Building Power-Efficient CoAP Devices for Cellular Networks

draft-arkko-core-cellular-00

J. Arkko, A. Eriksson, A. Keränen

IETF 84, Vancouver, BC, Canada

August 2nd, 2012

Ari Keränen

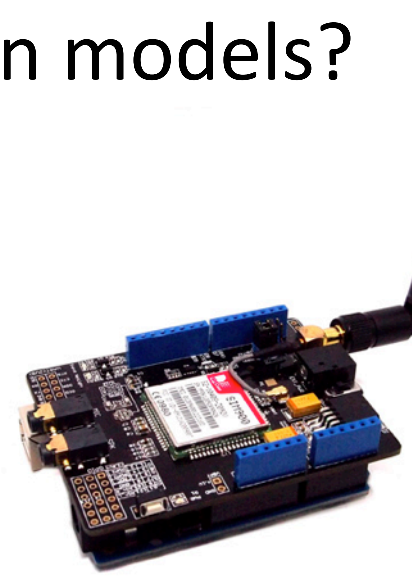
ari.keranen@ericsson.com

Scope

- Cellular networks
  - Large-scale, public, point-to-point, radio networks
- When power saving is important
  - Battery operation
  - Energy harvesting
  - …
- Optimize the system, not just the radio layer

Background

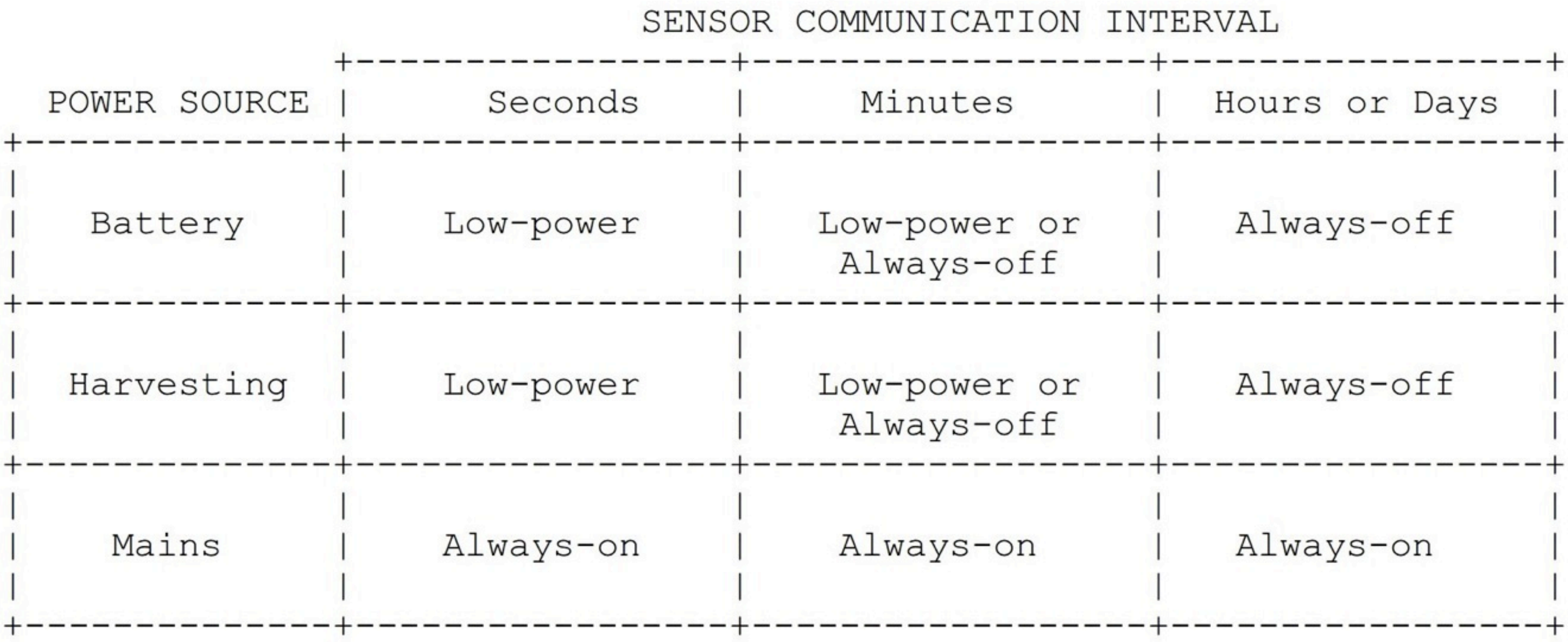
- Low-power cellular prototype
  - Arduino + GPRS shield + solar power cell + sensor = “infinite lifetime sensor”
  - With low-power CoAP Client
- Suitable communication models?



