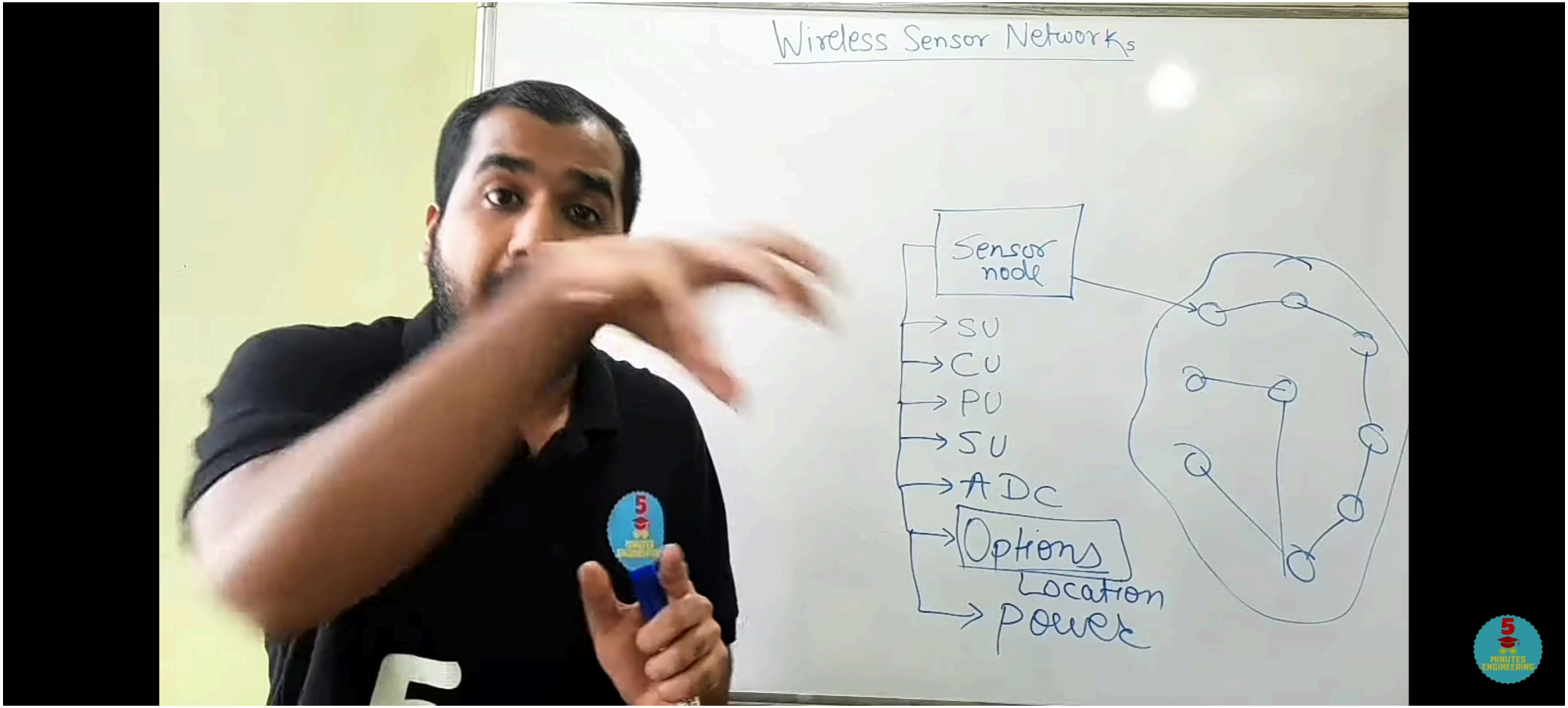

Wireless Sensor Networks
Sensor node
SU
CU
PU
SU
ADC
Options
Location
Power

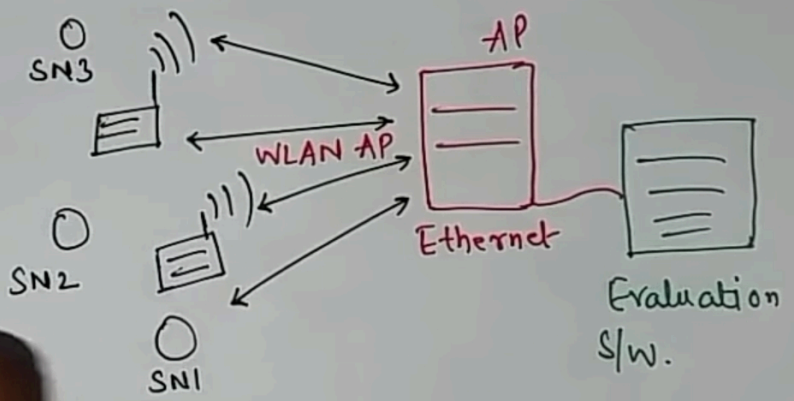
Wireless Sensor N/w (WSN):

Components:

(i) Sensors: They capture the measured variable in a data acquisition N/w. For further proce... sensor signal is converted into ...

