Mobile and Sensor Systems

Lecture 4: Wireless Sensor Systems

Dr. Sarfraz Nawaz

Dr. Cecilia Mascolo

UNIVERSITY OF CAMBRIDGE

In this lecture

- We will describe wireless sensor networks in general and the properties of sensor nodes.
- We will introduce sensor network MAC Layer issues and some solutions.

UNIVERSITY OF CAMBRIDGE

Wireless Sensor Networks?

- In many situations, we want to measure things to develop a better understanding of various phenomena.
- With this insight, we can then design novel or improved systems.

Example Application

- A vineyard farmer wants to measure soil moisture, so he can irrigate only those parts where soil moisture is low.
- He wants to measure humidity and temperature as well, so he can use pesticides only when these are most effective.
- Sensing allows him to save resources and can reveal previously unknown behaviour.

Many Applications

- Structural health monitoring
- Environmental monitoring
- Animal behaviour
- Warehouse logistics

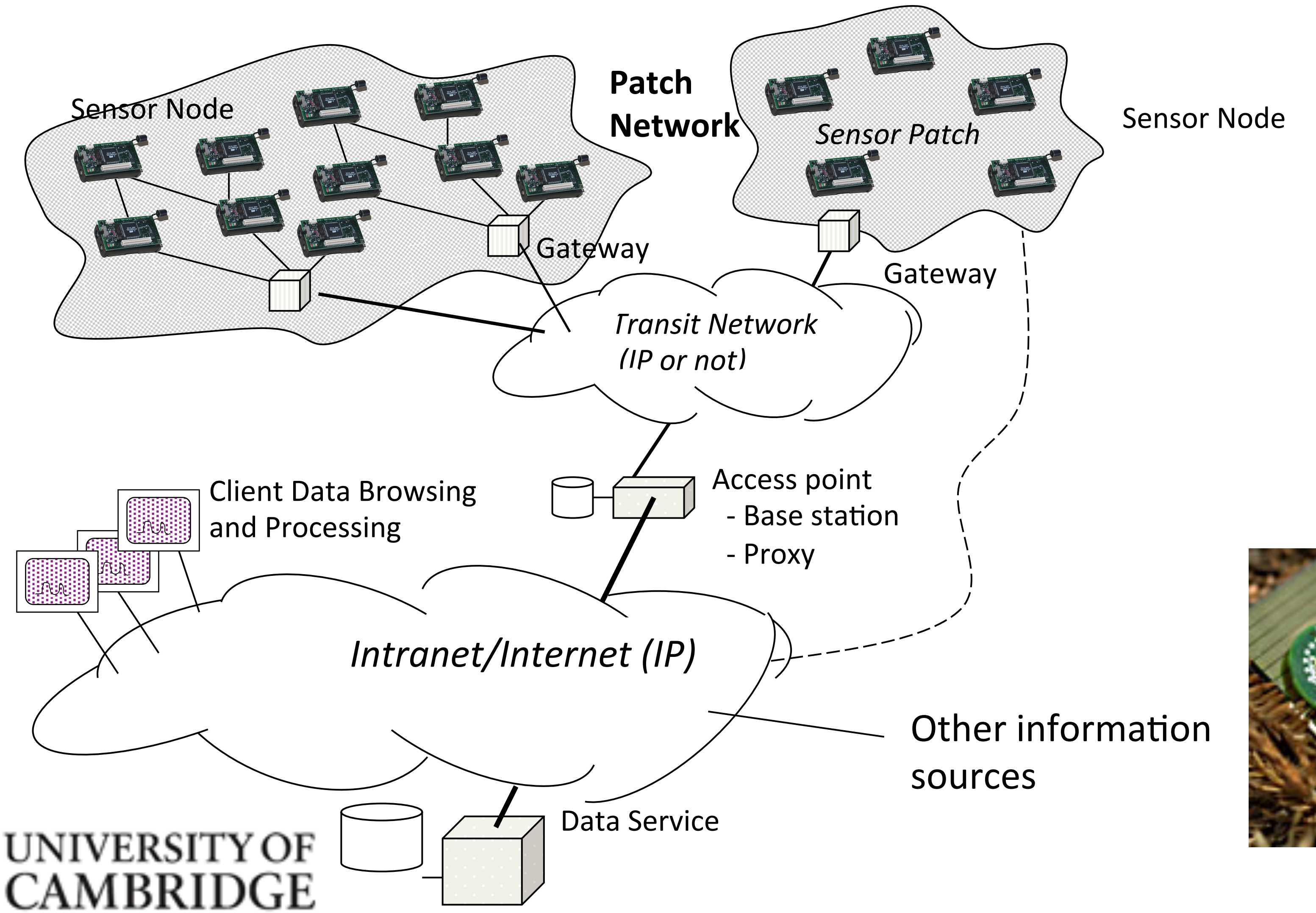
UNIVERSITY OF CAMBRIDGE

Characteristics

- Farmer wants to cover his entire vineyard
  - Large number of sensing devices.
- He wants to keep the cost down
  - Low cost, resource constrained.
- He cannot run wires to these many devices
  - Battery powered, wireless.

UNIVERSITY OF CAMBRIDGE

An Example of Sensor Network Architecture



UNIVERSITY OF CAMBRIDGE

Sensor Systems vs Standard or Mobile Systems

- Sensor nodes have limited computational resources and energy.
- Sensor nodes are prone to failures (especially because they are often deployed in challenging conditions).
- The topology of a sensor network might not change frequently:
  - Many deployments involve nodes with fixed locations.
  - Some deployments may have mobile sensors.