Power Usage Strategies

- Always-on – self-explanatory
- Always-off – wake-up infrequently, perform full attachment, communicate, detach, sleep
- Low-power – all other attempts to minimize power consumption while keeping some state and attachment status across periods of sleep

Types of Devices and Power Strategies



	SENSOR COMMUNICATION INTERVAL		
POWER SOURCE	Seconds	Minutes	Hours or Days
Battery	Low-power	Low-power or Always-off	Always-off
Harvesting	Low-power	Low-power or Always-off	Always-off
Mains	Always-on	Always-on	Always-on

Link-Layer

- Public, generic-use network
  - No app-specific discovery or configuration support
  - Possibly limited reachability (e.g., NATs)
- Point-to-point link
  - No multicast discovery
  - (Private APN)
- Long-range radio technology
  - Transmission takes significant amount of energy
  - Periodic checks for messages (paging)

Some Possible Recommendations

- Protocol: CoAP – less round trips; small packet size
- Data formats: JSON/SENML – smaller than XML; easier than binary
- Communications frequency – per application needs; possibly bundle
- Discovery – see next slides
- Communications model – see next slides

Discovery

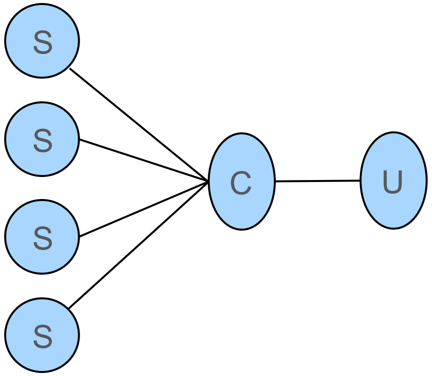
- No a priori address assignment in public networks
- Have to register device in a directory to be reachable
  - CoAP directory servers
  - CoAP mirror proxies
  - ...
- But how do we find the directory server?
- Not easy to provide application-specific configuration data via DHCP and other methods in public networks
- No easy solutions: manual configuration, manufacturer burned-in server address, indirection to the real server via the manufacturer [short-term preference], global discovery infrastructure [longer-term solution]
- More work needed

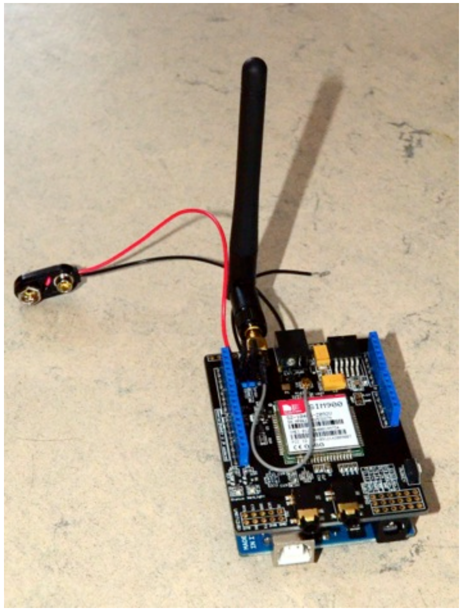
Communications Model

- Two types of devices:
  - Real-time reachable devices
  - Sleepy devices
- Sleepy devices have some freedom how often they need to communicate, e.g., many sensors fall in this category
- Real-time reachable devices are, e.g., light bulbs or other actuators that need to act after a very small delay
- For real-time reachable devices, there is not much choice about the communication model; they need to be servers that can be reached directly
- For sleepy devices, something else works better

Communications Model – Sleepy Devices

- The device should ideally sleep as much as possible
- One good way is the "client" communication model – sending results to a proxy node ("mirror proxy")
- Some cases: "server" model
  - With improved link layer characteristics; less energy is wasted on checking for incoming messages – but still some checking needs to happen
  - Availability signaling





Future Work

- Discovery procedures
- Details of the mirror proxy arrangement
- Understanding the tradeoffs between “low-power” and “always-off” strategies
- Understanding the tradeoffs between improving link layers vs. optimizing application communications better