Web resource identification

Universal Resource Name (URN)
urn:Sensei:sensinode.com:NanoSensor:N740:3a-43-ff-12-01-01

Universal Resource Identifier (URI)

Universal Resource Locator (URL)
http://www.example.org:8080/sensors?id=light

Scheme | Authority | Port | Path | Query
URL Resolution
Example: an HTTP request
From Web App to IoT nodes

- Web Object
  - HTTP
  - TLS / TCP
  - IP

- Binary Web Object
  - CoAP
  - DTLS / UDP
  - IP

- IoT backhaul

- Binary Web Object
  - CoAP
  - DTLS / UDP
  - 6LoWPAN

- IoT node network

Wireless Systems Lab - 2014
CoAP is an application layer protocol tailored for resource constrained devices and M2M applications.

- RESTful protocol designed from scratch
- Transparent mapping to HTTP
- Additional features for M2M scenarios (low overhead, multicast support)

**Request/Response Sub-layer**
- RESTful interaction

**Message Sub-layer**
- Reliability

**CoAP**
- GET, POST, PUT, DELETE
- URIs and Internet Media Types
- Deduplication
- Optional retransmissions
  - Confirmables “CON”
  - Non confirmable “NON”
  - Acknowledgment “ACK”
  - Reset “RST”
Constrained Application Protocol

Binary protocol
• Low parsing complexity
• Small message size

Options
• Numbers in IANA registry
• Type-Length-Value
• Special option header marks payload if present

4-byte Base Header
Version | Type | T-len | Code | ID

0 – 8 Bytes Token
Exchange handle for client

Options
Location, Max-Age, ETag, ...

Marker
0xFF

Payload
Representation

Wireless Systems Lab - 2014
CoAP Request Example

Confirmable Request
Piggy-backed Response
Dealing with packet loss
Separate Response

CoAP Client

CON [0x1b] GET /light Token: 0x31

ACK [0x1b]

takes too much time

CoAP Server

CON [0x823] 2.05 Content /light Token: 0x31 "<light>..."

ACK [0x823]

/light ready
bits and Byte..

CLIENT

---- CON [0x7d34] GET /temp ---->

SERVER

0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1
0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1
++-------------------------------++-------------------------------++
| 1 | 0 | 0 | GET = 1 | MID=0x7d34 |
|++-------------------------------++-------------------------------++
| 11 | 4 | "temp" (4 B) ...
++-------------------------------++-------------------------------++

CLIENT

<-------- ACK [0x7d34] 2.05 Content --------

SERVER

0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1
0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  0  1
++-------------------------------++-------------------------------++
| 2 | 0 | 2.05=69 | MID=0x7d34 |
|++-------------------------------++-------------------------------++
| "22.3 C" (6 B) ...
++-------------------------------++-------------------------------++

Wireless Systems Lab - 2014
RESTful group communication

GET /status/power

PUT /control/onoff

PUT /control/color
   #00FF00

all-lights.floor-d.example.com
Observing resources - NON mode

current representation of a resource over a period of time

Resource state at origin server

Server

Client

Replicated state at client

Observe illustration courtesy of Klaus Hartke

Wireless Systems Lab - 2014
Observing resources - CON mode

current representation of a resource over a period of time

Resource state at origin server

Server

GETObserv

Notification

Notification

Notification

Retransmission

Notification

Client

Replicated state at client

Observe illustration courtesy of Klaus Hartke
Resource discovery

Based on Web Linking (RFC5988)
Extended to Core Link Format (RFC6690)

GET /.well-known/core

</config/groups>;rt="core.gp";ct=39,
</sensors/temp>;rt="ucum.Cel";ct="0 50";obs,
</large>;rt="block";sz=1280,
</device>;title="Device management"
Security

Based on **DTLS** (TLS/SSL for Datagrams)
Focus on Elliptic Curve Cryptography (**ECC**)
Pre-shared secrets, certificates, or raw public keys

Hardware acceleration in IoT devices

IETF is currently working on
- Authentication/authorization (ACE)
- DTLS profiles (DICE)

e.g.,

Wireless Systems Lab - 2014
Status of CoAP

Proposed Standard since 15 Jul 2013

RFC 7252

Next working group documents in the queue

- Observing Resources
- Group Communication
- Blockwise Transfers
- Resource Directory
- HTTP Mapping Guidelines
Status of CoAP

In use by

- OMA Lightweight M2M
- IPSO Alliance
- ETSI M2M / OneM2M

- Device management for network operators
- Lighting systems for smart cities
Proxy

CoAP Server → Proxy → HTTP Client

CON GET /light
ACK max-age=30s 2.05 Content "<light>..."

