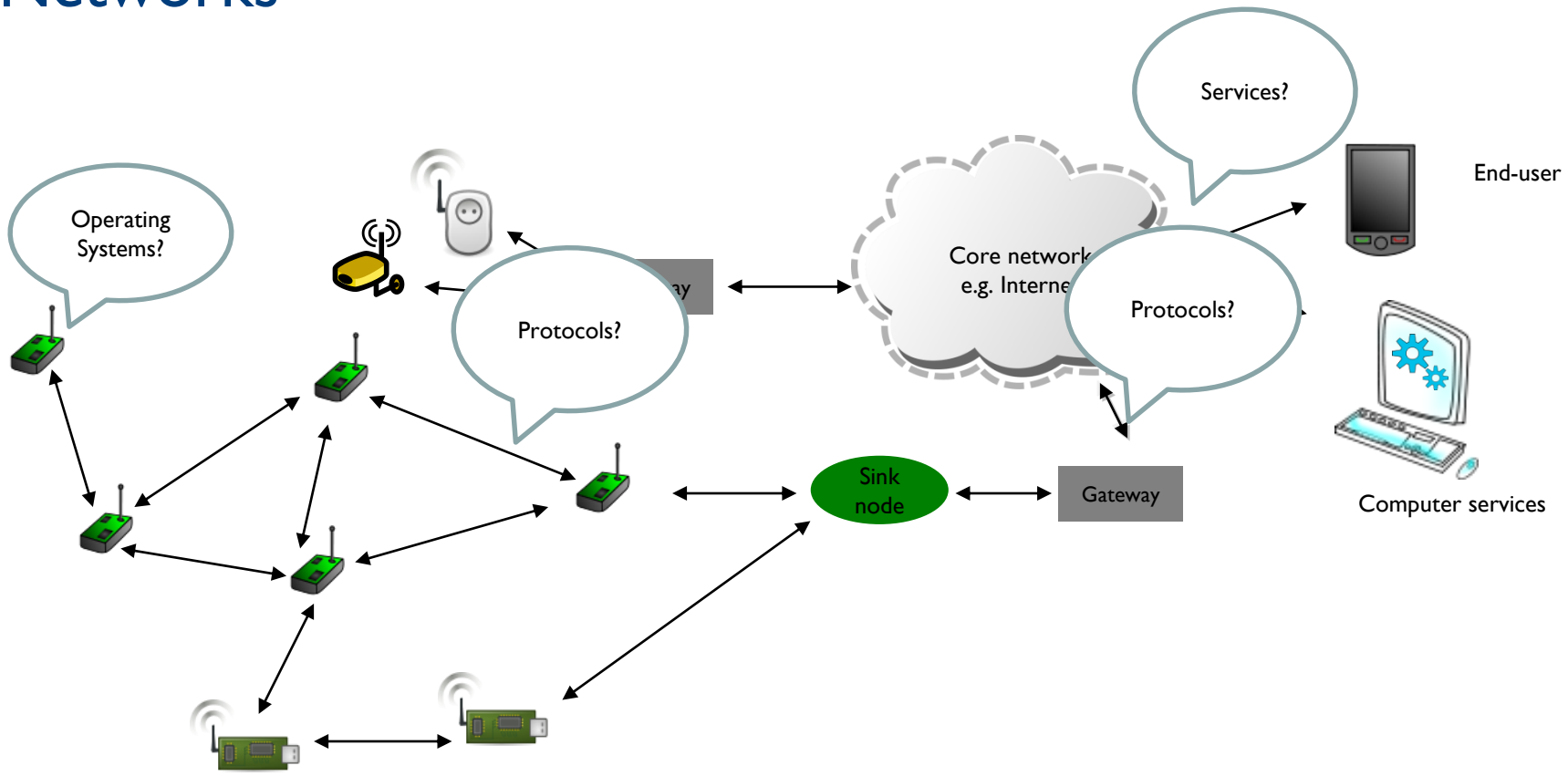
Internet of Things

Software platforms and services

Wireless Sensor (and Actuator) Networks



- The networks typically run Low Power Devices
- Consist of one or more sensors, could be different type of sensors (or actuators)

Operating Systems

- An Operating System (OS) in an embedded system is a thin software that resides between the node's hardware and the application layer.
- OS provides basic programming abstractions to the application developer.
- The main task of the OS is to enable applications to interact with hardware resources, to schedule and prioritise tasks and mediate between applications and services that try to use the memory resources.

