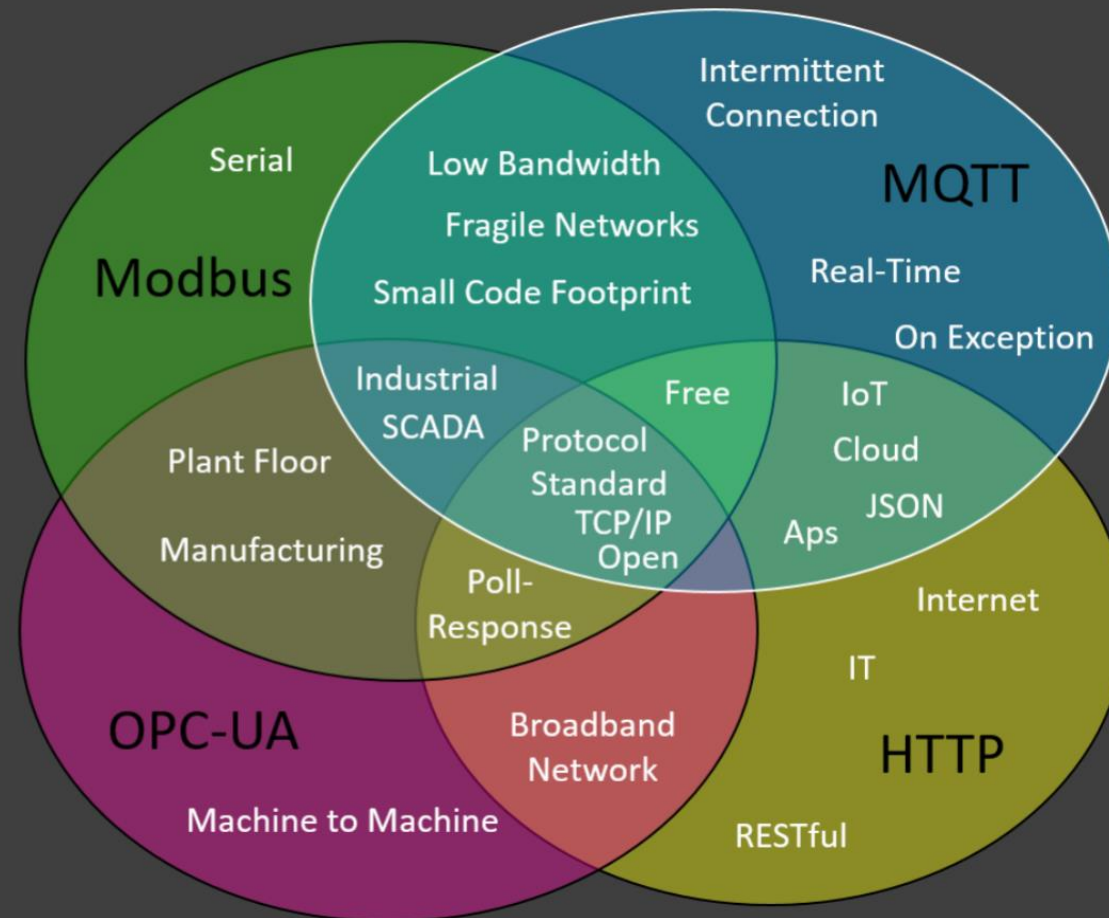


Efficient IIoT Communications

An OPC-UA, HTTP, Modbus and MQTT benchmarking discussion



Johnathan Hottell

April 30 2019

What's Better, Modbus or MQTT?

- That's a loaded question
- It depends...
- There's no way to make an objective comparison without framing better questions first

Agenda

- Discuss the issues that arise when thinking about how to benchmark protocols
- Sending a single value – what is the actual cost
- Sending Nine values and data concentration
- Real world - Sending 389 values for an Oil and Gas wellsite on change
- Final thoughts

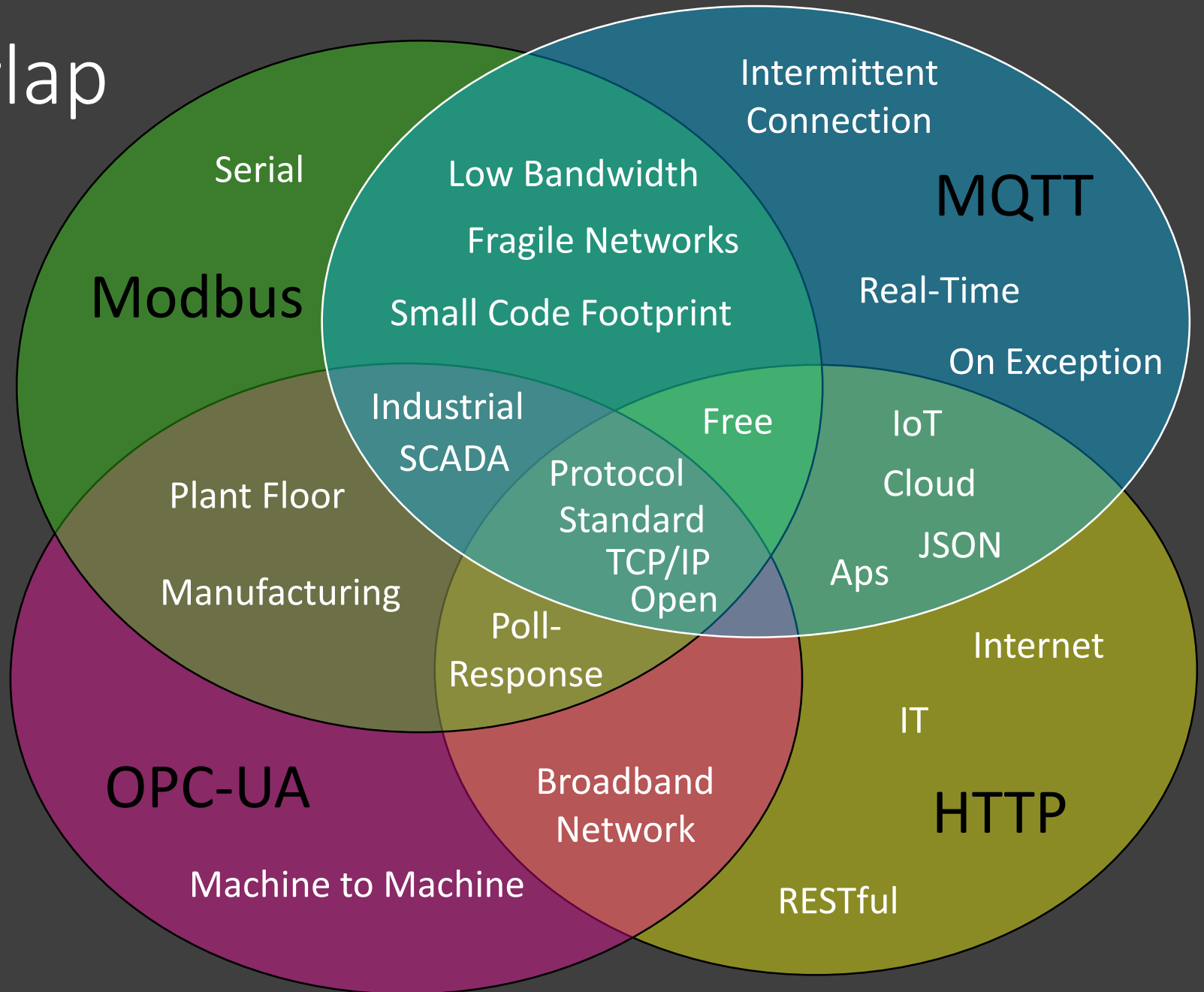
Start by framing the application more clearly

- Best for the Plant floor?
- Best for Interoperable interfaces on a wired network?
- Best for IoT Wired Networks?
- Best for IIoT – Constrained networks?
- Best for battery powered devices?
- Best for distributed real-time SCADA?

Each of these questions has a different answer... and the answers may not be straight forward...

Protocol Use Overlap

- Many protocols can be used in similar scenarios – that doesn't mean they're a perfect fit for them all
- What are your applications most important requirements?



Ideal Conditions vs The Real World

- Expectations vs reality – the results are usually more complicated than you think
- Ideal condition experiment – testing on the bench can be misleading – for instance testing using a 100/1000 network vs on a slow SCADA network
- Be careful to not test to win

Hardware and Software Used for Testing



- Hardware
 - Windows 10 Laptop x 2
 - Raspberry Pi
 - EZ Logix PLC
 - Thermo Scientific AutoPilotPro
 - Moxa Ethernet Switch
- OS
 - Windows
 - Linux Mint
 - Raspbian Stretch
- Software
 - Node-Red
 - Ignition
 - ACM
 - Kepware
 - Wireshark
 - MQTT.fx
 - MQTTSpy

Tag001	499.000000

Use case: Retrieve 1 Value

Retrieve 1 value via OPC, HTTP,
Modbus RTU Encapsulated,
and MQTT

Communication Events That Consume Bandwidth

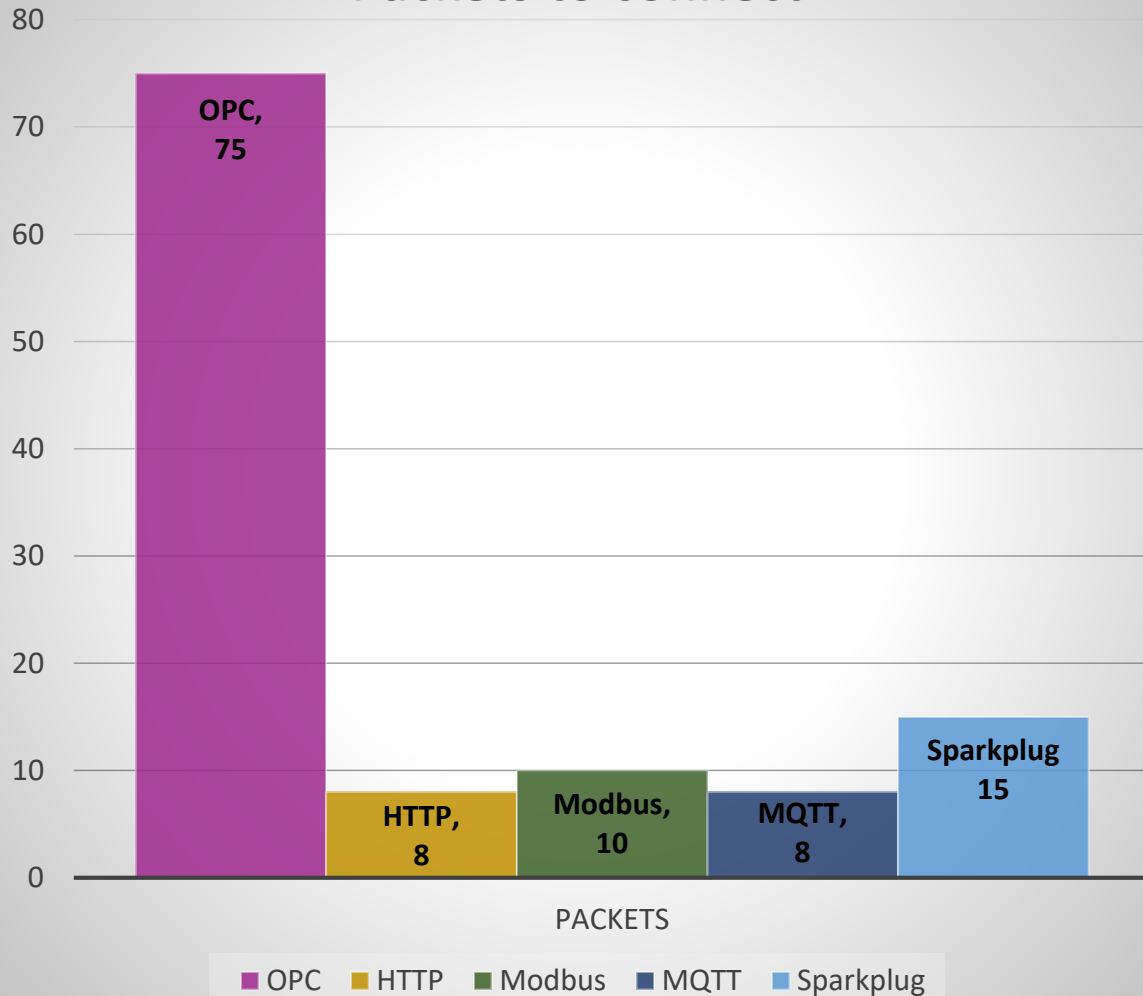
- Making the connection
- Transferring data
- Keeping the connection alive
- Tearing down the connection

Slow Connection Considerations

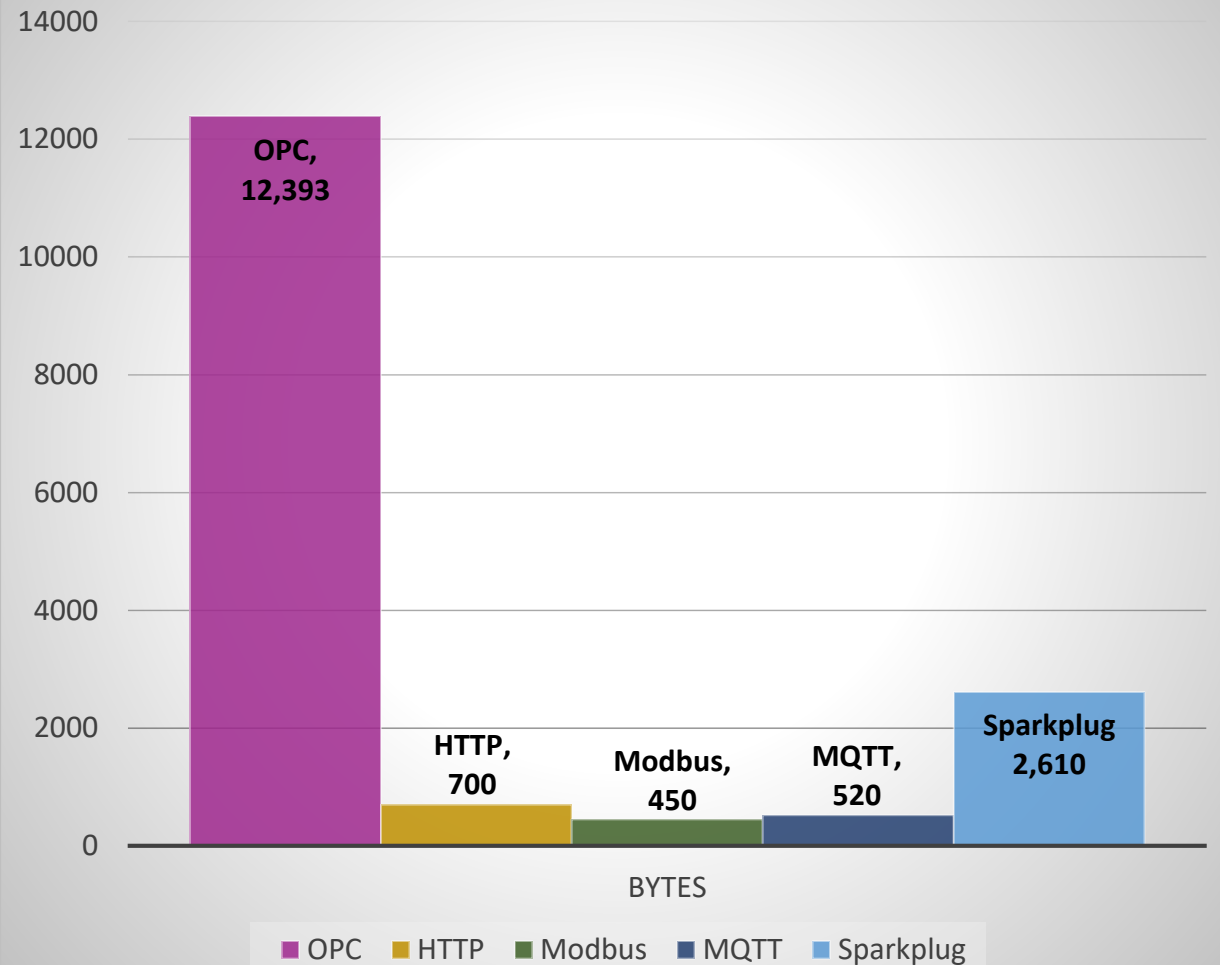
- If you are on a high speed network you may just consider data bytes and not think about the number of packets sent
- Packets sent is a critical metric on a slow network –total round trip latency can be a killer if a lot of small packets need to be sent vs fewer larger packets

