Best practices for HTTP-CoAP mapping implementation

draft-castellani-core-http-mapping-01

Angelo P. Castellani, Salvatore Loreto, Akbar Rahman, Thomas Fossati and Esko Dijk

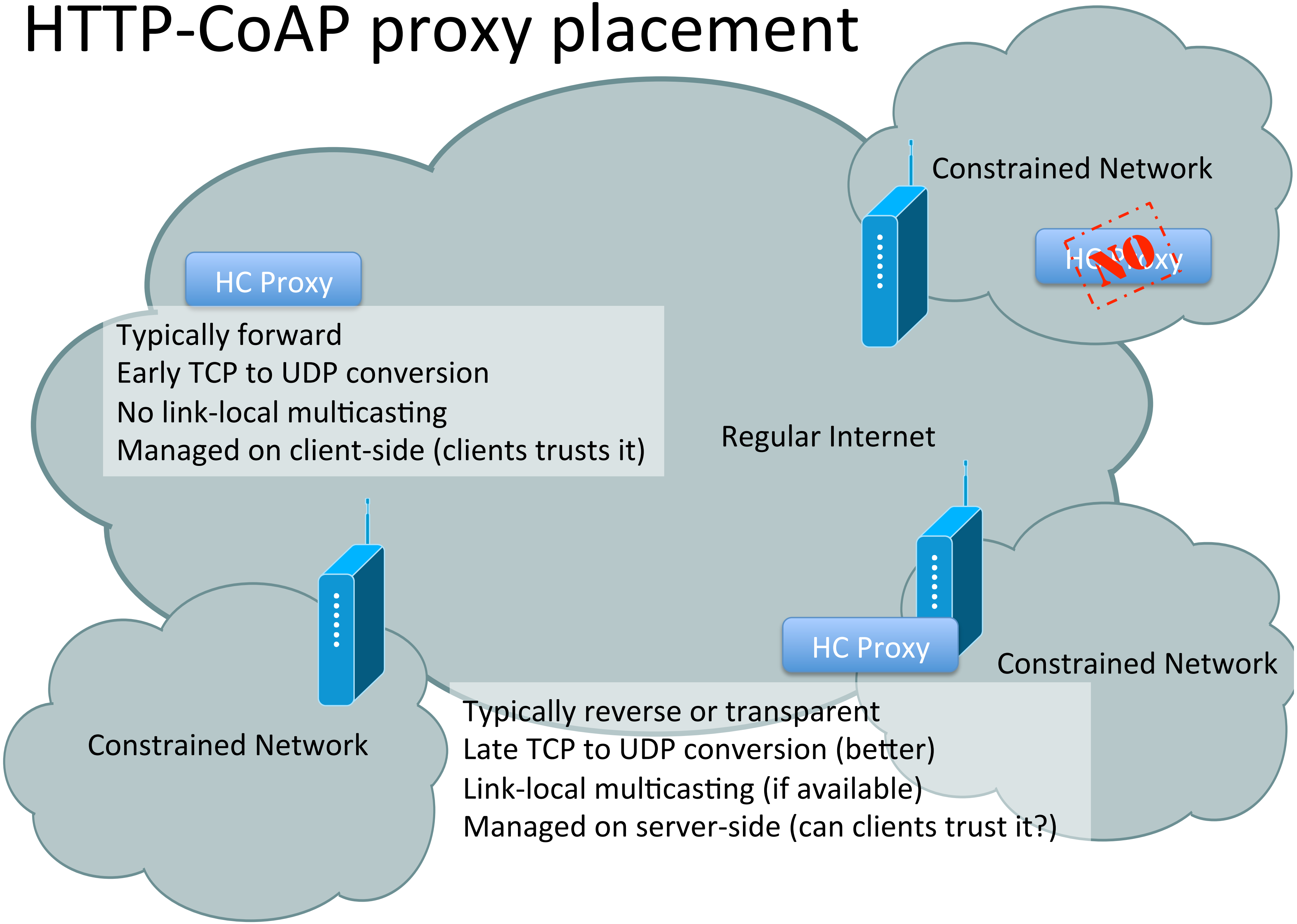
Introduction

- The I-D provides a base reference documentation for HTTP-CoAP (HC) proxy implementers
- It details deployment options, discusses possible approaches for URI mapping, and provides useful considerations related to protocol translation
- The HC proxy does NOT target running on a constrained device (Class 1 or 2)