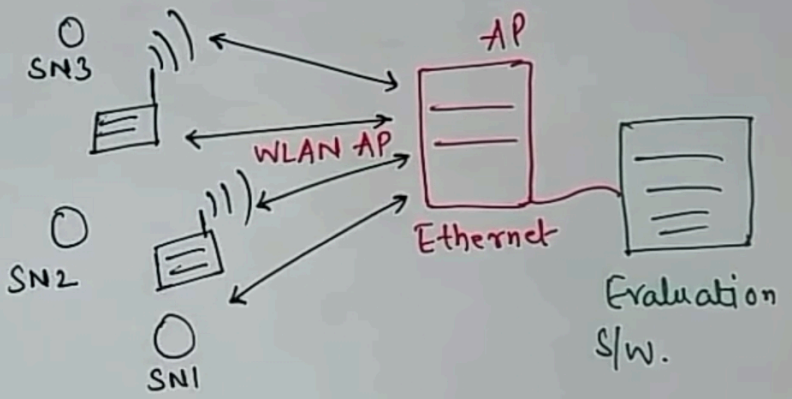
(ii) wireless Senso... They receive ... Sensors.

(iii) WLAN Acces... which are ... wirelessly.



(iv) Evaluation S/w: for any data analysis, WLAN access point is connected to an evaluation unit.

Wireless Sensor N/W (WSN):



Components:

(i) **Sensors**: They capture the measured variable in a data acquisition N/W. For further processing sensor signal is converted into electrical signal.

(ii) **Wireless Sensor nodes or Radio nodes**: They receive the Sensor data from Sensors.

(iii) **WLAN Access Point**: Receives sensor data which are transferred by sensor nodes wirelessly.

(iv) **Evaluation S/W**: For any data analysis, WLAN access point is connected to an evaluation unit.

Components of a wireless Sensor Node:

(i) MicroController: This is a Computer-on-a chip which is very small in size.

↳ Capable of doing powerful tasks including Controlling the func^n of other devices Connecting to it.

MicroController → Microprocessor
→ RAM Memory
→ Associated Peripherals.

(ii) Transreceiver: This is transmitter-receiver that is used for comm^n purposes to send and receive data, and commands.

(iv) External Memory: (WSN) nodes usually use flash memories.

- Small in Size
- Reasonable storage Capacity.

(v) Power Source: Power is Stored in the form of batteries.

Adhoc Wireless N/W Vs. Sensor N/W:

i) No. of nodes in Sensor N/W can be more as compared to adhoc N/W.

ii) Sensor Nodes are more prone to failure and energy drain.

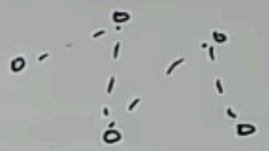
iii) Sensor Nodes → Data Centric

Adhoc N/W → Address Centric

iv) Most of the adhoc n/w routing Protocols can't be implemented to Sensor N/W.

Issues and Challenges in designing Sensor N/W:

(i) No Regular topology.



(ii) Infrastructure less.

(iii) Resource Constraints → Energy
→ bandwidth.

(iv) H/W should be energy efficient.
↳ micro Controller.

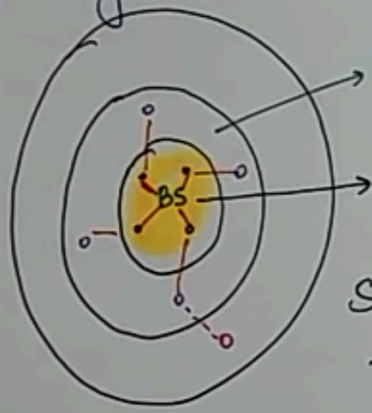
(v) Security Issues. → Enemy territory.
→ No Central tower.

(vi) Adaptible to changing Connectivity

Synchronization.

