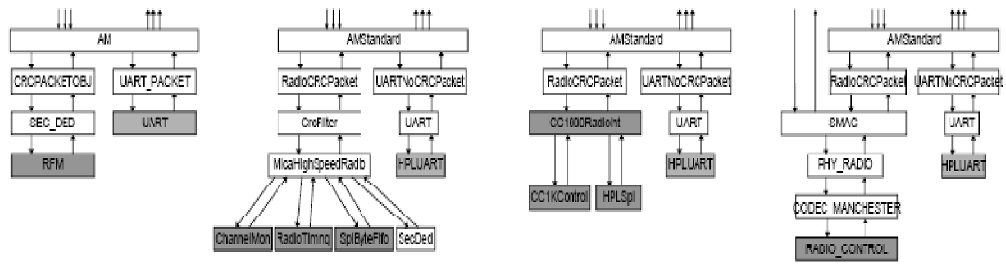
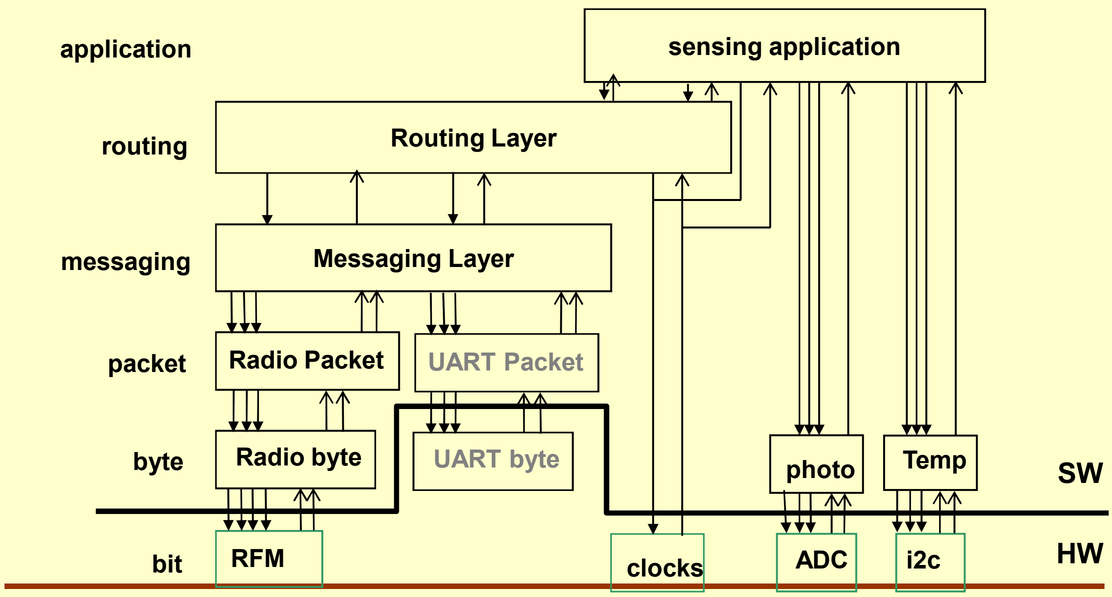


Figure 2: Typical single hop network stacks for three generations of motes. From left to right: rene, mica, mica2, and S-MAC on mica radio stacks. Grey components abstract hardware.

Adapted From: Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Alec Woo, Eric Brewer and David Culler, Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), San Francisco, March 2004.

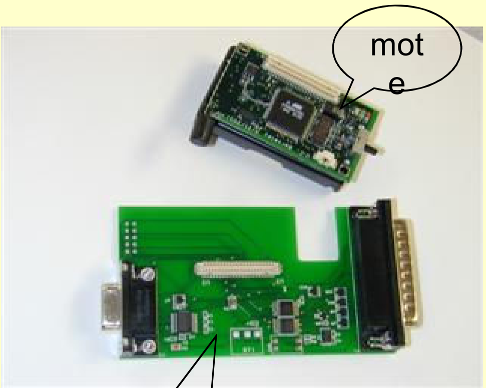
A TinyOS Application



91

Programming Environment

- OS: cygwin/Win2000 or gcc/Linux
- Software: atmel tools, java, perl



92

Programming Environment

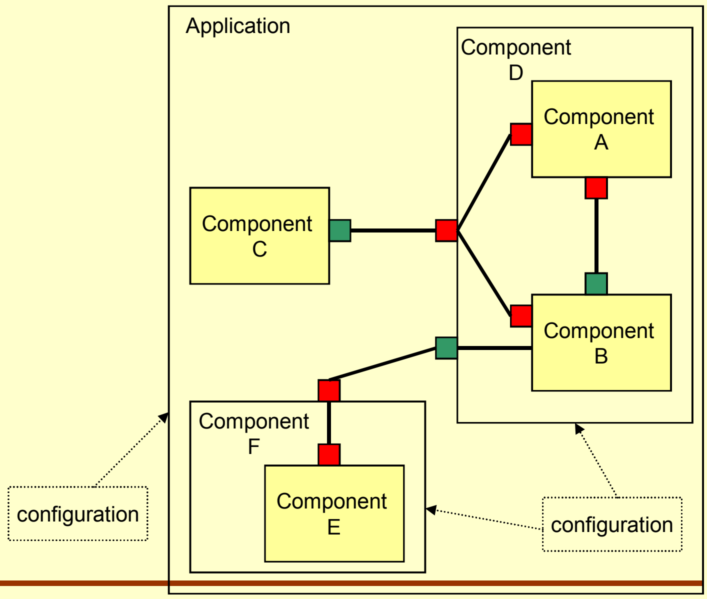
- download, install and build:
  - cygwin (http://www.cygwin.com)
  - nesC (http://nesc.sourceforge.net)
  - Java JDK (http://java.sun.com/j2se/1.4.1)
  - tinyOS distribution (http://sourceforge.net/projects/tinyos)
- build your application
  - code your components
  - $ make mica2 install.1
- debug your application with TOSSIM simulator:
  - $ make pc
  - $ build/pc/main.exe 25

NesC

- C-like programming language with component model support
  - Compiles into GCC-compatible C
- 3 types of files:
  - *Interfaces*
    - Set of function prototypes; no implementations or variables
  - *Modules*
    - Provide (implement) zero or more interfaces
    - Require zero or more interfaces
    - May define module variables, scoped to functions in module
  - *Configurations*
    - Wire (connect) modules according to requires/provides relationship

nesC

- the nesC model:
  - interfaces:
    - uses
    - provides
  - components:
    - modules
    - configurations
- application:= graph of components