Making the Connection

Packets to connect

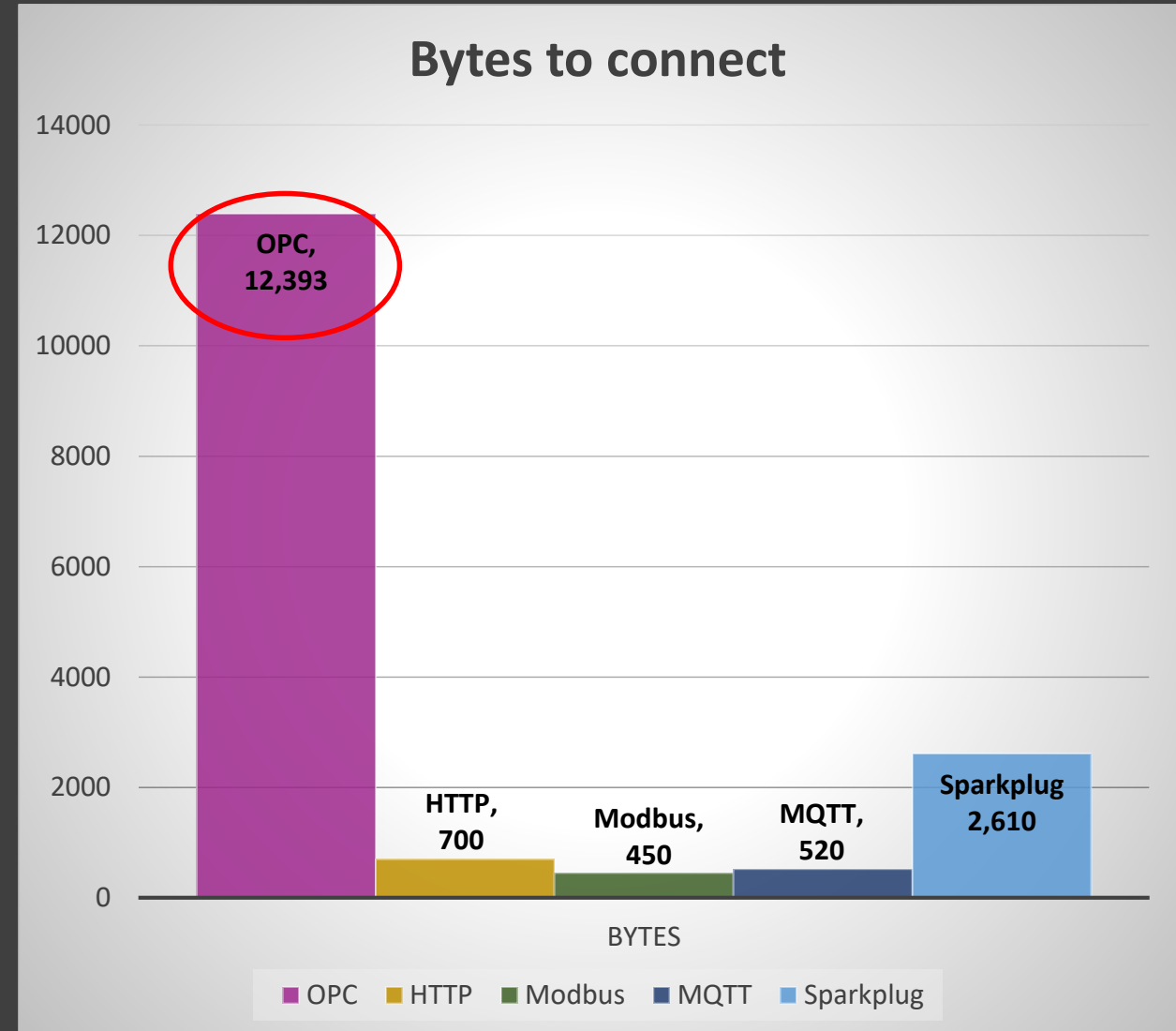


Bytes to connect



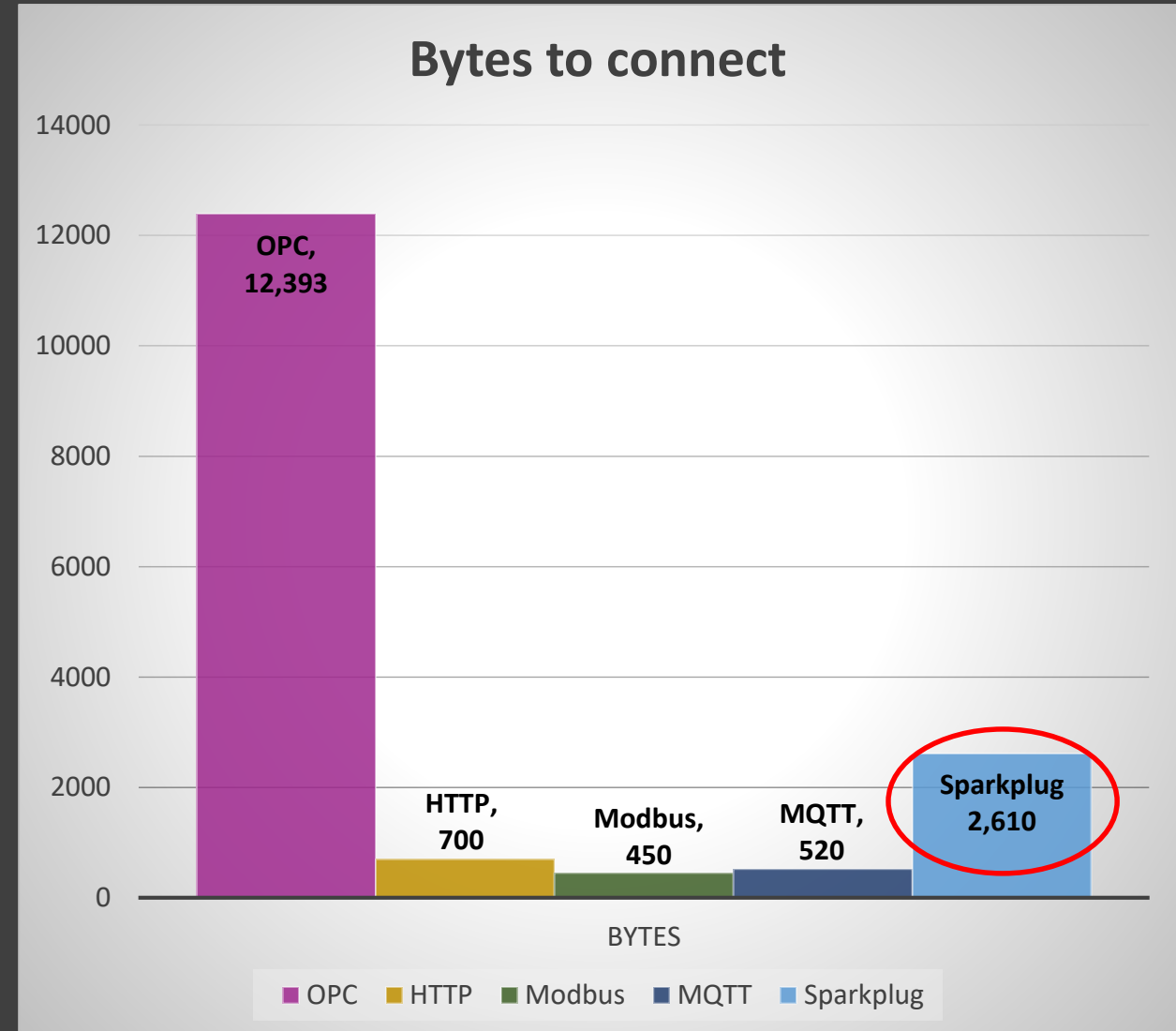
Cost to Connect

- Protocols must connect before transferring data
- OPC send's a lot of data on connection.
- OPC's best use cases may not be over slow, constrained, or fragile networks...



Cost to Connect

- Sparkplug B sends a bit more data on connection than some of the other protocols, this is because of the “Birth” message

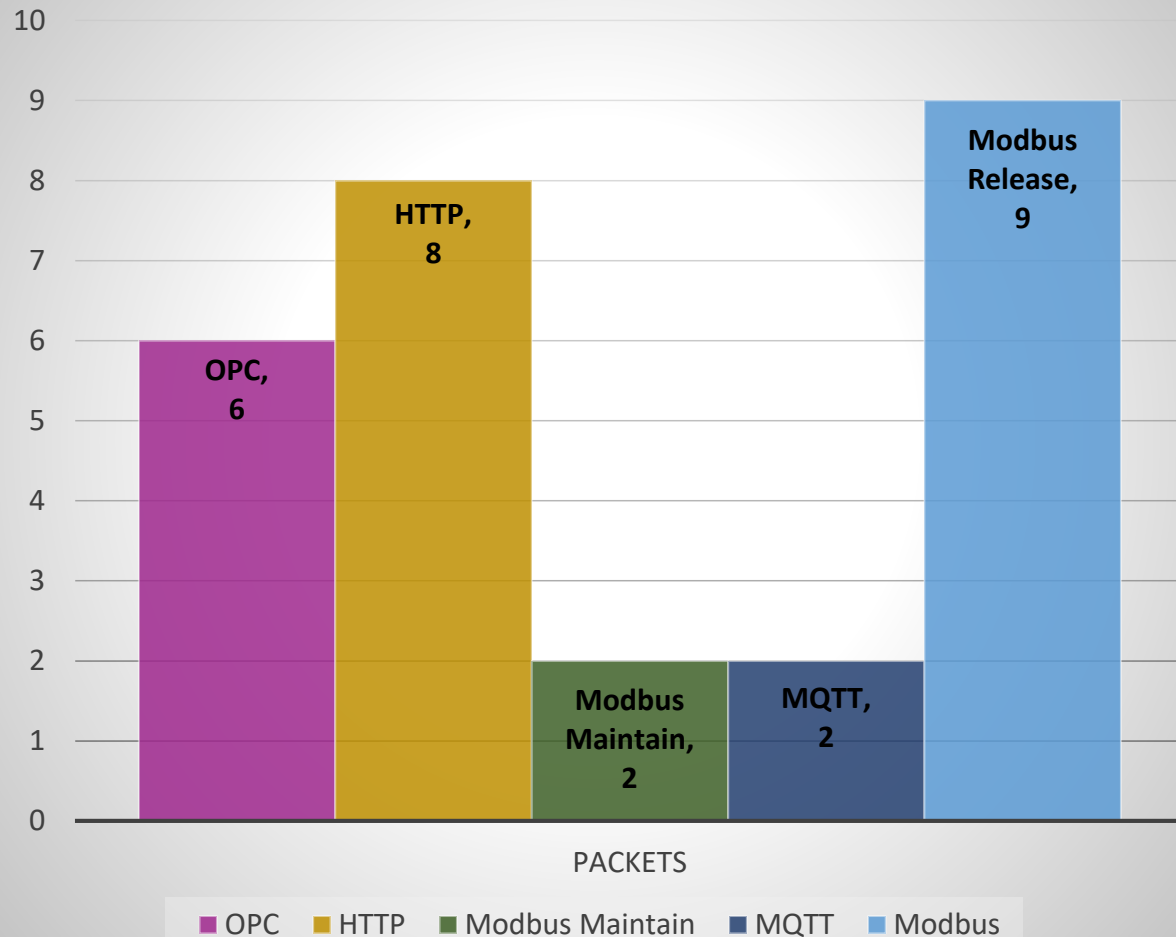


Payload

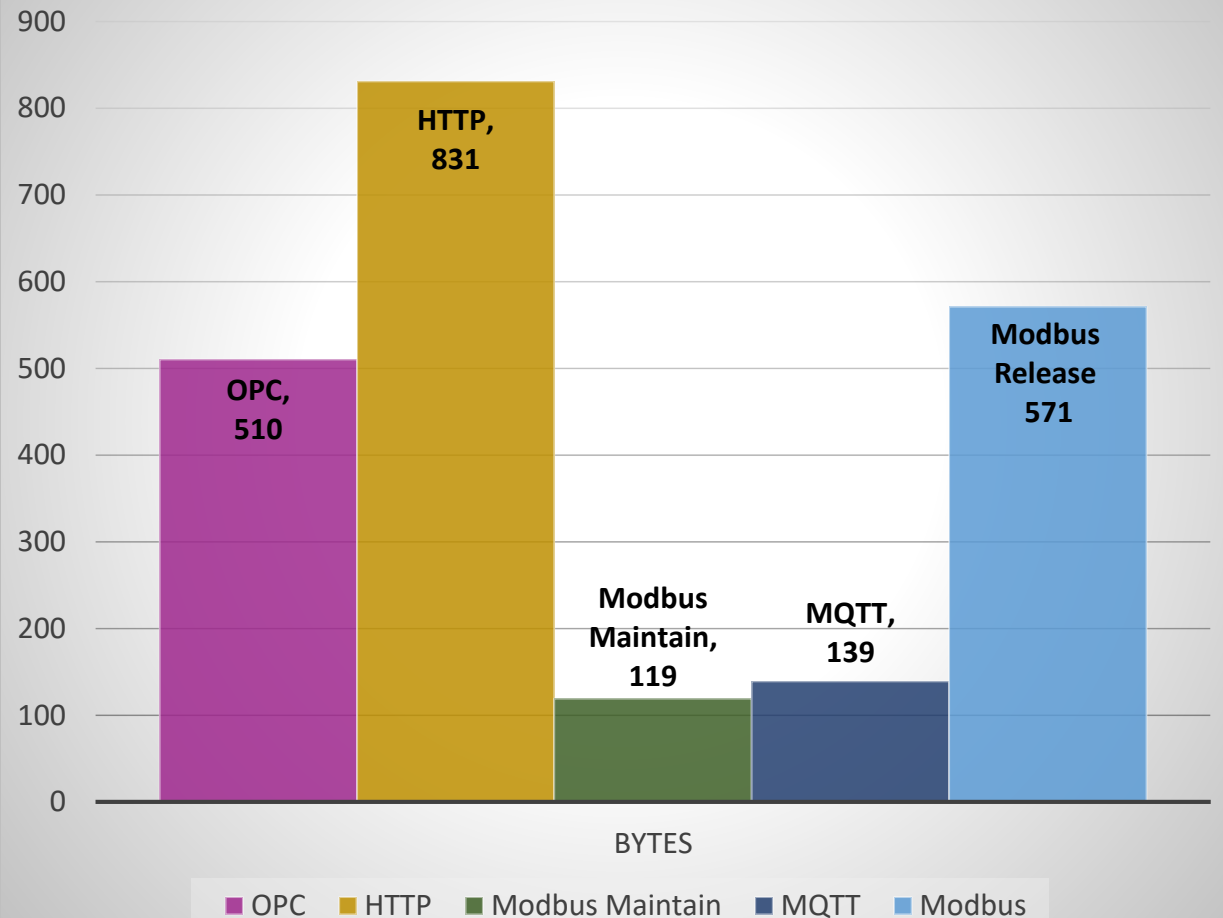
- Now that the connection is made, how much data is used to send one value