Sensor N|w Architecture:

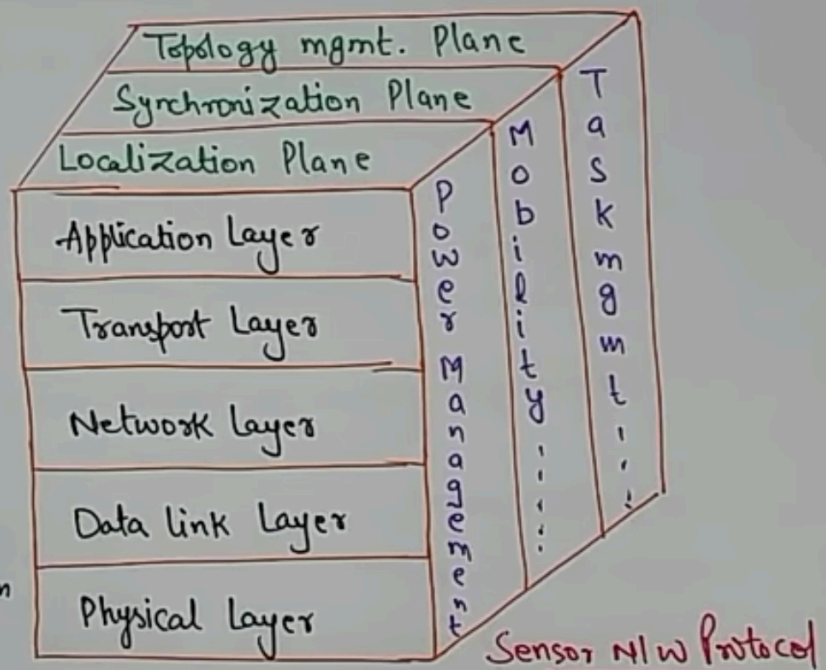
(i) Layered N|w Architecture: It includes five Layers and three cross Layers.



Two hop Layer

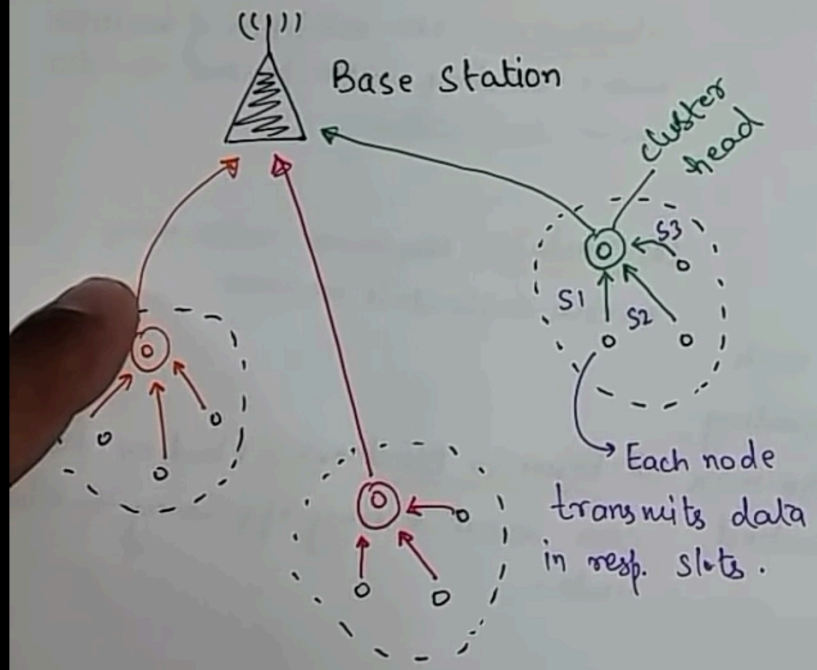
One hop Layer

Scalable,
fault tolerance,
Power Consumption is less



Sensor N|w Protocol Task

(ii) Clustered N/W Architecture:



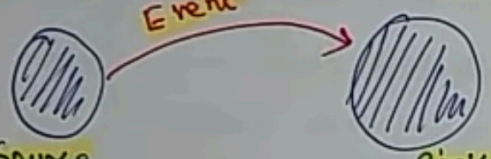
LEACH Protocol: Low Energy Adaptive clustering hierarchy.

- ↳ 2-tier hierarchy clustering architecture
- ↳ distributed algo - to organize the Sensor nodes into clusters.
- ↳ cluster head nodes create TDMA Schedules.
- ↳ Energy Efficiency = Data fusion.