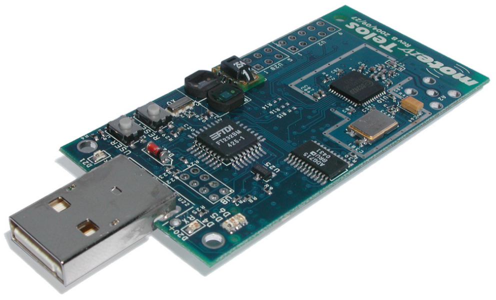
UNIVERSITY OF CAMBRIDGE

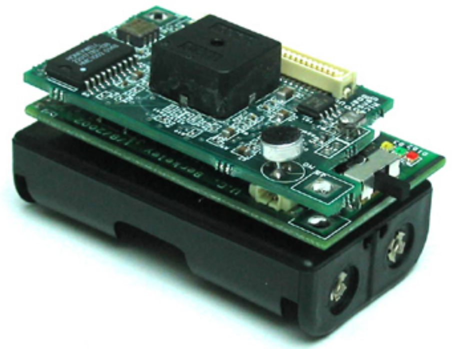
Sensor Node

- A typical sensor node is composed of,

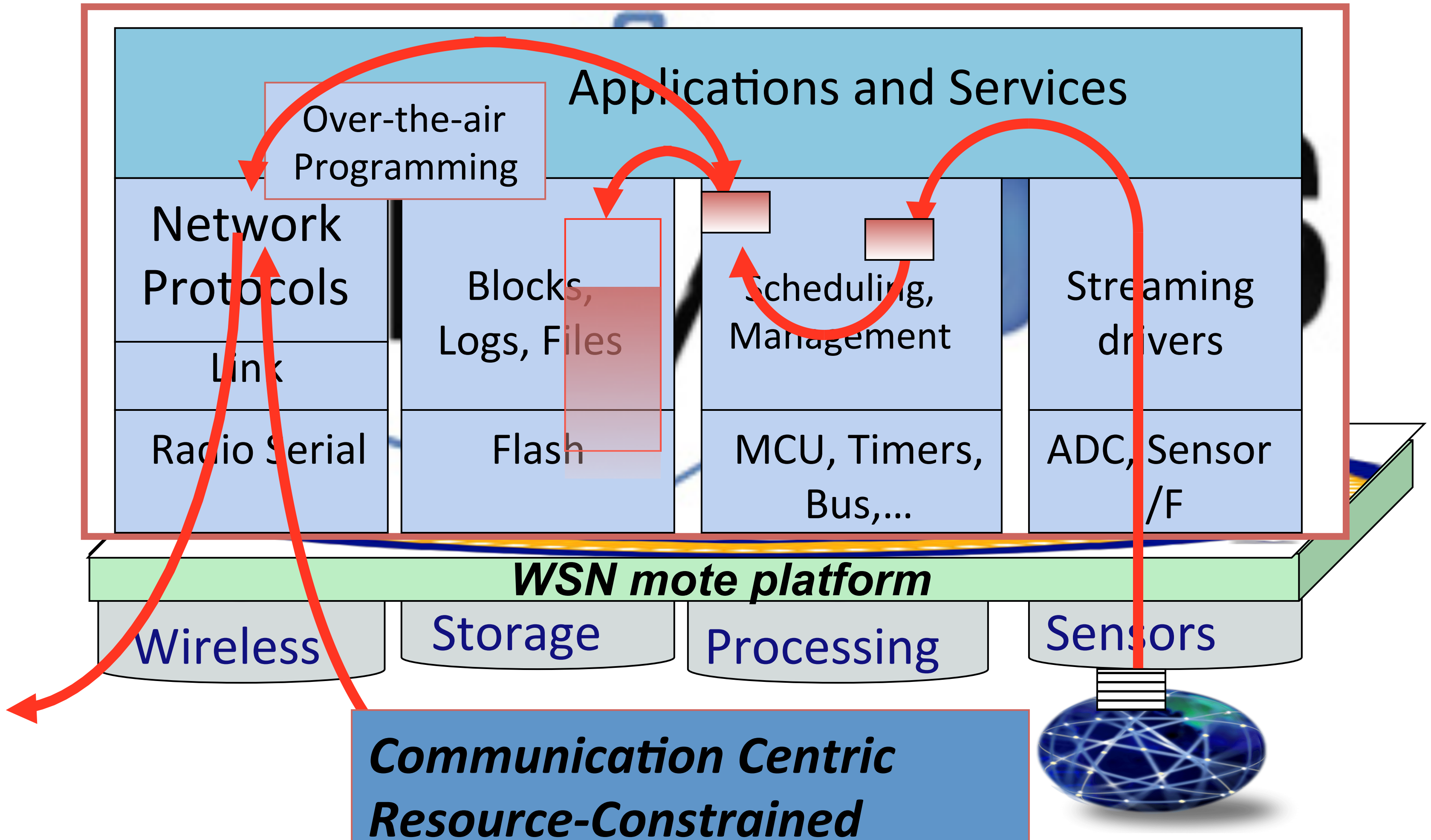
– Sensing device	(Temperature, Humidity)
– Small processor	(16bit, 8Mhz Microcontroller)
– Low power radio	(250 kb/s Zigbee)
– Battery	(Two AA Batteries)
– Small storage	(128 kB flash)