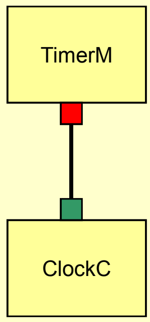


Interfaces

- used for grouping functionality, like:
  - split-phase operation (send, sendDone)
  - standard control interface (init, start, stop)
- describe bidirectional interaction:

```
interface Clock {
  command result_t setRate (char interval, char scale);
  event result_t fired ();
}                                                      Clock.nc
```

- interface provider must implement commands
- interface user must implement events



96

Interfaces

- examples of interfaces:

```
interface StdControl {
   command result_t init ();
   command result_t start ();
   command result_t stop ();
}
                                    StdControl.nc
```

```
interface Timer {
   command result_t start (char type,
                                   uint32_t interval);
   command result_t stop ();
   event result_t fired ();
}                                          Timer.nc
```

```
interface SendMsg {
   command result_t send (uint16_t addr,
                           uint8_t len,
                           TOS_MsgPtr p);
   event result_t sendDone ();
}
                                       SendMsg.nc
```

```
interface ReceiveMsg {
   event TOS_MsgPtr receive (TOS_MsgPtr m);
}
                                    ReceiveMsg.nc
```

97

Modules

- implements a component's specification with C code:

```
module MyComponent {
   provides interface X;
   provides interface Y;
   uses interface Z;
}
implementation {
...// C code
}
```

MyComponent.nc

- a thread of control crosses components only through their specifications

Modules

- parameterised interfaces:

```
module GenericComm {
   provides interface SendMsg [uint8_t id];
   provides interface ReceiveMsg [uint8_t id];
   …
}
implementation {…
}
```

GenericComm.nc

  - i.e., it provides 256 instances of SendMsg and RecvMsg interfaces
  - they are not strictly necessary – the handler ID can be passed as an argument to the *send* method

Modules

- implementing the specification:
  - simple interfaces, (e.g. interface Std of type StdControl):

```
module DoesNothing {
  provides interface StdControl as Std;
}
implementation {
  command result_t Std.init() {
    return SUCCESS;
  }
  command result_t Std.start() {
    return SUCCESS;
  }
  command result_t Std.stop() {
    return SUCCESS;
}
```

DoesNothing.nc

Modules

- calling commands and signaling events
  - simple interface:

```
module TimerM {
   provides interface StdControl;
   provides interface Timer[uint8_t id];
   uses interface Clock;…
}

implementation {
   command result_t StdControl.stop() {
      call Clock.setRate(TOS_I1PS, TOS_S1PS);
   }
   …
}                                              TimerM.nc
```

Modules

- posting tasks:

```
module BlinkM {…
}
implementation {…
  task void processing () {
    if(state) call Leds.redOn();
    else call Leds.redOff();
  }

  event result_t Timer.fired () {
    state = !state;
    post processing();
    return SUCCESS;
  }…
}
```

BlinkM.nc

Configurations

- implements a component by wiring together multiple components:

```
configuration MyComponent {
   provides interface X;
   provides interface Y;
   uses interface Z;
}
implementation {
…// wiring code
}
```

MyComp.nc

- wiring := connects interfaces, commands, events together

Configurations

- connected elements must be compatible (interface-interface, command-command, event-event)
- 3 wiring statements in nesC:
  - $endpoint_1$ = $endpoint_2$
  - $endpoint_1$ -> $endpoint_2$
  - $endpoint_1$ <- $endpoint_2$ (equivalent: $endpoint_2$ -> $endpoint_1$)

104

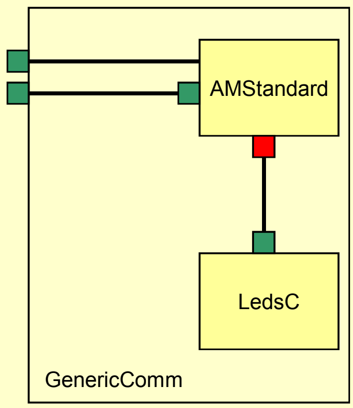
Configurations

- wiring example:

```
configuration GenericComm {
      provides interface StdControl as Control;
      command result_t activity();…
}

implementation {
   components AMStandard, LedsC;

   Control = AMStandard.Control;
   AMStandard.Leds -> LedsC.Leds;
   activity = AMStandard.activity;
}                                                        GenericComm.nc
```



Example

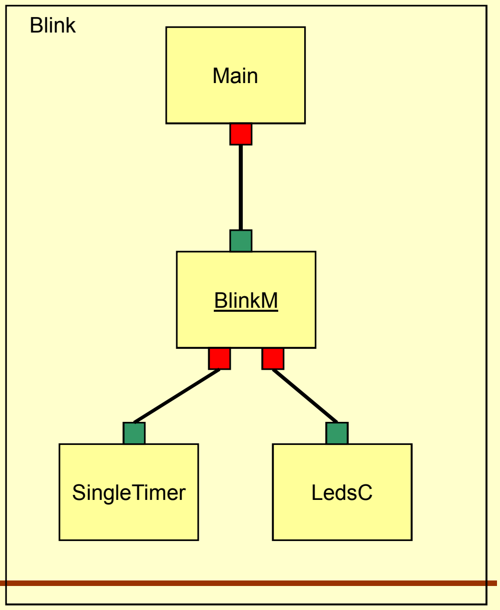
- Blink application

```
configuration Blink {
}

implementation {
   components Main, BlinkM, SingleTimer,
     LedsC;

   Main.StdControl->BlinkM.StdControl;
   BlinkM.Clock->SingleTimer.Timer;
   BlinkM.Leds->LedsC;
}
```

Blink.nc



Example

- BlinkM module:

```
module BlinkM {
  provides interface StdControl;
  uses interface Clock;
  uses interface Leds;
}

implementation {
  bool state;

  command result_t StdControl.init() {
    state = FALSE;
    call Leds.init();
    return SUCCESS;
  }
```

Blink.nc

```
  command result_t StdControl.start() {
    return call Clock.setRate(128, 6);
  }

  command result_t StdControl.stop() {
    return call Clock.setRate(0, 0);

  event result_t Clock.fire() {
    state = !state;
    if (state) call Leds.redOn();
    else call Leds.redOff();
  }
}
```