Data used to send a single value

Packets for a single value

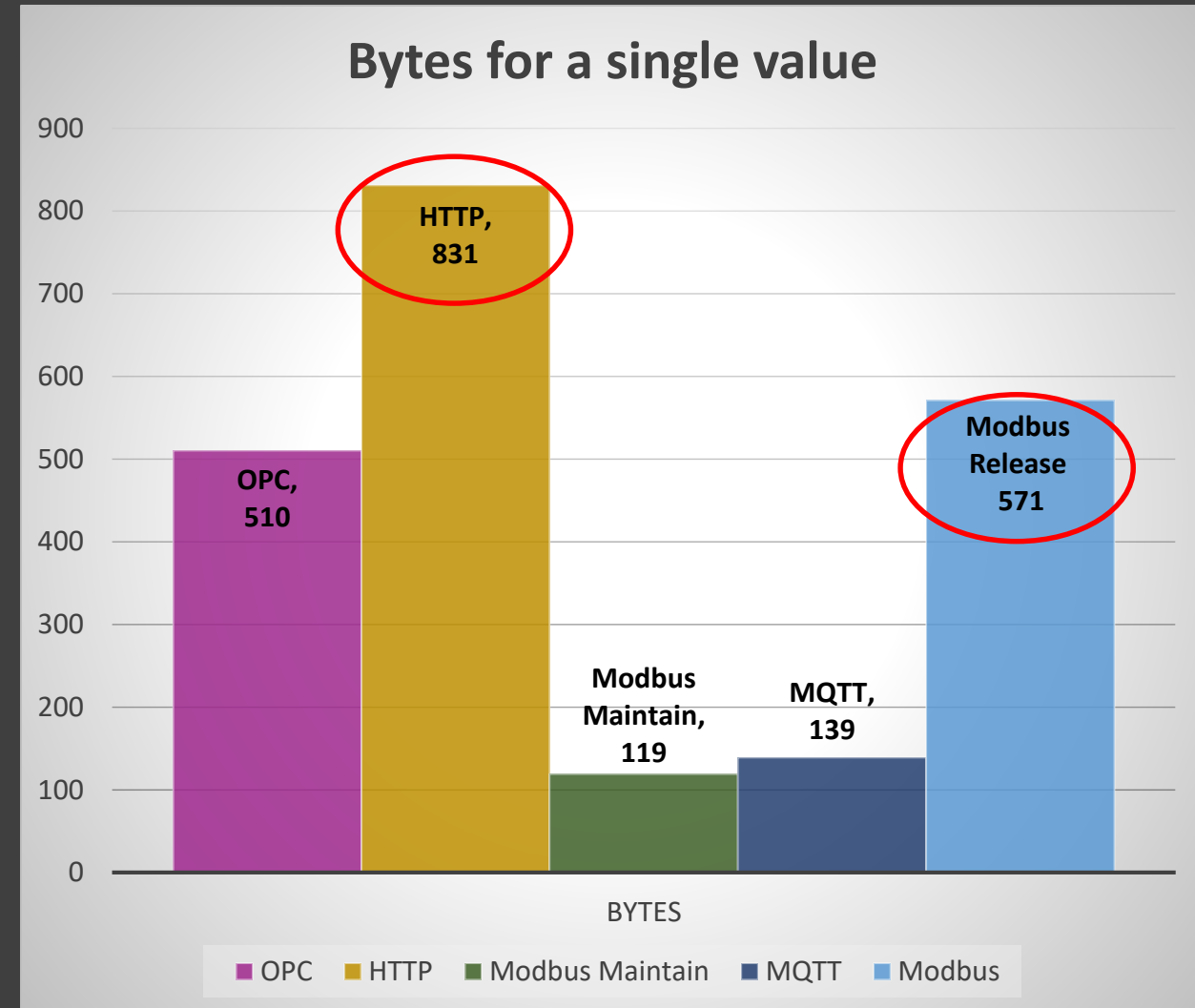


Bytes for a single value



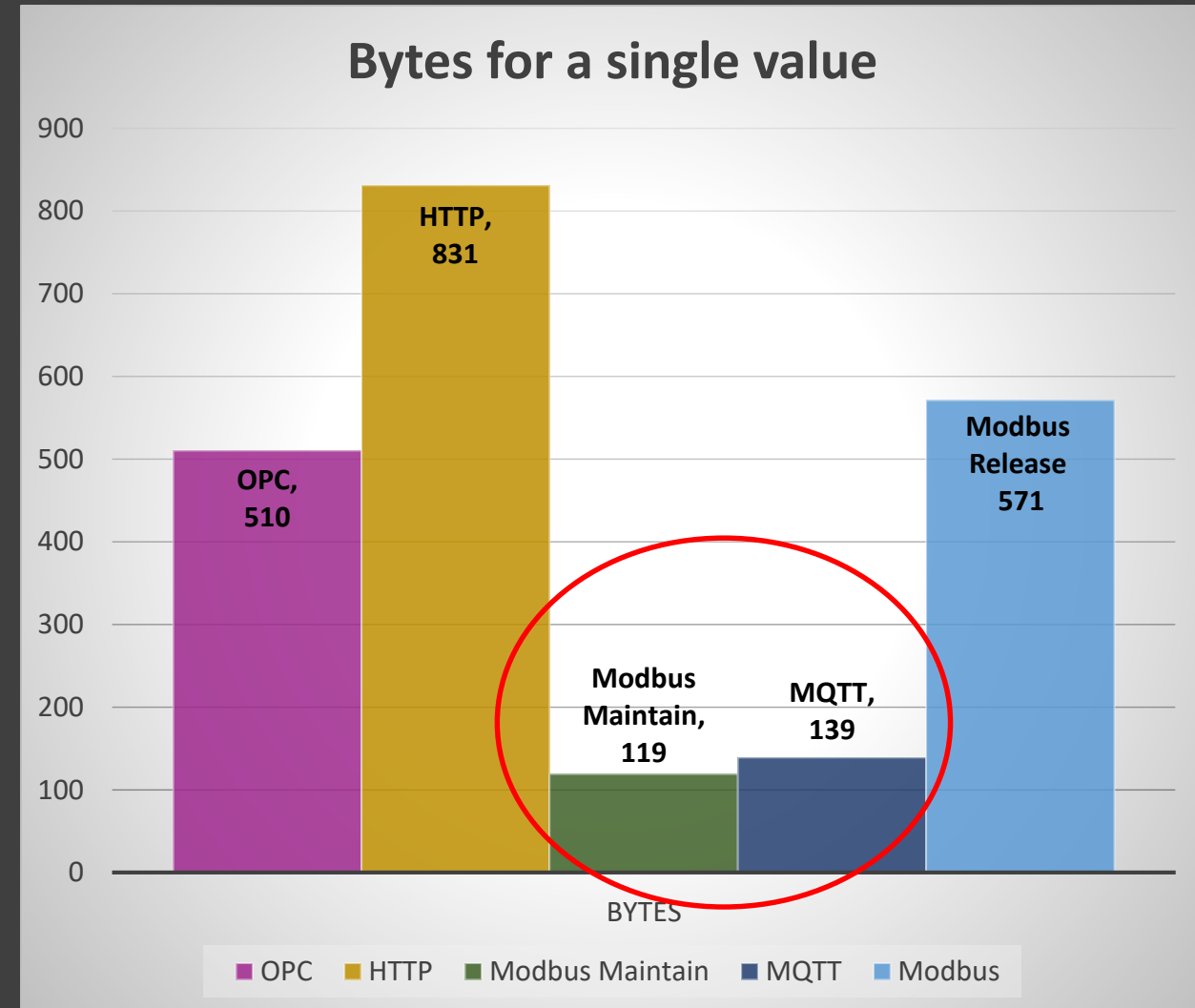
Maintaining a Connection vs Disconnecting When Done

- HTTP usually connects and releases when done, this means it is going to be more inefficient for a single value than if the connection is held open
- Similarly, Modbus usually releases the port when done though in some cases it can be set up to maintain the connection

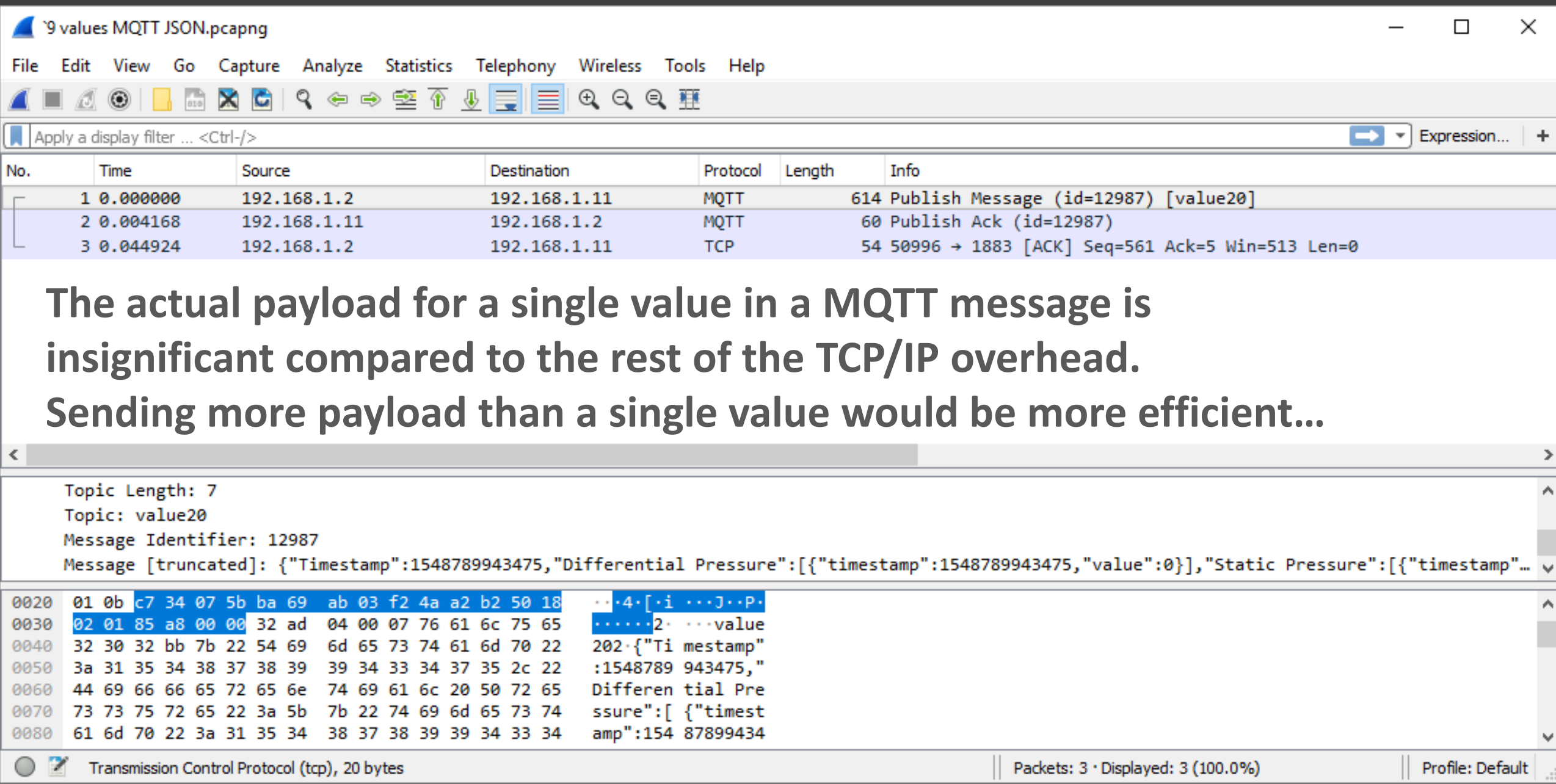


Maintaining a Connection vs Disconnecting When Done

- MQTT is intended to connect then maintain the connection
- Modbus and MQTT are both fairly efficient when you hold the connection open



MQTT Wireshark



The screenshot shows a Wireshark capture of an MQTT JSON payload. The packet list shows three packets: a Publish Message (id=12987) from 192.168.1.2 to 192.168.1.11, a Publish Ack (id=12987) from 192.168.1.11 to 192.168.1.2, and a TCP ACK from 192.168.1.2 to 192.168.1.11. The packet details pane shows the MQTT message structure, including the Topic Length (7), Topic (value20), Message Identifier (12987), and the Message payload (truncated). The packet bytes pane shows the raw data of the TCP segment, with the MQTT message structure highlighted in blue.

9 values MQTT JSON.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	192.168.1.11	MQTT	614	Publish Message (id=12987) [value20]
2	0.004168	192.168.1.11	192.168.1.2	MQTT	60	Publish Ack (id=12987)
3	0.044924	192.168.1.2	192.168.1.11	TCP	54	50996 → 1883 [ACK] Seq=561 Ack=5 Win=513 Len=0

The actual payload for a single value in a MQTT message is insignificant compared to the rest of the TCP/IP overhead. Sending more payload than a single value would be more efficient...

Topic Length: 7
Topic: value20
Message Identifier: 12987
Message [truncated]: {"Timestamp":1548789943475,"Differential Pressure":[{"timestamp":1548789943475,"value":0}], "Static Pressure":[{"timestamp":...

0020 01 0b c7 34 07 5b ba 69 ab 03 f2 4a a2 b2 50 18 ...4.[.i ...J..P.
0030 02 01 85 a8 00 00 32 ad 04 00 07 76 61 6c 75 652. ...value
0040 32 30 32 bb 7b 22 54 69 6d 65 73 74 61 6d 70 22 202·{"Ti mestamp"
0050 3a 31 35 34 38 37 38 39 39 34 33 34 37 35 2c 22 :1548789 943475,"
0060 44 69 66 66 65 72 65 6e 74 69 61 6c 20 50 72 65 Differen tial Pre
0070 73 73 75 72 65 22 3a 5b 7b 22 74 69 6d 65 73 74 ssure":[{"timest
0080 61 6d 70 22 3a 31 35 34 38 37 38 39 39 34 33 34 amp":154 87899434

Transmission Control Protocol (tcp), 20 bytes

Packets: 3 · Displayed: 3 (100.0%)

Profile: Default

Modbus TCP Release When Done Wireshark

The payload for a single value in a Modbus Release When Done message is even more insignificant compared the rest of the connection and TCP/IP overhead...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.100.100.99	10.100.100.187	TCP	78	47544 → 10502 [PSH, ACK] Seq=1 Ack=1 Win=229 Len=1
2	0.000505802	10.100.100.187	10.100.100.99	TCP	66	10502 → 47544 [ACK] Seq=1 Ack=13 Win=227 Len=0 TSv
3	0.002763853	10.100.100.187	10.100.100.99	TCP	85	10502 → 47544 [PSH, ACK] Seq=1 Ack=13 Win=227 Len=
11	29.054608923	10.100.100.99	10.100.100.187	TCP	66	47544 → 10502 [ACK] Seq=25 Ack=30 Win=229 Len=0 TS

> Frame 8: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
 > Ethernet II, Src: WistronI_c9:4e:1e (54:ee:75:c9:4e:1e), Dst: Raspberr_57:81:23 (b8:27:eb:57:81:23)
 > Internet Protocol Version 4, Src: 10.100.100.99, Dst: 10.100.100.187
 > Transmission Control Protocol, Src Port: 47544, Dst Port: 10502, Seq: 13, Ack: 20, Len: 12
 > Data (12 bytes)

0000	b8 27 eb 57 81 23 54 ee 75 c9 4e 1e 08 00 45 00	· · · W · # T · u · N · · · E ·
0010	00 40 30 6e 40 00 40 06 2c 64 0a 64 64 63 0a 64	· @ 0 n @ · @ · , d · d d c · d
0020	64 bb b9 b8 29 06 fc e7 75 a3 48 29 63 92 80 18	d · · ·) · · · u · H) c · · ·
0030	00 e5 de 18 00 00 01 01 08 0a 58 58 e1 a5 be e1	· · · · · · · · · · X X · · · ·
0040	a1 93 00 04 00 00 00 06 01 01 00 01 00 05	· · · · · · · · · ·