UNIVERSITY OF CAMBRIDGE


TelosB sensor node


MicaZ sensor node

What happens in the node



Communication Centric
Resource-Constrained
Event-driven Execution

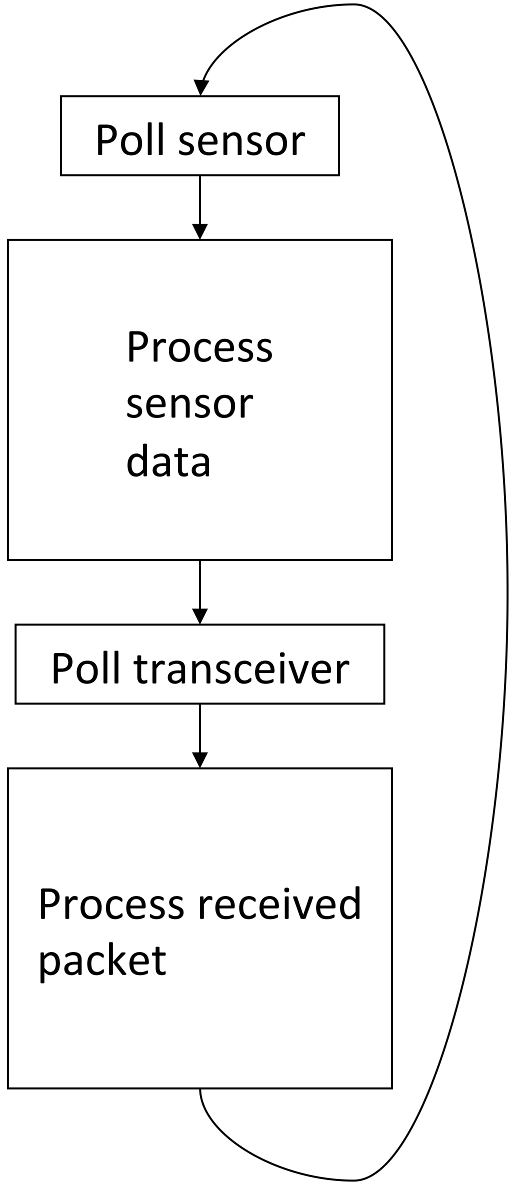
UNIVERSITY OF CAMBRIDGE

What Operating System runs on a sensor?

- Operating system useful to simplify programming tasks and to allow more control over operations of the system
- But what can we do with such a constrained device?
- Given the kind of applications needed it is important to support concurrency…[frequent and parallel collection from different sensors]

Main issue: How to support concurrency

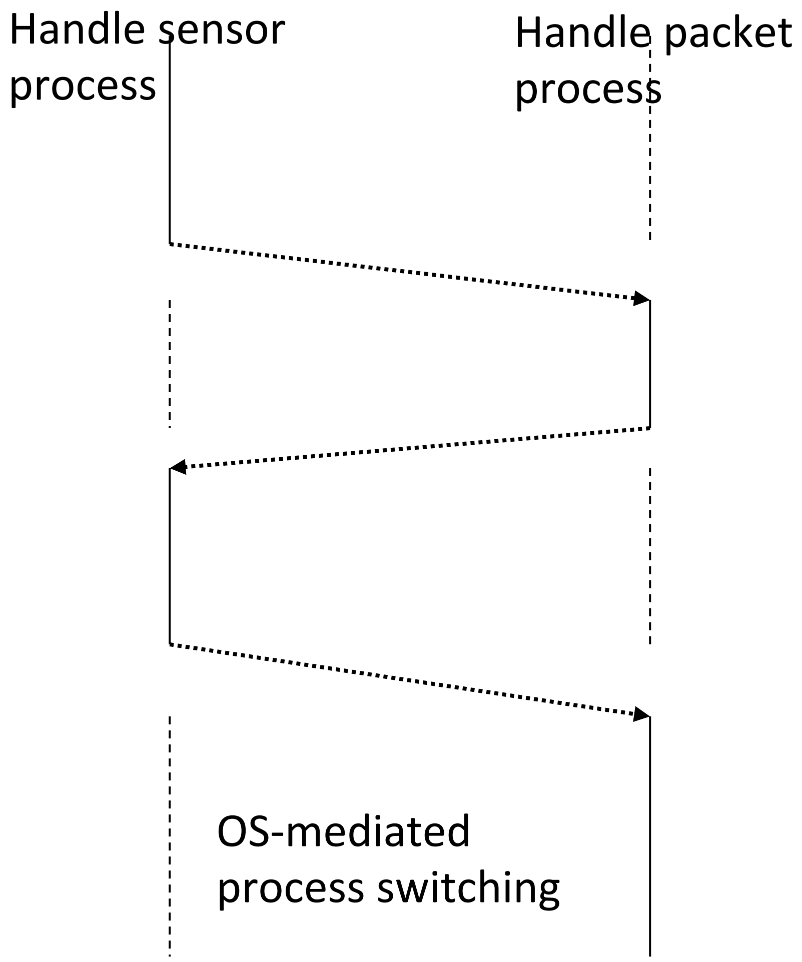
- Simplest option: No concurrency, sequential processing of tasks
  - Not satisfactory: Risk of missing data (e.g., from transceiver) when processing data, etc.
  - Interrupts/asynchronous operation has to be supported
- Why concurrency is needed
  - Sensor node's CPU has to service the radio modem, the actual sensors, perform computation for application, execute communication protocol software, etc.