Cross-protocol proxies taxonomy

- Forward
  - It is explicitly known by the client
- Reverse
  - Acts as if it was the origin server
  - It knows explicitly the servers that is proxying
- Interception [RFC3040]
  - Receives requests through network interception
  - Zero configuration or discovery of the endpoints

HTTP-CoAP proxy placement



Cross-protocol URI

- Protocol-aware
  - Client uses the scheme specific to the protocol
    - **Example**: An HTTP client accesses coap://node.something.net/foo directly
- Protocol-agnostic
  - Client uses its natively supported scheme
    - **Example**: An HTTP client accesses coap://node.something.net/foo at an http: URI
      - The client does not even need to know the coap: URI
  - Requires cross-protocol URI mapping

URI mapping

- It is a mechanism to map a URI across two different scheme domains
  - Example: coap://node.something.net/foo is mapped to http://something.net/node/foo
- Could be complex in general
  - **Static**: the mapping does NOT change over time
  - **Dynamic**: the mapping can change over time

URI mapping examples

- Homogeneous
  - Only the scheme part of the URI changes, authority and path stay the same
    - **Example**: coap://node.something.net/foo is mapped to http://node.something.net/foo
    - Interception proxy deployments MUST use this mapping
- Embedded
  - All but the scheme part of the URI is embedded as-is in the mapped URI
    - **Example**: coap://node.something.net/foo is mapped to http://example.com/node.something.net/foo
    - Reduces mapping complexity in reverse proxy deployments

Cross-protocol URI handling

- Identification of cross-protocol URIs
  - **Example**: the proxy knows that http://node.something.net/foo is a HTTP-CoAP resource and should be mapped
- Apply correct URI mapping
  - **Example**: the mapping required by that URI is homogeneous, the final coap URI is coap://node.something.net/foo

Cross-protocol URI handling (cont.)

- RFC 3986, Appendix B says:
  - Any URI can be completely parsed through a POSIX regular expressions
- Regexp-based URL rewriting approach
  - Matching and saving parts of the URI:
    - **Example**: ^http://(.*)
  - Apply saved parts to the destination URI:
    - **Example**: coap://$1
  - Example implements Homogeneous mapping
  - More complex static mappings can easily be done

