h2ot: HTTP/2 for IoT

96 IETF Berlin, 22-Jul-2016

Gabriel Montenegro
Sandra Céspedes
Salvatore Loreto
Robby Simpson

Communication Patterns in IoT

	Constrained Network scenario	*Internet scenario*
Node-to-node	X	
Node to gateway	X	
Gateway to cloud		X
Node to cloud		X

NOTE: Internet traffic is assumed to be carried over TLS

Motivation for HTTP/2 for IoT

Or, *why we should give mainstream protocols a chance*

- Lessons from WAP
  - “wireless is different” ➔ creation of a purpose-built stack for mobile (cellular) networks
- Current IoT landscape
  - Multiple purpose-build stacks and protocols because “IoT is different”
  - Some of this is going on within the IETF
- Proliferation of purpose-built stacks is really bad for security, the #1 problem with IoT
- Less obvious in Internet scenario, yet stacks also seen there

Common stack elements

- HTTP/2 as application transport
- DNS-SD, mDNS multicast for discovery
- Authentication
  - OAUTH profile under way in ACE
- Data Models discussions ongoing
  - *Potentially* independent of transport
  - Not so in reality: HTTP/2 binding for LWM2M not defined (e.g., Server PUSH for pub/subscribe functionality)

	HTTP/1.1 (over TLS)	HTTP/2 (over TLS)	MQTT	AMQP	CoAP
General Protocol (vs vertical app protocol)	General	General	Vertical (e.g., automotive)	Vertical (financial services etc messaging middleware)	General
Standards ready (yes, no, partially)	Partially (Works, but not optimized)	Partially (Works, but not optimized)	Yes (but ongoing)	Yes	Yes (but ongoing)
Developer Mind Share (Eclipse survey in 2016 and 2015)	61%, 63%	19%, 0%	52%, 53%	14%, 11%	21%, 21%
Transport Used (UDP vs TCP)	TCP	TCP UDP being defined via QUIC	TCP (UDP experimental)	TCP	UDP (TCP being defined)
Compact (e.g., binary)	No	Yes	Yes	Yes	Yes
Class of devices targeted (RFC 7228)	Class 2	Class 2, maybe Class 1	Class 2, maybe Class 1 (e.g., impl <30k)	Unknown, but maybe Class 2	Class 1
Firewall issues (Many, few, some)	Few	Few	Some	Some	Many

Notes

Eclipse IoT Developer Survey

- *2016: http://iot.ieee.org/images/files/pdf/iot-developer-survey-2016-report-final.pdf*
- *2015: http://www.slideshare.net/IanSkerrett/iot-developer-survey-2015*

Classes of devices per http://tools.ietf.org/html/rfc7228#section-3:

```
+-------------+-----------------------+-------------------------+
| Name        | data size (e.g., RAM) | code size (e.g., Flash) |
+-------------+-----------------------+-------------------------+
| Class 0, C0 | << 10 KiB             | << 100 KiB              |
|             |                       |                         |
| Class 1, C1 | ~ 10 KiB              | ~ 100 KiB               |
|             |                       |                         |
| Class 2, C2 | ~ 50 KiB              | ~ 250 KiB               |
+-------------+-----------------------+-------------------------+
```

Importance of Protocol Reuse

- Security is more challenging than usual (no physical security, constrained devices)
  - Lots of research and attention
- Several protocol stacks at different maturity levels at play and coexisting in some nodes (gw's, cloud, etc)
  - Issues other than cryptography
  - Software engineering and silly bugs
  - Already commonly identified (shodan) and expected to become much worse (surveillance agencies and others are salivating)
- Many stacks impose the use of gateways for the foreseeable future

HTTP/2: the best *general* alternative

- By far, the most reliable alternative for internet scenario (firewall issues)
  - Best bet: TCP on port 443
- Only alternative suitable for both *constrained* and *internet* scenarios.
  - Given the limits of code space, constrained devices benefit from a single stack for multiple scenarios.
  - Security argument: Better to have only one stack and not twice the attack surface
- The power of mainstream (yes, given current deployment/usage numbers) analogous to benefits of IP in https://tools.ietf.org/html/rfc4919#section-3
  - Use of existing infrastructure
  - well-known technology
  - implementations and libraries available
  - tools for diagnostics etc available
  - no need for intermediaries so e2e option is available