UNIVERSITY OF CAMBRIDGE

Traditional concurrency: Processes

- Traditional OS: processes/threads
  - Based on interrupts, context switching
  - But: memory overhead, execution overhead
- concurrency mismatch
  - One process per protocol entails too many context switches
  - Many tasks in WSN are small with respect to context switching overhead

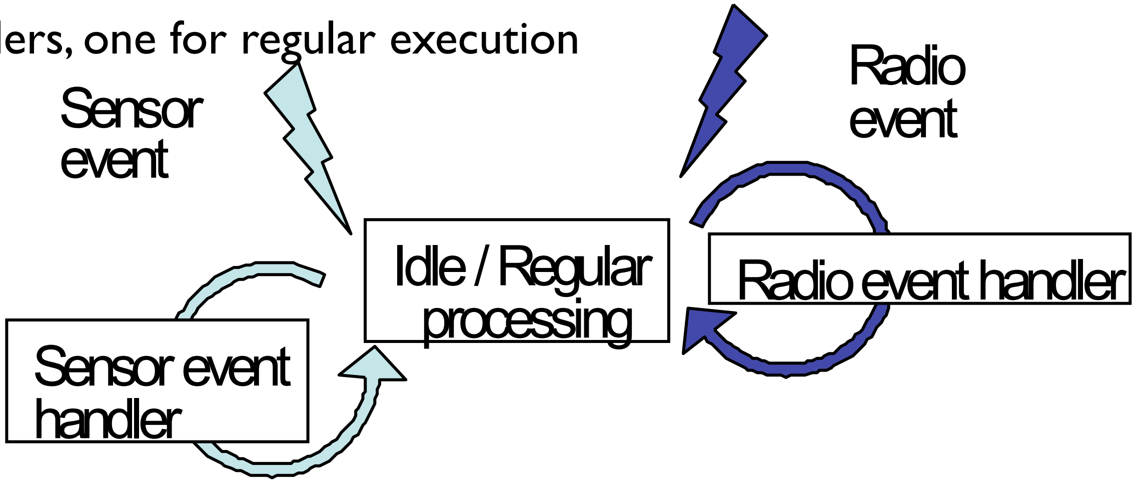


UNIVERSITY OF CAMBRIDGE

Event-based concurrency

- Alternative: Switch to ***event-based programming model***
  - Perform regular processing or be idle
  - React to events when they happen immediately
  - Basically: interrupt handler
- Problem: must not remain in interrupt handler too long
  - Danger of losing events
  - Only save data, post information that event has happened, then return

  ! ***Run-to-completion*** principle
  - Two contexts: one for handlers, one for regular execution



TinyOS: Tasks and Command/Event Handlers

- TinyOS: an OS for sensor networks
- Event handlers must run to completion:
  - Must not wait an indeterminate time.
  - Only a ***request*** to perform some action.
- Tasks can perform arbitrary, long computation;
  - Also have to be run to completion.
  - But can be interrupted by handlers.

  ! No need for stack management, tasks are atomic with respect to each other.

UNIVERSITY OF CAMBRIDGE

Energy Management

- Local computation does not consume significant amount of energy.
- **The main source of energy consumption is the radio.**
- Current draw on Telosb,
  - Microcontroller ON, Radio OFF 1.8mA
  - Microcontroller ON, Radio ON 21mA

Energy Management

- In order to save energy, limit the number of radio transmissions.
- Idle listening consumes as much power as transmission.
- Current draw on Telosb,
  - Idle listening 23mA
  - Transmitting 21mA
- Idle listening is wasteful when average data rate is low.
- Switch off the radio when idle.
- Transmissions from other sensor nodes are lost.

UNIVERSITY OF CAMBRIDGE

Radio Duty Cycling

- Switch off the radio of all sensors at specific intervals:
  - Very precise synchronization.
  - Still probable idle time for sensors which do not communicate.
- More refined strategy:
  - Wave of switch off time depending on topology.
  - Still an overestimate of the communication needs of some sensors (traffic might be varying across the network).

UNIVERSITY OF CAMBRIDGE

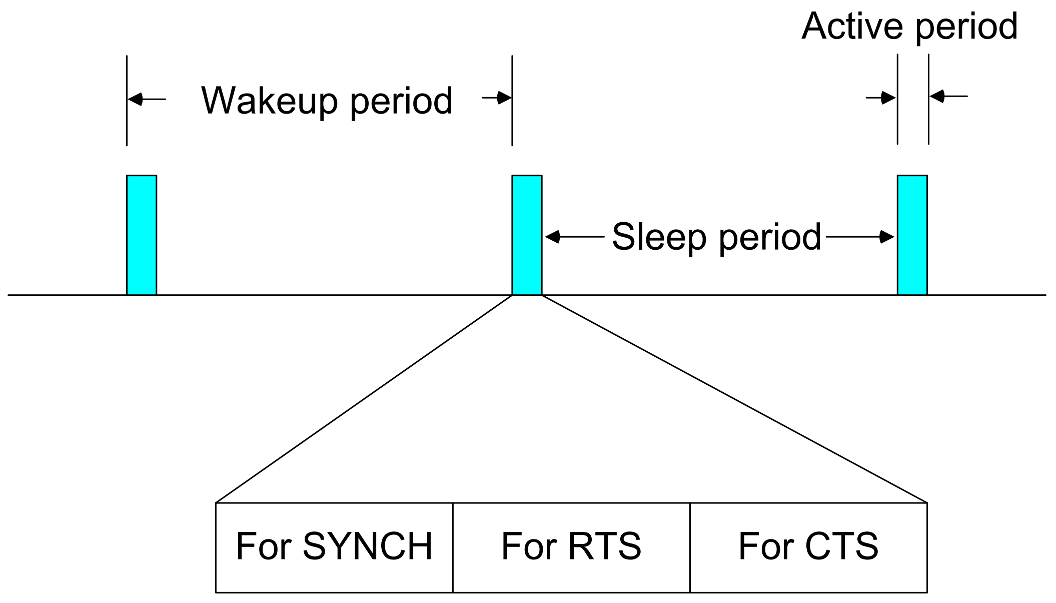
Dynamic duty cycling

- More refined strategies have been proposed which aim to allow sensors with more packets to stay awake longer and others to sleep more.
  - Synchronized (e.g. S-MAC)
  - Asynchronous (e.g. B-MAC, X-MAC)

- Synchronized protocols try to **negotiate a schedule** among neighboring nodes.
- Asynchronous protocols rely on **preamble sampling** to connect a transmitter to receivers.