Features of the OS in embedded systems

- Memory management
- Power management
- File management
- Networking
- Providing programming environment and tools (commands, interpreters, compiler, etc.)
- Providing entry points to access sensitive resources such as writing to input components.
- Providing and supporting functional aspects such as scheduling, multi-threading, handling interrupts, memory allocations.

Threads and events

- A “thread” in a programming environment is the smallest sequence of programmed instructions that can be managed and run independently by a scheduler.
- An event driven program typically runs an event loopss. It keeps waiting for an event, e.g. input from internal alarms. When an event occurs, the program collects data about the event and dispatches the event to the event handler software to deal with it.

Thread-based vs Event-based Programming

- In WSN it is important to support concurrent tasks, in particular tasks related to I/O systems.
- Thread-based programming uses multiple threads and a single address space.
- This way if a thread is blocked by an I/O operation, the thread can be suspended and other tasks can be executed in different threads.
- The programmer must protect shared data structures with locks, coordinate the execution of threads.
- The program written for multiple threading can be complex, can include bugs and may lead to deadlocks.

Thread-based vs Event-based Programming- II

- The event-based programming uses events and event handlers.
- Event handlers are registered at the OS scheduler and are notified when a named event occurs.
- The OS kernel usually implements a loop function that polls for events and calls relevant event handlers when an event is occurred.
- An event is processed by an event handler until completion unless it reaches a blocking operation.
- In the case of reaching a blocking operation it registers a new call back and returns control to scheduler.

Embedded Operating Systems

- OS running on devices with restricted functionality
  - In the case of sensor nodes, there devices typically also have limited processing capability
  - e.g. TinyOS
- Restricted to narrow applications
  - industrial controllers, robots, networking gear, gaming consoles, metering, sensor nodes…
- Architecture and purpose of embedded OS changes as the hardware capabilities change (i.e. mobile phones)

Source: The Web of Things, Marko Grobelnik, Carolina Fortuna, Jožef Stefan Institute.

TinyOS

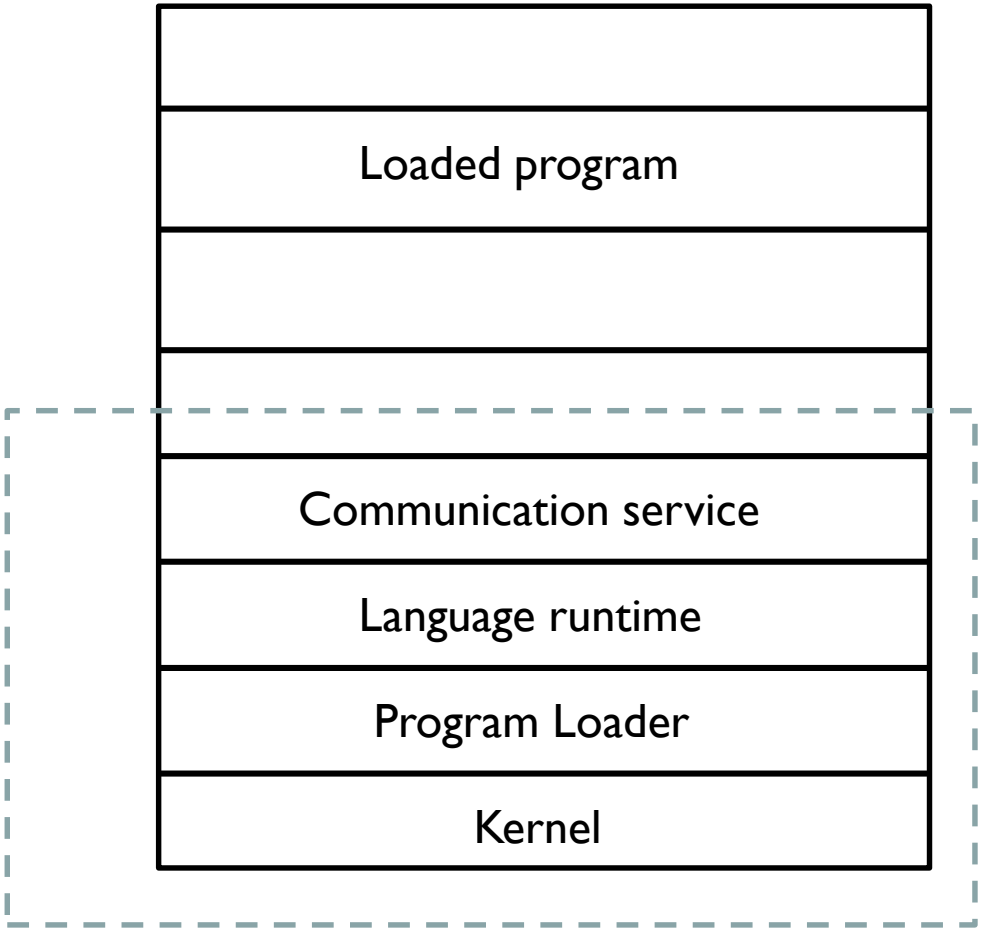
- "TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks ."
- TinyOS applications are developed using nesC
- nesC is a dialect of the C language that is optimised for the memory limits of sensor networks.