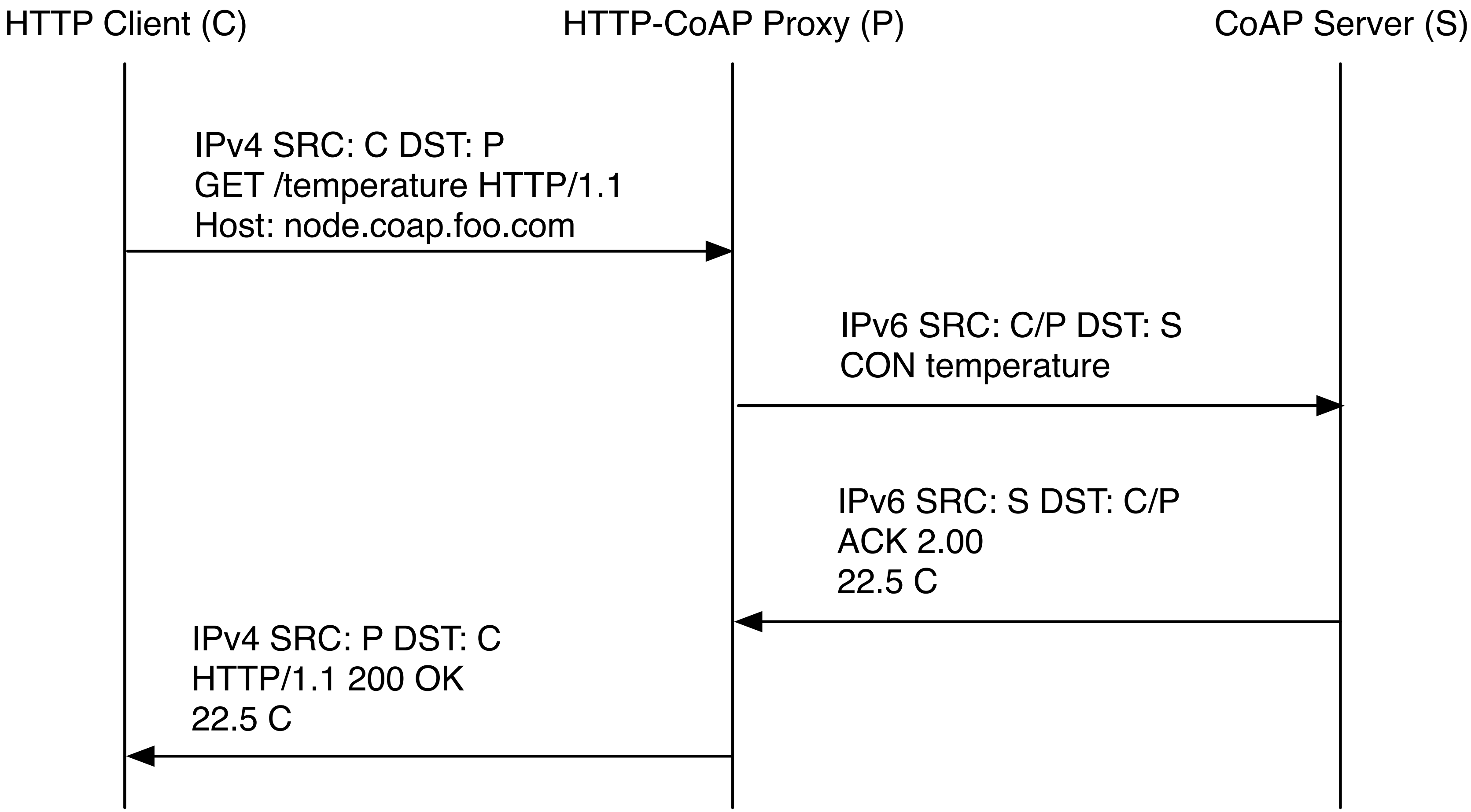
HTTP-CoAP caching and congestion

- An HTTP-CoAP (HC) proxy using caching reduces load on CoAP servers
  - e.g. avoiding duplicate requests
- Observe relationship can be established towards "popular" resources
  - See draft-ietf-core-observe-02
- HC proxy may apply aggregate congestion control towards the same constrained network
  - See draft-eggert-core-congestion-control-01

Cache implementation

- It can be implemented using a combination of:
  - RAM, i.e. using hash maps
  - Disk, i.e. a file per-object
  - VMM/`mmap`'ing, i.e. memory mapped to a big file
- It should implement a mechanism to rate the popularity of the cached resource
  - Most popular resources that are accessed at least every X seconds should be "observed"
  - What is a suitable value for X?

HTTP-CoAP v4/v6 use case



DNS A record for node.coap.foo.com points to P

or P is Forward

HTTP unicast --> CoAP multicast

- Identification and mapping
  - The HC proxy understands whether an URI identifies a multicast resource
  - Maps the request to the relevant multicast group
  - The mapping depends on the multicast communication technology in use
    - see draft-rahman-core-groupcomm-06