Sensor-MAC (S-MAC)

- Idea: Switch nodes off, ensure that neighboring nodes turn on simultaneously to allow packet exchange (rendez-vous)
  - Packet exchange occurs only in these ***active periods***
  - Need to also exchange wakeup schedule between neighbors
  - When awake, essentially perform RTS/CTS
  - Use SYNCH, RTS, CTS phases



UNIVERSITY OF CAMBRIDGE

S-MAC

- SYNC phase divided into time slots with CSMA and backoffs to send schedule to neighbours.
- Y chooses a slot and if no signal is received in this slot, it will transmit its schedule to X otherwise it will wait for next wake up of X.
- RTS phase: X listens for RTS packets (CSMA contention).
- CTS phase: X sends one and extends its wake up time.

UNIVERSITY OF CAMBRIDGE

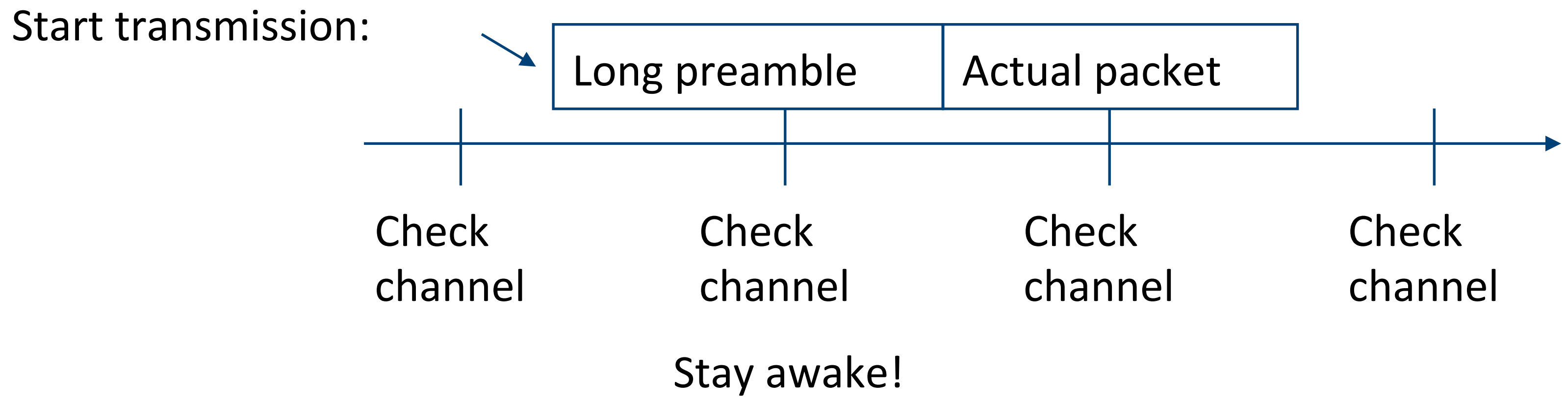
S-MAC synchronized islands

- Nodes try to pick up schedule synchronization from neighboring nodes.
- If no neighbor found, nodes pick some schedule to start with.
- If additional nodes join, some node might learn about two different schedules from different nodes
  - "Synchronized islands".
- To bridge this gap, it has to follow both schemes and use more energy.

UNIVERSITY OF CAMBRIDGE

Preamble Sampling

- So far: Periodic sleeping supported by some means to synchronize wake up of nodes to ensure rendez-vous between sender and receiver.
- Alternative option: Don't try to explicitly synchronize nodes:
  - Have receiver sleep and only periodically sample the channel.
- Use ***long preambles*** to ensure that receiver stays awake to catch actual packet. Example: B-MAC and WiseMAC.



UNIVERSITY OF CAMBRIDGE

Problems with this technique

- Overhearing
  - All receivers listening to the preamble have to stay awake to find out who is the intended receiver.
- Energy Consumption
  - Long preamble causes increased energy consumption at both the transmitter and the receiver.
- Latency
  - Long preamble introduces per-hop latency.

