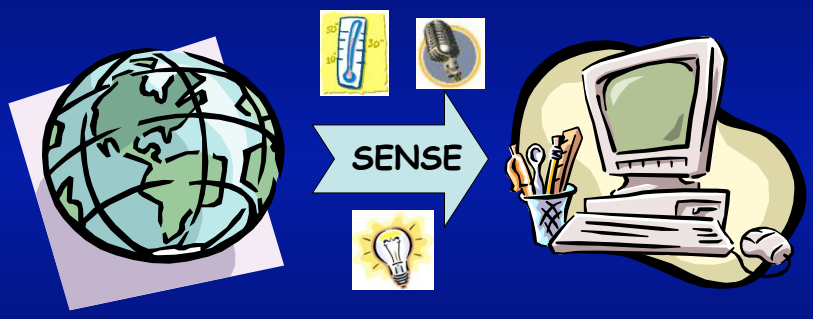
WSN Security

Javier Lopez
Computer Science Department
University of Malaga
Spain

Sensor node

FOSAD'09

Real World → Computer



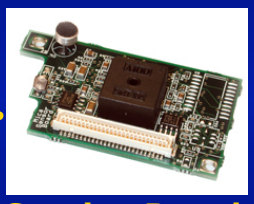
Real World

Computer World

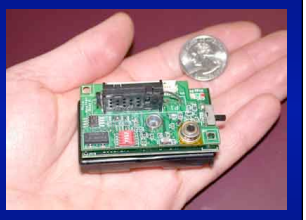
FOSAD'09

Sensor nodes

Autonomous Computer



Sensing Board

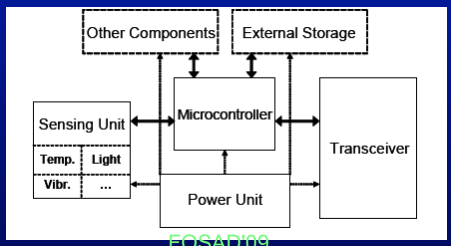


SENSOR NODE

FOSAD'09

Components of the sensor node

- A sensor node (also known as mote) is typically made up of four basic components:
  - Sensing unit: array of sensors that can measure the physical characteristics of its environment <feel>
  - Processing unit: in most cases, a microcontroller
    - can be considered as a highly constrained computer, with just the memory and interfaces necessary to create simple applications <think>
  - Transceiver: send and receive messages wirelessly <talk>
  - Power unit: provides the energy required by all components <subsist>



FOSAD'09

Components of the node: Transceiver (talking)

- One of the foundations of the sensor network paradigm is distributed collaboration, hence any node has to "converse" with other nodes
- Most of nodes have a limited energy supply, thus a transceiver has to offer:
  - an adequate balance between a low data rate (e.g. 19.2 Kbps to 250 Kbps) and a small energy consumption
  - allowing the node to live for an extended period of time
- Radio frequency communication is ideal in most of cases
  - it is not limited by line of sight
  - current technology allows implementation of low-power radio transceivers

FOSAD'09

Components of the node: Transceiver (talking)

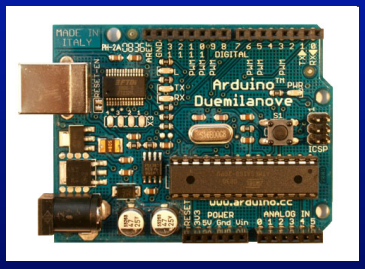
- What transceiver?
  - After the appearance in 2003 of the IEEE 802.15.4 standard for low-rate wireless personal area networks (PANs), most sensor nodes started to use transceivers that complied with this standard
- Energy consumption of the transceiver is far greater than the energy consumption of the microcontroller
  - thus sensor nodes are encouraged to do as much in-network processing as possible

Components of the node: Microcontroller (thinking)

- A sensor node use a microcontroller instead of a microprocessor
- A microcontroller is especially suitable for sensors due to its cost-effectiveness:
  - It has enough computational capabilities and memory for executing simple tasks while consuming as less energy as possible.
- What microcontroller? It depends on what has to provide to the node in terms of:
  - energy consumption
  - instructions memory and RAM memory
  - storage
  - speed
  - external I/O ports

Components of the node: Microcontroller (thinking)

- Classification of microcontrollers used in sensor nodes:
  - Class I: Very limited capabilities. Barely support the de-facto standard operating system for sensor nodes, TinyOS
  - Class II: Most common. Resource-constrained but powerful enough to run relatively complex applications
  - Class III: PDA-like capabilities. Can host complex operating systems or Java-based virtual machines



Components of the node: Microcontroller (thinking)

	Speed	RAM	ROM	Energy
Class I	4 Mhz	1 KB	4-16 KB	1.5 mA
Class II	4-8 Mhz	4-10 KB	48-128 KB	2-8 mA
Class III	13-180 Mhz	256-512 KB	4-32 MB	40 mA

- Other factors to consider when selecting a microcontroller:
  - low active current, wide operating voltage range, a 16-bit sleep timer, fast wakeup from sleep, direct memory access (DMA) channels to operate while CPU sleeps

Components of the node: Power Unit (subsisting)

- Protocols and services that run in a sensor have to take energy consumption into consideration.
  - Most class II nodes are powered by AA batteries
  - Class III sensor nodes are usually powered by high energy density batteries (e.g. based on lithium-ion).

- It is also possible to harvest energy from the environment (power scavengers)
  - Main sources of ambient energy:
    - solar (generated by sunlight or artificial light)
    - mechanical (generated by the movements of objects)
    - thermal (generated by temperature differences between two objects)

Features of specific commercial sensor nodes

- For the case of *Mica* family (*Mica2*, *Mica2dot*, *MicaZ*), and *Telos* nodes:
  - Processor:
    - 8-bit Atmel ATmega processor
    - Telos: 16-bit TI MSP430 processor
  - Memory:
    - 128 KB ROM and 4 KB RAM
    - Telos: 48 KB ROM and 10 KB RAM
  - Speed:
    - Mica2dot: 4 MHz
    - Mica2 and MicaZ: 7.37 MHz
    - Telos: 8MHz

Features of specific commercial sensor nodes

- Communications:
  - Mica2dot and Mica2 deliver up to 20 kbps on a single shared channel, with a range of up to around a hundred meters
  - MicaZ and Telos deliver up to 250 kbps.
- Software:
  - *TinyOS* operating system
    - Highly optimized (small, fast,…)
    - Support real-time tasks (multi-threaded, events-oriented)
  - C variant called *nesC* for programming purposes
    - featuring an event-driven concurrency model