Blink.nc

107

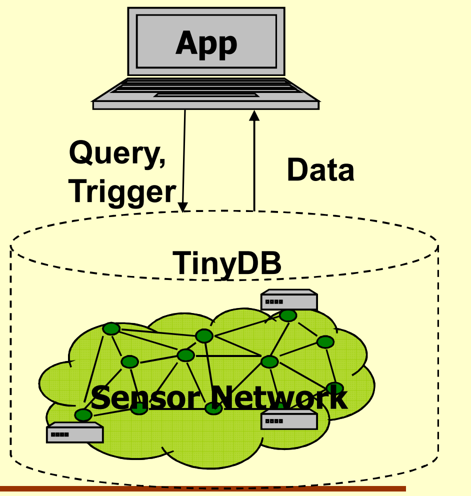
Summary/Discussion

- small memory footprint +
- concurrency intensive application, event-driven architecture +
- power conservation +
- modular, easy to extend +
- simplistic FIFO scheduling -> no real-time guarantees -
- bounded number of pending tasks -
- no process management -> resource allocation problems -
- software level bit manipulation. HW implementation can provide speed up and power saving. -
- no hardware timer support. It is done in software, which is lost during sleep. -
- better OS race conditions support. -

TinyDB

- High level abstraction:
  - Data centric programming
  - Interact with sensor network as a whole
  - Extensible framework
- Under the hood:
  - Intelligent query processing: query optimization, power efficient execution
  - Fault Mitigation: automatically introduce redundancy, avoid problem areas

```
SELECT MAX(mag)
FROM sensors
WHERE mag > thresh
SAMPLE PERIOD 1024 ms
```

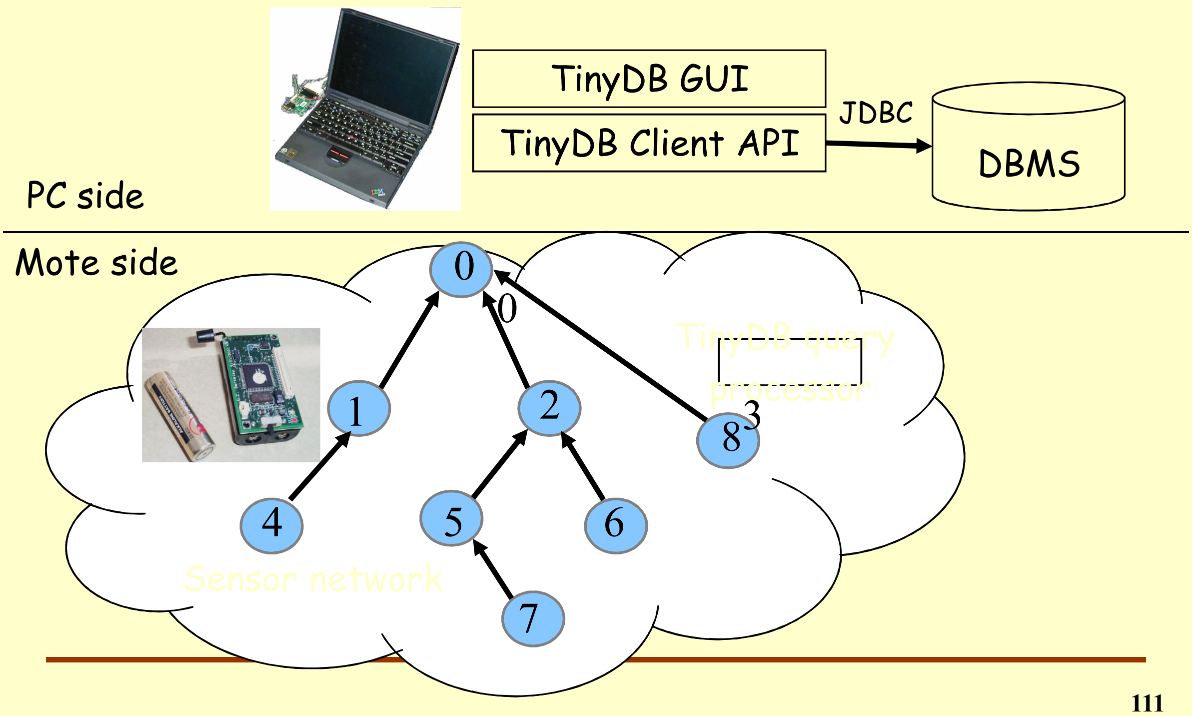


109

Feature Overview

- Declarative SQL-like query interface
- Multiple concurrent queries
- Network monitoring (via queries)
- In-network, distributed query processing
- Extensible framework for attributes, commands and aggregates
- In-network, persistent storage

Architecture



111

Data Model

- Entire sensor network as one single, infinitely-long logical table: *sensors*
- Columns consist of all the *attributes* defined in the network
- Typical attributes:
  - Sensor readings
  - Meta-data: node id, location, etc.
  - Internal states: routing tree parent, timestamp, queue length, etc.
- Nodes return NULL for unknown attributes
- On server, all attributes are defined in catalog.xml
- Discussion: other alternative data models?

Query Language (TinySQL)

SELECT <aggregates>, <attributes>
[FROM {sensors | <buffer>}]
[WHERE <predicates>]
[GROUP BY <exprs>]
[SAMPLE PERIOD <const> | ONCE]
[INTO <buffer>]
[TRIGGER ACTION <command>]

Comparison with SQL

- Single table in FROM clause
- Only conjunctive comparison predicates in WHERE and HAVING
- No subqueries
- No column alias in SELECT clause
- Arithmetic expressions limited to *column op constant*
- Only fundamental difference: SAMPLE PERIOD clause

TinySQL Examples

"Find the sensors in bright nests."

1

SELECT nodeid, nestNo, light
FROM sensors
WHERE light > 400
EPOCH DURATION 1s

Sensors

Epoch	Nodeid	nestNo	Light
0	1	17	455
0	2	25	389
1	1	17	422
1	2	25	405

TinySQL Examples (cont.)

② SELECT AVG(sound)

FROM sensors

EPOCH DURATION 10s

"Count the number occupied nests in each loud region of the island."