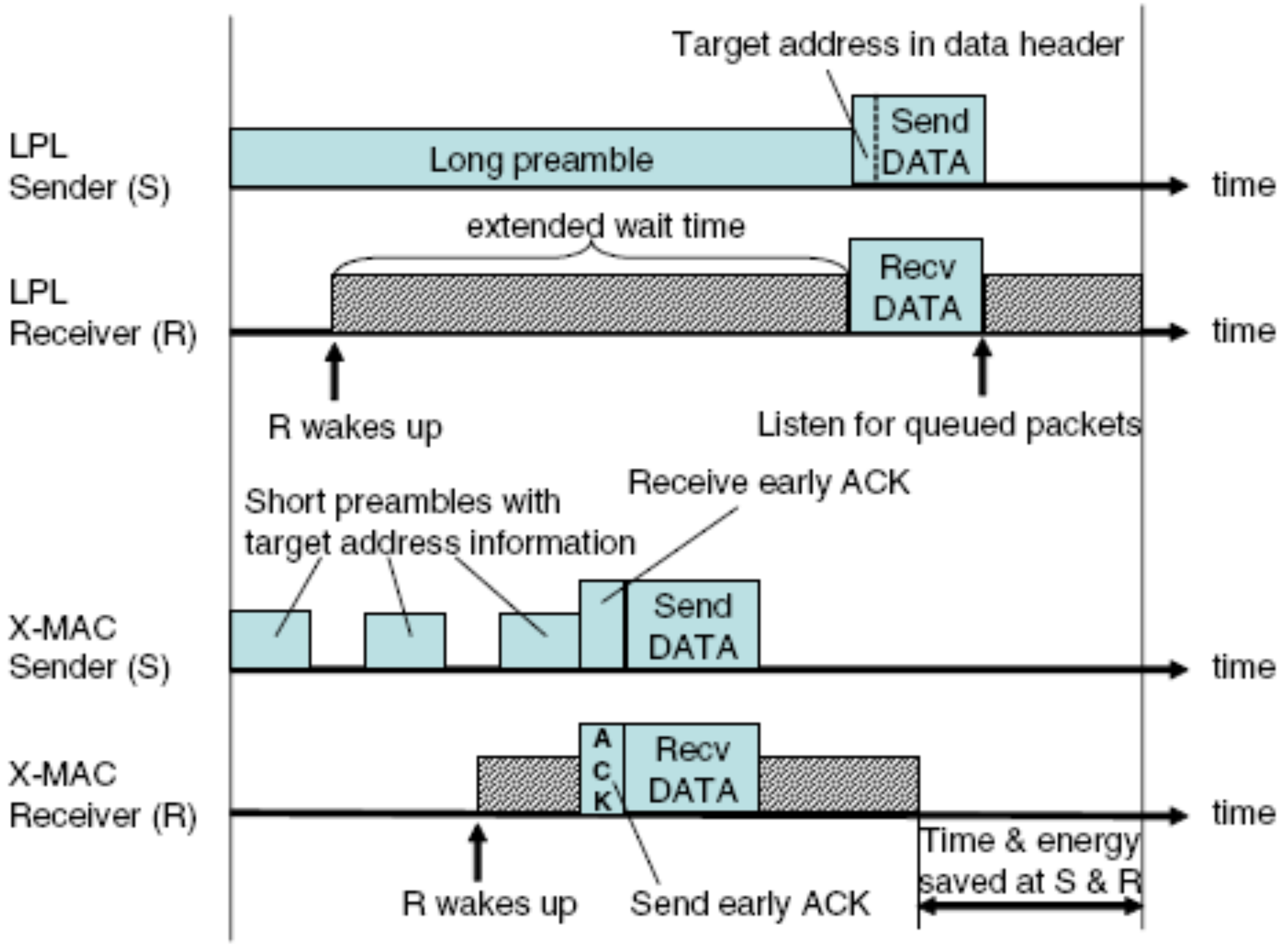
UNIVERSITY OF CAMBRIDGE

X-MAC

- Short preamble
  - Reduce latency and reduce energy consumption
- Target in preamble
  - Minimize overhearing problem.
- Adding wait time between preambles
  - Reduces latency for the case where destination is awake before preamble completes.