Features of specific commercial sensor nodes

	Btnode 3	mica2	mica2dot	micaz	telos A	tmote sky	EYES
Manufacturer	Art of Technology	Crossbow			Imote iv		Univ. of Twente
Microcontroller	Atmel Atmega 128L				Texas Instruments MSP430		
Clock frequency	7.37 Mhz		4 MHz	7.37 MHz	8 MHz		5 MHz
RAM (KB)	64 + 180	4	4	4	2	10	2
ROM (KB)	128	128	128	128	60	48	60
Storage (KB)	4	512	512	512	256	1024	4
Radio	Chipcon CC1000 315/433/868/916 MHz 38.4 Kbauds			Chipcon CC2420 2.4 GHz 250Kbps IEEE 802.15.4			RFM TR1001868 MHz 57.6 Kbps
Max Range (m)	150-300			75-100			
Power	2 AA batteries		Coin cell	2 AA Batteries			
PC connector	Through PC-connected programming board				USB		Serial Port
OS	Nut/OS	TinyOS					PEEROS
Transducers	On acquisition board				On board		On acquisition board
Extras	+ Bluetooth radio						

Influence of components on security

- The different hardware components of the node have a great influence on security primitives and protocols
- As for the transceiver: the main influence factors are:
  - Bandwidth: the speed of the wireless channel will:
    - influence on the completion time of the security protocols
    - determine the overhead produced by confidentiality, integrity, and authentication services
  - Energy consumption:
    - if the transceiver spends too much energy sending and receiving, it is necessary to compensate by reducing both the message size and number of steps of the security protocols
  - Channel error rate:
    - reliability of the wireless channel will affect the design of the security protocols, as they must be robust against failures in the communication

Influence of components on security

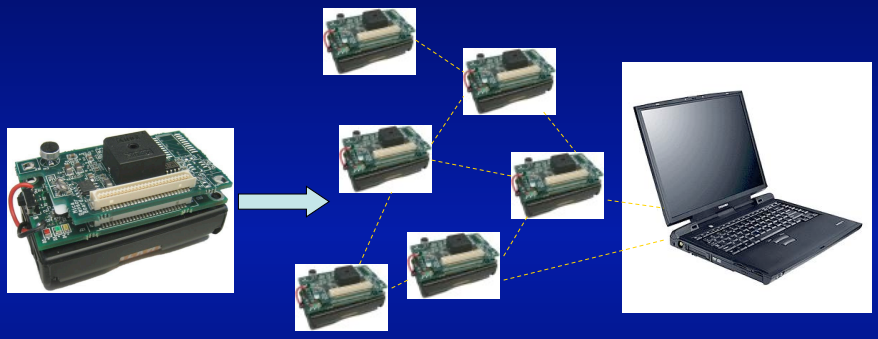
- As for the microcontroller:
  - The amount of memory dictates how many mechanisms, both security-related and application-related, can be included inside it
    - If application is too complex, little room for security mechanisms
    - If security mechanisms occupy too much space, very difficult to implement the application logic
  - Amount of memory also dictates if it is necessary to optimize the use of the security primitives
    - For instance, using AES it is possible to obtain message authentication codes through the CMAC mode of operation
  - Finally, memory is also important for holding important security data such as credentials
    - Precisely, the low amount of memory available has made very active the research field of "key management systems"

From sensors to WSN

Sensors limitations

- If sensor nodes are so constrained devices, why are they so relevant?
- Their intrinsic nature to communicate among them and create a Wireless Sensor Network (WSN), makes them one of the key technologies of the ubiquitous computing visions
- Moreover, despite the resource limitations, their tiny size makes them feasible (and, most probably, unique) for ubiquitous and real-time embedded applications
- It is precisely this combination (of certainly contradictory characteristics) what gives rise to new research challenges:
  - design of different types of communication protocols
  - development and deployment of applications and
  - specification and design of new security models and solutions

From sensor nodes to sensor networks (WSN)



(Collaboration, Event-driven processing, ...) = Distributed Applications

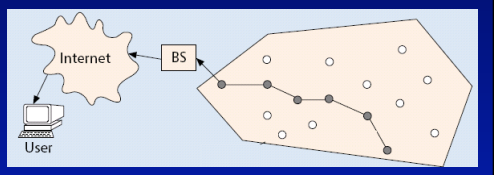
WSN basics

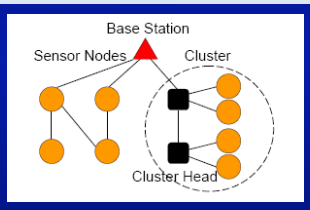
- Sensors in a WSN operate and cooperate in an ad hoc manner using their radio interfaces, resulting in a mesh architecture where nodes:
  - communicate directly only with nodes nearby due to limited power
    - some nodes communicate with a base station
  - support multiple communication paths
  - provide routing capabilities

what turns out to be an advantage in comparison with 802.11 and Bluetooth.

WSN basics

- The base station collects the data from the sensors, aggregate and send it to the outside world:
  - A central computing system where the information is stored for different purposes (analysis, control decision making, etc.)





- Contrarily to the case of the sensors, it is supposed that the base station has no limited resources
  - not only for all necessary computations but for all internal and external communications to the WSN

WSN Applications

- The evolution of sensor networks has opened a wide range of application possibilities, though WSN
  - are not especially suitable for very complex applications
  - or applications with strong demands of Quality of Service (QoS)
- Nevertheless, WSNs can be used in applications where sensors are unobtrusively embedded into systems, involving operations like:
  - monitoring
  - tracking
  - detecting
  - collecting
  - reporting

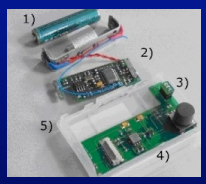
WSN Applications

- By sectors, WSNs can be used in:
  - agricultural
  - business
  - critical infrastructure protection
  - environment
  - health care
  - homeland security
  - industrial
  - military applications
  - etc.