HTTP/2 as a good match for IoT

- A more modern transport
  - Binary and compact: 9 byte header
  - small code size
  - resource-friendly header compression
  - reuse of a single TCP connection
  - PUSH for subscriptions
- transport security negates advantages (at least in Internet scenario)
  - Multicast often unusable
    - DICE WG entertained multicast extensions for DTLS
    - From a security point of view this is a HUGE undertaking, has been tried before, and may never pan out
  - After adding DTLS/TLS overhead (**12 octets or so**), fixed Header size difference is a smaller portion, e.g.:
    - HTTP/2 header: 9 octets ➔ 21 octets
    - CoAP only: 4 (plus 1+ with options) ➔ 16+ octets
    - NOTE: QUIC apparently improving upon this
- Reliability, congestion control
  - Other end up reinventing much of the TCP wheel
  - If one wishes to do so, QUIC is probably the best bet

IoT Profile for HTTP/2

- HTTP/2 parameter considerations
  - SETTINGS_HEADER_TABLE_SIZE: e.g., 512 (versus 4096)
  - SETTINGS_ENABLE_PUSH: 1 (this is the default, but 0 ok in some scenarios)
  - SETTINGS_MAX_CONCURRENT_STREAMS: value: 1 or 2 or 3? (versus infinite)
  - SETTINGS_INITIAL_WINDOW_SIZE: value: few kb (versus 64K)
  - SETTINGS_MAX_FRAME_SIZE: could leave large (e.g., 16K) and use flow control
  - SETTINGS_MAX_HEADER_LIST_SIZE: few kb (versus infinite)

HTTP/2 as an important component in IoT

- This draft is just a beginning
- Asking for others interested to work together
- Performance measurements and comparisons
- Implementations
- Longer-term HTTP improvements for IoT
- Please contact us: draft-montenegro-httpbis-h2ot@ietf.org

Extra Slides

Other Convergence points

- web linking RFC 6690
  - In Web usage, links are transported in an HTTP header
  - Of course, sending links within the payload (per CoRE's RFC6690) is also possible
- Object compression and encoding (CBOR, etc)
  - Work on data objects is reusable
- DTLS profile: https://tools.ietf.org/html/draft-ietf-dice-profile/
  - Profile applies to authentication modes, hence to TLS itself
  - Reusable for HTTP/2

Application Transport Alternatives and their strengths: CoAP (1/2)

*21% of devs in April 2015/2016 survey**

- Beginning of IoT within the IETF: 6lowpan base publications (2007-2012)
- Need for application layer solution identified early on
- Requirements not met by HTTP/1.1
- ➔ CoAP defined (base publications: 2014-ongoing)

Application Transport Alternatives and their strengths: CoAP (2/2)

- popular in intranet/constrained scenario (node to node, node to gateway)
- UDP is limiting for internet scenario and firewall traversal
- Support for group communication based on experimental multicast mechanism (typically used for discovery).
- Not generally available in cloud services
- Several related drafts to complete the picture:
    - BLOCK draft for TCP-like functionality to transfer large blocks (in RFC Ed queue)
    - OBSERVE draft similar to HTTP/2 PUSH (RFC7641)
    - congestion control in core coap and in separate drafts
    - HTTP mapping draft, etc

Application Transport Alternatives and their strengths (cont…)

- HTTP/1.1: 63% of developers in 2015 survey, 61% in 2016 (!!!)
  - VERY popular still despite its terrible characteristics
  - Widespread know-how
  - Many implementations, tools, support, etc
  - The power of mainstream
- MQTT: 53% of devs in 2015 survey, 52% in 2016
  - Publish/subscribe, created by IBM, now in OASIS
  - popular in internet scenario (node to cloud, gateway to cloud)
  - Nice and small
  - But SSL is nowadays customary on the internet, so some advantage is lost anyways
  - Uses port 8883 for MQTT-over-SSL (1883 without SSL)
  - Firewall issues