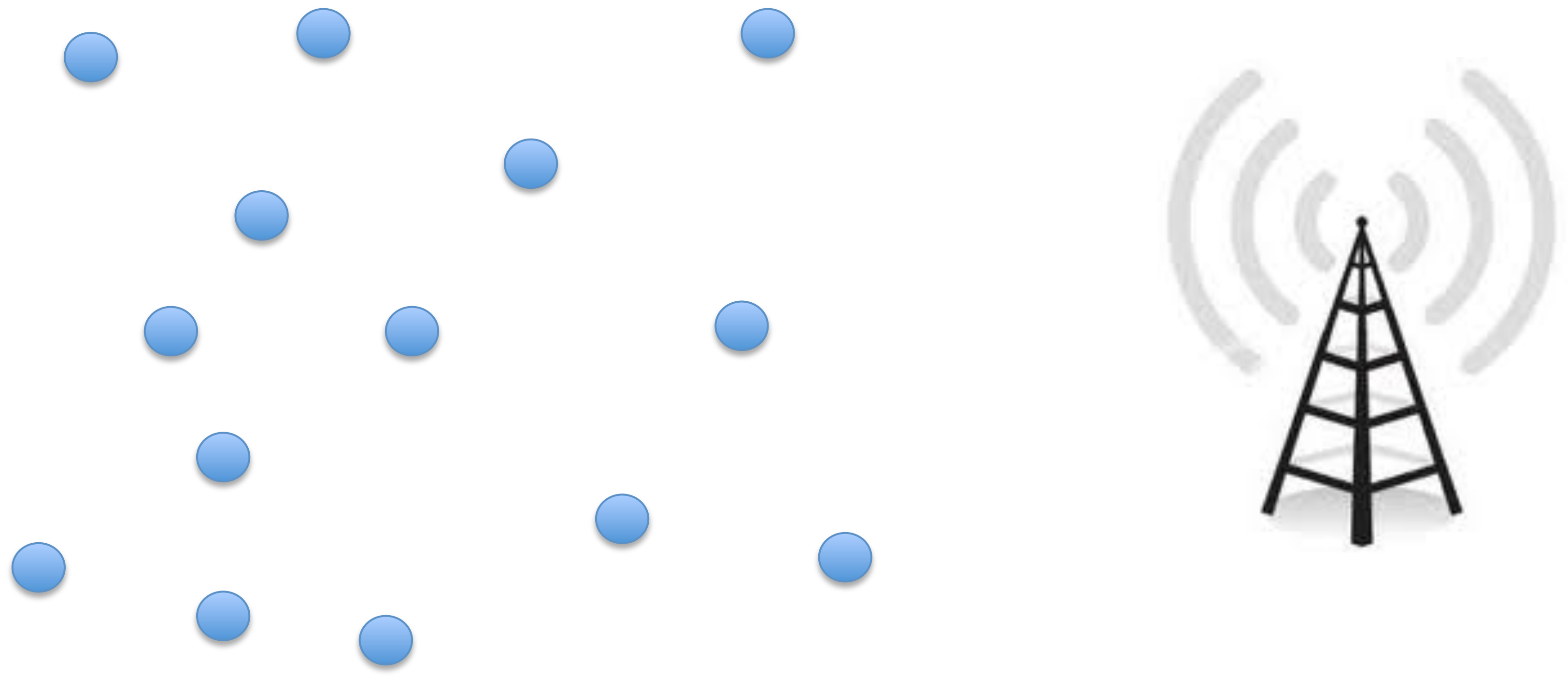
UNIVERSITY OF CAMBRIDGE

X-MAC



UNIVERSITY OF CAMBRIDGE

Low-Energy Adaptive Clustering Hierarchy (LEACH)

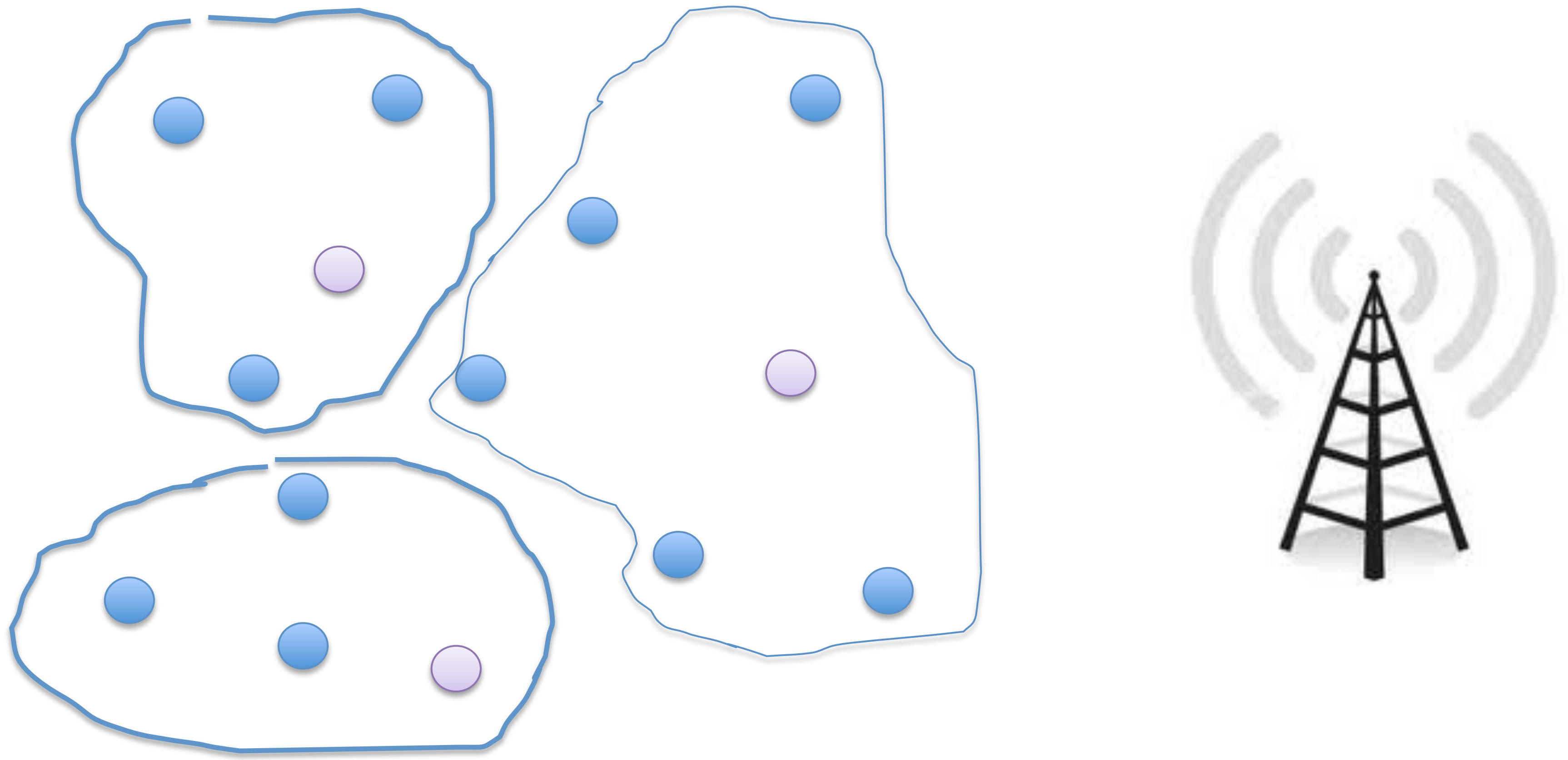


UNIVERSITY OF CAMBRIDGE

Low-Energy Adaptive Clustering Hierarchy (LEACH)