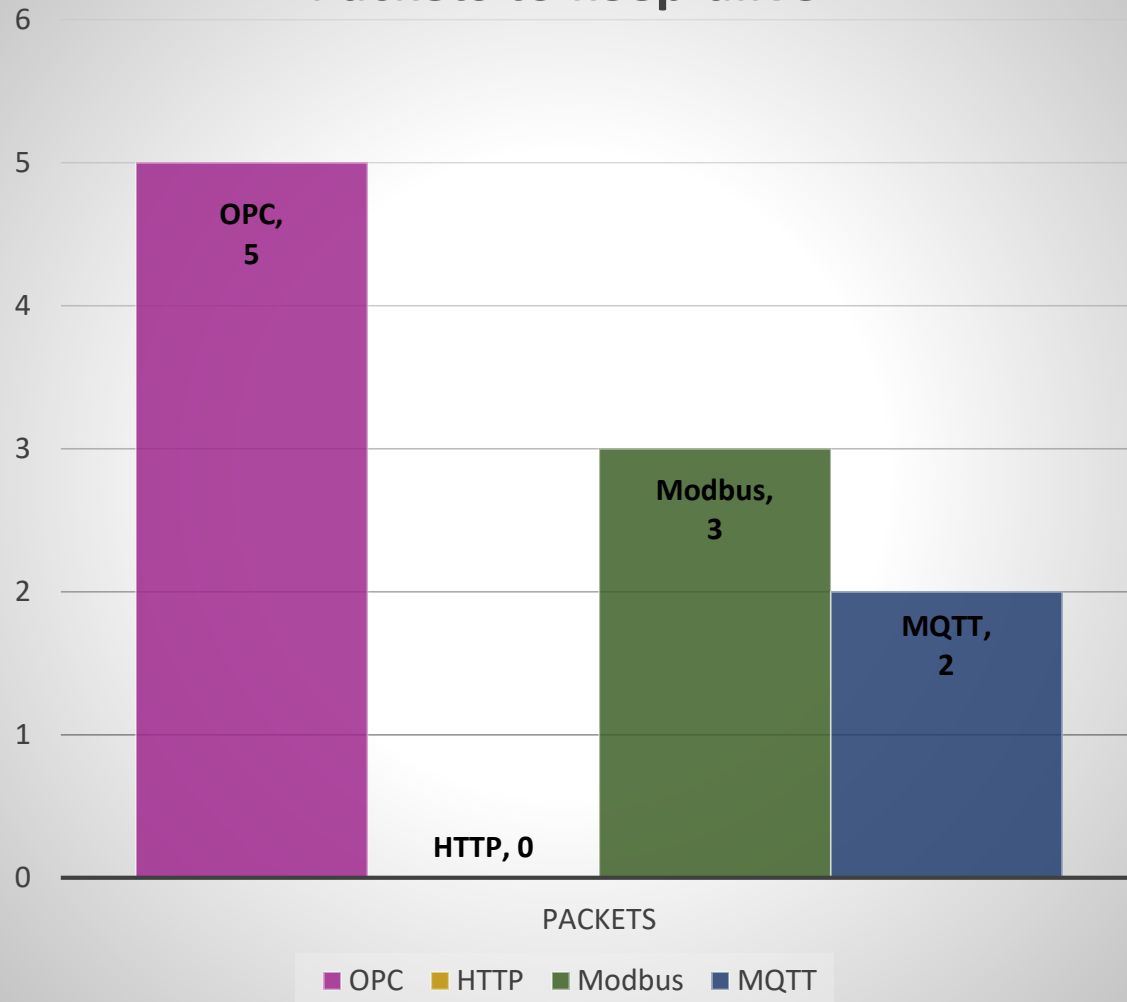
Maintaining the connection

- Maintaining a connection unfortunately is not free
- There is a keep-alive ping sent on an interval
- Some implementations let you adjust this keep-alive ping frequency
- How often you ping depends on the criticality of knowing when a device goes offline vs conserving bandwidth

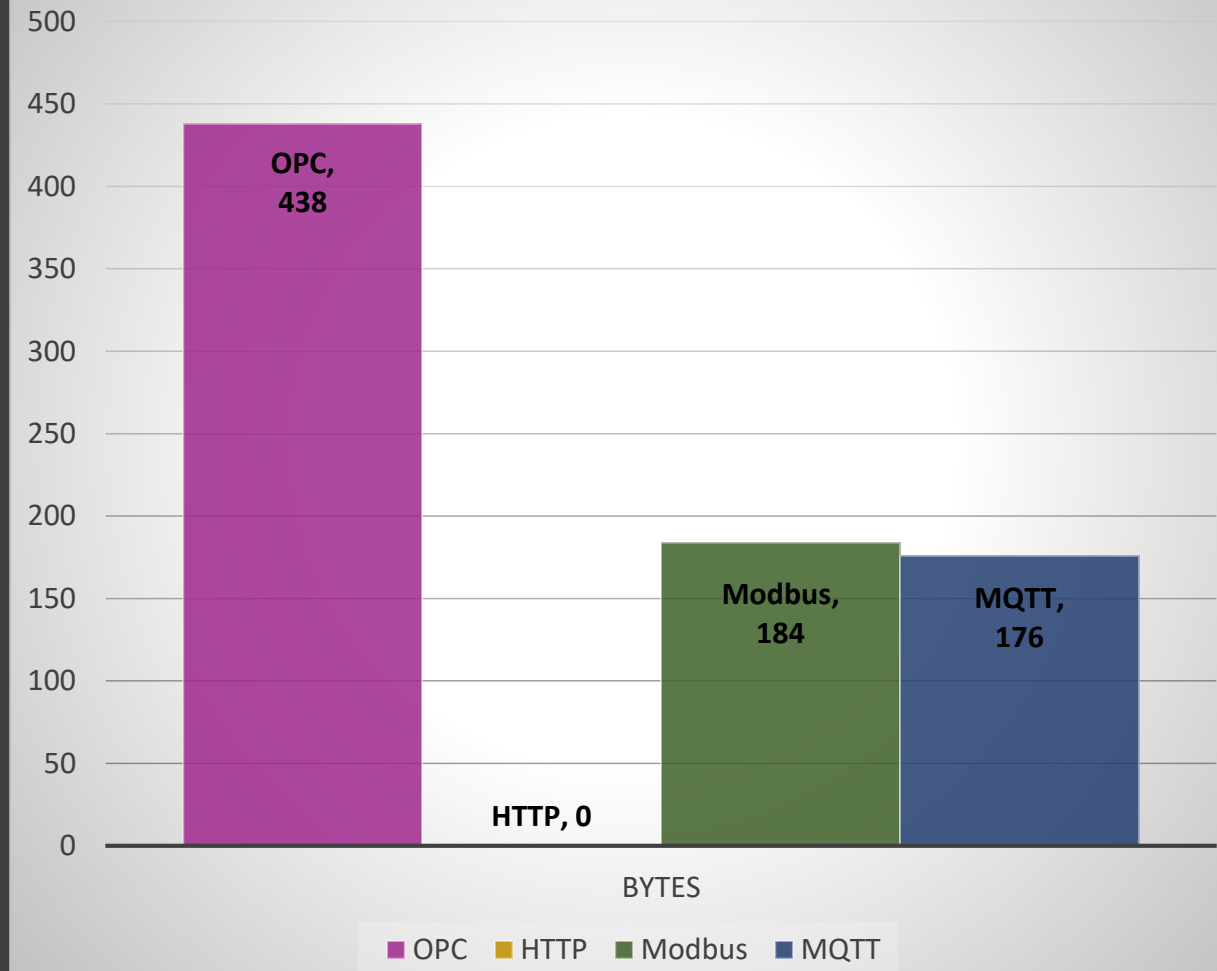
Cost to Maintain a Connection

Unfortunately you can't always adjust the keep-alive time, you can often set it with MQTT as it was designed with bandwidth usage in mind.

Packets to keep-alive



Bytes to keep-alive



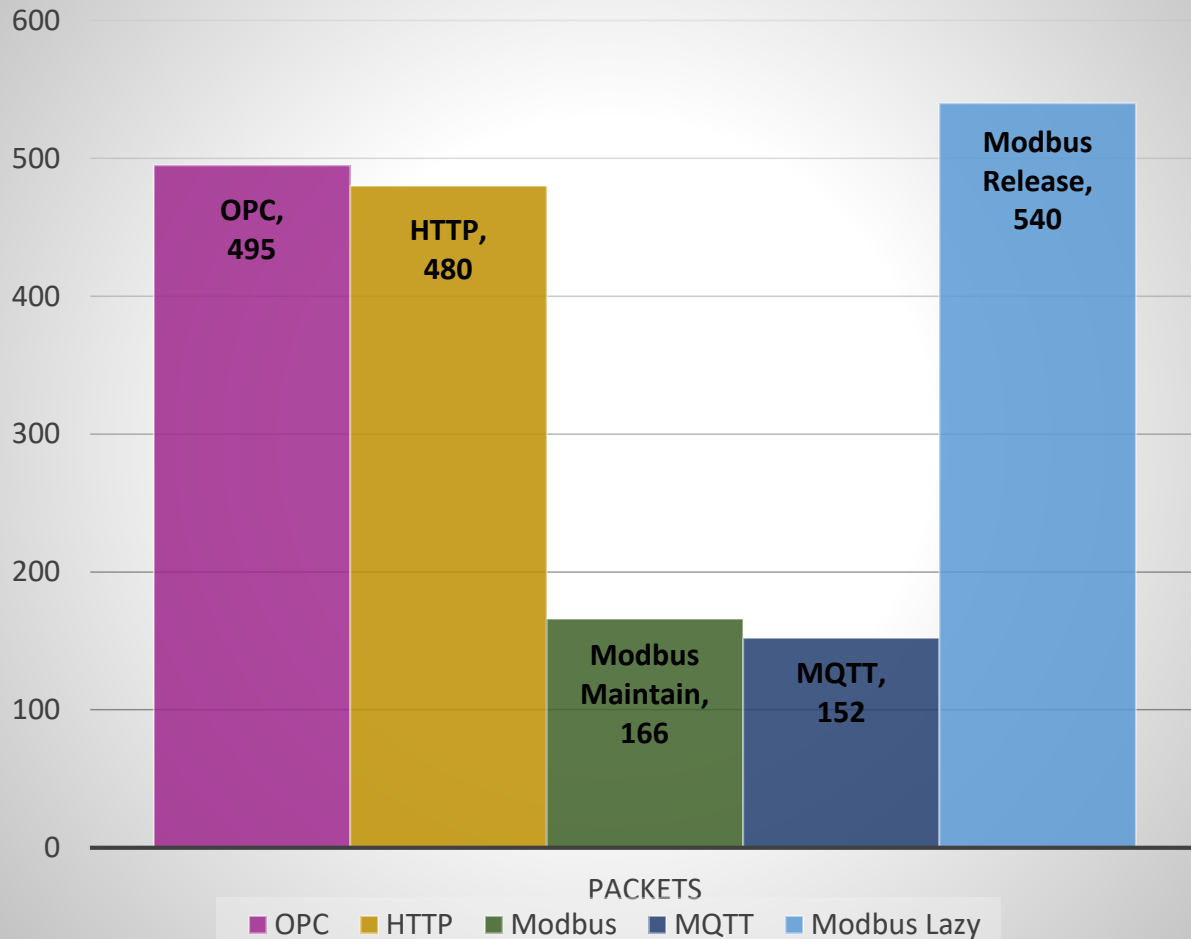
So keeping the things we have learned in mind;
What would the “Total Cost of Ownership”(TCO)
be to send one value every minute for an hour (60
records), with connection and keepalive cost
included?

AKA “Minute Data”