Negotiating the HTTP/2 usage profile

- Constrained usage profile:
  - ND option similar to 6CO and ABRO (potentially in DHCPv6 option as well)
  - Signal:
    - Use of HTTP/2
    - Use of TCP header compression
      - TBD, e.g., https://tools.ietf.org/html/draft-aayadi-6lowpan-tcphc/
    - Optional reuse of lower-layer security services (e.g., for 802.15.4)
    - In-the-clear but no Upgrade dance: “prior” knowledge (obtained from HTTP/2 ND option)

- Internet usage profile:
  - ALPN (no longer used for token binding, so less explosion, but still some concern)
  - Prior knowledge based on the application
  - Initial setup based on first message exchange
    - Simpler than general HTTP/2 case: no in-the-clear Upgrade path means the client is always in control of first message

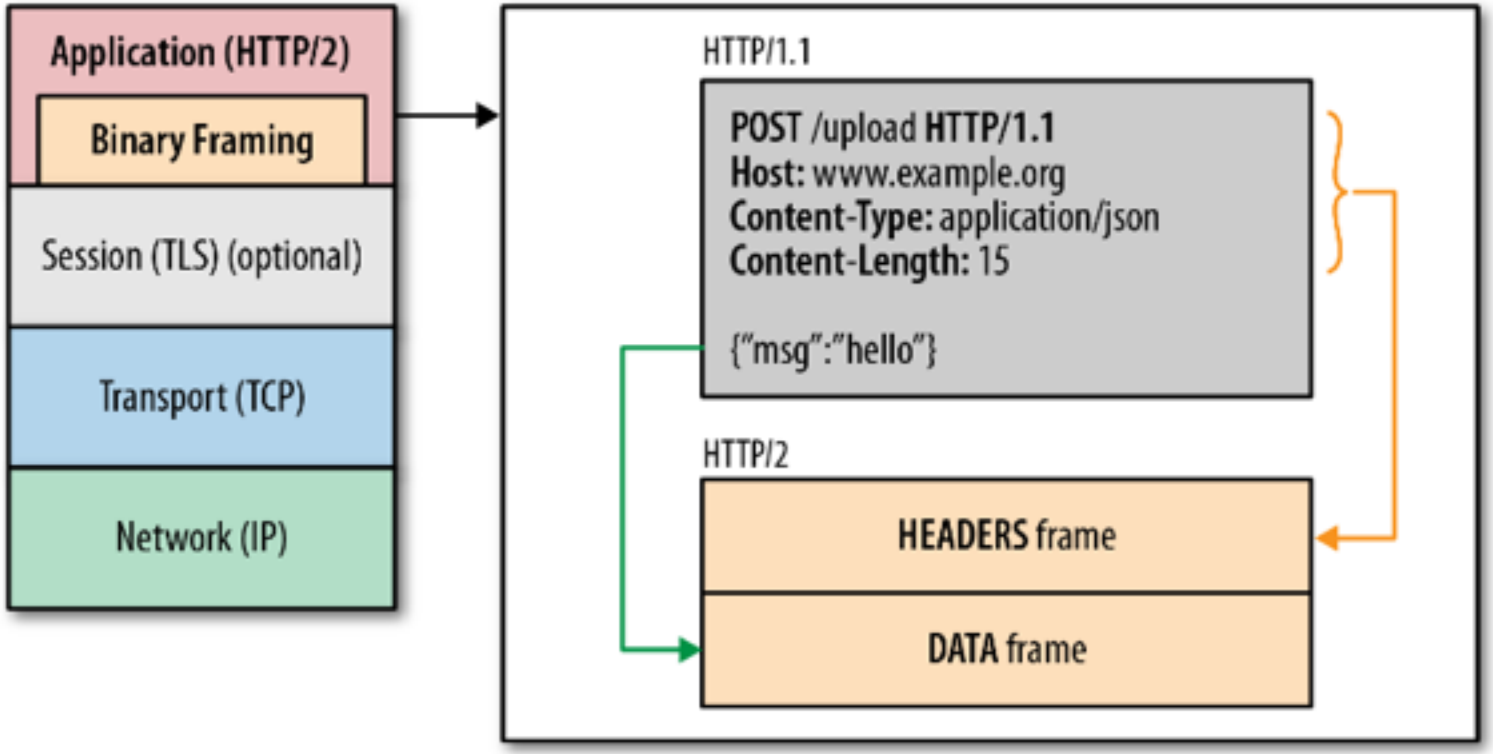
Issues with HTTP/2 for IoT

- Must relax HTTP/2 position on TLS_PSK_WITH_AES_128_CCM_8
  - Preferred for IoT per [I-D.ietf-dice-profile]
  - Black listed by HTTP/2 [RFC7540]
  - Precedence: IPsec requirement for IPv6 was relaxed for RFC4944
- Making the static table truly alphabetical
  - Error prone – some developers may not realize the list is not currently alphabetical
  - Savings – Efficiency gains searching the static table as well as in memory representation
- Adding default values for items in the static table (many do not have default values)
  - Default values allow for much more compact encoding over the wire when available vs a minor tradeoff in additional codespace
