Connectivity 2

Internet refresher …

TCP/IP

IP

Private & public addresses

Routing, NAT and Firewalls

Host names (DNS)

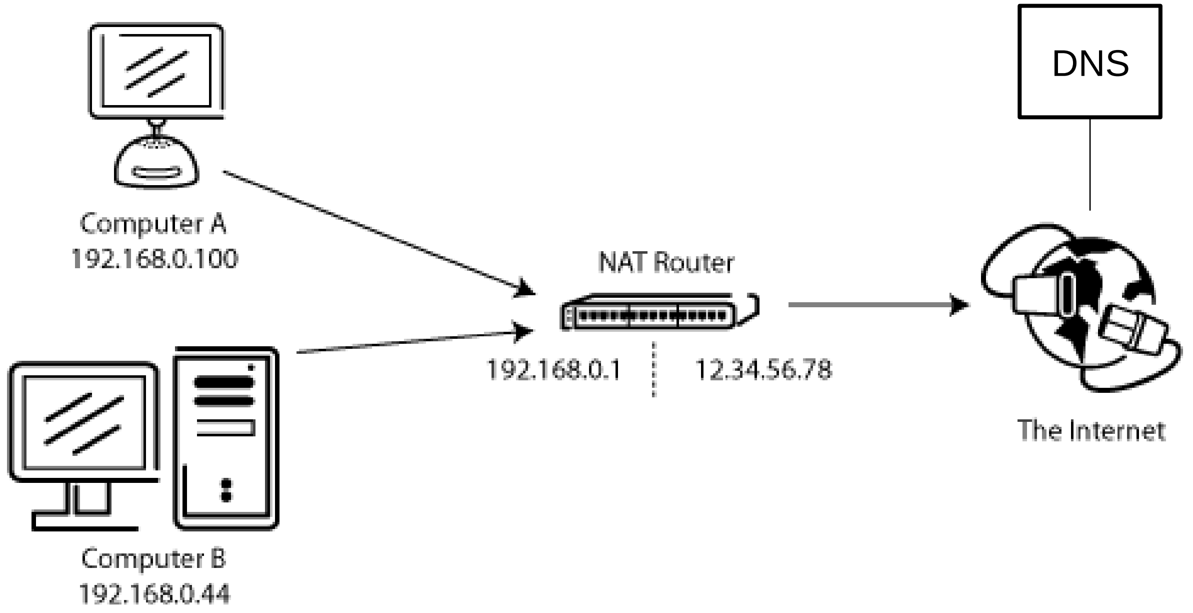
Load balancing

- DNS round robin
- Virtual IP

TCP & UDP

Ports (services)

Error control & ordering



HTTP Request

Method

GET, POST, PUT, DELETE …

Headers

Host

Accept (content type, encoding ….)