WSN Applications

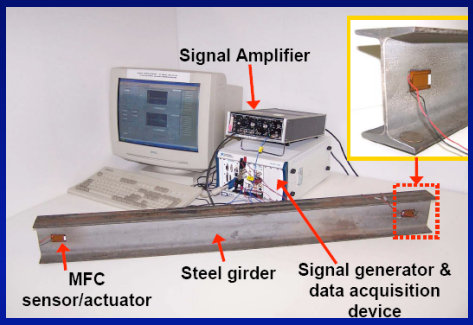
- Classification:
  - Monitoring space. The sensor network simply monitors the physical features of a certain environment.
    - environmental and habitat monitoring, precision agriculture, indoor climate control, surveillance, treaty verification, and intelligent alarms
  - Monitoring things. The sensor network controls the status of a physical entity.
    - structural monitoring, ecophysiology, condition-based equipment maintenance, medical diagnostics, and urban terrain mapping
  - Monitoring interactions. The sensor network monitors the interactions of things (both inanimate and animate) with each other and the encompassing space
    - wildlife habitats, disaster management, critical (information) infrastructure systems, emergency response, asset tracking, healthcare, and manufacturing process flow

WSN Applications



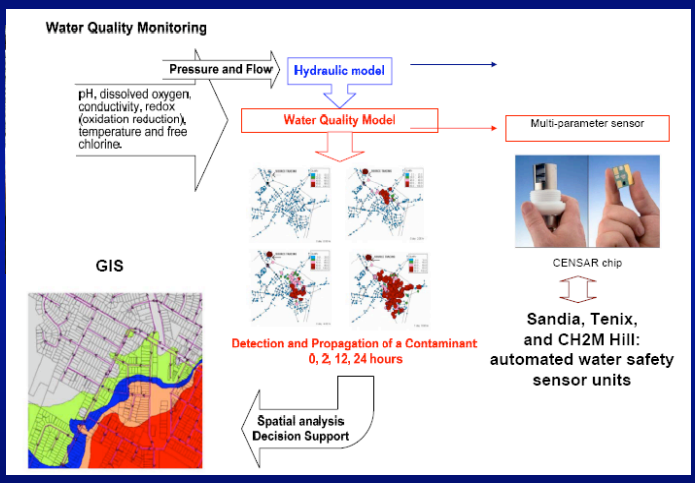
FOSAD'09

WSN Applications

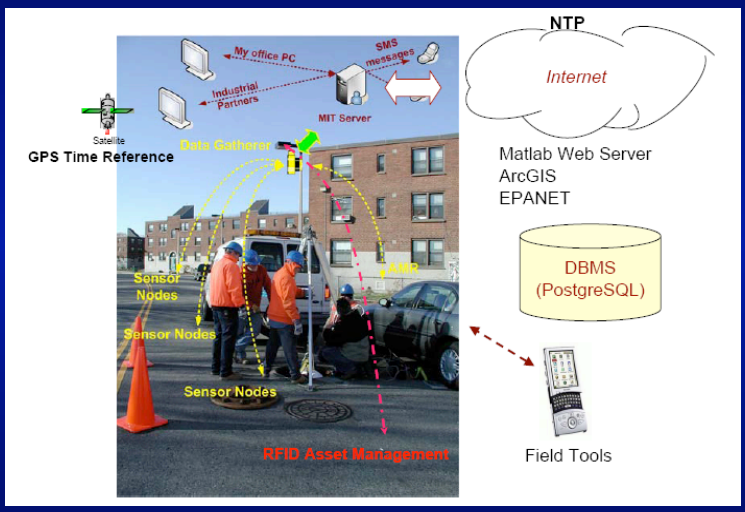


FOSAD'09

WSN Applications



WSN Applications



WSN Applications … for Internet

- Still a wide range of applications to come when sensors can globally exchange information with entities on the Internet:
  - reaching, for instance, home environments.
  - creating what already has been called “Internet of Things“ (also “tangible Internet”)

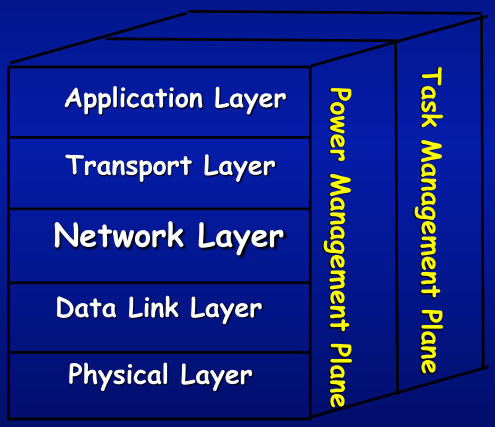
WSN Applications and Security

- Security is also influenced by the characteristics of an application and its context
  - First step: obtain the security requirements of the application and to quantify the risks and consequences of a failure in the security of the WSN
  - Second step: From requirements, it is possible to know not only what security mechanisms are needed, but also the actual importance of every mechanism
  - Third step: There are different approaches for implementing the security mechanisms but not all approaches can be optimally applied to a certain scenario
    - It is therefore necessary to choose the implementations of the security mechanisms that are more suitable for the context of a specific application

Communication Architecture

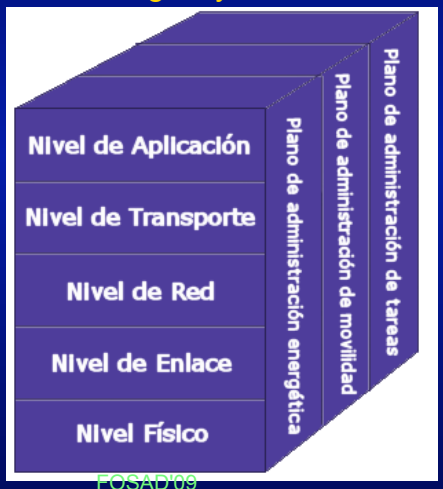
WSN Communication Architecture

- The communication architecture may be initially considered in the following way



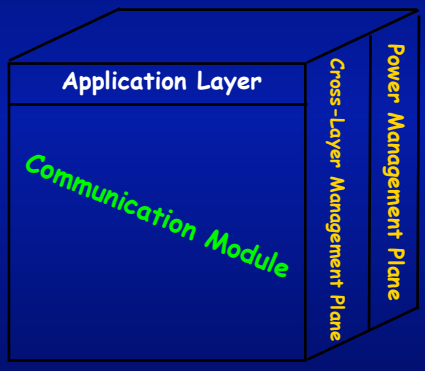
WSN Communication Architecture

- The communication architecture may be initially considered in the following way



WSN Communication Architecture

- Due to cross-layer melting, it is evolving to the following



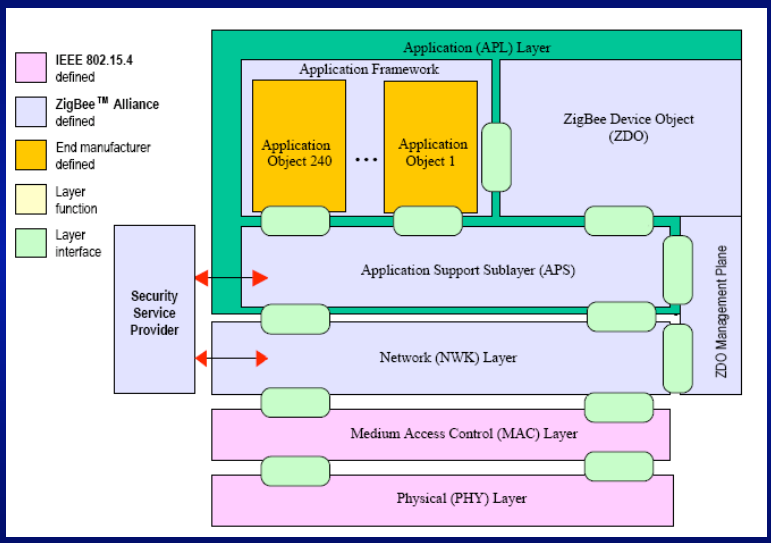
WSN Communication Architecture

- Cross-layer contributes to autonomy and self-configuration of the nodes
  - Because any component can directly access to resources and processes provided by another component
- Flexible access to information and control is convenient because of:
  - Inherently restrictions of sensors
  - Specific applications requirements