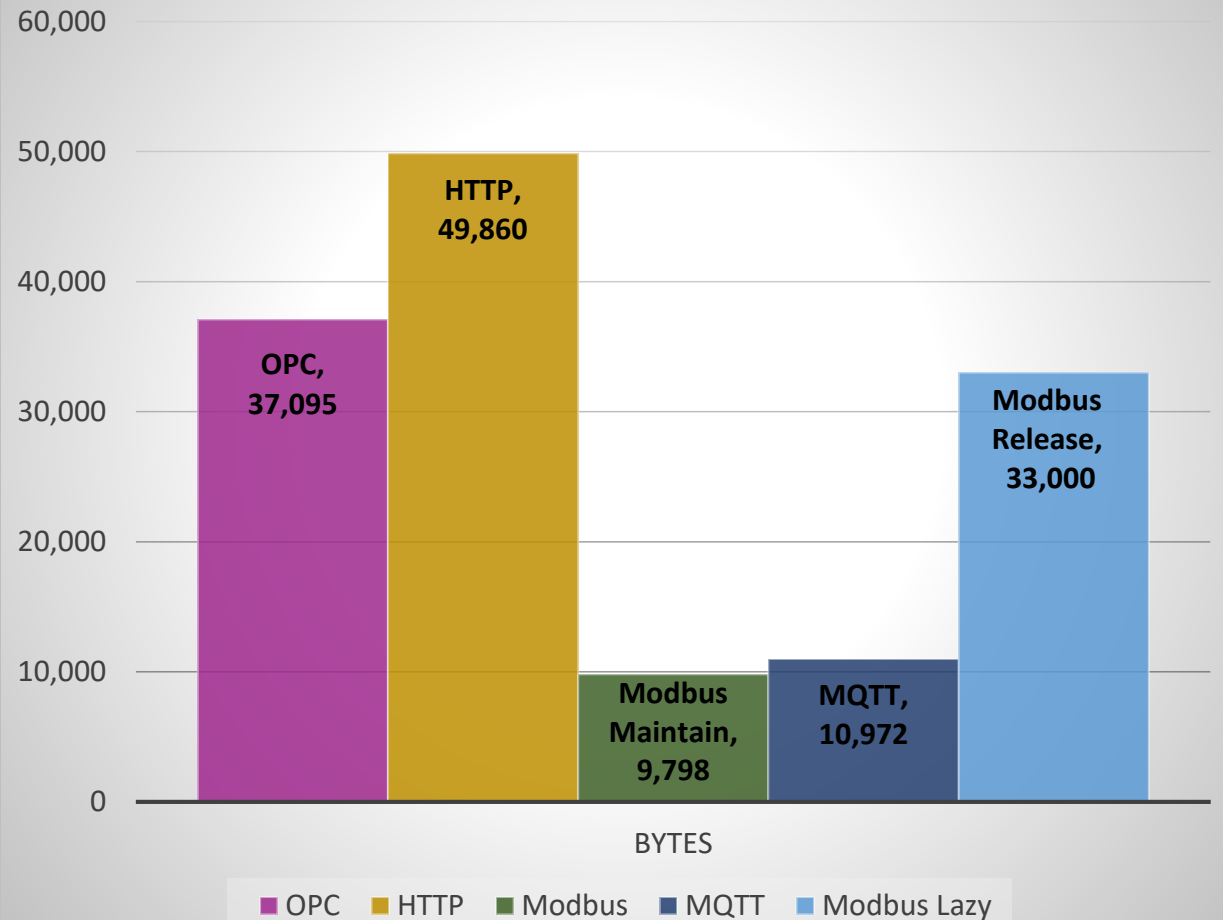
TCO of an hours worth of minute data

Assuming keep-alive can be set to 5 minutes on OPC, Modbus, and MQTT

Packets

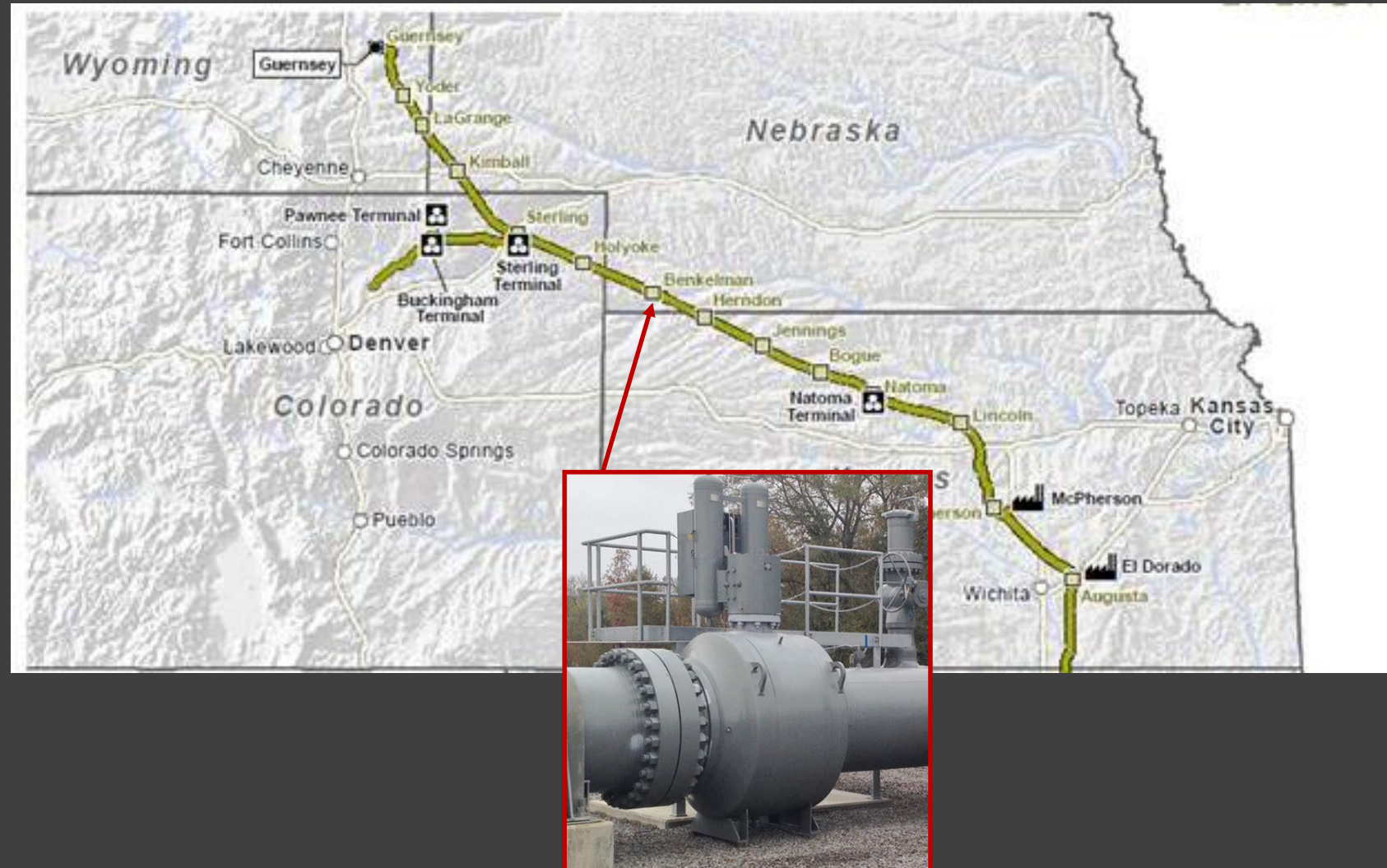


Bytes



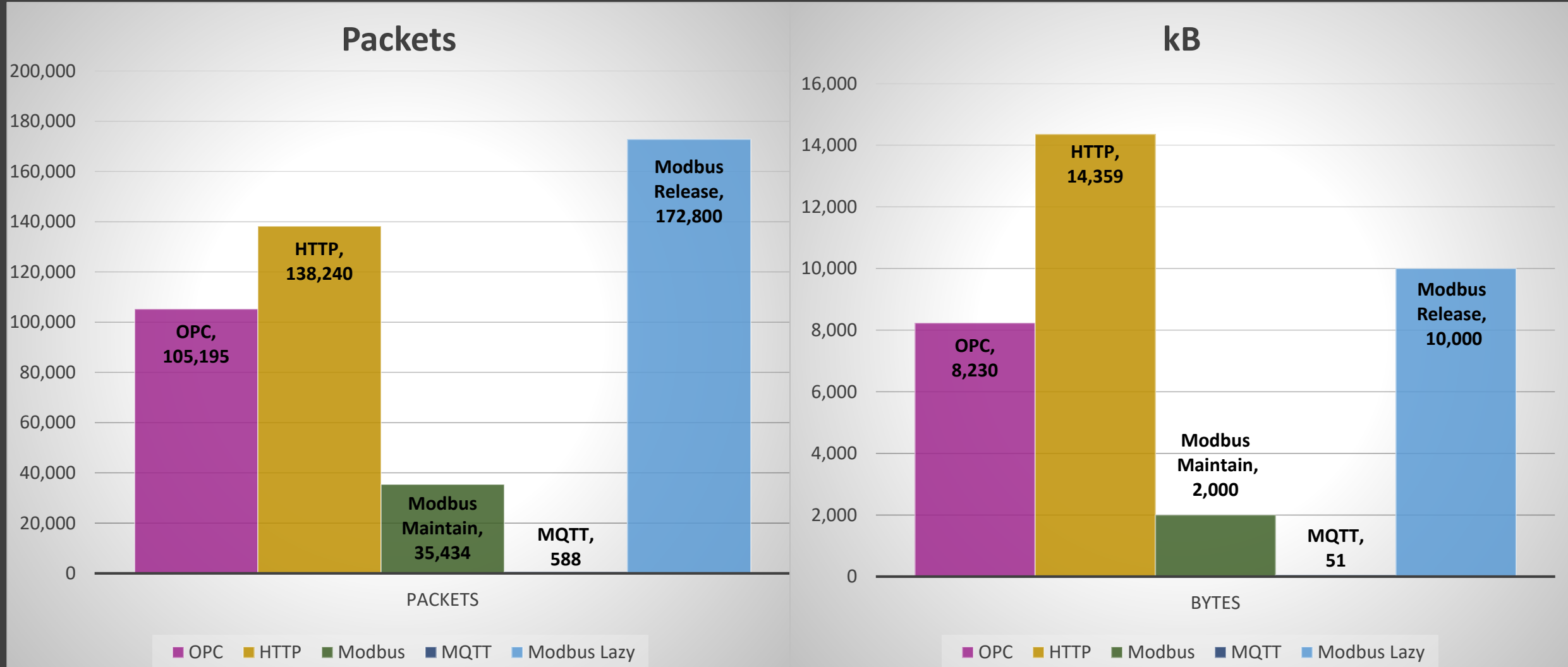
Data On Change Example For a Valve Station

OK, so now what if
we needed to know
within five seconds
when a valve started
moving?
(Valve moves twice
per day)
(Single tag – not a
perfect example)



TCO of a days worth of 5 second data

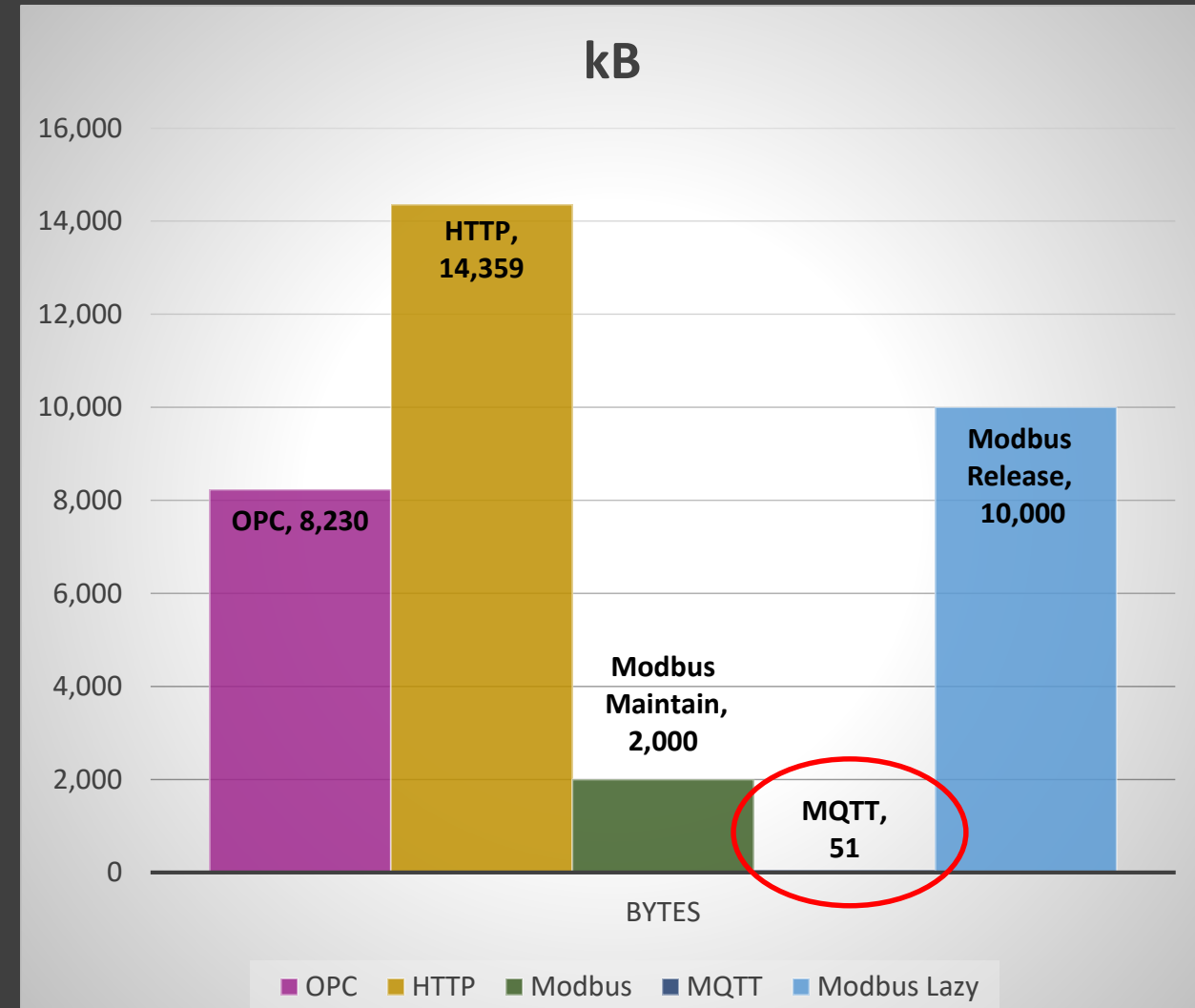
Assuming keep-alive can be set to 5 minutes, data changes twice per day



TCO of a days worth of 5 second data

Assuming keep-alive can be set to 5 minutes, data changes twice per day

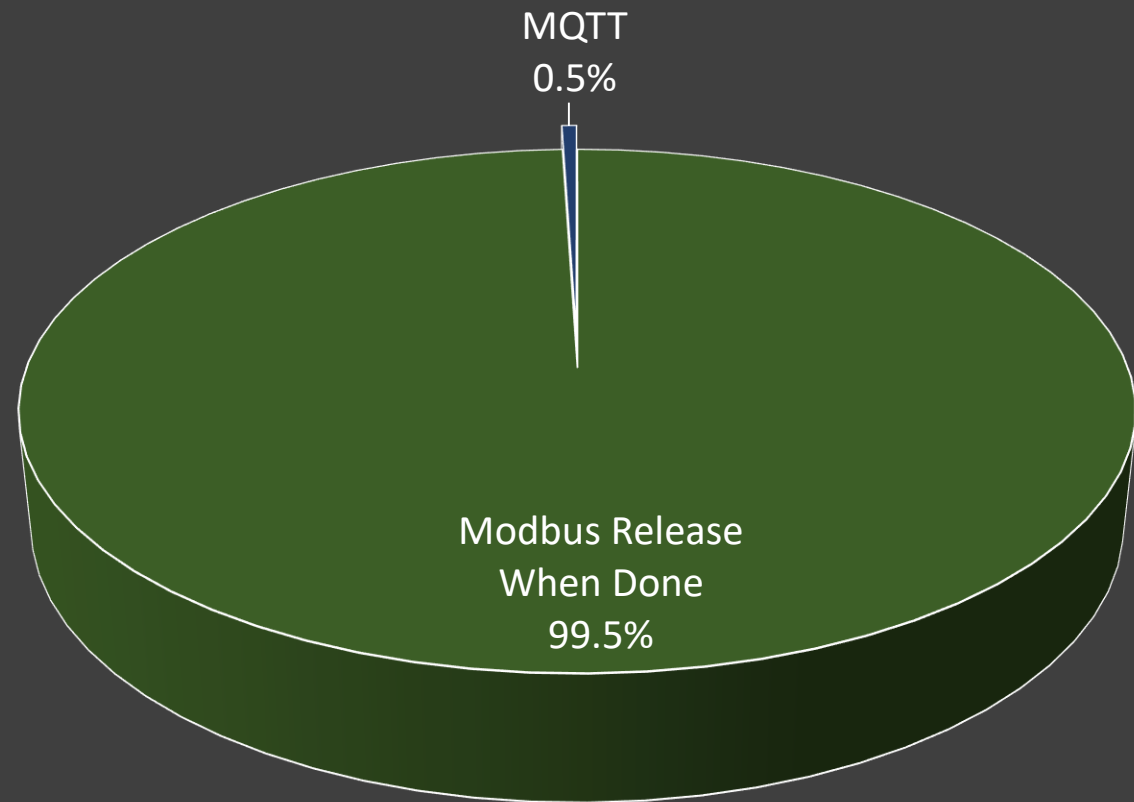
MQTT data consumption is significantly lower than the others in this case because the data is only sent when the value changes.



We are starting to detect that MQTT can be quite efficient when set up to send data when it changes v.s. traditional polling

More on this later...

kB MQTT on change vs Modbus Release When Done for a single value.
5 second resolution and data changes once per day



■ ■ ■ Modbus ■ MQTT

Differential Pressure	0.000000
Static Pressure	499.000000
Temperature	73.361351
MCFD	0.000000
MCF CD	0.000000
MCF PD	0.000000
Sales Valve	0.000000
Tubing Pressure	565.820007
Casing Pressure	576.809998

Use case: Retrieve 9 Values

Retrieve 9 values every minute
via Modbus RTU Encapsulated
vs MQTT vs MQTT Sparkplug B

Considerations

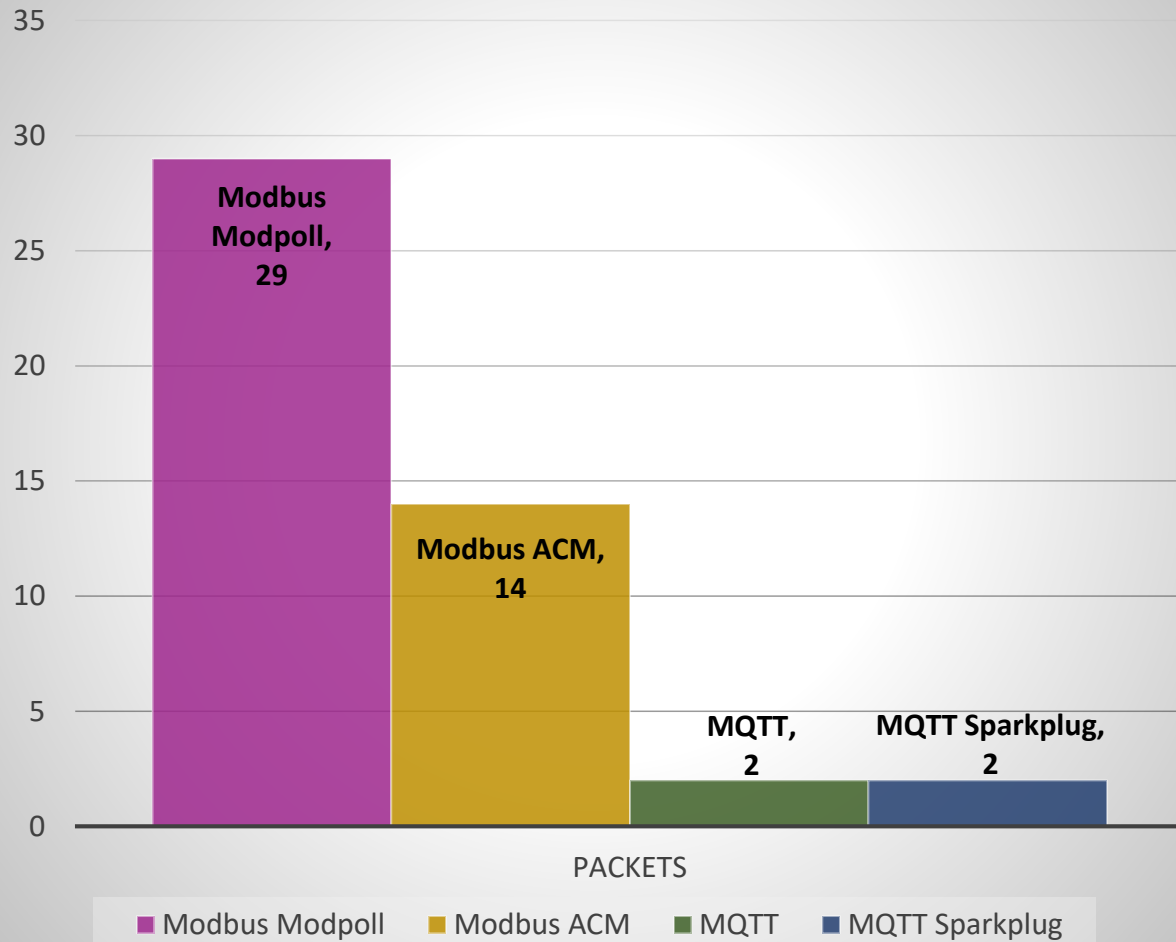
- These values are not in a contiguous Modbus block so Modbus has to poll three different ranges of registers
- Not all drivers for the same protocol will give the same results
- Sparkplug can be more efficient if it is implemented as intended vs sending unnecessary data