  - Avoids possible need to add default value to dynamic table
- Allow the piggybacking of SETTINGS ACKs with SETTINGS
  - Constrained devices will likely need to exchange SETTINGS
  - Avoids sending frames simply for ACK
  - Potentially avoids round-trip wait for SETTINGS ACK (should confirmation be desired prior to data transfer)
- Multicast
  - Yes, it's a can of worms (reliability, security, etc.)
  - However, many IoT use cases (e.g., lighting) require the use of multicast
  - Perhaps achievable with multiple unicasts (similar to 802.11 position on multicast)
- TCP optimizations for IoT
  https://tools.ietf.org/html/draft-gomez-core-tcp-constrained-node-networks

HTTP/2 Status and info

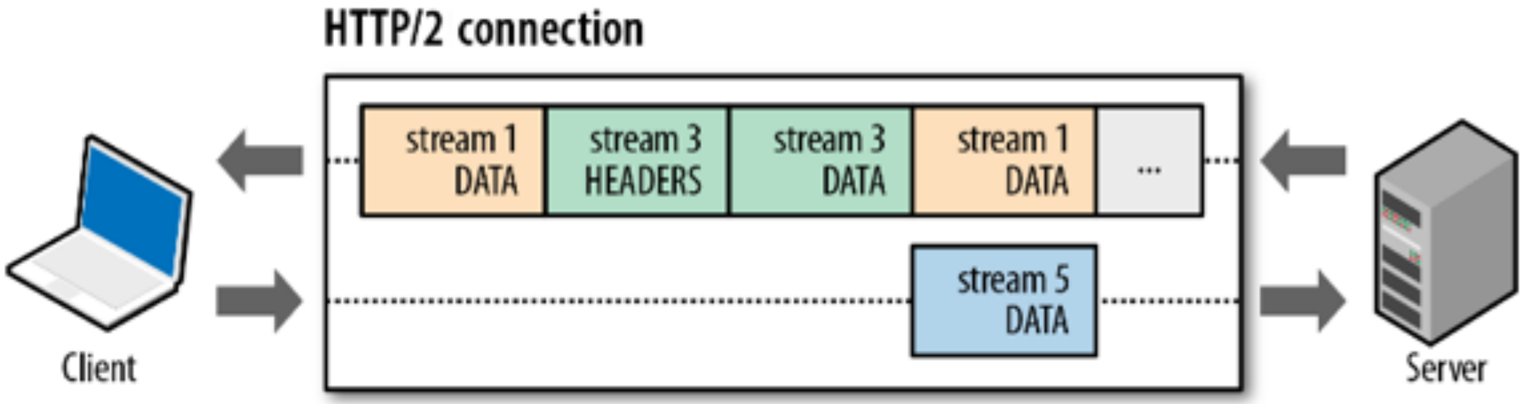
- HTTP/2 page on github maintained by IETF HTTPbis WG:
  http://http2.github.io/
- HTTP/2 is defined by:
  - Hypertext Transfer Protocol version 2 - RFC7540
  - HPACK - Header Compression for HTTP/2 - RFC7541
- Supported in major browsers, clients, servers, proxies, etc
  - https://github.com/http2/http2-spec/wiki/Implementations
- HTTP/2 and IoT
  - On a CC3200 Launchpad board
    http://robbysimpson.com/2015/02/16/first-iot-device-with-http2/
  - Relevant blogs:
    http://robbysimpson.com/2015/01/26/http2-and-the-internet-of-things/
    http://www.limmat.co/2015/02/18/http-2-the-new-iot-protocol/
  - Good intro in *High Performance Computing* by Ilya Grigorik:
    http://chimera.labs.oreilly.com/books/1230000000545/ch12.html