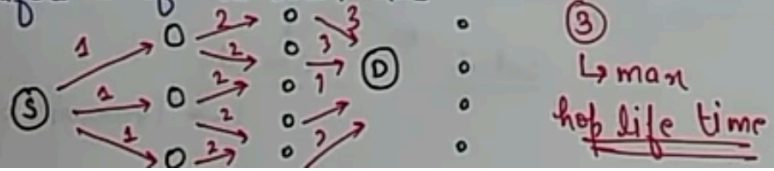
LEACH

- Setup
- Steady
  - ↳ Transmiss

Data Dissemination: It is the process by which queries or data are routed in the Sensor N/w. → Info" to be reported/ sent.

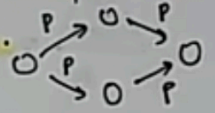


FLOODING: Simplest Design. In this each node receiving data repeats it by broadcasting the data to every neighbour unless the max hop lifetime of the data has been reached.

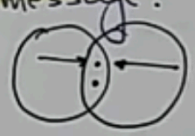


Shortcomings of Flooding:

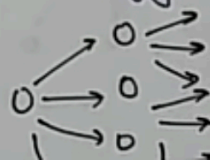
(i) **Implosion**:- No retriction of multiple nodes sending same packet to the same destination.



(ii) **Overlap**:- Neighbour nodes may receive duplicated message.
↳ overlapping regions.



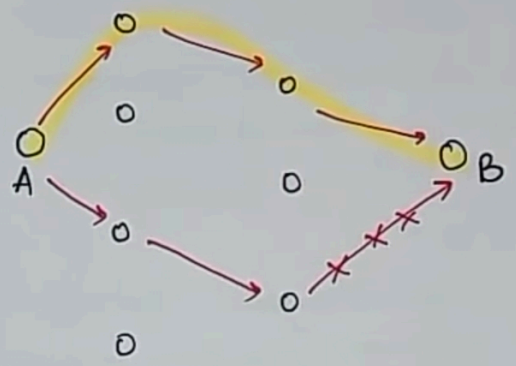
(iii) **Resource Blindness**:- Flooding doesn't care about energy efficiency of the nodes.



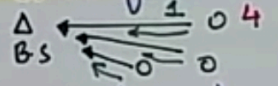
GOSSIPING: It is the enhancement of Flooding. In this, when a node receives data, it randomly chooses a neighbour and sends the data to it.

↳ It avoids the problem of Implosion,

↳ Contributes to the latency of N/w.



Data Gathering in Sensor N/W:- Objective is to transmit the sensed data from each Sensor node to a BS.



One Round = BS Collecting data from all sensor nodes once.

Algorithm Goal = Maximize the no. of rounds before node dies and N/W becomes inoperable.

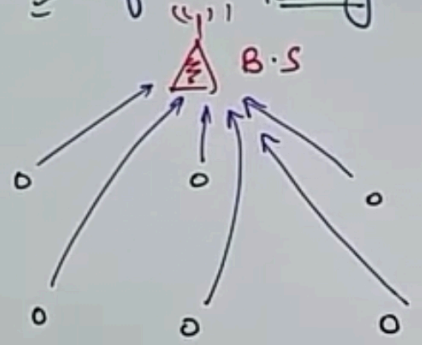
↳ Minimize Energy and reduce delay.

(i) **Direct Transmission:** All the Sensor nodes transmits their data directly to the BS.

↳ expensive in terms of energy Consumed. (Energy ↑)

↳ Media Delay is also large

↳ Performs poorly.



(ii) Power-Efficient Gathering for Sensor Info^n Systems: (PEGASIS)

Enhancement over (LEACH) protocol.

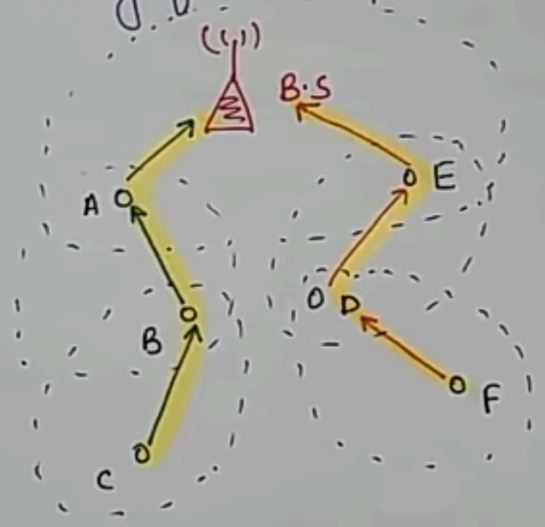
↳ Based on the assumption that all the Sensor nodes know the loc^n of every other node.