WSN Communication Architecture
The case of Zigbee

- ZigBee: Specification for WSN
  - Built upon IEEE 802.15.4
    - Standard for WPAN
    - Low energy consumption, low transmission rate (250kbps), low cost
  - Security: AES-128
- Hierarchical model
  - But with limited support to cross-layer
    - Management
    - Security

WSN Communication Architecture
The case of Zigbee



Security threats

Security concerns

- The reasons why security becomes an essential issue in WSN are:
  - sensitive nature of many of those applications
  - untrusted environment where the sensors are deployed
- Hence, a WSN must be adequately protected against threats that can affect its functionality
  - Given the role of sensor networks as a “sensory system", any disturbance may have consequences in the real world

Security concerns

- Vulnerabilities arise because of sensor intrinsic features:
  - Constrained in terms of computational capabilities, memory, communication bandwidth, and battery power
    - hence, it is challenging to implement and use the cryptographic algorithms and protocols required for security services
  - In many cases, it is easy to physically access sensor nodes
    - they are located near the physical source of the events, and once found they can be reprogrammed (no tamper-resistant) or destroyed
  - Information exchange can be intercepted
  - Difficult to monitor and control the status of the sensors due to the inherent distributed nature of the WSN
    - Any failure in any of its elements may remain unnoticed
    - A sensor network can be attacked at any physical point

Specific threats

- Denial of service attack:
  - Can range from simply jamming the sensor's communication channel to more sophisticated attacks
    - more alarming is the projected use of sensor networks in highly critical and sensitive applications
  - Simple jamming is the transmission of a radio signal that interferes with the radio frequencies being used by the sensor network
  - Retransmission of packets deplete a sensor node's power supply by forcing too many retransmissions

Specific threats

- Impersonation attack:
  - A malicious device illegitimately takes multiple identities (sibyl attack)
  - It is effective against routing algorithms, data aggregation, etc.
    - Regardless of the target, it functions similarly
  - For instance, to attack the routing protocol, the sybil attack would rely on a malicious node taking on the identity of multiple nodes, thus routing multiple paths through one malicious node

Specific threats

- Traffic Analysis:
  - For an adversary to effectively render the network useless, the attacker can simply disable the base station.
    - The base station can be identified (with high probability) without even understanding the contents of the packets (if the packets are themselves encrypted)
  - Nodes closest to the base station tend to forward more packets. An attacker need only monitor to whom a node sends its packets.

Specific threats

- Node compromise:
  - Sensor networks typically operate in hostile outdoor environments.
    - The small form factor of the sensors, and the unattended and distributed nature of their deployment, become a problem.
  - Attackers can:
    - extract cryptographic secrets,
    - tamper with the associated circuitry,
    - modify programming in the sensors,
    - replace them with malicious sensors under the control of the attacker,
    - etc.

Specific threats

- Node replication:
  - An attacker seeks to add a node to an existing sensor network by copying (replicating) the ID of an existing node.
    - Packets can be corrupted or even misrouted.
  - An attacker can copy cryptographic keys to the replicated sensor and can also insert the replicated node into strategic points in the network
    - could easily manipulate a specific segment of the network

Specific threats

- Attack against privacy:
  - Sensor networks aggravate the privacy problem because they make large volumes of information easily available through remote access.
  - Adversaries need not be physically present to maintain surveillance
    - They can gather information in a low-risk, anonymous manner.
    - Remote access also allows a single adversary to monitor multiple sites simultaneously.

Security requirements

Security Requirements

- After the overview of the potential security threats, it is possible to argue about the different security requirements for WSN applications
- Data Confidentiality
  - A sensor network should not leak sensor readings to its neighbors (especially in a military application, the data stored in the sensor node may be highly sensitive).
  - Sensor identities and public keys should also be encrypted
  - Key distribution is extremely important to build a secure channel.
- Authentication
  - The receiver needs to ensure that the data used in any decision-making process originates from the correct source

Security Requirements

- Data Integrity
  - With confidentiality, an adversary may be unable to steal information. However, it can change the data, so as to send the sensor network into disarray.
  - For example, a malicious node may add some fragments or manipulate the data within a packet, that is later sent to the original receiver.
- Data Freshness
  - It is necessary to ensure that the data is recent and that no old messages have been replayed.
  - Especially important when there are shared-key strategies employed in the design.

Security Requirements

- Availability
  - Adjusting the traditional security algorithms to fit within the WSN is not free, and will introduce some extra costs.
    - Additional computation consumes additional energy.
    - Additional communication also consumes more energy.
  - A single point of failure will be introduced if using the central point scheme, what greatly threatens the availability of the network.
- Self-Organization
  - If self-organization is lacking in a sensor network, the damage resulting from an attack or even the hazardous environment may be devastating.
  - But also self-organization is necessary to support multihop routing, and to conduct key management and building trust relations.

Security Requirements

- Time Synchronization
  - In order to conserve power, an individual sensor's radio may be turned off for periods of time.
  - Furthermore, sensors may wish to compute the end-to end delay of a packet as it travels between two pairwise sensors.
  - A more collaborative sensor network may require group synchronization for tracking applications, etc.
- Secure Localization
  - Often, the utility of a sensor network will rely on its ability to accurately and automatically locate each sensor in the network.
  - For instance, a sensor network designed to locate faults will need accurate location information in order to pinpoint the location of a fault.

Security Requirements

- Auditing
  - The elements of a sensor network must be able to store any significant events that occur inside the network.
  - As users do not operate the sensor nodes directly, but through the base station, they may not be able to know about the existence of a certain event unless the nodes store it.
  - In case of failure, auditing information can be used to analyze the behaviour of the system prior to the failure.
- Non-repudiation
  - A node can not deny sending a message previously sent
- Forward secrecy
  - A sensor should not be able to read any future message after it leaves the network
- Backward secrecy
  - A joining sensor network should not be able to read any previously transmitted message

Security Requirements

- Once that we define the security requirements of our specific application, we have to decide on the security mechanisms to use
  - We hence need security primitives as building blocks in order to create the mechanisms
- As mentioned, it is questionable if primitives traditionally used in other networking scenarios are suitable for WSN
- Cryptographic operations must be designed to minimize the use of memory.
  - Also, design of secure protocols should consider that each bit transmitted consumes as much power as executing hundreds of instructions.