TinOS - programming

- "TinyOS is completely non-blocking:
  - It has one stack.
  - All I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback.
  - "To enable the native compiler to better optimise across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically."

TinyOS home page: http://www.tinyos.net/
TinyOS tutorial: http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials

The Contiki OS



Loadable programs
Can be easily updated

Core: single binary
Usually never modified

Sensor Network Programming

- Sensor Network programming can be: node centric or it can be application centric.
- Node-centric approaches focus on development of a software for nodes (on a per-node level).
- Application-centric approaches focus on developing software for a part or all of the network as one entity.
- The application centric programming will require collaboration among different nodes in the network for collection, dissemination, analysis and/or processing of the generated and collected data.
- While in node centric programming the main focus is on developing a software on a per-node level.

Nodes and Applications in Wireless Sensor Networks

- Sensor Networks consist of nodes with different capabilities.
  - Large number of heterogeneous sensor nodes
  - Spread over a physical location
  - It includes physical sensing, data processing and networking
- In ad-hoc networks, sensors can join and leave due to mobility, failure etc.
- Data can be processed in-network, or it can be directly communicated to the endpoints.

Types of nodes

- Sensor nodes
  - Low power
  - Consist of sensing device, memory, processor and radio
  - Resource-constrained
- Sink nodes
  - Another sensor node or a different wireless node
  - Normally more powerful/better resources
- Gateway
  - A more powerful node
  - Connection to core network
  - Could consist service representation, cache/storage, discovery and other functions

Types of applications

- Event detection
  - Reporting occurrences of events
  - Reporting abnormalities and changes
  - Could require collaboration of other nearby or remote nodes
  - Event definition and classification is an issue
- Periodic measurements
  - Sensors periodically measure and report the observation and measurement data
  - Reporting period is application dependent
- Approximation and pattern detection
  - Sending messages along the boundaries of patterns in both space/time
- Tracking
  - When the source of an event is mobile
  - Sending event updates with location information

Requirements and challenges

- Fault tolerance
  - The nodes can get damaged, run out of power, the wireless communication between two nodes can be interrupted, etc.
  - To tolerate node failures, redundant deployments can be necessary.
- Lifetime
  - The nodes could have a limited energy supply;
  - Sometimes replacing the energy sources is not practical (e.g. underwater deployment, large/remote field deployments).
  - Energy efficient operation can be a necessity.