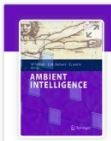
GOALS of PEGASIS:

↳ minimize node tr^n distance.
↳ Lower overhead of broadcasting.
↳ minimize no. of messages that need to be Sent to the BS.
↳ Equal distribution of energy Consumption across all nodes.

Data Gathering with Pegasis:

↳ Greedy algo is used.
→ Chain of Sensor nodes is Constructed Starting from the farthest node from BS.





Ambient Intelligence pp 115–148 | Cite as

TinyOS: An Operating System for Sensor Networks

P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer & D. Culler

Chapter

2806 Accesses | 572 Citations | 6 Altmetric

Abstract

We present TinyOS, a flexible, application-specific operating system for sensor networks, which form a core component of ambient intelligence systems. Sensor networks consist of (potentially) thousands of tiny, low-power nodes, each of which execute concurrent, reactive programs that must operate with severe memory and power constraints. The sensor network challenges of limited resources, event-centric concurrent applications, and low-power operation drive the design of TinyOS. Our solution combines flexible, fine-grain components with an execution model that supports complex yet safe concurrent operations. TinyOS meets these challenges well and has become the platform of choice for sensor network research; it is in use by over a hundred groups worldwide, and supports a broad range of applications and research topics. We provide a qualitative and quantitative evaluation of the system, showing that it supports complex, concurrent programs with very low memory requirements (many applications fit within 16KB of memory, and the core OS is 400 bytes) and efficient, low-power operation.We present our experiences with TinyOS as a platform for sensor network innovation and applications.

Keywords

Sensor Network | Wireless Sensor Network | Power Management | Execution Model | Data Race

These keywords were added by machine and not by the authors. This process is experimental and the keywords may be updated as the learning algorithm improves.

Network Layer

This layer is used to send datagrams from the source network to the destination network. we use IPv4 and IPv6 protocols as host identification that transfers data in packets.

IPv4

This is a protocol address that is a unique and numerical label assigned to each device connected to the network. an IP address performs two main functions host and location addressing. IPv4 is an IP address that is 32-bit long.

IPv6

It is a successor of IPv4 that uses 128 bits for an IP address. it is developed by the IETF task force to deal with long-anticipated problems.

Link Layer

Link-layer protocols are used to send data over the network's physical layer. it also determines how the packets are coded and signaled by the devices.

Ethernet

IPv6

It is a successor of IPv4 that uses 128 bits for an IP address. it is developed by the IETF task force to deal with long-anticipated problems.

Link Layer

Link-layer protocols are used to send data over the network's physical layer. it also determines how the packets are coded and signaled by the devices.

Ethernet

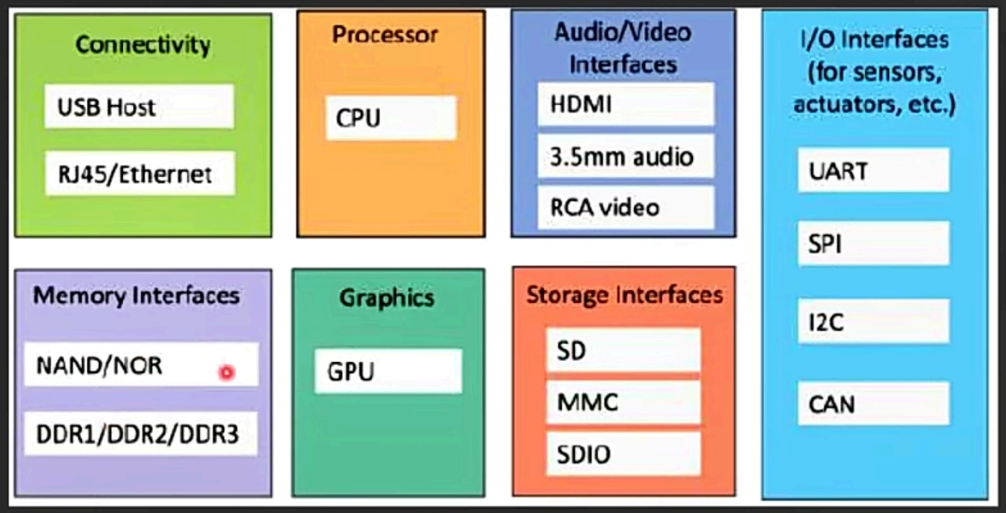
It is a set of technologies and protocols that are used primarily in LANs. it defines the physical layer and the medium access control for wired ethernet networks.