Security Primitives: symmetric-key based and public-key based

Symmetric-key based

Symmetric-key based

- Implementations of security primitives for sensor nodes is a very advanced research field
- In the area of Symmetric Key Cryptography for WSN:
  - Block ciphers are more flexible and powerful
  - Stream ciphers are simpler and faster
- AES is not one of the fastest primitives, but quite used.
  - One of the most optimized software implementation of AES-128 achieves an encryption speed of 286.35 Kbps
    - a RAM requirement of 260 bytes
    - and a code size of 5160 bytes
    - running on a 8 Mhz Texas Instruments' MSP430 microcontroller
  - Skipjack is slightly less secure due to its key size (80 bits)
    - but has achieved a reasonably low encryption overhead per byte (25 microsec)
    - and a low memory overhead (code size of 2600 bytes)

Symmetric-key based

- Regarding stream ciphers, one of the most known ciphers is RC4. Very simple and high speed.
  - It needs just 428 bytes of code size, but its inherent weaknesses (mainly in the initialization phase) suggests the use of other stream ciphers
- The eSTREAM project (organized by the EU ECRYPT network) aimed to identify new stream ciphers that could be used even in constrained devices.
  - Salsa 20/12 algorithm requires 1412 bytes of code size and it provides a throughput of 43700 bytes per second
  - Sosemanuk requires more memory (9092 bytes of code size) but provides a higher throughput (67660 bytes per second)

Symmetric-key based

- In most of cases, symmetric algorithms are used after following secret keys pre-distribution procedure among sensors
  - previous to the deployment phase so that neighbouring sensors can later establish encrypted communications (hop-by-hop encryption at link level).
- In some sense, that criteria is followed because the low amount of memory precludes sensor nodes from storing a large number of keys.
- Some researchers argue about difficulties of pre-distribution, but in principle is not a major issue
  - sensors in a WSN usually belong to one domain and can be managed by the same entity before the deployment.

Symmetric-key based

- However, the number of nodes is large in many scenarios, hence:
  - end-to-end encryption becomes unrealistic
  - and affects the scalability.
- Other authors argue that physical security of the nodes is not possible
  - Thus, the protection of the key material is not guaranteed and an eventual compromise of a specific node would allow an attacker:
    - to produce encrypted datagrams and
    - decrypt the secret information directed to that node.

Symmetric-key based

- Open discussion regarding scalability issue, key distribution, key management, communication with external parties, etc.
- To some extent, the underlying problem here is the typical key management shortcomings of symmetric-key algorithms.
- The use of public key cryptography would eliminate the need for complicated protocols.
- Nonetheless, public-key cryptography, has been considered too expensive and impractical
  - because of the amount of computation required in contrast with the very limited memory and power that sensors offer.

Public-key based

Statements regarding PKC in WSN

"Many current sensor devices have limited computational power, making public-key cryptographic primitives too expensive in terms of system overhead"

Communications of the ACM, June 2004

"Public key cryptography is prohibitively expensive for sensor networks in terms of computation and energy consumption"

ACM Conference on Embedded Networked Systems, Nov. 2004

"Traditional public key cryptography is not going to work in this environment"

ISC'05 Keynote: *Security in Sensor Webs*, Sept. 2005

Symmetric vs. Asymmetric Crypto

- The challenge is to overcome the considerable computational complexity of standard public key encryption algorithms and make public key encryption possible in self powered sensor nodes.
- Solutions?

Emulation of asymmetric crypto primitives

- Protocols like SNEP and µTESLA provide secure authentication using only symmetric key techniques
- In order to provide authentication to insecure nodes µTESLA has to emulate asymmetry through a delayed disclosure of symmetric keys
- The emulation of an asymmetric cryptographic primitive requires that is each node:
  - is time synchronized with the base station
  - has key management functions
  - has ample storage

Emulation of asymmetric crypto primitives

- Keys shared among all nodes need to be updated in regular intervals
  - Requiring broadcasts from the base station to all nodes
  - As in many settings the base station can not directly communicate with all nodes, these keys need to be forwarded from node to node
  - There is a protocol overhead (increased energy consumption of the nodes) as keys and key management messages need to be transmitted frequently
- Complex key management and high storage requirements for multiple keys and messages put a considerable burden on the power consumption of the nodes.

Use of real asymmetric primitives

- RSA, El-Gamal or DSA seem to require considerable amounts of resources because they consume a lot of memory and computing time
- However, elliptic curve cryptography (ECC) based algorithms seem to suit better for wireless environments, running faster and providing equivalent security.
- ECC can reach the same level of security with smaller keys.
  - More precisely, 160 and 224-bits ECC keys provide the same level of security as 1024 and 2048-bits RSA keys, respectively.
  - Smaller keys imply benefits in processing time, storage space, bandwidth and power consumption.

Use of real asymmetric primitives

- Elaborating a little bit more on this, when trying to compare RSA and ECC, it is necessary to contrast the hard mathematical problems in which they rely on for their security.
- It is well known that the best algorithm that solves the integer factorization problem is sub-exponential
- While the best algorithm that solves the elliptic curve discrete logarithm problem is exponential.
- Because of this reason, ECC can reach the same level of security with smaller keys.
  - More precisely, 160 and 224-bits ECC keys provide the same level of security as 1024 and 2048-bits RSA keys, respectively.

Use of real asymmetric primitives

- It is obvious that smaller keys imply benefits in processing time, storage space, bandwidth and power consumption.
- A further advantage in ECC is that key sizes scale linearly, what is not the case for RSA.
  - Thus, ECC may perform even better than RSA when we want to increase the security of the system by using longer keys.
- These features have attracted recently a lot of attention on ECC as it seems particularly convenient for constrained devices, like sensors.