Requirements and challenges – Cont'd

- Scalability
  - A WSN can consists of a large number of nodes
  - The employed architectures and protocols should scale to these numbers.
- Wide range of densities
  - Density of the network can vary
  - Different applications can have different node densities
  - Density does not need to be homogeneous in the entire network and network should adapt to such variations.

Requirements and challenges – Cont'd

- Programmability
  - Nodes should be flexible and their tasks could change
  - The programmes should be also changeable during operation.
- Maintainability
  - WSN and environment of a WSN can change;
  - The system should be adaptable to the changes.
  - The operational parameters can change to choose different trade-offs (e.g. to provide lower quality when energy efficiency is more important)

Required mechanisms

- Multi-hop wireless communications
  - Communication over long distances can require intermediary nodes as relay (instead of using high transmission power for long range communications).
- Energy-efficient operation
  - To support long lifetime
  - Energy efficient communication/dissemination of information
  - Energy efficient determination of a requested information
- Auto-configuration
  - Self-xxx functionalities
  - Tolerating node failures
  - Integrating new nodes

Required mechanisms

- Collaboration and in-network processing
  - In some applications a single sensor node is not able to handle the given task or provide the requested information.
  - Instead of sending the information form various source to an external network/node, the information can be processed in the network itself.
    - e.g. data aggregation, summarisation and then propagating the processed data with reduced size (hence improving energy efficiency by reducing the amount of data to be transmitted).
- Data-centric
  - Conventional networks often focus on sending data between two specific nodes each equipped with an address.
  - Here what is important is data and the observations and measurements not the node that provides it.

Communication and Network Protocol Support

Communication Protocols

- Wired
  - USB, Ethernet
- Wireless
  - Wifi, Bluetooth, ZigBee, IEEE 802.15.x
- Single-hop or multi-hop
  - Sink nodes, cluster heads…
- Point-to-Point or Point-to-Multi Point
- (Energy) efficient routing

Wireless Communications

- Mostly performed in unlicensed bands according to open standards
  - Standard: IEEE 802.15.4 -Low Rate WPAN
    - 868/915 MHz bands with transfer rates of 20 and 40 kbit/s, 2450 MHz band with a rate of 250 kbit/s
    - Technology: ZigBee, WirelessHART
  - Standard: ISO/IEC 18000-7 (standard for active RFID)
  - 433 MHz unlicensed spectrum with transfer rates of 200kbit/s
    - Technology: Dash7

Wireless Communications - continued

  - Standard: IEEE 802.15.1 –High Rate WPAN
    - 2.40GHzbands with transfer rates of 1-24 Mbit/s
    - Technology: **Bluetooth** (BT 3.0 Low Energy Mode)
  - Standard: IEEE 802.11x –WLAN
    - 2.4, 3.6 and 5GHz with transfer rates 15-150 Mbit/s
    - Technology: **Wi-Fi**

- Licensed bands
  - Standard: 3GPP –WMAN, WWAN cellular communication
  - 950 MHz, 1.8 and 2.1 GHz bands with data rate ranging from 20 Kbit/s to 7.2 Mbit/s, depending on the release
  - Technology: **GPRS, HSPA**

IEEE 802.15.4 WPAN

- IEEE standard for WPAN applications
- MAC protocol
  - Single channel at any one time
  - Combines contention-based and schedule-based schemes
  - Asymmetric: nodes can assume different roles
- It does not define other higher-level layers and interoperability sub-layers are
  - ZigBee is built on this standard
  - TinyOS stack also uses some items of IEEE 802.15.4 hardware.