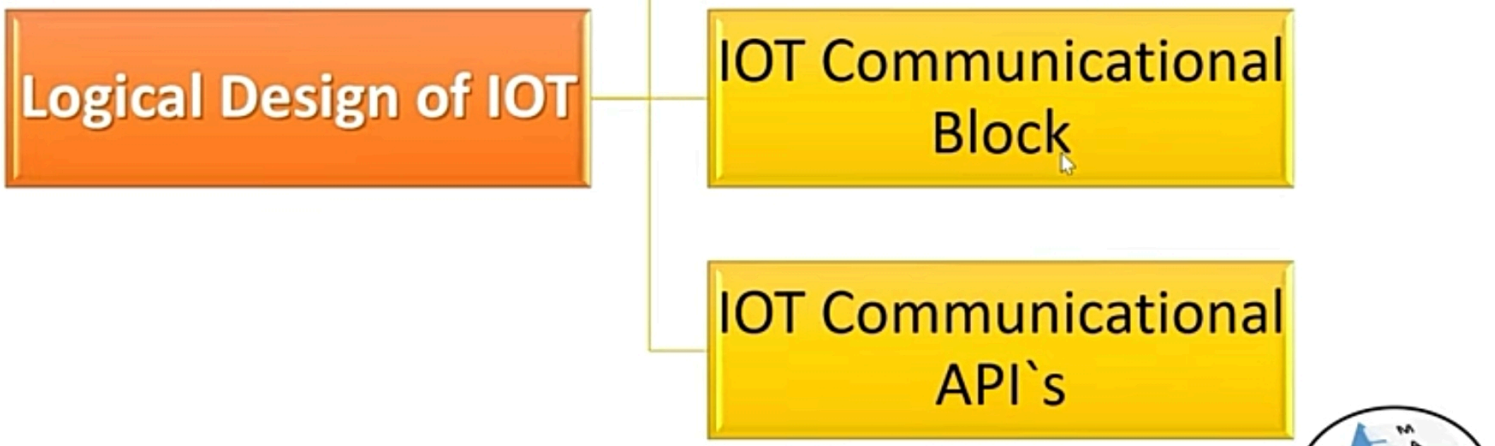
WiFi

It is a set of LAN protocols and specifies the set of media access control and physical layer protocols for implementing wireless local area networks.

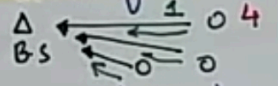
Read other IoT tutorials

Physical Design of IoT - Generic Block Diagram




Logical Design of IOT
IOT Functional Bloc
IOT Communicational Block
IOT Communicational API`s

Data Gathering in Sensor N/W:- Objective is to transmit the sensed data from each Sensor node to a BS.



One Round = BS Collecting data from all Sensor nodes once.

Algorithm Goal = Maximize the no. of rounds before node dies and N/W becomes inoperable.

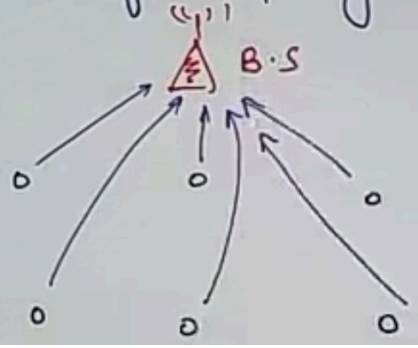
↳ Minimize Energy and reduce delay.

(i) **Direct Transmission**: All the Sensor nodes transmits their data directly to the BS.

↳ expensive in terms of energy Consumed. (Energy ↑)

↳ Media Delay is also large

↳ Performs poorly.



(ii) Power-Efficient Gathering for Sensor Info^n Systems: (PEGASIS)

Enhancement over (LEACH) protocol.

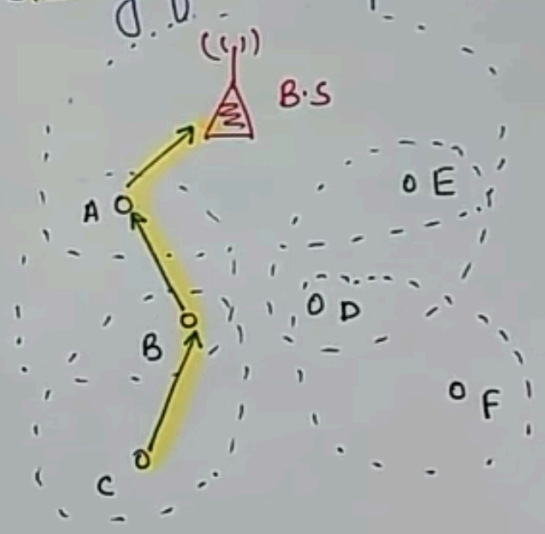
↳ Based on the assumption that all the Sensor nodes know the loc^n of every other node.