Optimization: hardware/architecture

- There are some experiences where cell phones, PDAs, etc. use efficient elliptic curve based algorithms which execute faster than traditional schemes
  - like RSA or ElGamal
  - while preserving the same level of security
- Gaubatz et al. consider that this comes at the price of much more complex arithmetic primitives
  - The heterogeneous structure and larger storage requirements of ECC makes it less scalable
  - And less attractive for energy efficient low-power implementations

Optimization: hardware/architecture

- Therefore, they propose a custom hardware assisted approach using:
  - Right selection of algorithms
    - Rabin's scheme
    - NTRUEncrypt
  - Right selection of associated parameter
    - Depending on the application it is possible to fix the public key to a constant value
  - Careful optimization
  - Low-power design techniques

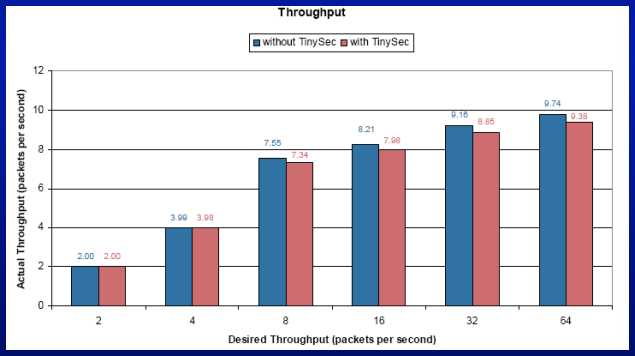
Optimization: hardware/architecture

- Two architectures, each implementing one of previous algorithms

	Rabin	Ntru
Equivalent security	60 bits	57 bits
Area (equ. gates)	16725	2850
Delay (avg. #cycles)	1440	29225
Avg. power @500kHz	148.18 µW	19.13 µW
Throughput (encrypted)	177.8 kbits/s	4.52 kbits/s

Optimization: Algorithms

- TinyOS offers MICA2 security capabilities through TinySEC (link layer security mechanism based on Skipjack)
- Impact on TinySec on MICA2's performance is reasonable



Optimization: Algorithms

- However, key distribution for applications based on MICA2 is still a problem.
- Thus, Malan et al. provided the first implementation of ECC for sensor networks, based on MICA2 mote.
  - Proved that D-H performance was slow.
  - Tried with two different ECC implementations.
    - Main result: public keys generated within 34 seconds.

FOSAD'09

Optimization: Algorithms

- However, Blaβ et al. have recently performed another implementation of asymmetric encryption and signature generation schemes for the MICA2 platform.
- The implementation is also based on elliptic curve cryptography.
  - Algorithms like Diffie-Hellman, El-Gamal and DSA based on ECC, offering the same security but less memory and computing power.

FOSAD'09

Optimization: Algorithms

- Optimization key points:
  - Saving memory by moving unchangeable data from RAM to flash-ROM or EEPROM (supported by MICA2 platform)
    - Offline precomputation and pre-deployment distribution of the constant multiplication matrix of ECC (saves 22% of RAM)
    - The same for field inversion operation (saves additional 28% of RAM)
  - Handcrafting the source code for the target platform
    - Avoiding loop checks
    - Moving outside the loops computations that do not change
    - Etc.
- Result: A total of 73KBytes of flash-ROM is permanently used for ECC operations (approx. 57%)
  - 55KByte for normal sensor code

Optimization: Algorithms

Operation	Time [s]	Malan et al. [est. s.]
Point multiplication (fixed)	6.74	~34
Point multiplication (random)	17.28	~34
Key generation	6.74	~34
Complete D-H key exchange	17.28	~68
El-Gamal encryption	24.07	~68
El-Gamal decryption	17.87	~34
ECDSA signature	6.88	~34
ECDSA verification	24.17	~68

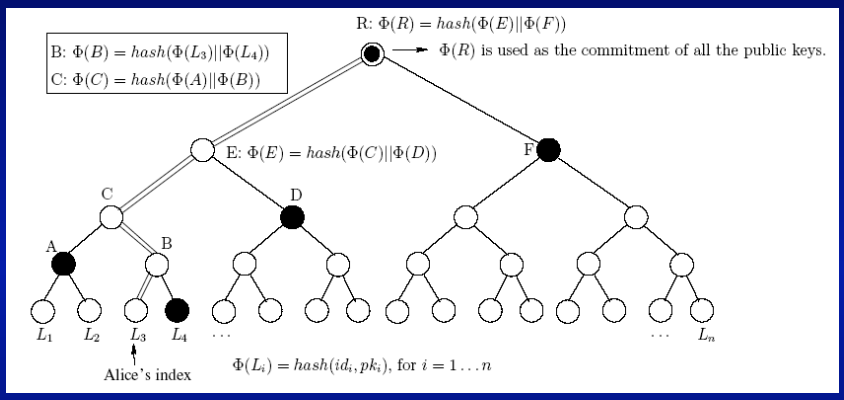
Optimization: network properties

- Du et al claim that previous results show that PKC is close to being practical in sensor nodes, but still expensive in terms of energy consumption
  - Underlying point: it is necessary to maximize the lifetime of sensors.
- Main idea:
  - Certificates are meant for user with no pre-established trust relation, but if users meet, they can interchange public keys personally.
  - Sensor nodes meet each other during the deployment phase because they usually belong to the same administrative entity, thus, they can exchange public keys securely.

Optimization: network properties

- Naive solution: Each node stores all other nodes' public keys.
  - Problem: Not enough memory on the sensor.
- Improved naive solution: Each node stores one-way hash values of all other nodes' public keys
  - Later, when A sends to B her public key, B checks that the hash value is the same one that he stores.
  - This means to replace public key authentication with symmetric key operations (using one-way hash functions).
  - Problem: Still not enough memory for large networks
- Memory-efficiency solution: Use Merkle tree technique for memory usage problem.

Optimization: network properties



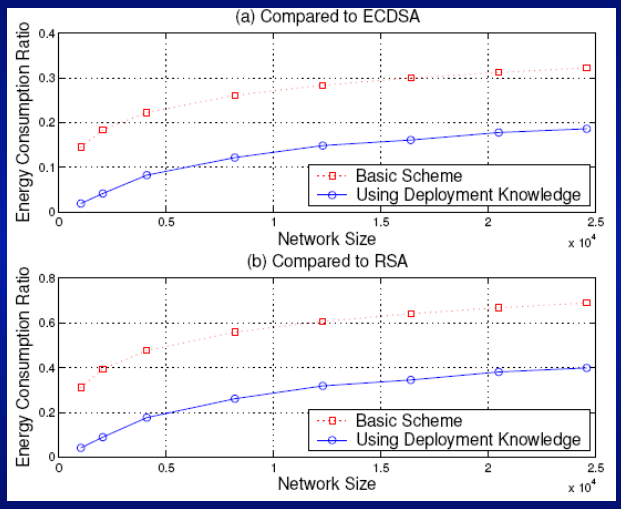
- Each leaf corresponds to a sensor node and contains the bindings between its identity and public key
- In comparison with the naive solutions, the communication overhead is increased (L x log N)

Optimization: network properties

- Deployment knowledge solution: Reduce the communication overhead by trimming down the single Merkle tree to a number of shorter trees.
- Policy:
  - If B is more likely to be A's neighbour, B should be in a shorter tree of A; otherwise, B can be put in a taller tree of A.