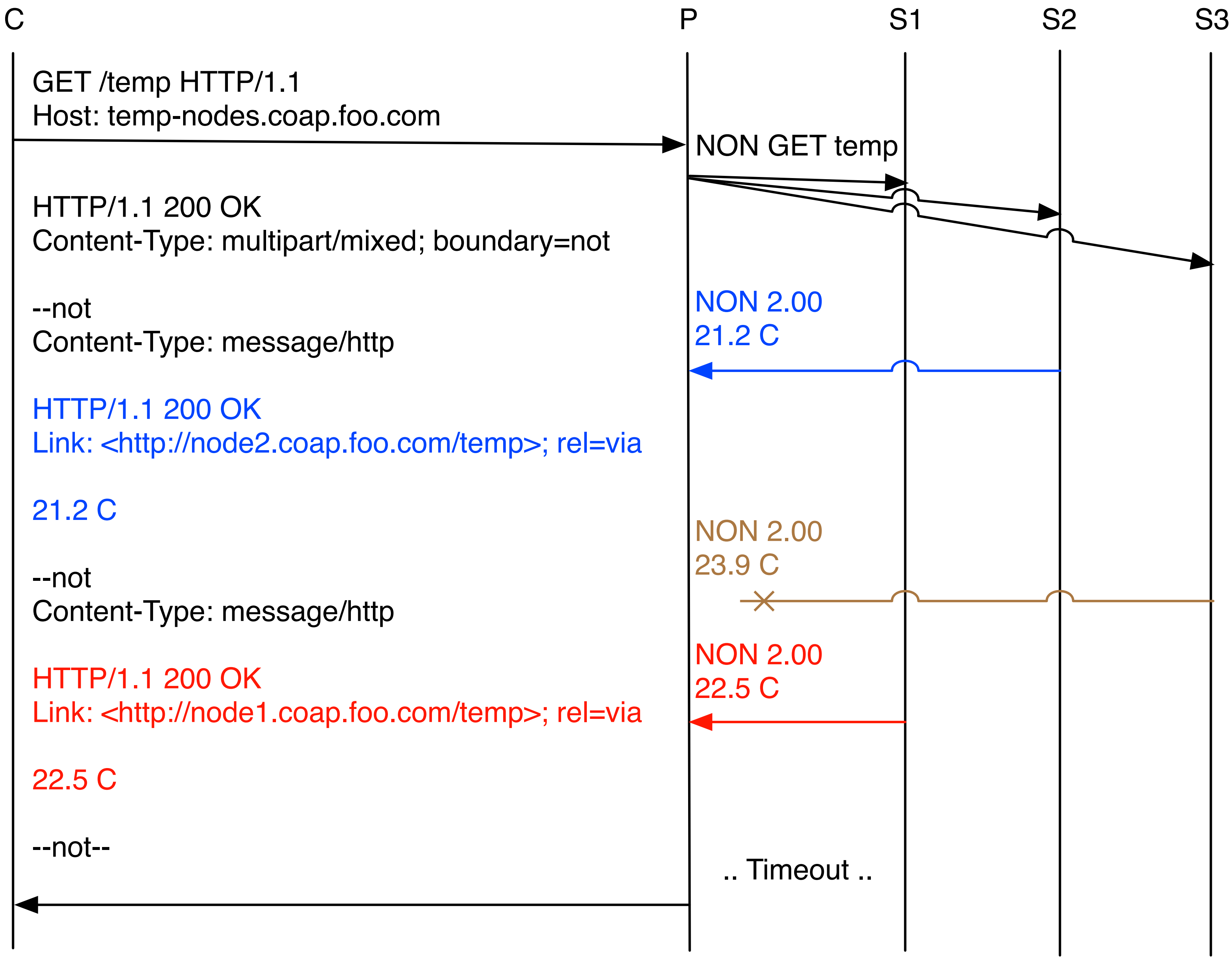
HTTP unicast --> CoAP multicast (cont.)

- Request handling
  - Involves the following tasks
    - Distributing the request
    - Collecting the responses
    - Timeout handling
    - Responses aggregation and delivery
  - Some tasks depend on the multicast communication technology in use

HTTP unicast --> CoAP multicast (cont.)

- Useful features from related standards
  - MIME media type multipart/*
    - Allows to represent multiple CoAP responses in a single HTTP payload.
  - Transfer-Encoding: chunked (HTTP streaming)
    - Enables immediate delivery of responses as soon as they arrive at the proxy.
  - Link format
    - Permits to pair with each actual response the URI of the actual source of that response (otherwise lost)

HTTP unicast --> CoAP multicast (cont.)



Security considerations

- **Availability**
  - **Risk**: Multicast amplification attacks
  - **Countermeasure**: Only known/authorized clients may access multicast resources

  - **Risk**: An high number of subscriptions can cause resource exhaustion
  - **Countermeasure**: Limit the number of concurrent subscription requests

Security considerations (cont.)

- **Integrity**
  - **Risk**: Cache poisoning on the CoAP side by an evil mote spoofing the response (feasible when using NoSec or even SharedKey).
  - **Countermeasure**: Use MultiKey with 1:1 identity binding, or SharedKey with procedurally secure mote crypto enrollment.

Security considerations (cont.)

- **Confidentiality**
  - A resource requested via a secure channel by the source SHOULD be mapped to a secure request (if possible) or rejected.