GOALS of PEGASIS:

↳ minimize node tr^n distance.
↳ Lower overhead of broadcasting.
↳ minimize no. of messages that need to be Sent to the BS.
↳ Equal distribution of energy Consumption across all nodes.

Data Gathering with Pegasis:

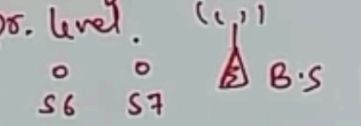
↳ Greedy algo is used.
→ Chain of Sensor nodes is Constructed Starting from the farthest node from BS.



(iii) Binary Scheme: Chain-based Scheme, (iv) Chain-Based Three Level Scheme:

...ich nodes are classified into ...nt levels.

...ot applicable for non-CDMA Sensor ...odes. At each level no. of nodes are reduced from pr. level.



S1 → S2 S3 → S4 S5 → S6 S7 B.S

(L2) S2 → S4 S6 → S7

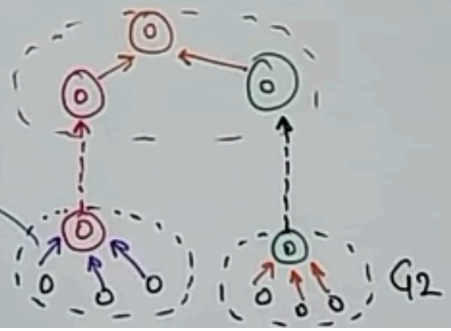
(L3) S4 → S7

(L4) S7 → B.S

Each node transmits alone.

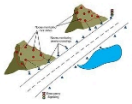
It is applicable for non-CDMA Sensor nodes.

↳ CHAIN is Constructed, but chain is divided into no. of groups to space out Simultaneous transmission.



About: Virtual sensor network

An Entity of Type: Thing, from Named Graph: http://dbpedia.org, within Data Space: dbpedia.org



A virtual sensor network (VSN) in computing and telecommunications is an emerging form of collaborative wireless sensor networks. In contrast to early wireless sensor networks that were dedicated to a specific application (e.g., target tracking), VSNs enable multi-purpose, collaborative, and resource efficient WSNs. The key idea difference of VSNs is the collaboration and resource sharing. By doing so nodes achieve application objectives in a more resource efficient way. These networks may further involve dynamically varying subset of sensor nodes (e.g., when the phenomenon migrates sensors that detect the phenomenon changes with time) and/or users (users that are accessing the network changes with time).A VSN can be formed by providing logical connectivity among collaborative sensors. Nod

Property	Value
dbo:abstract	• A virtual sensor network (VSN) in computing and telecommunications is an emerging form of collaborative wireless sensor networks. In contrast to early wireless sensor networks that were dedicated to a specific application (e.g., target tracking), VSNs enable multi-purpose, collaborative, and resource efficient WSNs. The key idea difference of VSNs is the collaboration and resource sharing. By doing so nodes achieve application objectives in a more resource efficient way. These networks may further involve dynamically varying subset of sensor nodes (e.g., when the phenomenon migrates sensors that detect the phenomenon changes with time) and/or users (users that are accessing the network changes with time).A VSN can be formed by providing logical connectivity among collaborative sensors. Nodes can be grouped into different VSNs based on the phenomenon they track (e.g., rock slides vs. animal crossing) or the task they perform. VSNs are expected to provide the protocol support for formation, usage, adaptation, and maintenance of subset of sensors collaborating on a specific task(s). Even the nodes that do not sense the particular event/phenomenon could be part of a VSN as far as they are willing to allow sensing nodes to communicate through them. Thus, VSNs make use of intermediate nodes, networks, or other VSNs to efficiently deliver messages across members of a VSN. (en)
dbo:thumbnail	• wiki-commons:Special:FilePath/Rock_Sliding_&_Animal_Monitoring.jpg?width=300
dbo:wikiPageExternalLink	• http://www.cnrl.colostate.edu/Projects/VSNs/vsns.html • http://www.cse.mrt.ac.lk/~dilumb/docs/pubs/VSN_WMSN_08.pdf • https://dx.doi.org/10.1016/j.comnet.2009.10.017 • https://arxiv.org/ftp/arxiv/papers/1501/1501.07135.pdf • https://arxiv.org/ftp/arxiv/papers/1501/1501.07139.pdf • https://arxiv.org/ftp/arxiv/papers/1503/1503.01676.pdf • https://arxiv.org/ftp/arxiv/papers/1709/1709.06486.pdf
dbo:wikiPageID	• 20514603 (xsd:integer)
dbo:wikiPageLength	• 5198 (xsd:nonNegativeInteger)
dbo:wikiPageRevisionID	• 1104988730 (xsd:integer)
dbo:wikiPageWikiLink	• dbr:Computing • dbr:Sensor_node • dbr:Telecommunications • dbr:System_deployment • dbr:Wireless_sensor_network • dbc:Wireless_sensor_network • dbr:Raouf_Boutaba • dbr:Animal_crossing • dbr:Phenomenon • dbr:Terrain • dbr:Mountainous • dbr:Phenomena • dbr:Protocol_(computing)

A **virtual sensor network** (VSN) in computing and telecommunications is an emerging form of collaborative wireless sensor networks.[1] In contrast to early wireless sensor networks that were dedicated to a specific application (e.g., target tracking), VSNs enable multi-purpose, collaborative, and resource efficient WSNs. The key idea difference of VSNs is the collaboration and resource sharing. By doing so nodes achieve application objectives in a more resource efficient way. These networks may further involve dynamically varying subset of sensor nodes (e.g., when the phenomenon migrates sensors that detect the phenomenon changes with time) and/or users (users that are accessing the network changes with time).

A VSN can be formed by providing logical connectivity among collaborative sensors. Nodes can be grouped into different VSNs based on the phenomenon they track (e.g., rock slides vs. animal crossing) or the task they perform. VSNs are expected to provide the protocol support for formation, usage, adaptation, and maintenance of subset of sensors collaborating on a specific task(s). Even the nodes that do not sense the particular event/phenomenon could be part of a VSN as far as they are willing to allow sensing nodes to communicate through them. Thus, VSNs make use of intermediate nodes, networks, or other VSNs to efficiently deliver messages across members of a VSN.

TinyOS

Article Talk

This article needs additional citations for verification.
(December 2006) Learn more

TinyOS is an embedded, component-based operating system and platform for low-power wireless devices, such as those used in wireless sensor networks (WSNs), smartdust, ubiquitous computing, personal area networks, building automation, and smart meters. It is written in the programming language nesC, as a set of cooperating tasks and processes. It began as a collaboration between the University of California, Berkeley, Intel Research, and Crossbow Technology, was released as free and open-source software under a BSD license, and has since grown into an international consortium, the TinyOS Alliance.

TinyOS

TinyOS applications are written in the programming language nesC, a dialect of the C language optimized for the memory limits of sensor networks. Its supplementary tools are mainly in the form of Java and shell script front-ends. Associated libraries and tools, such as the nesC compiler and Atmel AVR binutils toolchains, are mostly written in C.

TinyOS programs are built of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

TinyOS is fully non-blocking: it has one call stack. Thus, all input/output (I/O) operations that last longer than a few hundred microseconds are asynchronous and have a callback. To enable the native compiler to better optimize across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically. While being non-blocking enables TinyOS to maintain high concurrency with one stack, it forces programmers to write complex logic by stitching together many small event handlers. To support larger computations, TinyOS provides tasks, which are similar to a Deferred Procedure Call and

TinyOS is fully non-blocking: it has one call stack. Thus, all input/output (I/O) operations that last longer than a few hundred microseconds are asynchronous and have a callback. To enable the native compiler to better optimize across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically. While being non-blocking enables TinyOS to maintain high concurrency with one stack, it forces programmers to write complex logic by stitching together many small event handlers. To support larger computations, TinyOS provides tasks, which are similar to a Deferred Procedure Call and interrupt handler bottom halves. A TinyOS component can post a task, which the OS will schedule to run later. Tasks are non-preemptive and run in first in, first out order. This simple concurrency model is typically sufficient for I/O centric applications, but its difficulty with CPU-heavy applications has led to developing a thread library for the OS, named TOSThreads. TOSThreads are unmaintained and have been deprecated.[1]

TinyOS code is statically linked with program code and is compiled into a small binary, using a custom GNU toolchain. Associated utilities are provided to complete a development platform for working with TinyOS.