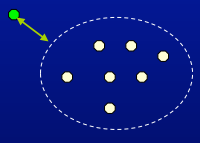
	RSA	ECC	Du et al. (SHA1)
Key or hash size	1024	160	160
Communication overhead (bit)	1024	320	160 x k
Computation time (ms)	430	1620	7.2 x k

Optimization: network properties



Optimization: PKI-like

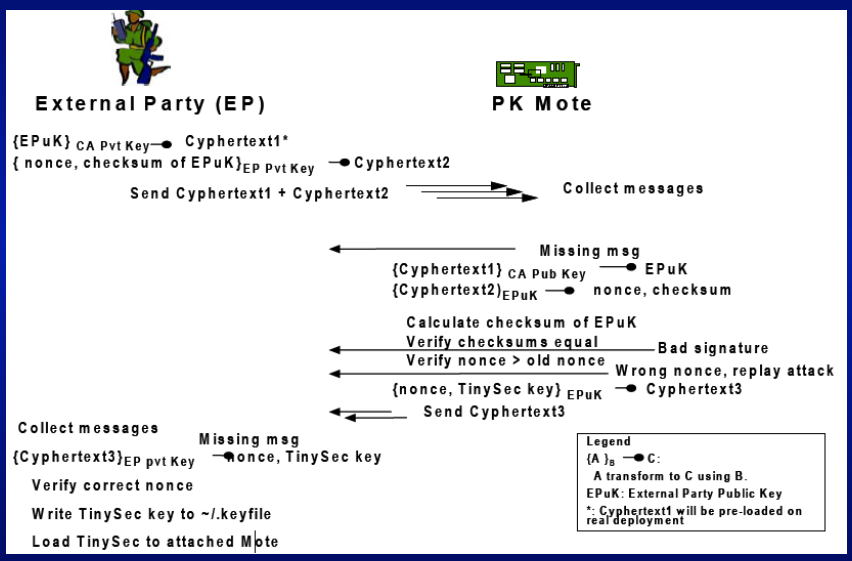
- Watro et al. propose TinyPK for authentication and key exchange between an external party and a sensor network.
- In order to make TinyPK practical, protocols require only public key operations on the sensor.
  - TinyPK is based on RSA cryptosystem, using e=3 as the public exponent.
    - The basic public operation is to cube a 1024-bit number and to take its residue modulo a large prime.



Optimization: PKI-like

- TinyPK requires a Certification Authority.
- Every node is pre-installed the CA's public key.
- Any external party that wishes to interact with the nodes also requires its own public/private key pair
  - And must have its public key signed by the CA's private key, thus establishing its identity.
- The scheme does not make use of certificates because nodes are assumed to not have enough processing power to make use of certificates
  - No real-time access to the CA infrastructure.
  - No revocation issues
- Protocol based on challenge-response

Optimization: PKI-like



Optimization: PKI-like

- Implemented on MICA1 Motes.
  - Microcontroller running at 4 MHz, with 4KB of RAM, and 128KB of flash memory.
- Implementation using TinyOS development environment and the NesC programming language.

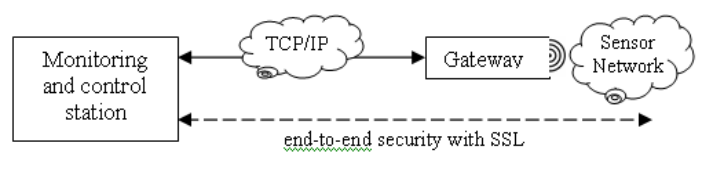
RSA Key Size	Time (sec)
512	3.8
768	8.0
1024	14.5

Optimization: End-to-End Security

- Gupta et al. developed Sizzle ("Slim SSL"):
  - A secure web server stack that runs on the Mica2dot mote.
- Goal: embed a secure web server in an array of tiny devices while using a web browser as the monitoring/ controlling application.
- Scenarios proposed range from home appliances to personal medical devices, where monitorization is done via Internet.

Optimization: End-to-End Security

- Devices in the WSN are connected via a gateway.
- The secure web server within each device of the WSN is mapped to different TCP ports at such gateway
  - from where access to the sensor nodes is controlled.
- The connection from the gateway to the nodes uses a special purpose simple and reliable protocol



Optimization: End-to-End Security

- Based on highly optimized, assembly language implementations of PKC
  - and integrates ECDH and ECDSA in SSL.
- Uses a persistent HTTP connection
  - keeps the TCP connection open for a configurable duration so that other arriving requests are serviced in the same connection.
    - saves CPU time and memory,
    - reduces network congestion,
    - improves response time
- Makes use of an abbreviated SSL handshake

Recent results

- Recent advances have focused more and more on ECC.
- One of the most well known software implementations of ECC, TinyECC, implements ECC-based signature generation and verification (ECDSA), encryption and decryption (ECIES), and key agreement (ECDH).
- The computational and memory requirements of these algorithms are not small
  - ECDSA requires 19308K ROM and 1510K RAM for the MICAz, generating a signature in 2s. and verifying it in 2.43s
- Improvements on the implementations of ECC primitives have allowed the existence of more complex PKC primitives in sensor nodes
  - such as identity-based cryptography (IBC)

Defensive measures

Defensive measures

- Defending against DoS attacks
  - Identify the jammed part of the WSN and route around.
- Secure broadcasting and multicasting
  - Based on encryption techniques and key management techniques.
- Defending against attacks on Routing Protocols
  - For instance, employing redundancy. Multiple identical messages are routed between the source and the destination (supported by an authentication scheme).
- Defending against the Sybil attack
  - For instance, by using a trusted node that validates identity of the other nodes.

Defensive measures

- Detecting node replication
  - Randomized multicast and line-selected multicast
- Defending against attacks on sensor privacy
  - Anonymity mechanisms to protect location information
- ….

More Security Issues

Routing

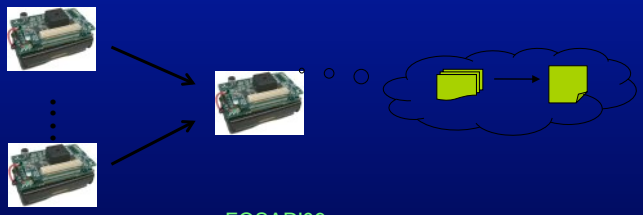
- Maximum transmission distance of current generation of sensor nodes ranges between 100 and 300 mts
  - Thus, messages can not be transmitted directly between any two nodes
  - A routing infrastructure is needed
- Algorithms should work:
  - Even when nodes start to fail due to energy issues
  - With any network size and node density
  - Providing a certain quality of service
  - Minimizing the memory usage, speed and energy consumption
- And Security must be considered!!!