Actual MQTT Payload Sent In This Test

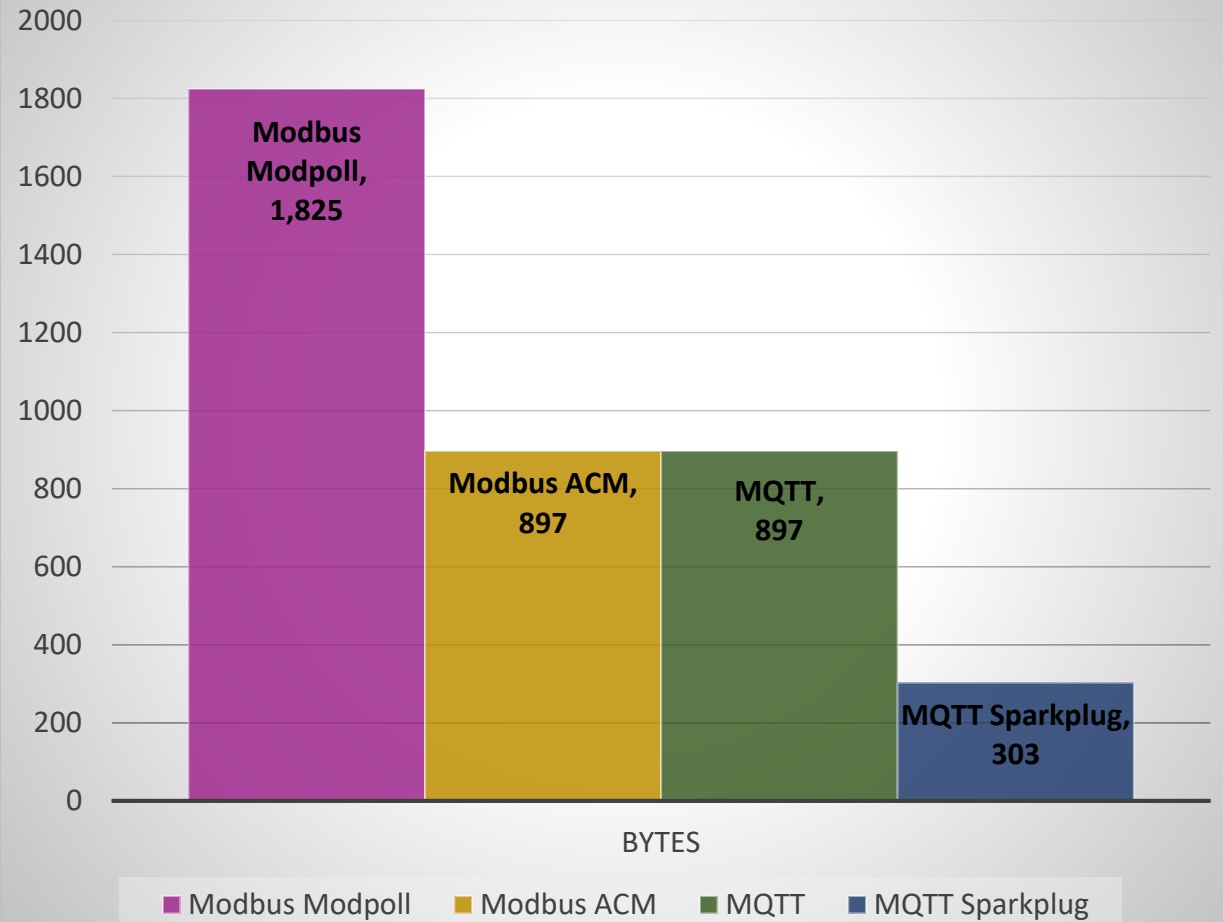
- {"timestamp":1520722891000,"metrics":[{"name":"","alias":10,"timestamp":1520722891000,"value":1},{"name":"","alias":11,"timestamp":1520722891000,"value":1},{"name":"","alias":12,"timestamp":1520722891000,"value":1},{"name":"","alias":13,"timestamp":1520722891000,"value":1},{"name":"","alias":14,"timestamp":1520722891000,"value":1},{"name":"","alias":15,"timestamp":1520722891000,"value":1},{"name":"","alias":16,"timestamp":1520722891000,"value":1},{"name":"","alias":17,"timestamp":1520722891000,"value":1},{"name":"","alias":18,"timestamp":1520722891000,"value":1}],"seq":2}

Real Device, 9 values

Packets for 9 values

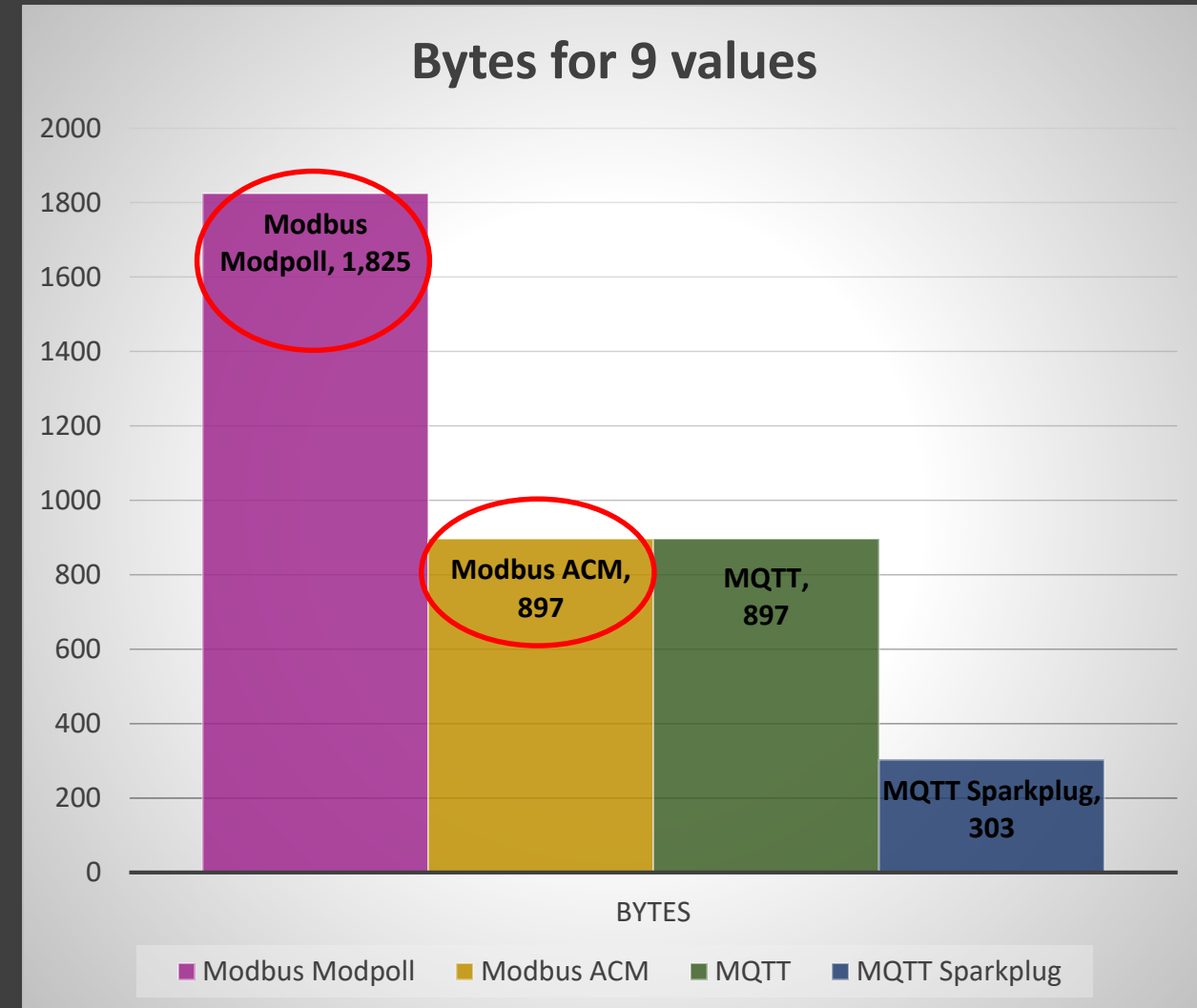


Bytes for 9 values



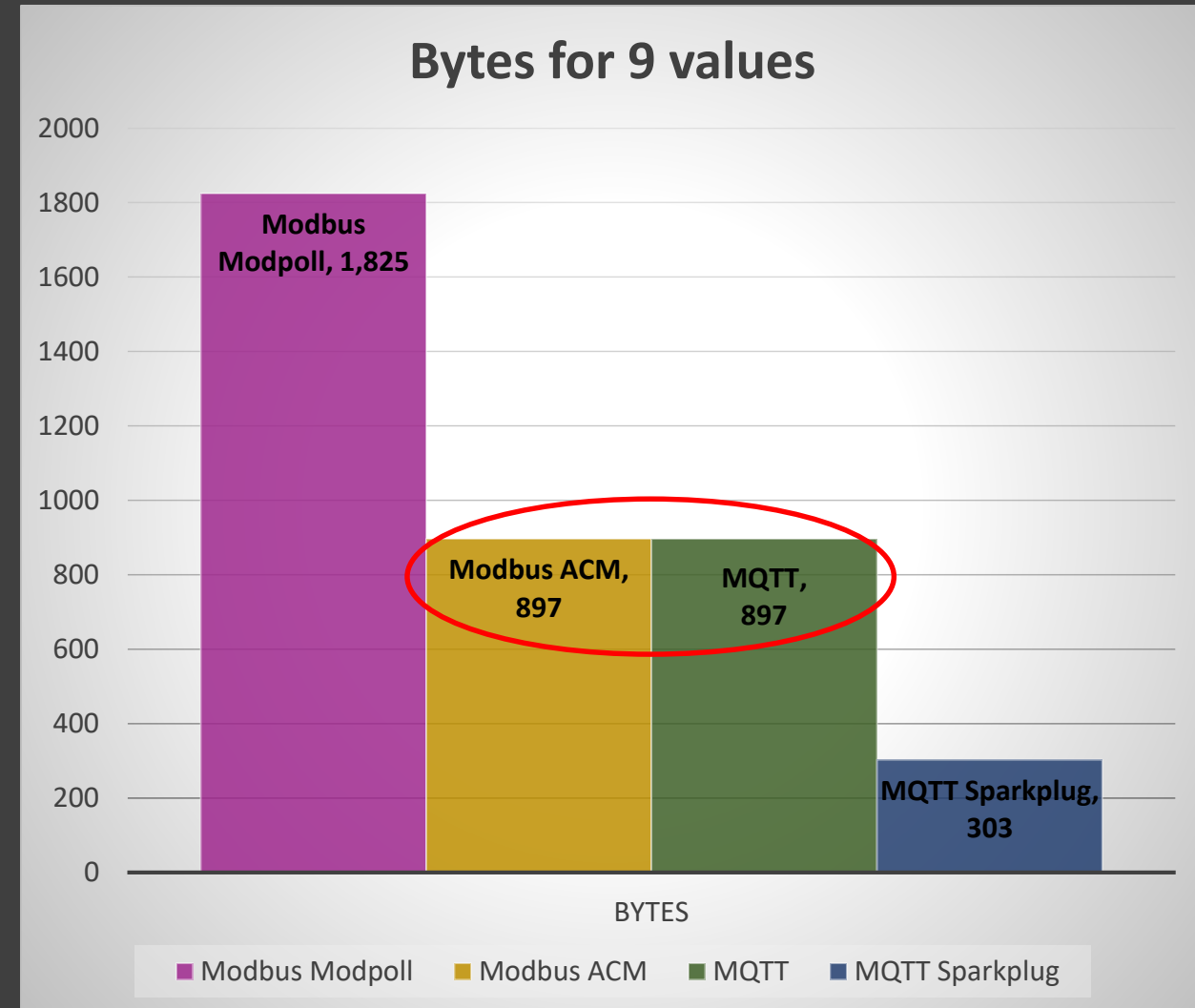
Different Modbus Drivers

Different modbus drivers seem to return different results, in this case Modpoll seems to send the data twice



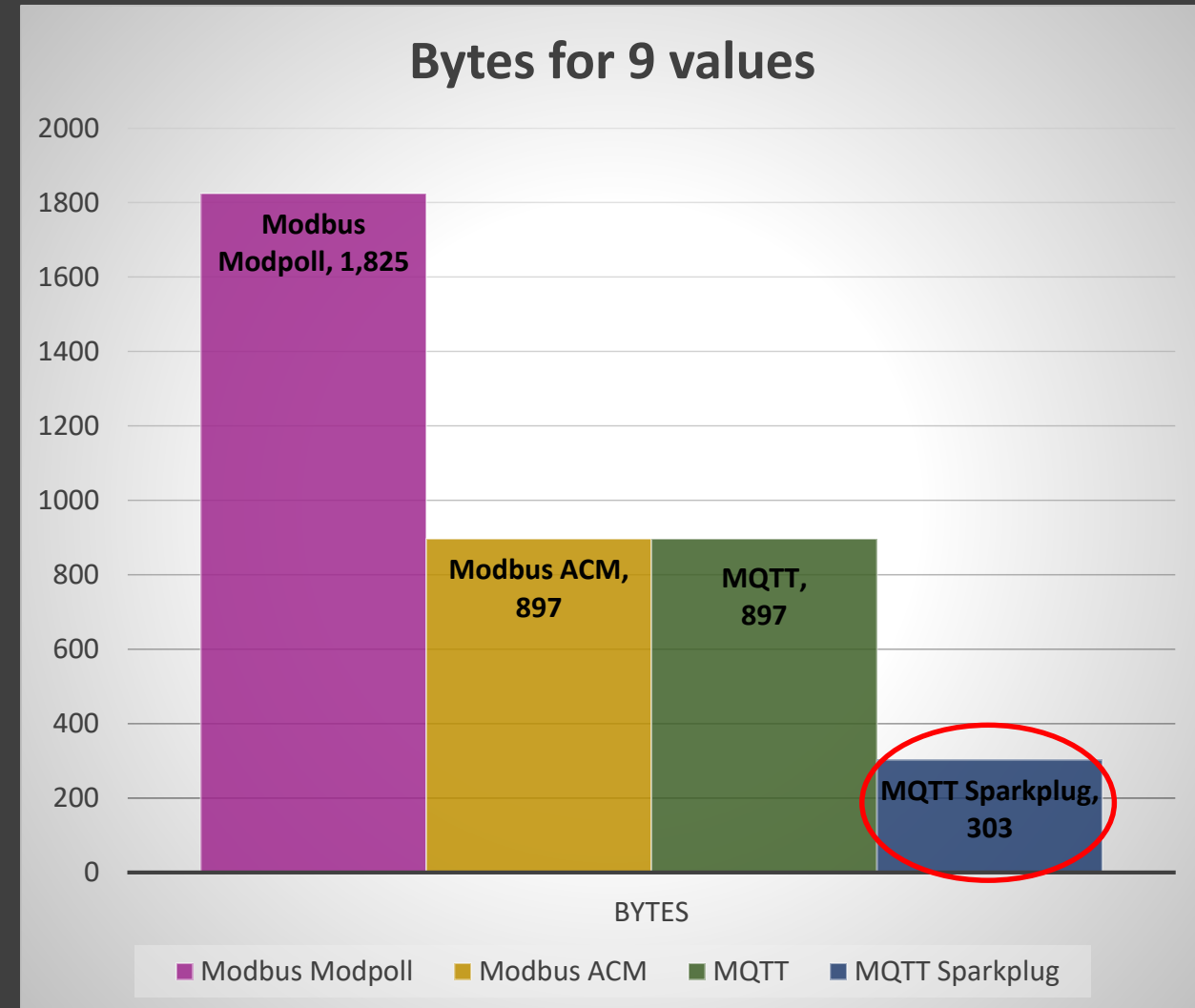
MQTT VS Modbus

MQTT Uses the same amount of bandwidth as Modbus ACM, even though it is sending 3x more data (timestamp, alias, and value)