Authorization

Cache-Control

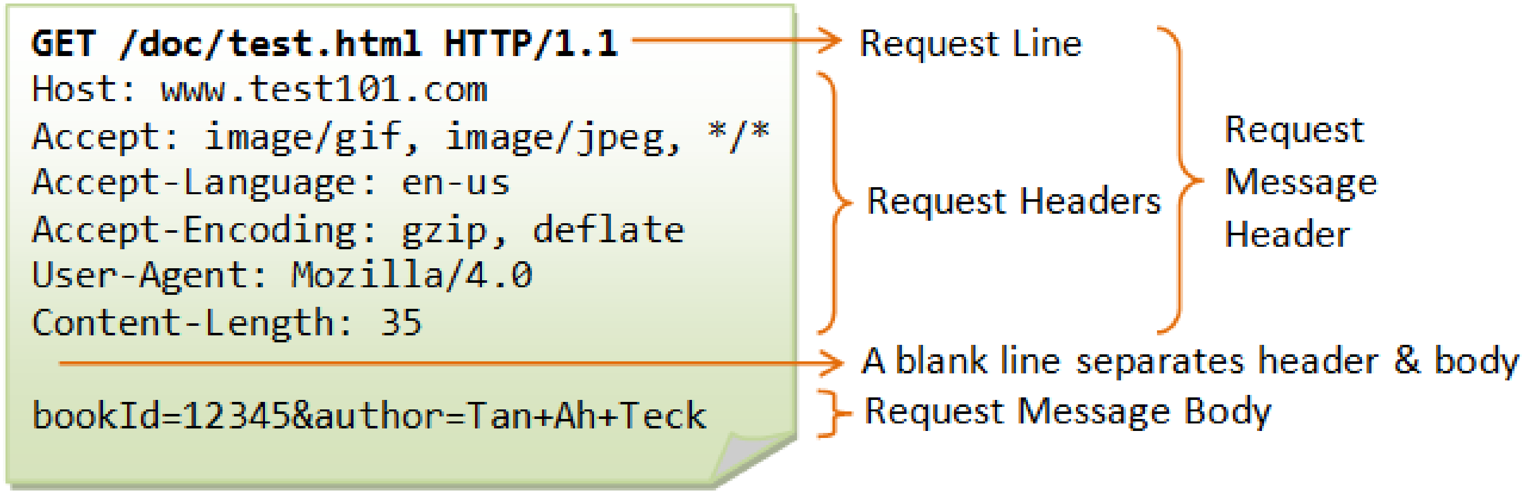
Cookies

Content-Type

Body

Application specific (e.g. JSON, XML …)

POST and PUT methods only



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

HTTP Response

Status line

Protocol version

Status

Headers

Access-Control-Allow-Origin

Cache-Control

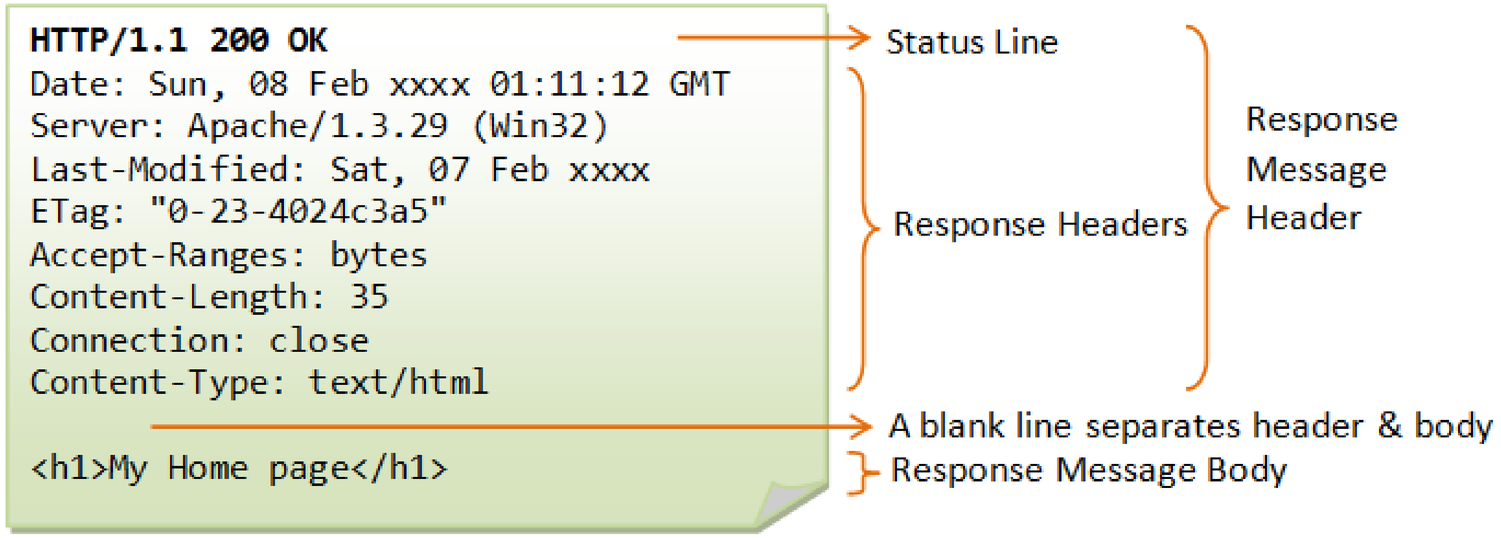
Content-Type

Set-Cookie

…

Body

Application specific (e.g. JSON …)



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

MQTT

MQTT (ISO/IEC PRF 20922)

Overview

TCP/IP based

- MQTT-SN (UDP)