Network protocols

- The network (or OSI Layer 3 abstraction) provides an abstraction of the physical world.
- Communication protocols
  - Most of the IP-based communications are based on the IPV.4 (and often via gateway middleware solutions)
  - IP overhead makes it inefficient for embedded devices with low bit rate and constrained power.
  - However, IPv6.0 is increasingly being introduced for embedded devices
    - 6LowPAN

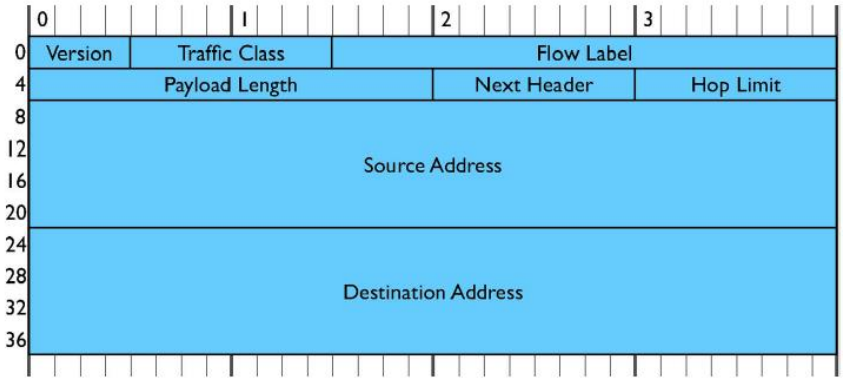
IPv6 over Low power Wireless Personal Area Networks (6LowPAN)

- 6LoWPAN typically includes devices that work together to connect the physical environment to real-world applications, e.g., wireless sensors.
- Small packet size
  - the maximum physical layer packet is 127 bytes
  - 81 octets (81 * 8 bits) for data packets.
- Header compression
- Fragmentation and reassembly
  - 6LoWPAN defines a header encoding to support fragmentation when IPv6 datagrams do not fit within a single frame and compresses IPv6 headers to reduce header overhead.
- Support for both 16-bit short or IEEE 64-bit extended media access control addresses.
- Low bandwidth
  - Data rates of 250 kbps, 40 kbps, and 20 kbps for each of the currently defined physical layers (2.4 GHz, 915 MHz, and 868 MHz, respectively).

Source: Jonathan W. Hui and David E. Culler, IPv6 in Low-Power Wireless Networks, Proceedings of the IEEE (Volume:98 , Issue: 11).

6LowPAN

- IPv6 requires the link to carry a payload of up to 1280 Bytes.
- Low-power radio links often do not support such a large payload - IEEE 802.15.4 frame only supports 127 Bytes of payload and around 80 B in the worst case (with extended addressing and full security information).
- the IPv6 base header, as shown, is relatively large at 40 Bytes.