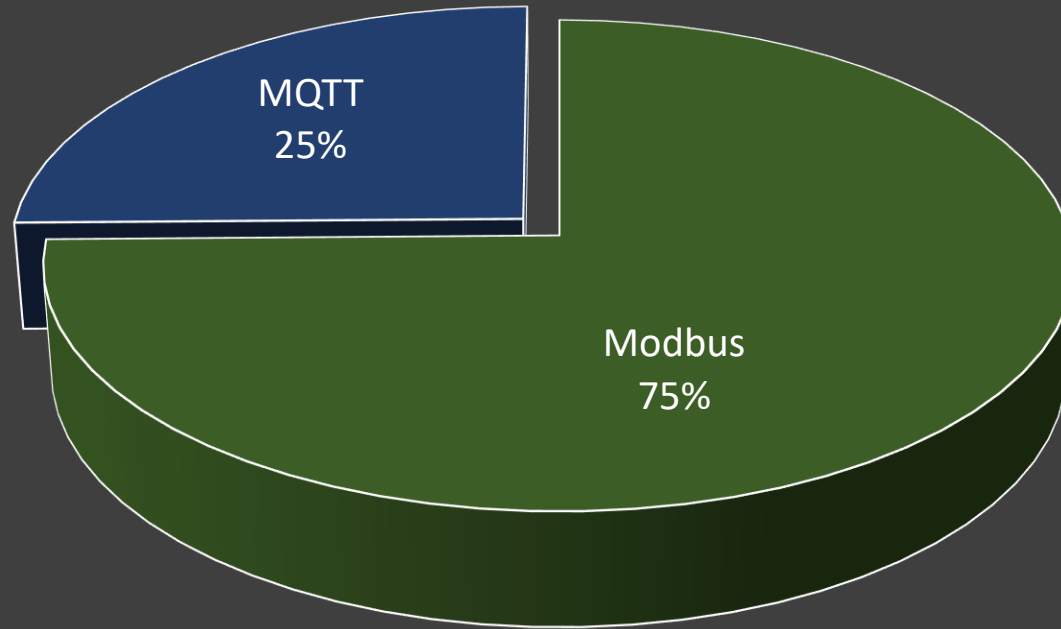


Sparkplug B Compresses Data 3x

MQTT Sparkplug B is 3x more efficient than regular MQTT sending the exact same payload because it compresses the data.



MQTT Sparkplug B vs Modbus Poll Response



Modbus MQTT

Sparkplug B Data on Birth

On birth (connection) Sparkplug B publishes important information about the tags that will not be sent in subsequent value updates

```
birth
{
  "name": "",
  "alias": 13,
  "timestamp": 1520722891000,
  "dataType": "UInt16",
  "value": 1
}

then
{
  "alias": 13,
  "timestamp": 1520722891000,
  "value": 1
}
```

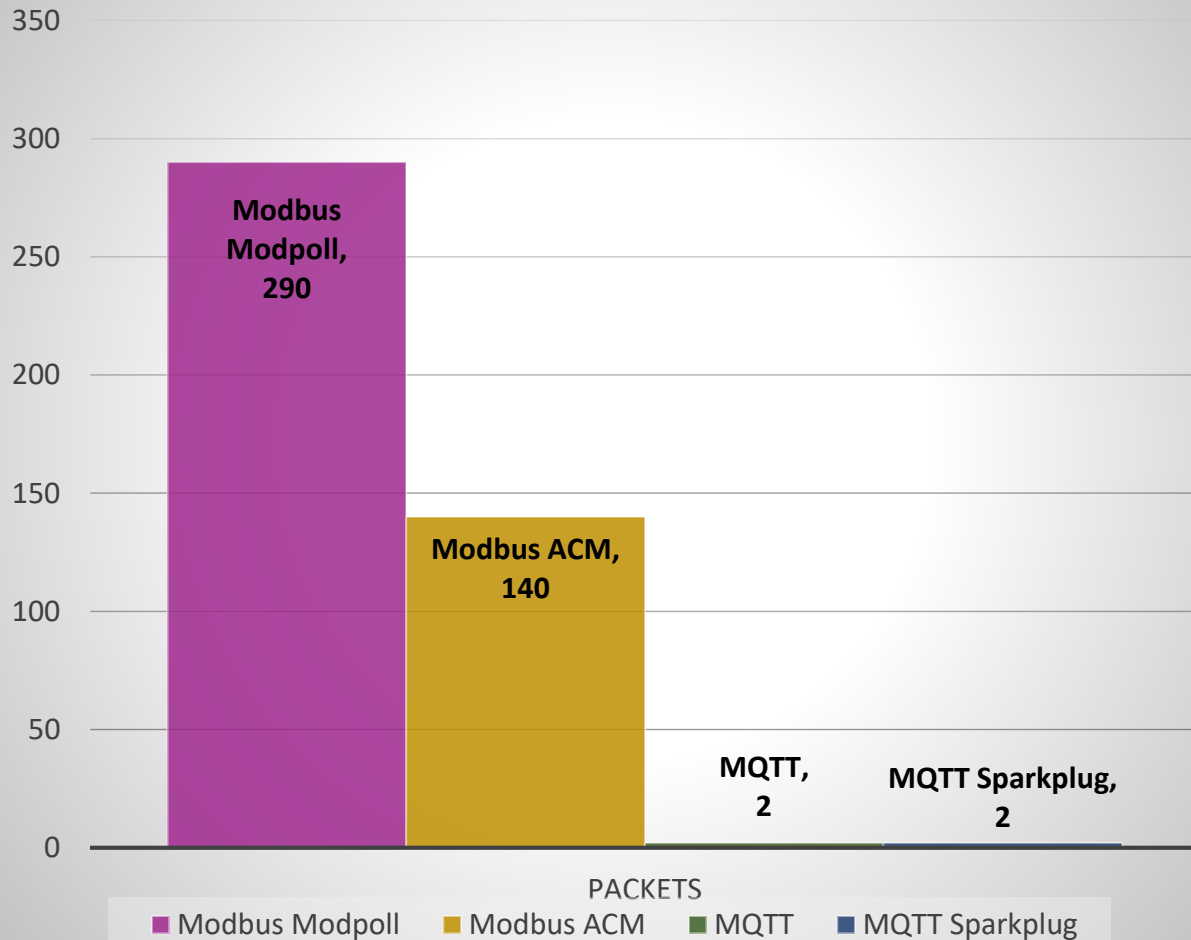
Data Concentration

Now what if the same minute data was buffered and shipped via MQTT every 10 minutes instead of every minute?

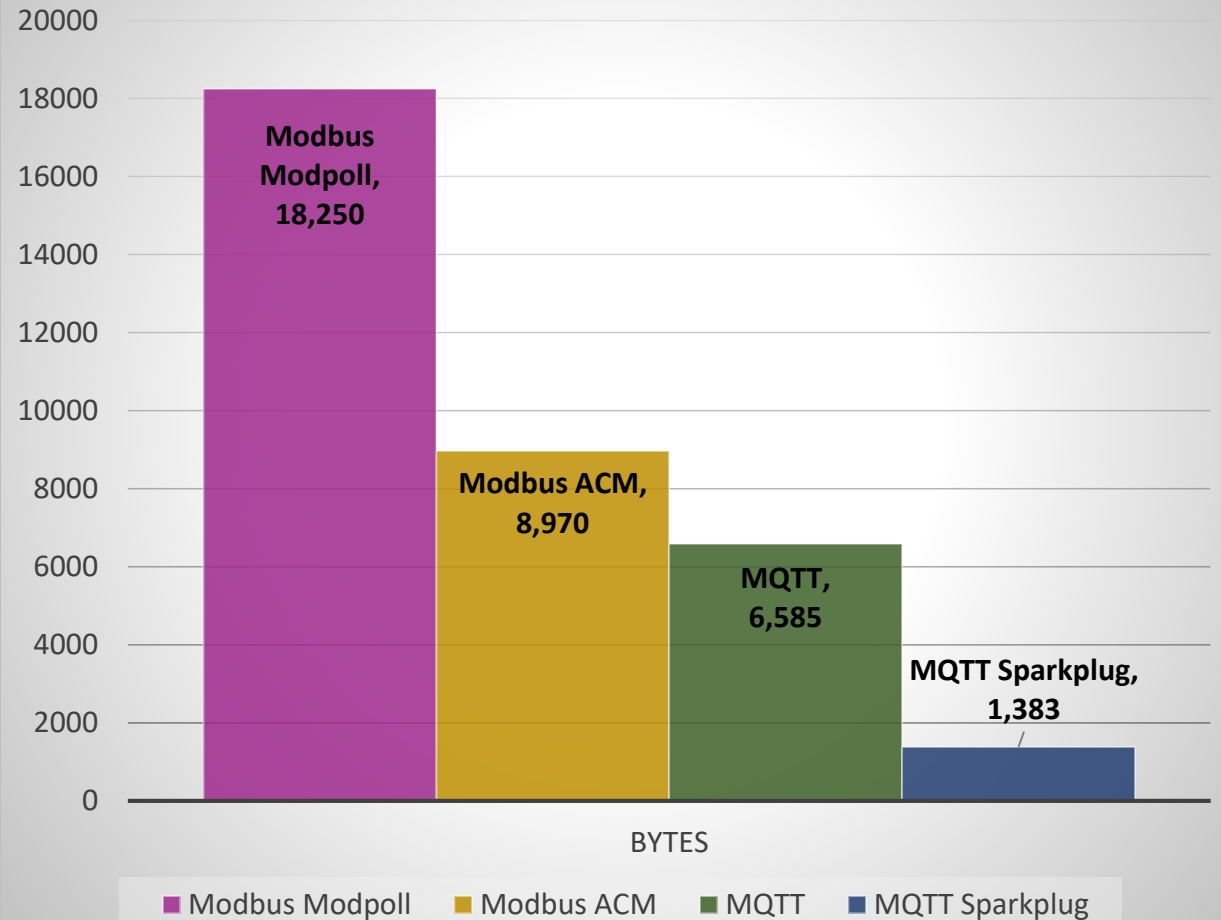
Same Device, 90 values sent every 10 minutes

(Modbus isn't buffering, it is still polling every minute)

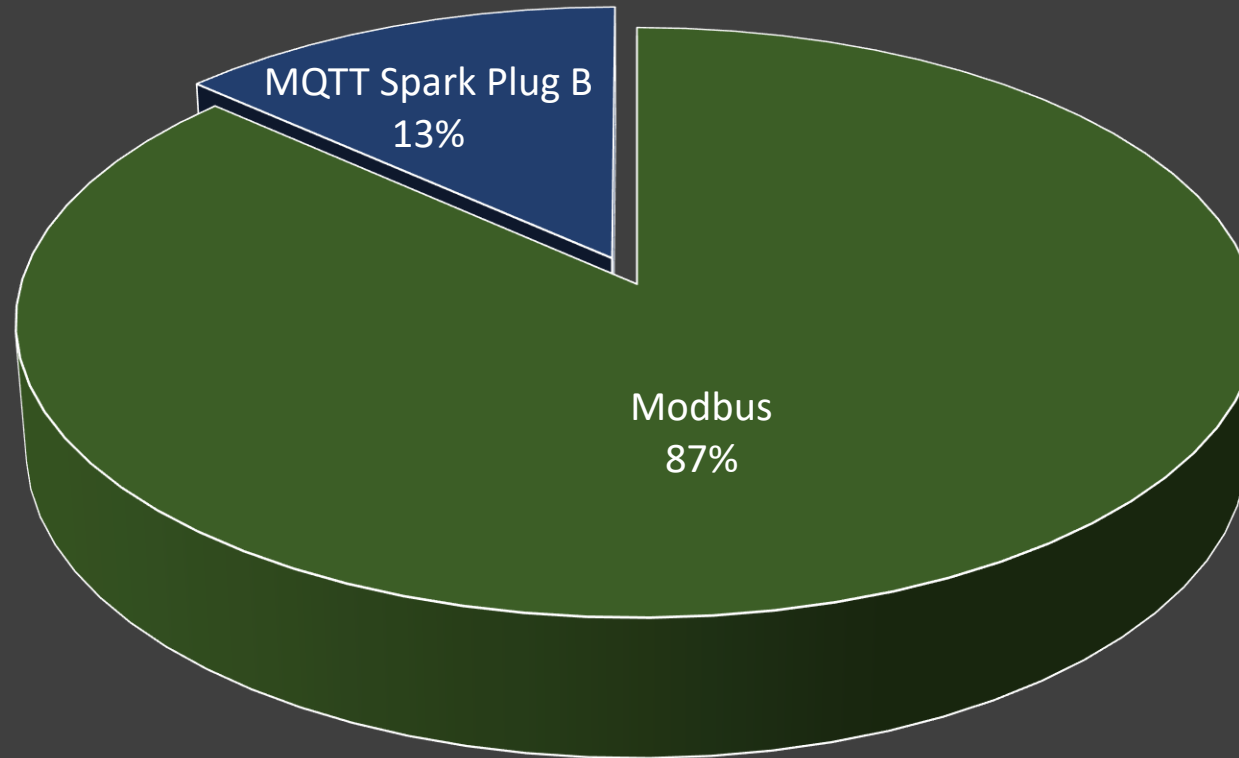
Packets for 90 values



Bytes for 90 values



MQTT Sparkplug B x10 buffered vs Modbus Poll Response



■ ■ ■ Modbus ■ MQTT Spark Plug B

	B	C	D	N	O	P	Q	R
	item	Tag Path	Changes per hour	MQTT	MQTT All	Modbus	HTTP	OPC
1	60	/Valves/PID01/DeadBand	0	0	574.2	831.6	1709.4	2706
2	60	/Valves/PID01/Derivative	0	0	574.2	831.6	1709.4	2706
3	60	/Valves/PID01/Integral	0	0	574.2	831.6	1709.4	2706
4	60	/Valves/PID01/Local SetPoint	0	0	574.2	831.6	1709.4	2706
5	60	/Valves/PID01/Max Output	0	0	574.2	831.6	1709.4	2706
6	60	/Valves/PID01/MaxSP	0	0	574.2	831.6	1709.4	2706
7	60	/Valves/PID01/PID Enabled	0	0	574.2	831.6	1709.4	2706
8	60	/Valves/PID01/PID Mode	0	0	574.2	831.6	1709.4	2706
9	60	/Valves/PID01/Proportional	0	0	574.2	831.6	1709.4	2706
10	60	/Valves/PID01/Ramp	0	0	574.2	831.6	1709.4	2706
11	60	/Valves/PID01/Remote SetPoint	0	0	574.2	831.6	1709.4	2706
12	60	/Valves/SSD/Status	0	0	574.2	831.6	1709.4	2706
13	60	/WaterRun01/Water BBL PD	0	0	574.2	831.6	1709.4	2706
14	60	/APPlunger/A STATE	1	9.57	574.2	831.6	1709.4	2706
15	60	/APPlunger/CONSECUTIVE EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
16	60	/APPlunger/Consecutive Non Arrivals	1	9.57	574.2	831.6	1709.4	2706
17	60	/APPlunger/COUNT FAST ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
18	60	/APPlunger/COUNT LATE ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
19	60	/APPlunger/COUNT NO ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
20	60	/APPlunger/COUNT NORMAL ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
21	60	/APPlunger/COUNT SLOW ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
22	60	/APPlunger/EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
23	60	/APPlunger/History/ArrvITime1	1	9.57	574.2	831.6	1709.4	2706
24	60	/APPlunger/History/ArrvITime10	1	9.57	574.2	831.6	1709.4	2706

Oil and Gas RTU Data On Change



Retrieve 389 tags for a wellsite. Data on exception and limited to minute resolution - Modbus vs MQTT on change.