Secure Routing

- Key infrastructure may help in the defense by authenticating nodes and protecting the routing infrastructure, but this is not enough:
  - Malicious nodes and denial of service still possible
- It is essential to make the routing algorithm robust against attacks
- Some work that focus on protection of existing routing protocols
- Others focus on designing new protection techniques
- Challenge: (almost) no protocols with security in mind from scratch!

Secure Aggregation

- Main purpose of Sensor networks: Send data to users
  - Large amounts of raw data
  - Dense networks => Redundant data
- Costly! (energy, time,…). Solution: Aggregate (summarize) data
  - (Data, Data, … , Data) → Report
- Who? Aggregators (Cluster heads, Special nodes,…)



Secure Aggregation

- Aggregation is prone to be attacked
  - Normal
    - Data injection, Data integrity
  - Internal adversaries
    - False Data (Nodes)
    - False Reports (Aggregator)
    - Data on Transit (Routing)

Auditing

- User/Admin can only access to Base Station (directly or not)
  - Base station only collects data from nodes
  - Impossible to know, for instance, state of the nodes (energy!)
- Solution: Audit subsystem
  - Able to inform about the internal state of a node/group
- Based on audit information: Intrusion Detection Systems
  - IDS: Monitor network, detects problematic situations, alerts users
  - Tools: Anomaly detection, Misuse detection
- Challenge: Provide IDS solutions

Privacy

- Two types of privacy
  - Network Privacy
    - Privacy of the network itself (nodes, information)
    - Sometimes important (battlefield), sometimes not (earthquake)
  - Social Privacy
    - Privacy of the subjects under surveillance

Privacy

- Threats to network privacy
  - Content Privacy
    - Meaning of a communication exchange? Messages, Context
  - Identity Privacy
    - Deduce identities of nodes in a communication
  - Location Privacy
    - Infer (or approximate) physical position of node
- Nodes will get smaller, cheaper…
  - Easy to create "surveillance" network
  - Get data about subjects at a "safe" distance
  - Automatic data collection, analysis and event correlation!

Other Issues

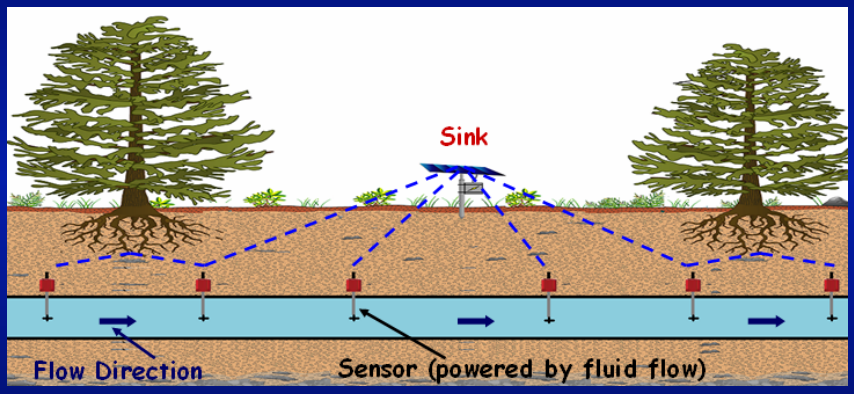
- Mobile Agents
  - Could be useful on a Sensor Network context
  - Constrained environment, no protection
- Delegation between the Base Station and the Sensor Nodes
  - All previous cases: static environments
- Automatic reaction against external/internal problems
  - Denial of Services attacks
- Challenges: All above

Further scenarios

Underground sensor networks

FOSAD'09

Underground sensor networks



FOSAD'09

Underwater sensor networks

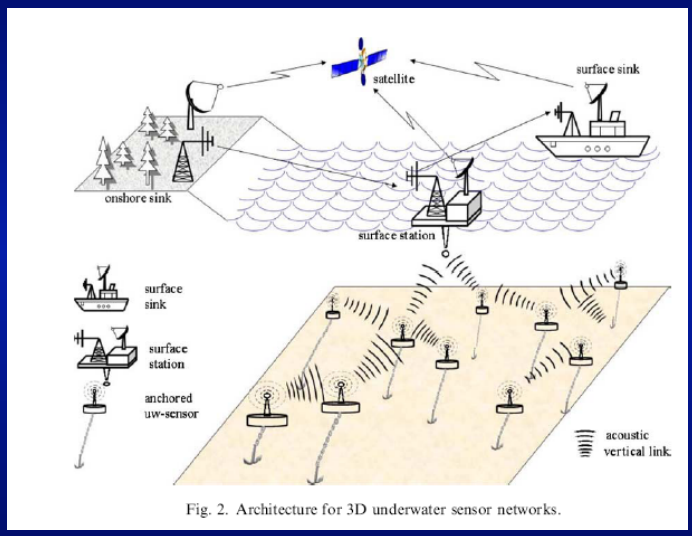
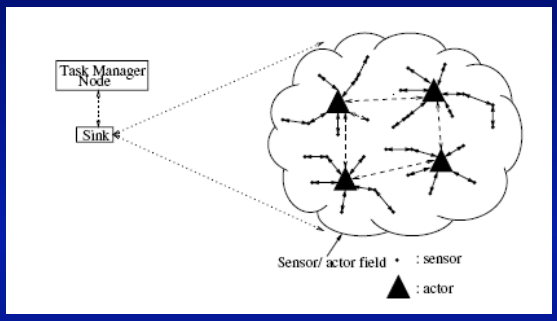


Fig. 2. Architecture for 3D underwater sensor networks.

FOSAD'09

Wireless Sensor and Actuator Networks

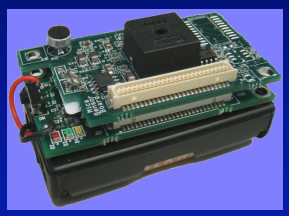


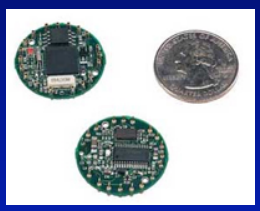
FOSAD'09

Final remark

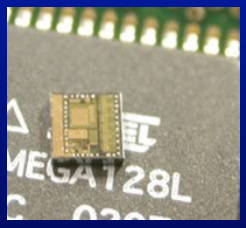
FOSAD'09

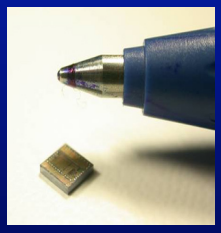
Final remark





FOSAD'09





FOSAD'09

Thanks for your attention!

Javier Lopez
Computer Science Department
University of Malaga
Spain

jlm@lcc.uma.es

FOSAD'09