HTTP GET /light
200 OK "<light>..."

cache /light

HTTP GET /light
200 OK "<light>..."

cache fresh

Wireless Systems Lab - 2014
Getting started with CoAP

There are many open source implementations available

- Java CoAP Library Californium
- C CoAP Library Erbium
- libCoAP C Library
- jCoAP Java Library
- OpenCoAP C Library
- TinyOS and Contiki include CoAP support

- CoAP is already part of many commercial products/systems
  - Sensinode NanoService
  - RTX 4100 WiFi Module

- Firefox has a CoAP plugin called Copper
- Wireshark has CoAP dissector support
- Implement CoAP yourself, it is not that hard!
Using CoAP on Android

1. Create POST requests
2. Use POST method on leds
3. Send HTTP request to the proxy
4. Turn on/off TelosB leds

Proxy: [http://192.168.0.10:8080/proxy/](http://192.168.0.10:8080/proxy/)

Coap: coap://[aaaa::212:7400:1024:1bf3]:5683/actuators/leds?color=r/b/g

POST payload: mode=on/off
Some hints

<uses-permission android:name="android.permission.INTERNET" />

USE: HttpClient and HttpPost

post.setEntity(new UrlEncodedFormEntity(pairs));

pairs can be a list<NameValuePair> or JSONObject

You should use threads to handle communication:
  new Thread(new Runnable(){your code}).start

Finally you should execute your HttpPost on your HttpClient
something like: client.execute(post);
Wireless Sensor Network

Made of motes:

– Battery powered (e.g., 2xAA alkaline)
– Wireless (2.4Ghz, 868Mhz, 433Mhz,...)
– Limited computational power
– Limited memory
– Communication interface

• External transducers (e.g., temperature, humidity, pressure,...)
• External transceivers
• External flash memory
WSN motes architecture

Additional units
- GPS
- Motor
- Power generator

Processing unit
- Processor
- Memory

Sensing unit
- ADC
- Sensor

Transceiver unit

Power unit

GPS: global positioning system
ADC: analog to digital converter
Example: Telos B mote
Hardware characteristics

TelosB

- SPI
- UART
- I2C
- MSP430 Texas Instruments
- 16 bit
- 48 Kb ROM
- 10 Kb RAM
- 8 Mhz
- SPI serial bus
- CC2420 Texas Instruments
- 2.4Ghz – 802.15.4
- 250 kbps
- +0 dBm tx power

iPhone 5c

- Apple A6
- 8GB storage
dual-core CPU
- up to 1.5 GHz

Wireless Systems Lab - 2014
Energy consumption

iPhone 5C
- Power: Non-removable Li-Po
- Expensive!
- Battery lasts < 1 day

TelosB
- Power: 2xAA alkaline batteries
- Cheap!
- With power management, battery lasts up to a couple of years
• Open-source OS for embedded systems
• Open-source
  ○ Source code easily reusable
  ○ Large developers community
• Very small, flexible “programming framework”
  ○ Less than 400 bytes!
• Written in NesC (a dialect of C)
Why a dedicated operating system?

● Motes are intentionally tiny!
  ○ 1-MIPS processor
  ○ 10KB of storage
  ○ Low power

● Special needs for sensor networks
  ○ Real-time
  ○ Reactive concurrency
  ○ Flexibility
TinyOS

Native support for low-power operations:

• Microcontroller Power Management
  • Microcontrollers should always be in the lowest power state possible
  • TinyOS handles state transitions automatically to achieve maximum power saving

• Radio Power Management
  • Duty-cycle radio to save energy and extend network lifetime

• Peripheral Energy Management
  • Energy-efficient scheduling of sensing operation and peripheral access
Mobile network states

1. Radio Standby
   - Radio idle for 12 seconds
   - Current: 10mA

2. Radio Low Power
   - Current: 300mA
   - Transmitting
   - 2s latency

3. Radio Full Power
   - Current: 150mA
   - Not Transmitting (idle)
   - Radio idle for 5 seconds
   - 1.5s latency

4. Dormant

http://www.slideshare.net/littleeye/power-optimization-for-android-developers

Wireless Systems Lab - 2014
Hanging socket

- Program your apps to close sockets when done

If network is used 4 times per hour -> 10 hours of standby instead of 8 (20% improvement!)

Wireless Systems Lab - 2014
Ungrouped vs grouped network activity

- Ungrouped network (upper graph): ~9000 mA
- Grouped network (lower graph): ~5000 mA
Projects

● Build your own walking detector app:
  ○ Use one or more low-power sensors (sensors fusion)
  ○ Experiment with different features
  ○ Collect your own training dataset (friends can help)
  ○ Test it!
● Build an Indoor Localization app
  ○ RSSI
  ○ DR
  ○ Try some filter e.g., EKF for sensor fusion and navigation
  ○ Acoustic
● CoAP and sensors
  ○ TelosB + Android Localization