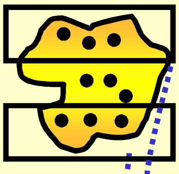
③ SELECT region, CNT(occupied)
AVG(sound)

FROM sensors

GROUP BY region

HAVING AVG(sound) > 200

~~EPOCH DURATION 10s~~



Epoch	region	CNT(…)	AVG(…)
0	North	3	360
0	South	3	520
1	North	3	370
1	South	3	520

Regions w/ AVG(sound) > 200

Using the Java API

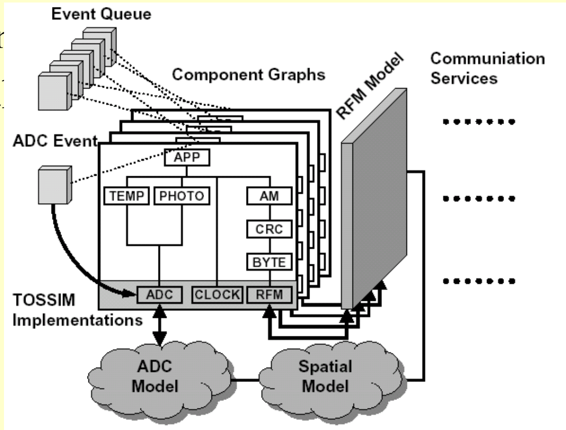
- SensorQueryer
  - translateQuery() converts TinySQL string into TinyDBQuery object
  - Static query optimization
- TinyDBNetwork
  - sendQuery() injects query into network
  - abortQuery() stops a running query
  - addResultListener() adds a ResultListener that is invoked for every QueryResult received
  - removeResultListener()
- QueryResult
  - A complete result tuple, or
  - A partial aggregate result, call mergeQueryResult() to combine partial results
- Key difference from JDBC: push vs. pull

Writing Scripts with TinyDB

- TinyDB's text interface
  - java net.tinyos.tinydb.TinyDBMain –run "select …"
  - Query results printed out to the console
  - All motes get reset each time new query is posed
- Handy for writing scripts with shell, perl, etc.

Scalable Simulation Environment - TOSSIM

- target platform: TOSSIM
  - whole application compiled for host native instruction set
  - event-driven execution mapped into event-driven simulator machinery
  - storage model mapped to thousar
- radio model and environmental
  - bit-level fidelity
- Sockets = basestation
- Complete application
  - including GUI



119

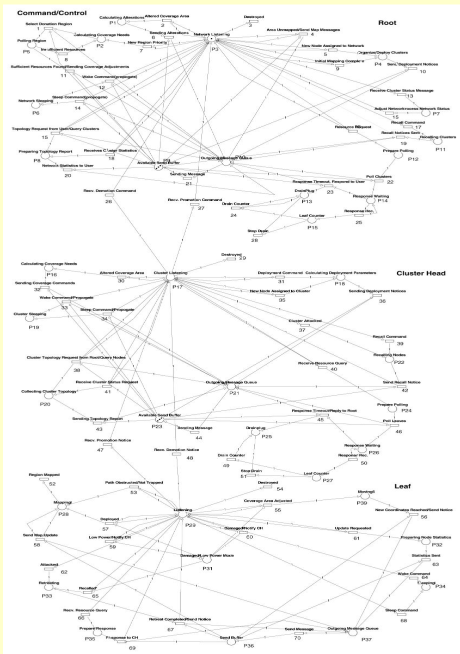
Surveillance Network Modeled by Petri Nets, Dr. Brooks, Dr. Mengxia, Dr. Iyengar and others

Design of adaptive control system for surveillance network

1.Collaborative sensing (collect data)

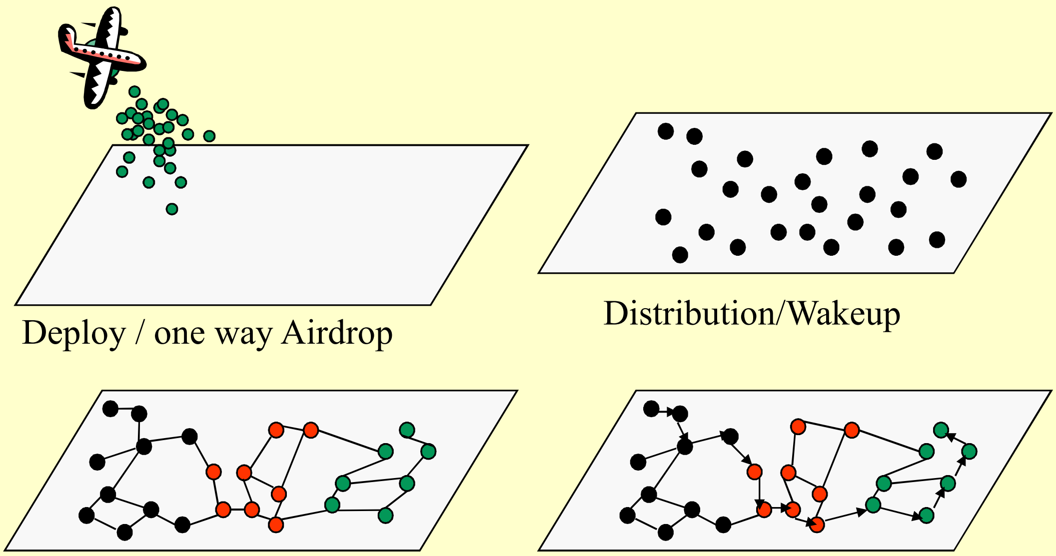
2.Network communication(Maintain Tree)

3.Operational command(User Command)



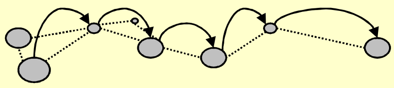
Carl Adam Petri (1962):A graphic mathematical model for describing information flow. This model is versatile in visualizing and analyzing the behavior of asynchronous, concurrent systems.

Sensor Network Initialization with flat/hierarchical Structure



Sensor Data Routing

- Routing allows a data path to be found from a source to a destination through a sequence of intermediate sensor nodes under the current network structure and traffic condition. Multi-Hop because of the communication range limit.



- **Table-Driven Routing Protocols**

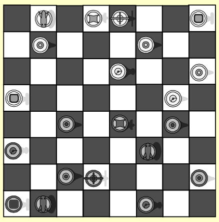
  Maintain consistent up-to-date routing information from each node to every other node in the network. Topology change is propagated throughout the network.
- **Source-Initiated On-Demand Routing**

  Routes are created only when required by source node. Start with Route discovery phase then followed by route maintenance phase until route inaccessible along path or route no longer desired.