Small footprint / low bandwidth

e.g. compared to HTTP

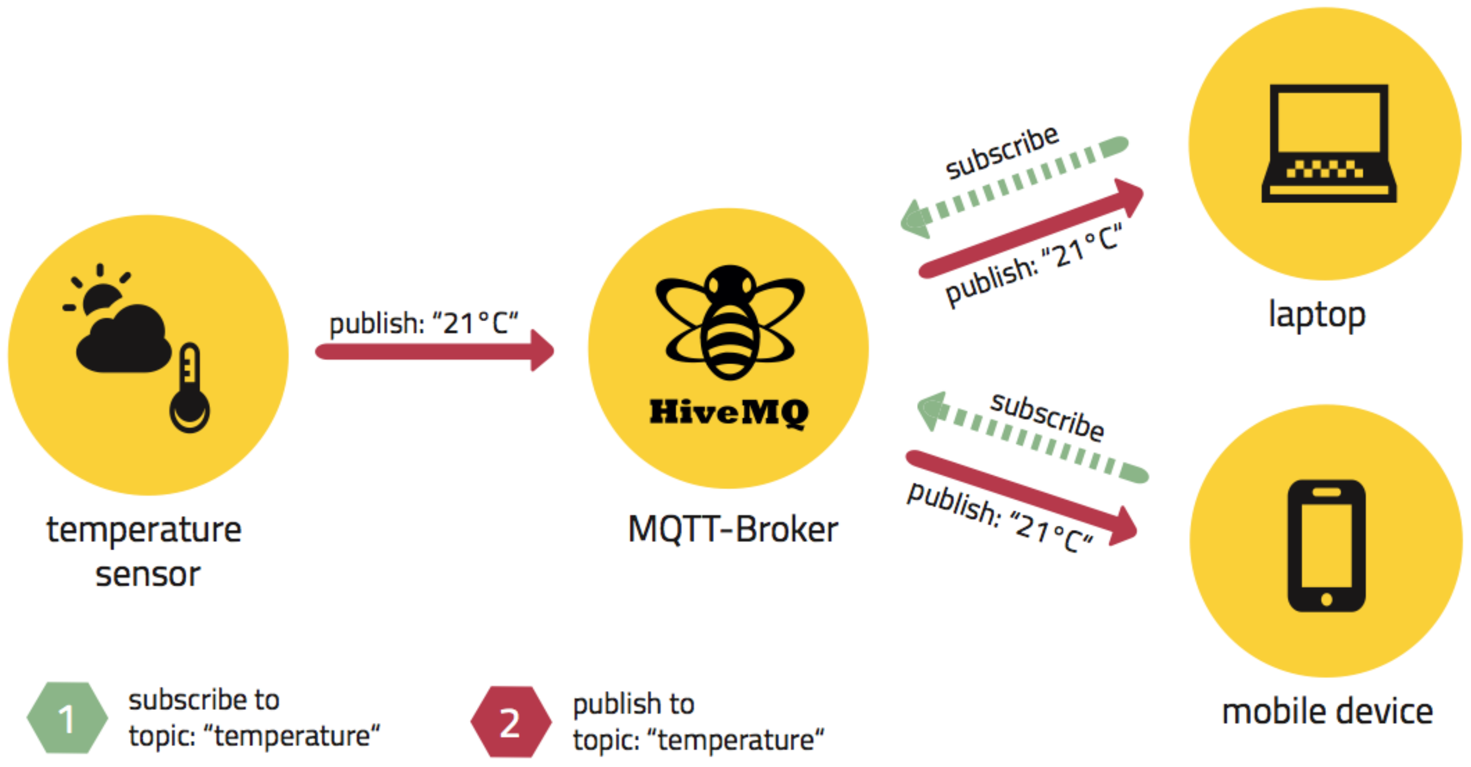
Pub/sub

Client connect to **broker**

And publish to **topics**

Or subscribe to **topics**

- Wildcard subscription
- Retained messages



MQTT implementations

Servers

Open source: Eclipse Mosquitto, HiveMQ, VerneMQ …

Cloud: AWS, Azure, GCloud …

Clients

ESP32: pubsubclient, lwmqtt, Paho

Linux: Paho

…

MQTT Quality of Service (QoS)

QoS Level

Sender defines message QoS

Subscriber defines its supported QoS during subscription

Does not apply to TCP, only to sender-receiver connection

Levels

- 0: fire and forget (at most once)
- 1: resend until acknowledged (at least once)
- 2: exactly once delivery (exactly once)

WARNING

Cloud providers may deviate from the MQTT QoS specification

Practical tips

- Use 0 when you don’t care too much about lost messages
- Use 1 when you can afford duplicate messages (e.g. deduplicating on the receiver side)
- Avoid 2
  – buffer locally (e.g. in-memory) when delivery fails
  – consider sending full actual/desired state with every message

MQTT security

Encryption

TLS on top of TCP/IP

Authentication

Username / password

Access key

Certificate / PK