Sample data on a real Thermo AutoPilot Pro RTU with plunger tags
Total 389 tags
Estimated how often the values would change based on usual activity and theoretical deadbands

223 discrete tags that would not change in an hour

111 tags would change hourly

19 intermittently used analog values

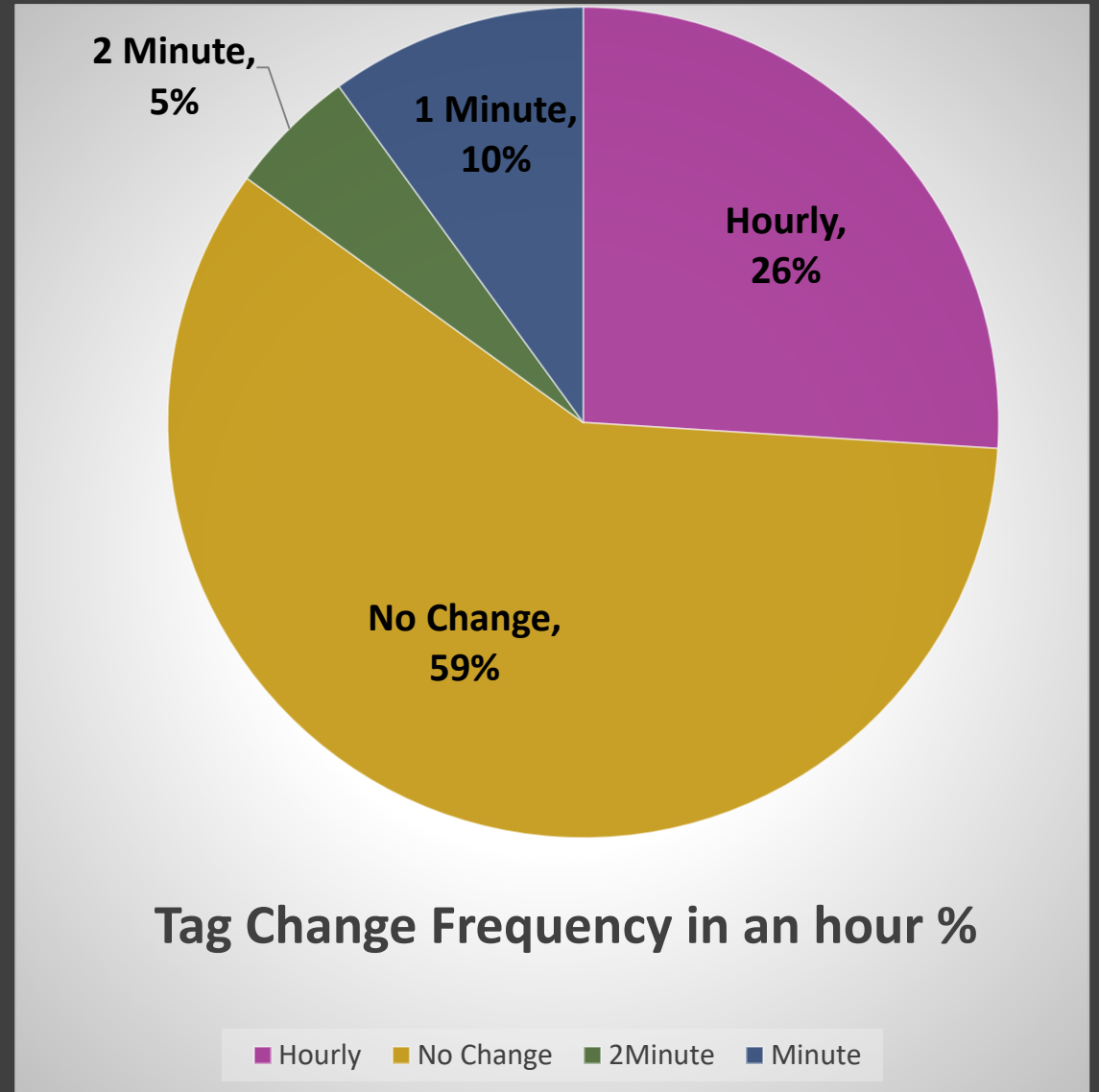
36 analog values would change each minute

	B	C	D	N	O	P	Q	R	
1	item	Tag Path	Changes per hour	MQTT	MQTT All	Modbus	HTTP	OPC	
211	60	/Valves/PID01/DeadBand	0	0	574.2	831.6	1709.4	2706	
212	60	/Valves/PID01/Derivative	0	0	574.2	831.6	1709.4	2706	
213	60	/Valves/PID01/Integral	0	0	574.2	831.6	1709.4	2706	
214	60	/Valves/PID01/Local SetPoint	0	0	574.2	831.6	1709.4	2706	
215	60	/Valves/PID01/Max Output	0	0	574.2	831.6	1709.4	2706	
216	60	/Valves/PID01/MaxSP	0	0	574.2	831.6	1709.4	2706	
217	60	/Valves/PID01/PID Enabled	0	0	574.2	831.6	1709.4	2706	
218	60	/Valves/PID01/PID Mode	0	0	574.2	831.6	1709.4	2706	
219	60	/Valves/PID01/Proportional	0	0	574.2	831.6	1709.4	2706	
220	60	/Valves/PID01/Ramp	0	0	574.2	831.6	1709.4	2706	
221	60	/Valves/PID01/Remote SetPoint	0	0	574.2	831.6	1709.4	2706	
222	60	/Valves/SSD/Status	0	0	574.2	831.6	1709.4	2706	
223	60	/WaterRun01/Water BBL PD	0	0	574.2	831.6	1709.4	2706	
224	60	/APPlunger/A STATE	1	9.57	574.2	831.6	1709.4	2706	
225	60	/APPlunger/CONSECUTIVE EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
226	60	/APPlunger/Consecutive Non Arrivals	1	9.57	574.2	831.6	1709.4	2706	
227	60	/APPlunger/COUNT FAST ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
228	60	/APPlunger/COUNT LATE ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
229	60	/APPlunger/COUNT NO ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
230	60	/APPlunger/COUNT NORMAL ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
231	60	/APPlunger/COUNT SLOW ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
232	60	/APPlunger/EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706	
233	60	/APPlunger/History/ArrvlTime1	1	9.57	574.2	831.6	1709.4	2706	
234	60	/APPlunger/History/ArrvlTime10	1	9.57	574.2	831.6	1709.4	2706	

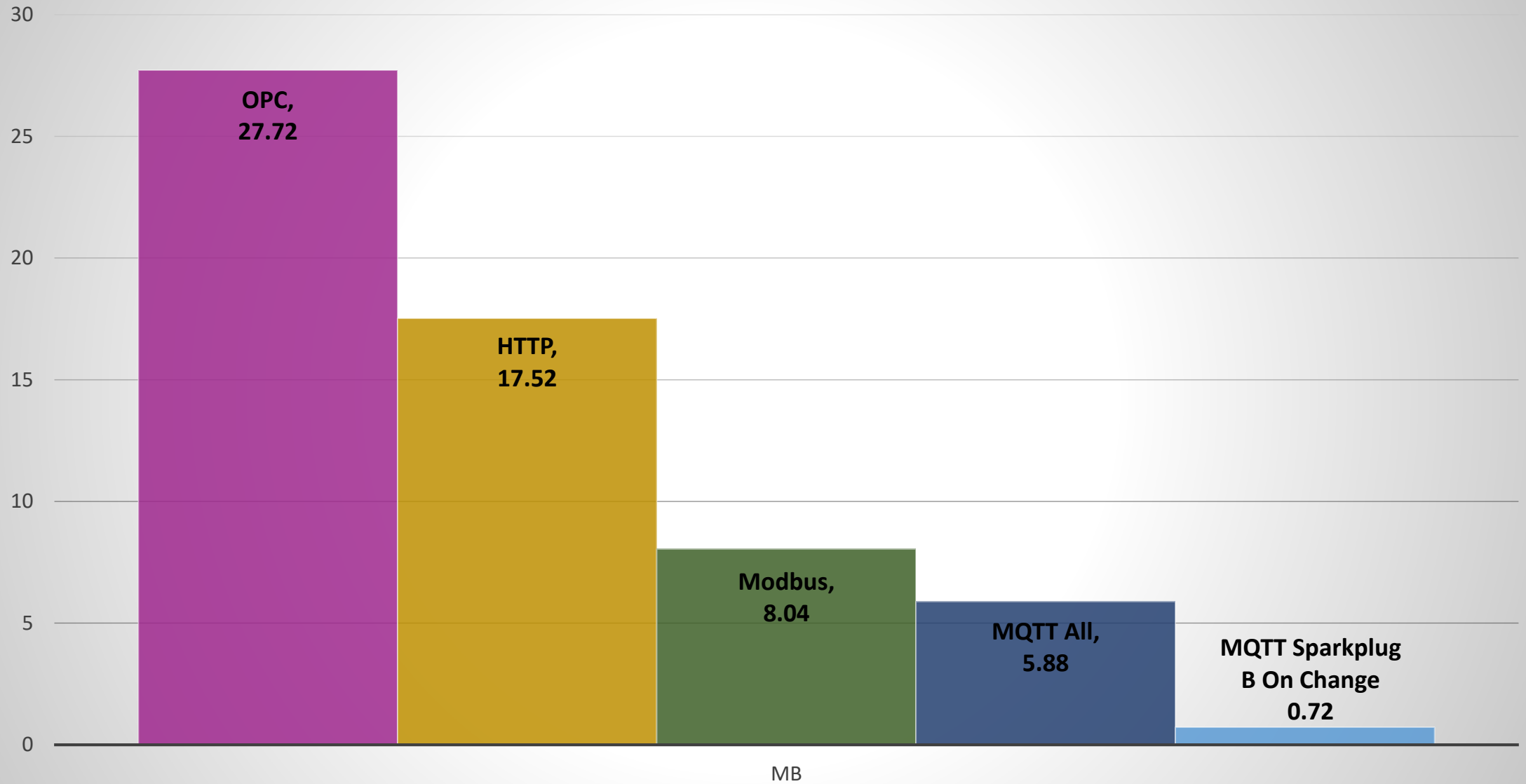
Facilities Tags

59% of the data doesn't change
even once in a hour

So why poll for it each time...

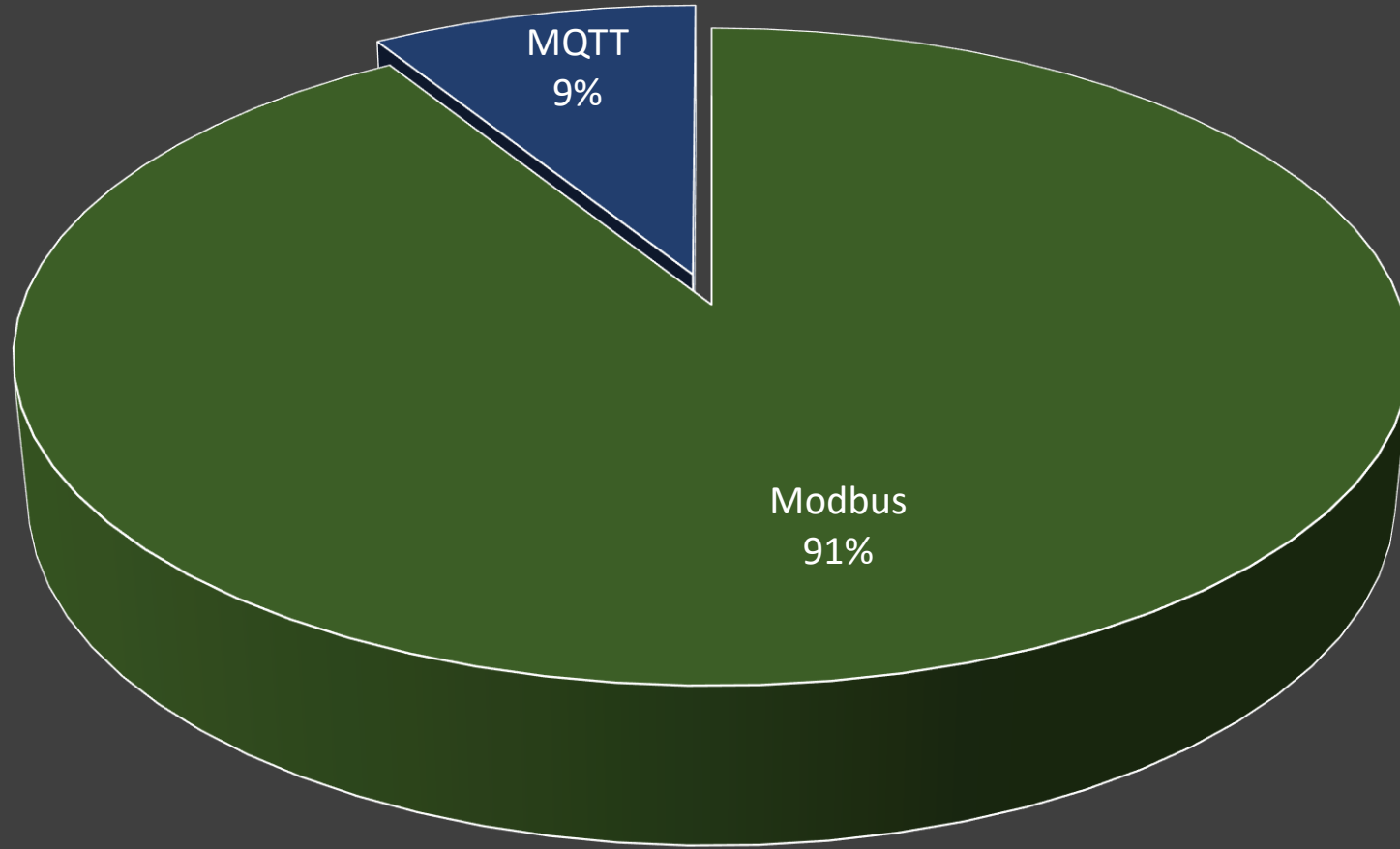


mB to poll minute resolution data on 389 tags over a 24 hour period



■ OPC ■ HTTP ■ Modbus ■ MQTT All ■ MQTT Sparkplug B On Change

mB MQTT Sparkplug B On Change vs Modbus Poll Response



■ ■ ■ Modbus ■ MQTT

Final Takeaways

- The most bandwidth savings comes from report by exception
- Sparkplug compresses 3x
- Trading Modbus for MQTT Sparkplug on exception can result in ~75% to ~99.5% network bandwidth savings but it will depend on your application, number of points, criticality of the data, and other factors
- Network bandwidth savings is always an estimate until you actually implement and test in the real world...

About The Author

Johnathan Hottell has over 18 years of industrial SCADA Automation experience and is currently the SCADA Supervisor at EXCO Resources, Inc.

Social

LinkedIn: Johnathan Hottell

Twitter: electronhacks

YouTube: electronhacks

The views expressed in this article do not necessarily reflect the views of EXCO Resources, Inc.