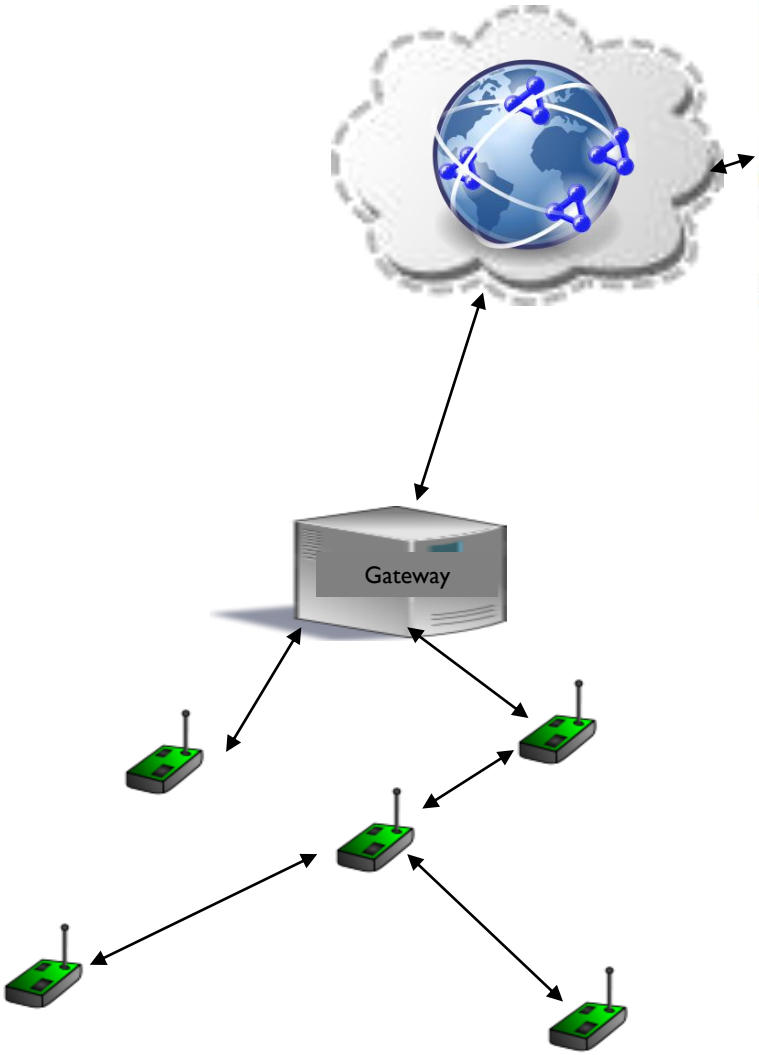


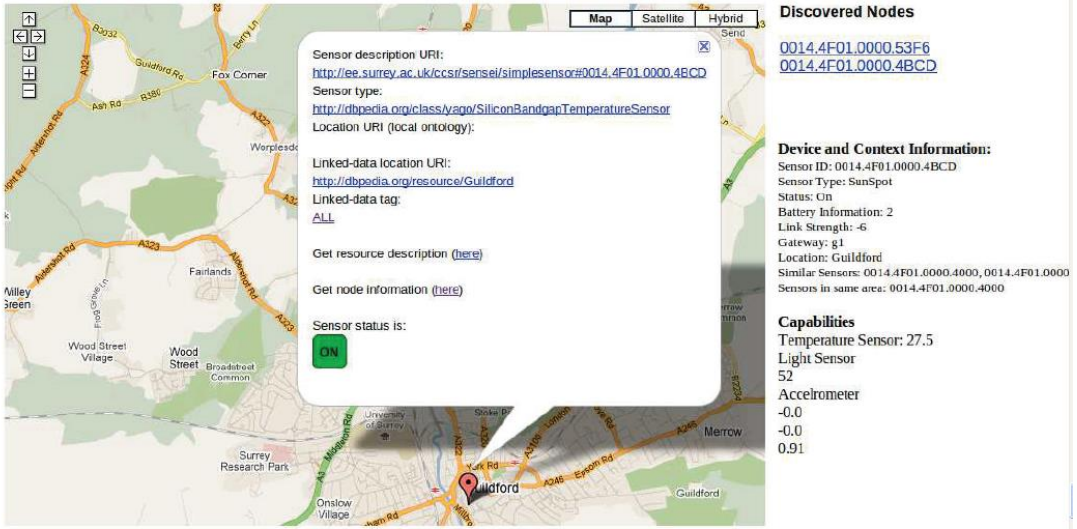
Source: Jonathan W. Hui and David E. Culler, IPv6 in Low-Power Wireless Networks, Proceedings of the IEEE (Volume:98 , Issue: 11).

Using gateway and middleware

- It is unlikely that everything will be IP enabled and/or will run IP protocol stack
- Gateway and middleware solutions can interfaces between low-level sensor island protocols and IP-based networks.
- The gateway can also provide other components such as QoS support, caching, mechanisms to address heterogeneity and interoperability issues.

Gateway and IP networks





Frieder Ganz, Payam Barnaghi, Francois Carrez and Klaus Moessner, "Context-aware Management for Sensor Networks", in the Fifth International Conference on COMmunication System softWAre and middlewaRE (COMSWARE11), July 2011.

Service interfaces to WSN

- Supporting high-level request/response interactions
- Asynchronous event notifications
- Identifying and accessing data
  - By location, by observed entity,
    - By semantically meaningful representations – "Room 35BA01"
- Accessibility of in-network processing functions
- Accessing node/network status information (e.g., battery level)
  - Security, management functionality, …
- There are emerging solutions and standards in this domain supported by Semantic Web technologies and Linked-data (we study some of these next week).