Our New routing notion is based on cellular automata theory.

Cellular Automata Theory

- Cellular Automata were introduced in the late 1940´s by John von Neumann and Stanislaw Ulam.



Cellular Automata

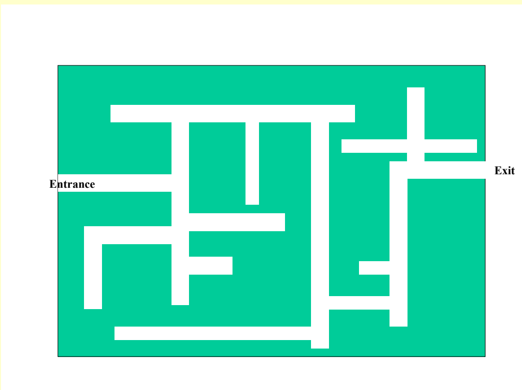
Cellular automaton views the world consisting of a population of interacting cells, each of which is a computer (automaton) with built-in rules.

Instead of describing a complex system with complex equations, but let the complexity emerge by interaction of individuals following simple rules.

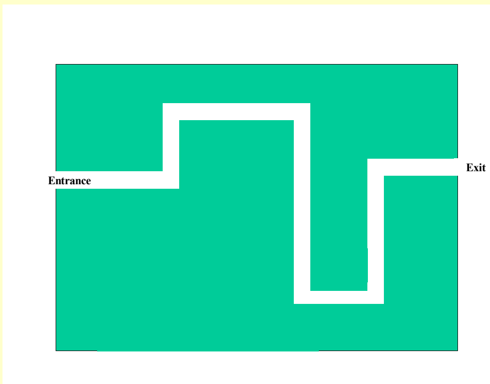
We can simulate many kinds of complex behaviors, ranging from the motion of fluids to outbreaks of starfish on a coral reef.

Maze Solving

- How to find a route from entrance to exit in a maze by CA?
- Divide grid into equally sized cells, either wall or corridor.
- Each cell has four neighbors.(east, west, south and north)
- CA rule: If three or four neighbors are wall, label it as wall.



Original Maze



Dead-ends removed

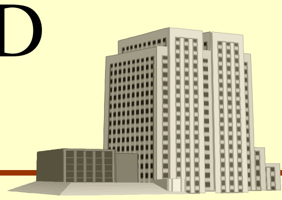
Three Cellular Automata Routing Models

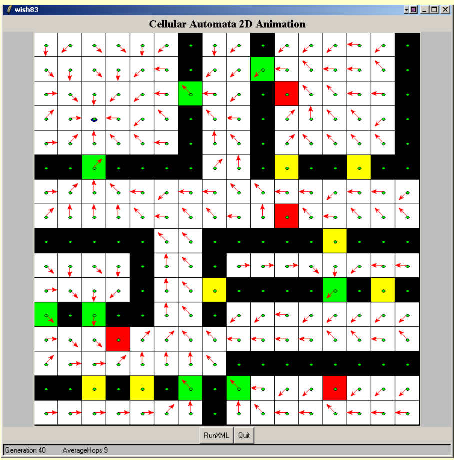
- **Spin-glass Model** Haken – Physics
  A variation of Ising model to simulate Ferro-magnetism. Every cell viewed as a small magnet, orientation of the spin depends on interaction between its local neighborhood and magnetic field.

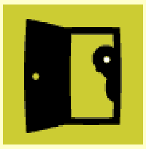
- **Multi-fractal Model** Prigogine – Chemistry
  A crystallization process initiates the growth of a routing tree stemming from the data-sink. Cell joins in the tree based on probability in terms of number of neighbors already in the tree.

- **Pheromone Model** Dorigo – Biology
  Inspired by insects colony behaviors. Ant released Pheromone and directs ant behavior in search for food. Shorter routes are found by following the strongest Pheromone trail.

Simulation Urban 2D Grid







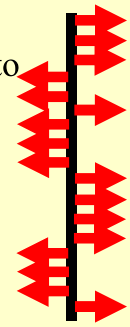
Black: Wall
Green: Open-door
Yellow: Closed-door
Red: Obstacle
White: FreeCell occupied by sensor node

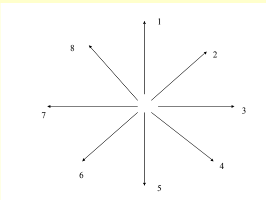
- The increasing possibility of urban combat prompts us to investigate a scenario which can represent a floor of a building and allow certain environmental changes, such as open doors, close doors, remove obstacles and place new obstacles etc.
- Weight: Represent energy cost to reach it via its neighbor.

  Default is one for free cell and open door. Infinite for wall, closed door and obstacle.

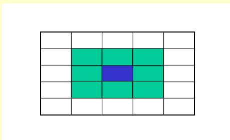
Spin-Glass Model

- Each cell(magnet) points in one of eight directions, data can be sent to any one of its eight neighbors.

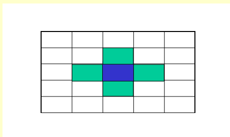




Moore Neighborhood



von Neumann Neighborhood



- A potential field defining the distance from sink is established (Lee algorithm,1961)
- Spin direction decided by potential field and kinetic factor.
- Shortest path each points to neighbor with lowest potential energy. Randomness and Error rate WSN

3	2	2	2	2
3	2	1	1	1
3	2	1	Data-sink	1
3	2	1	1	1

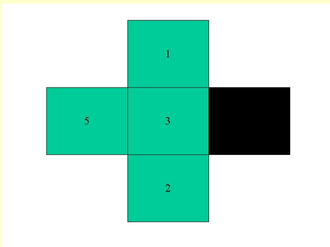
Spin-Glass Model

- Kinetic factor:Temperature tunes the balance between two physical competing factors: Energy minimization and entropy maximization
- High Temperature, Entropy maximization force dominates; Low Temperature, energy minimization force dominates.

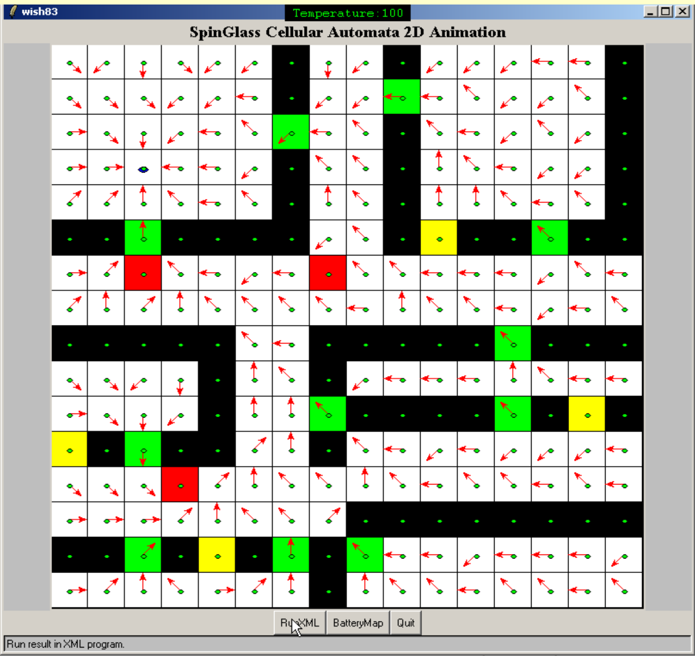
Probabilities of pointing to eight possible spin directions are given by Boltzmann probability distribution function as:

$$\frac{P[A]}{P[B]} = e^{-D/kT}$$

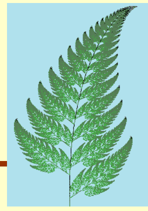
$$D = E(A) - E(B) < 0$$



Spin Glass Demo



Multi-fractal Model



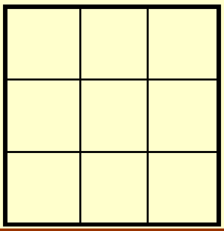
- Fractal: Benoit Mandelbrot father of fractal(1970's) a word from the latin "fractum" (broken leg). An irregular geometric object with an infinite nesting of structure at all scales. (coastline, cloud, river etc)
- fractal is self-similarity, an infinite nesting of structure on all scales and it has fractional dimension instead of integer.
- Dimension Calculation: line 1, plane 2, cube 3

D=log(#of self-similar pieces)/log(magnification factor)

- Fractal Dimension:is a measure of how complicated/rough a self-similar object is.

  mono-fractal:a single fractal dimension can characterize it.

  multi-fractal: a spectrum of fractal dimensions are needed.

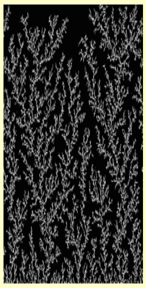


Log 9/log3=2log3/log3=2

Multi-fractal Model

- Electrophoresis: Solution of iron sulfate. Positive ions to cathode. Upon contact with the cathode, positive irons deposit and become iron molecules. And so on…

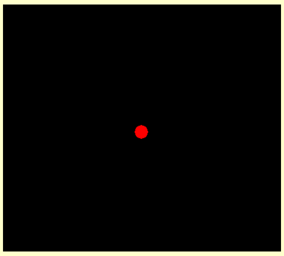
- The classic crystal-growing prototype for gas and fluid is called DLA. Diffusion Limited Aggregation

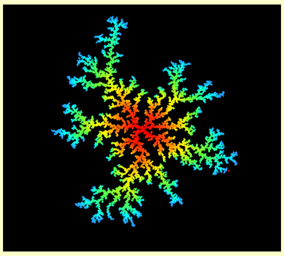


- Crystallization growth inhibition effect exerted by nearby already crystallized particles. Explained by interfacial surface tension and latent heat diffusion effect. In other words, more neighbors in crystallized state, less likely it would be crystallized.

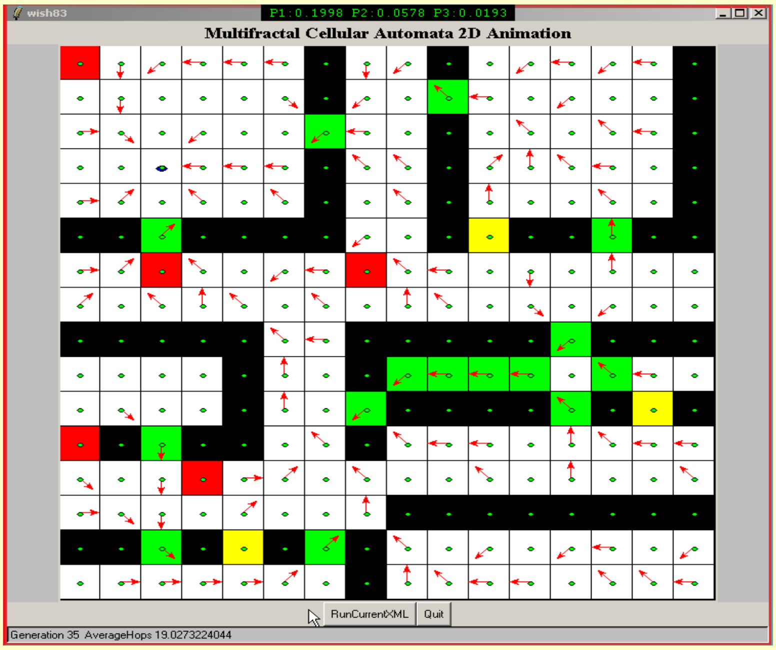
Multi-fractal Model

- Routing tree starts growing from data sink as a single seed
- Probabilities set defined for joining tree based on number of neighbors in tree. Number of neighbors in routing tree ranges from zero to eight for Moore neighborhood type.
- By specifying the numerical value the probabilities set, we can define how exactly inhibition effect increase as the number of neighbors in routing tree increases. Control the growing speed and tree structure.



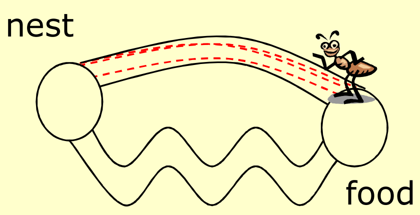


Multi-fractal Demo



Pheromone Model

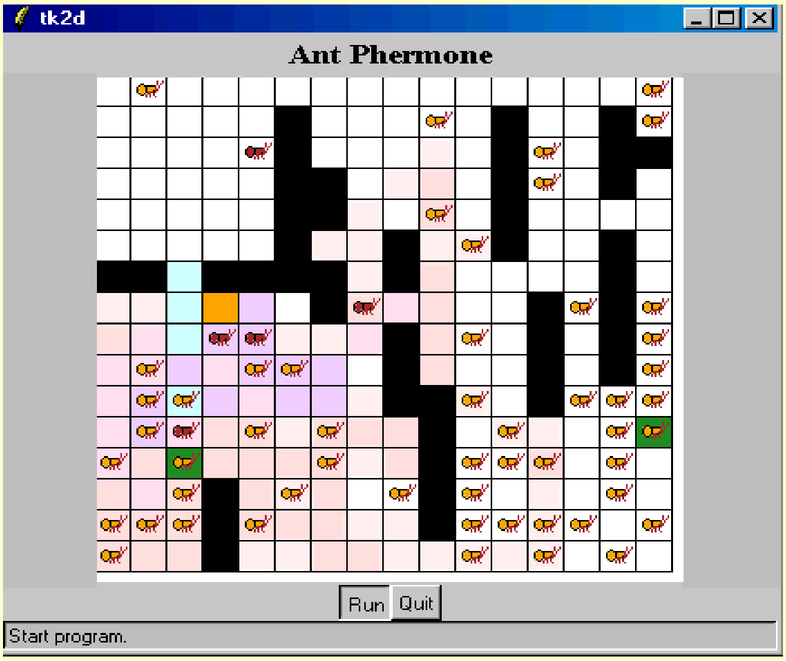
- Some of the most impressive natural self-organising systems are to be found in the world of colonial insects…
- with no central planning and very little communication complicated, coordinated, goal-directed behaviour often seems to arise spontaneously from the interactions of many simple insects.(forage food, construct nest and defeat enemy etc.)



As ants forage they deposit a trail of slowly evaporating pheromone. Those that reach the food first return before the others.

One pheromone trail is now stronger than the other, directing the ants to the food via the shorter route.

Ant Demo



Dr. Brooks, Dr. Mengxia, Dr. Iyengar

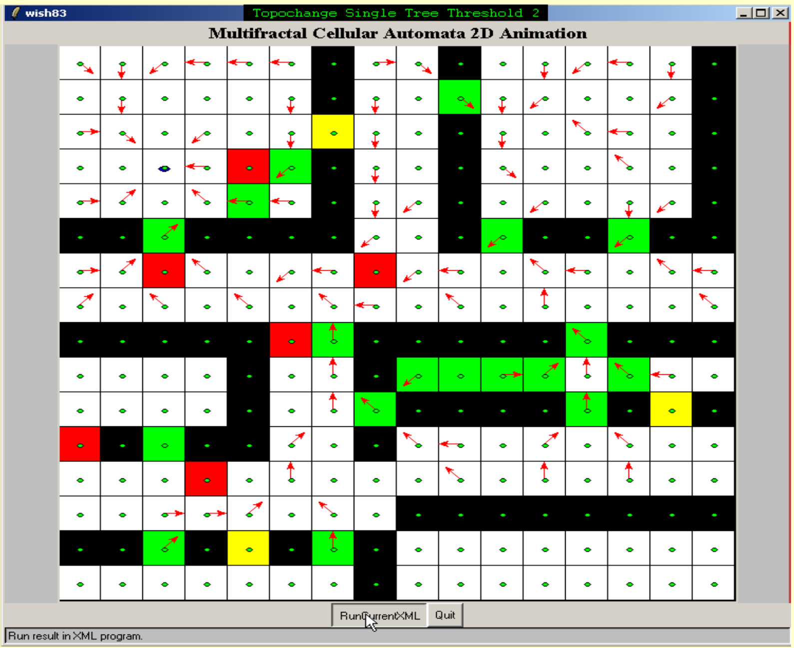
Adaptation to Topological Disturbance

- Spin Glass: A new potential field is established and new spin direction probabilities distribution is calculated.

  temperature variable tunes the system's ability to adapt. Low temperature, systems adapt slowly and dampen oscillations. High temperature, systems adapt quickly but can become unstable.

- Multi-fractal: Only pathological branches are removed, new branches substitute in response to new topology. It is based on a run and fix mode. Less effective than Spin Glass mode.

Adapt Routing

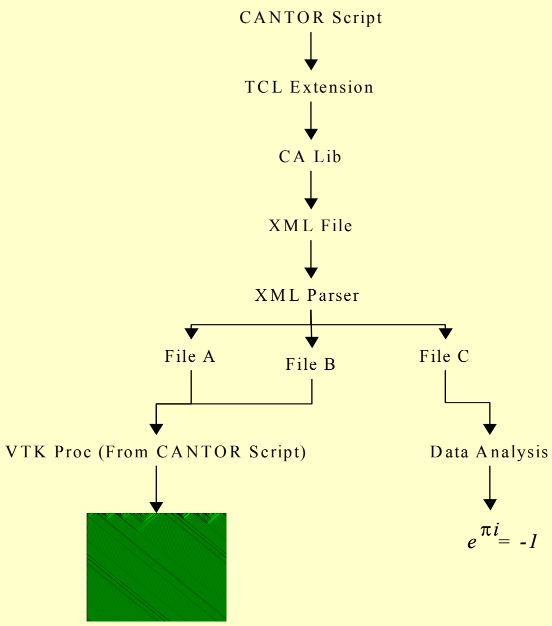


Dr. Brooks, Dr. Mengxia, Dr. Iyengar

Implementation

- The Cantor tool is a simulator based on the Generic Automata with Interacting Agents model. The tool is capable of modeling dynamic, discrete event systems using cellular automata.
- Cantor has three architecture subsystems: the Cellular Automata Library (static and dynamic), the TCL Extension and the VTk Library.
- The CA Library is written in C++ and models a cellular grid where each cell has a state, a set of neighbor cells and a set of behaviors. The TCL Extension is simply a wrapper for the CA Library. VTk Library is a series of TCL/VTk procedures and scripts designed to graphically display the evolution and interaction of cells and agents.

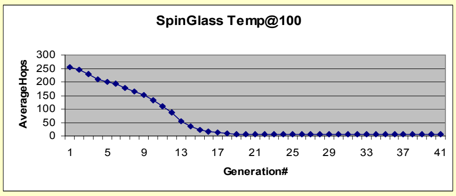
Cantor Data Flow

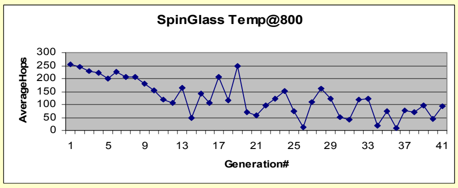


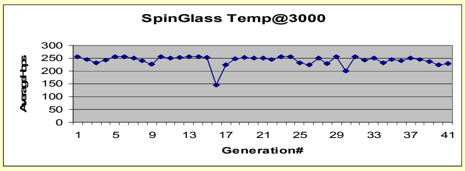
- Data flow in the Cantor modeling tool passes from the TCL script through the TCL extension to the CA Library. From there, state information is written to an XML history file that is parsed to produce data files for either a VTk visualization or data analysis.

Dr. Brooks, Dr. Mengxia, Dr. Iyengar

Comparison of Spin-Glass Result over a Range of Temperature





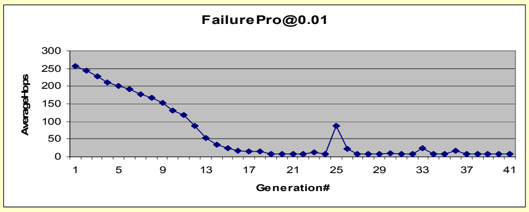


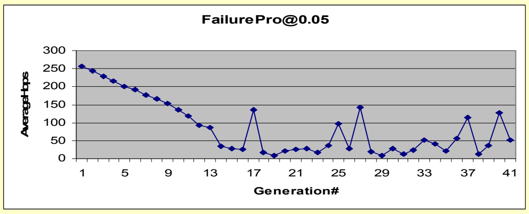
- Left-hand figures show generation number versus average-hops at different temperature in a spin-glass model.

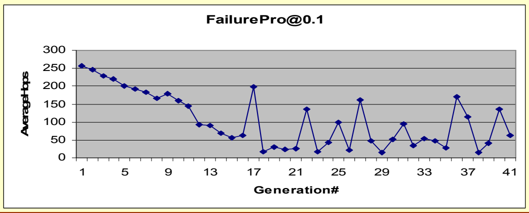
- The higher temperature, the larger the average-hops, indicating a increasing randomness in the system.

Dr. Brooks, Dr. Mengxia, Dr. Iyengar

Comparison of Results over a Range of Cell-failure probability





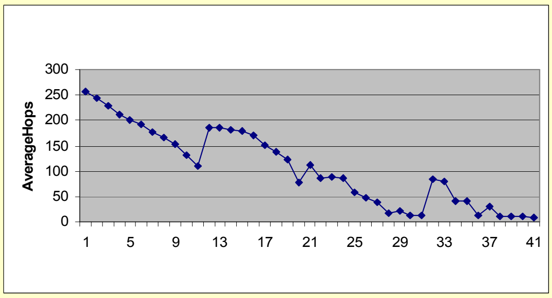


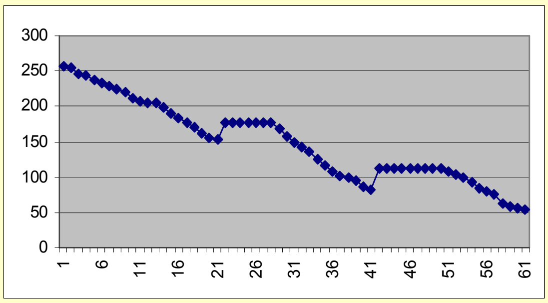
- Cell failure probability specify a probability for a cell to randomly select its direction instead of following the spin glass logic.

- The higher cell failure probability, more fluctuation in the stabilizing process.

Impact of Environmental Disturbance

- Introduce a sequence of environmental disturbance at certain generations, both system demonstrates a strong capacity of self-adapting.

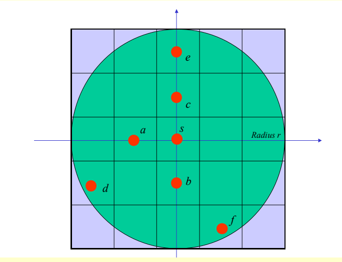


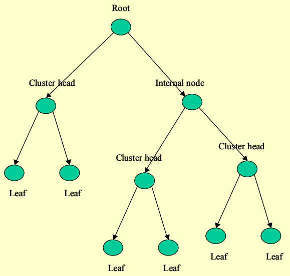


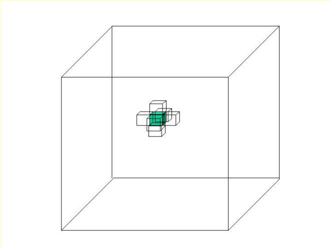
Dr. Brooks, Dr. Mengxia, Dr. Iyengar

Future Work

- Modifications need to be made to adapt to real WSN scenario. Cells may not all be occupied by sensor nodes.
- Arbitrary number of neighbors. Neighborhood table dynamically maintained by hello and acknowledge exchange. Dynamic Weight update(eavesdropping).
- Hierarchical structure with leaf node, cluster head and root
- 3-Dimensional animation







Dr. Brooks, Dr. Mengxia, Dr. Iyengar

Conclusion

- Our cellular automata routing models are distributed and self-adaptive, which come with many advantages over traditional routing approaches.
- Parallel update: All cells update their next step state simultaneously.
- Local communication: No global information is needed. Each cell updates itself solely based on its neighborhood's states and itself. Reduce the communication load and save energy and reduce network congestion.
- Homogeneity: Each cell follows the same update rule assigned to the whole grid. Make designing rules much easier.
- Adaptive: They can detect network structure changes and traffic congestion and adapt to it to achieve optimality under current condition without central supervisor.

References

1. Wireless Sensor Network Craig Ulmer George Institute of Technology

2. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks Elizabeth M. Rover. University of California, Santa Barbara

3.Deborah Estrin and Ramesh Govindan etc. " Next Century Challenges: Scalable Coordination in Sensor Networks ", USC / Information Sciences Institute

4. Nathan L. Orr Master Thesis entitled A Message-Based Taxonomy Of Mobile Code for Quantifying Network Communication August 2002

5. Richard J.Gaylord Kazume Nishidate Modeling Nature Cellular Automata Simulations with Mathematica

6. Stephen Wolfram A new Kind of Science 2002

7. C.Y. Lee, An algorithm for path connection and its application IRE Trans. Electronic Computer, vol. EC-10, 1961, pp. 346-365