HTTP/2 in one slide



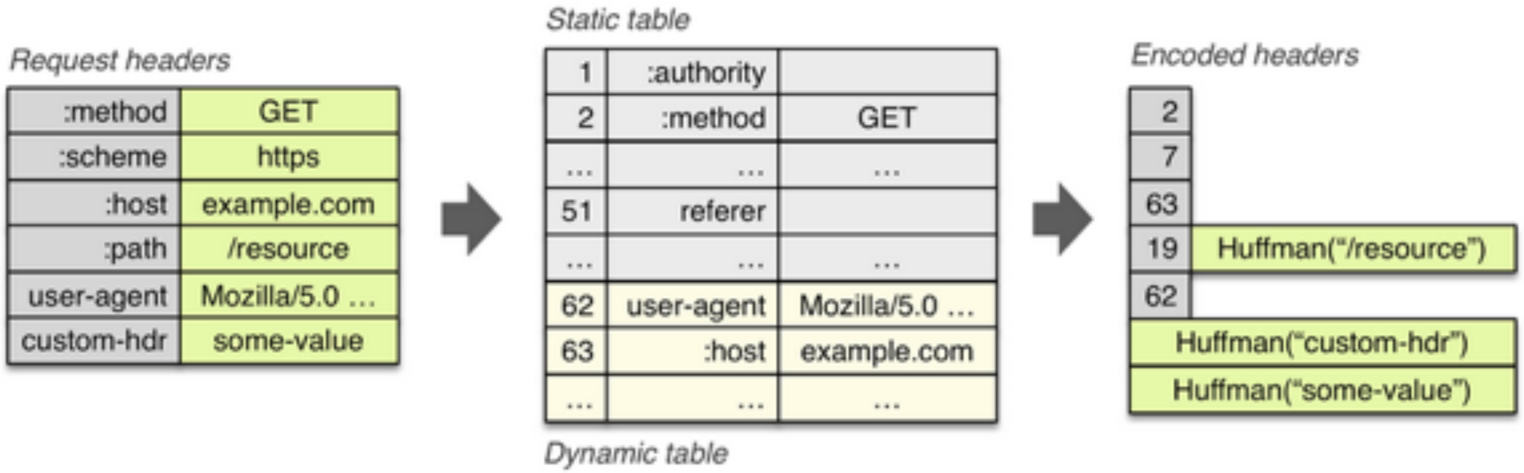
Source: *High Performance Computing* by Ilya Grigorik

HTTP/2 multiplexing



Source: *High Performance Computing* by Ilya Grigorik

HPACK for header compression



Source: *High Performance Computing* by Ilya Grigorik

Common 9-byte frame header

Bit	+0..7	+8..15	+16..23	+24..31
0	Length			Type
32	Flags			
40	R	Stream Identifier		
...	*Frame Payload*			

Source: *High Performance Computing* by Ilya Grigorik