CoAP Protocol Negotiation

draft-silverajan-core-coap-protocol-negotiation

Bill Silverajan TUT
Mert Ocak Ericsson

Summary of changes from -02

- Restructuring for easier editing
- Scenarios and examples added
- Node classification based on transport types
- CoAP transports can have "al" (active lifetime) attribute

Forthcoming change:
Usage of URI Templates

- Change this operation:

```
Client ----> GET /.well-known/core?tt=* ---> Server

Client <--- 2.05 Content, tt="tcp sms" <---  Server
```

- Into this operation:

```
Client ---- GET /.well-known/core?rt=core.pn ----> Server
          Content-Format: application/link-format

Client <-- 2.05 Content"</pn>;rt="core.pn";ct=40 <--- Server
```

- Introduce a discovery interface for CoAP transports:

  Method: `GET`
  URI Template: `/.well-known/core`
  URI Template: `/{+pn}{?q*}`

  Example Request: `GET /pn?tt="tcp"`

Proposal:
Client-Initiated Transport Negotiation

- In version -03, waking up an inactive transport is implicit:

```
Client ----> GET coap+sms://0012345/.well-known/core?tt=udp ----> Server

Client <--- 2.05 Content, <coap://example.org/>;rel_"altloc";al=120 <-- Server
```

- Work for version -04: New CoAP option
  - For clients to request activating server's inactive transport
  - Prevent transport from going inactive (eg by extending lifetime)

- Example 1:

```
Client ----> GET coap+sms://001234567/pn?tt=udp ----> Server

Client <--- 4.04 "Not Found" <-- Server
```

- Example 2:

```
Client ----> GET coap+sms://001234567/pn?tt=udp ----> Server
           OPTION ACTIVE_TRANSPORT

Client <-- 2.05 Content <coaps://example.org/>;rel="altloc";al=120;tt=udp <-- Server
```

- Alternatives to above approach?

Session continuation

- Mechanism for client to inform server to continue session over a different CoAP transport
  - Many pitfalls envisaged (Observe, Block Transfers, switching to less secure channel)
- Go/no-go decision to explore?