Service interfaces and Web connectivity

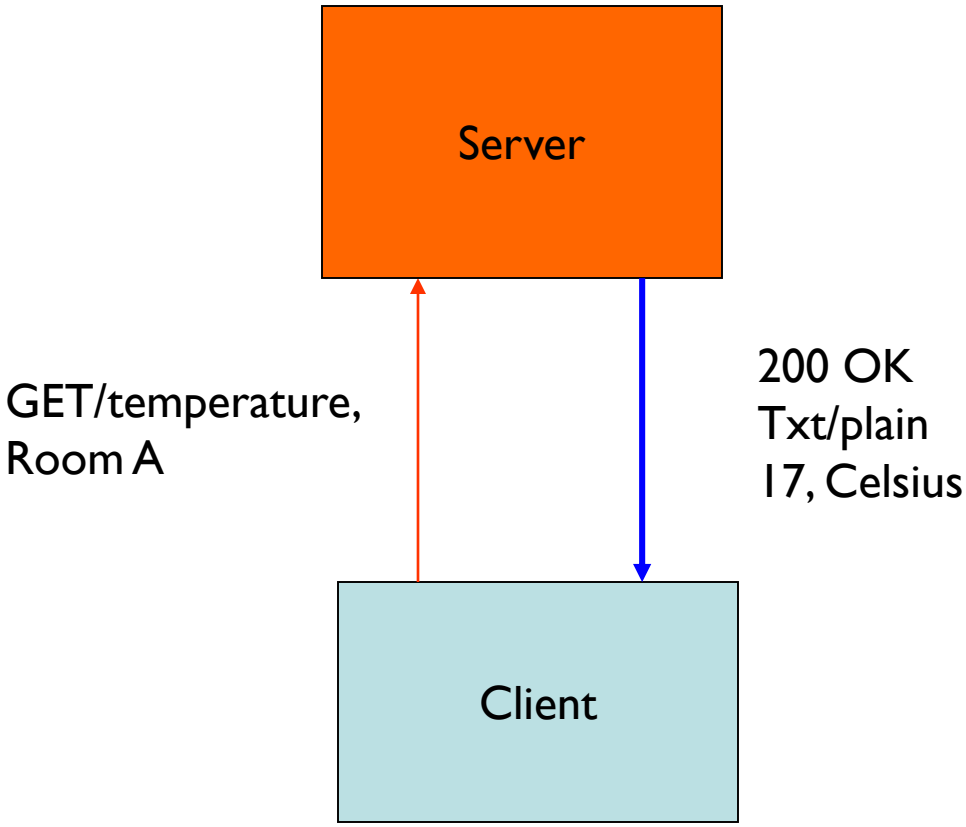
- WSN nodes are typically resource constrained
  - Memory and process limitations
  - Communication load
- Often none-IP or use 6LowPAN
- Using gateway and middleware is a clear solution
- Or can the nodes directly connect to the Web and or support service interfaces?

Constrained Application Protocol (CoAP)