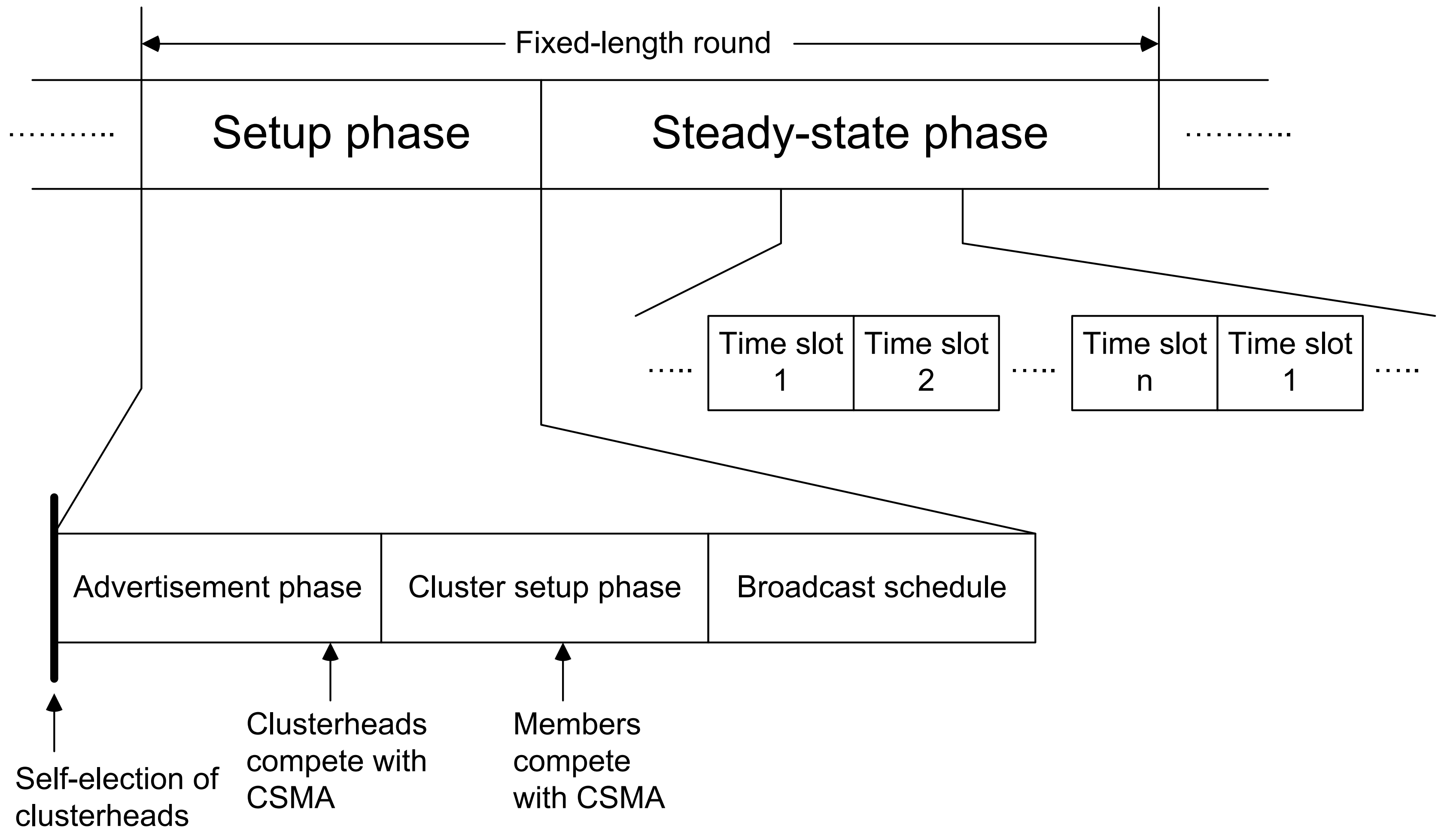
- Assumption: dense network of nodes, reporting to a central sink, each node can reach sink **directly**.
- Idea: Group nodes into "***clusters***", controlled by ***clusterhead:***
  - Setup phase; details: later.
  - About 5% of nodes become clusterhead (depends on scenario).
  - Role of clusterhead is rotated to share the burden.
  - Clusterheads advertise themselves, ordinary nodes join CH with strongest signal.
  - Clusterheads organize: CDMA code for all member transmission. TDMA schedule to be used within a cluster
- In steady state operation:
  - CHs collect & aggregate data from all cluster members.
  - Report aggregated data to sink using CSMA.

Low-Energy Adaptive Clustering Hierarchy (LEACH)



UNIVERSITY OF CAMBRIDGE

LEACH rounds



UNIVERSITY OF CAMBRIDGE

References

- TinyOS tutorial: http://www.tinyos.net/tinyos-1.x/doc/tutorial/
- SMAC: Ye, W., Heidemann, J., and Estrin, D. 2004. Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. Netw. 12, 3 (Jun. 2004), 493-506.
- WISEMAC: El-Hoiydi, A. and Decotignie, J. 2004. WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In Proceedings of the Ninth international Symposium on Computers and Communications 2004 Volume 2 (Iscc"04) - Volume 02 (June 28 - July 01, 2004). ISCC. IEEE Computer Society, Washington, DC, 244-251.
- X-MAC: M. Buettner, G.V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems* Boulder, Colorado, USA: ACM, 2006.
- LEACH: Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, Energy-Efficient Communication Protocols for Wireless Microsensor Networks, Proc. Hawaaian Int'l Conf. on Systems Science, January 2000.