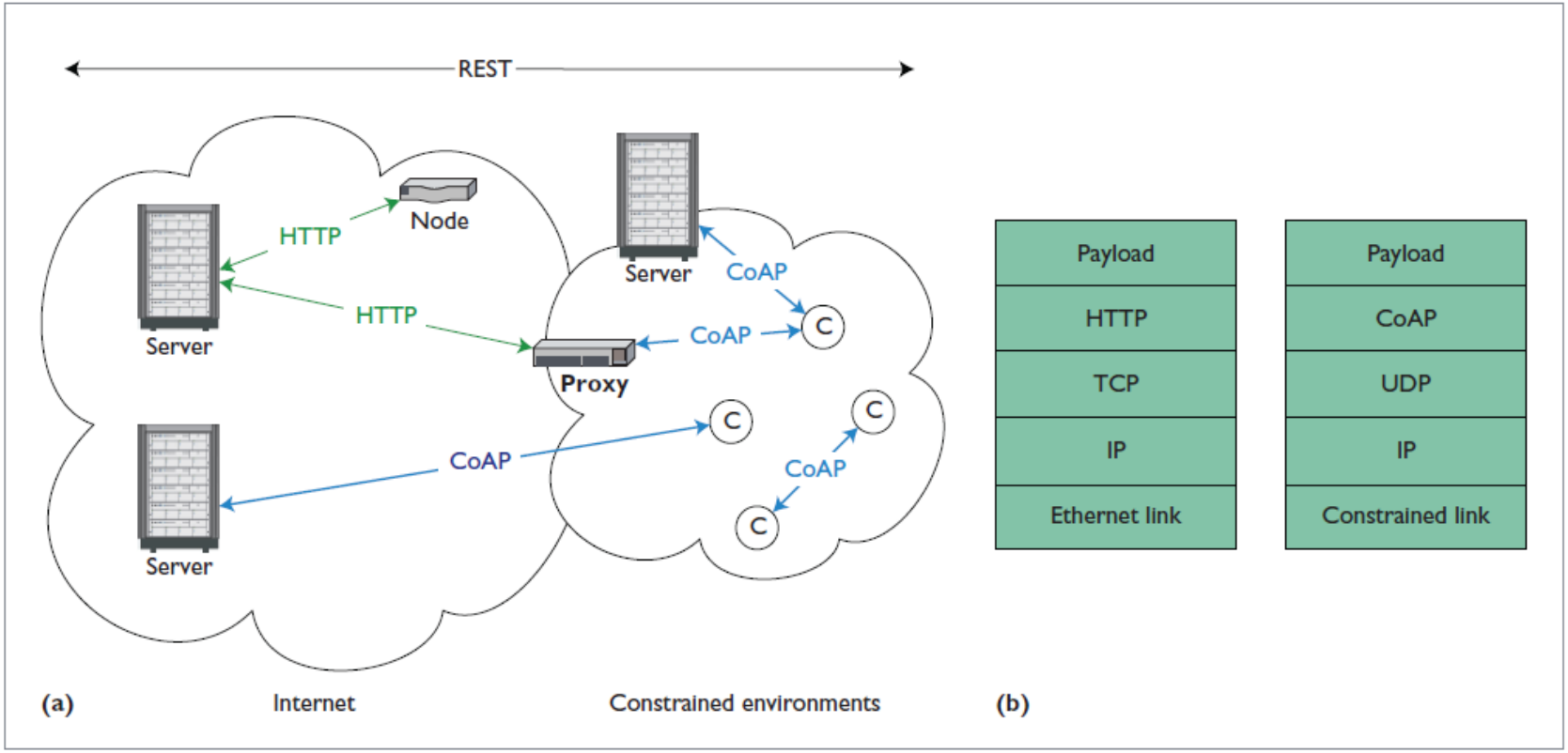
- CoAP is a transfer protocol for constrained nodes and networks.
- CoAP uses the Representational State Transfer (REST) architecture.
  - REST make information available as resources that are identified by URIs.
  - Applications communication by exchanging representation of these resources using a transfer protocol such as HTTP.
  - Clients access servicer controlled resources using synchronous request/response mechanisms.
    - Such as GET, PUT, POST and DELETE.
  - CoAp uses UDP instead of TCP and has a simple "message layer" for re-transmitting lost packets.
  - It also uses compression techniques.

C. Bormann, A. P. Castellani, Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," IEEE Internet Computing, vol. 16, no. 2, pp. 62-67, Feb. 2012, doi:10.1109/MIC.2012.29

Constrained Application Protocol (CoAP)



CoAP protocol stack and interactions



Implementing the Web architecture with HTTP and the Constrained Application Protocol (CoAP). (a) HTTP and CoAP work together across constrained and traditional Internet environments; (b) the CoAP protocol stack is similar to, but less complex than, the HTTP protocol stack.

C. Bormann, A. P. Castellani, Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," IEEE Internet Computing, vol. 16, no. 2, pp. 62-67, Feb. 2012, doi:10.1109/MIC.2012.29

Further reading and examples

- Processes and protothreads: https://github.com/contiki-os/contiki/wiki/Processes
- Examples: https://github.com/contiki-os/contiki/tree/master/examples
- Cooja simulator (if you are interested in advance programming and simulation): https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja