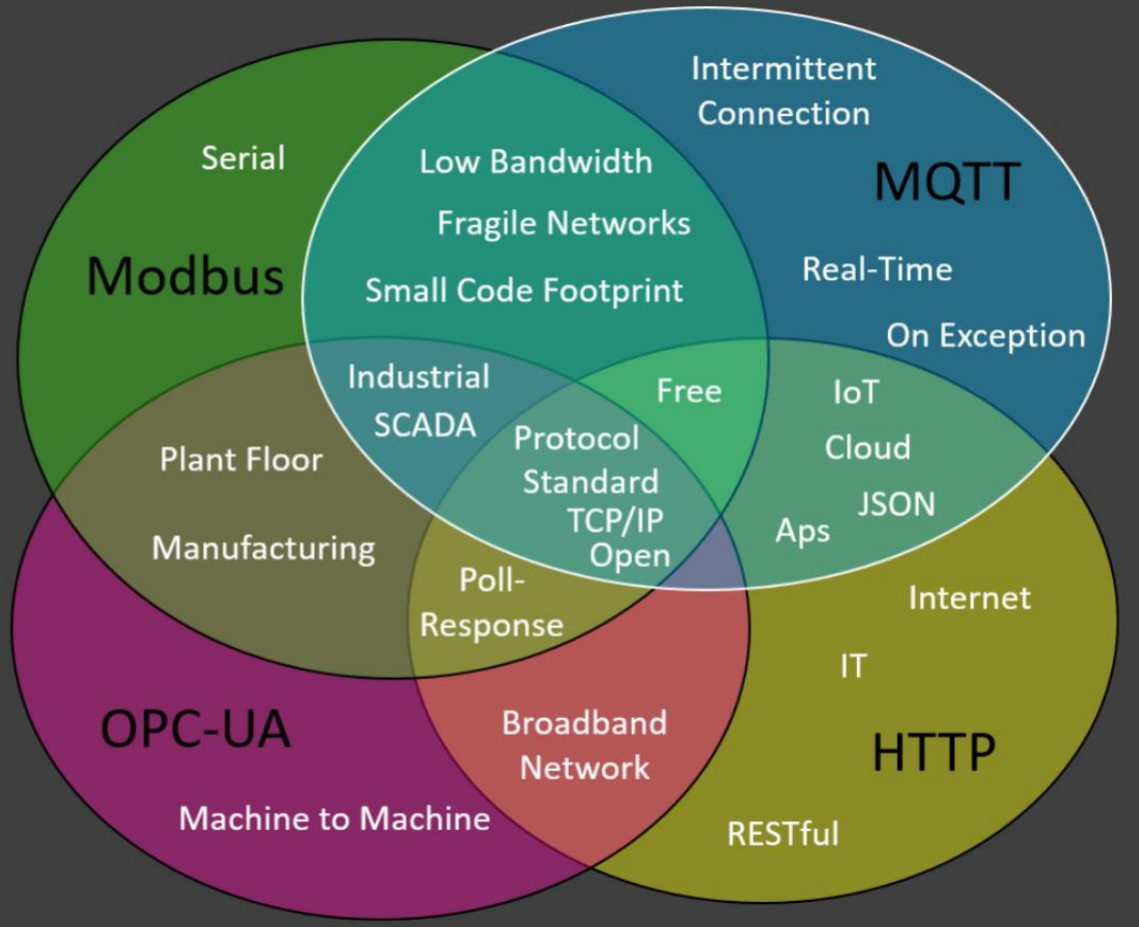
Efficient IIoT Communications

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

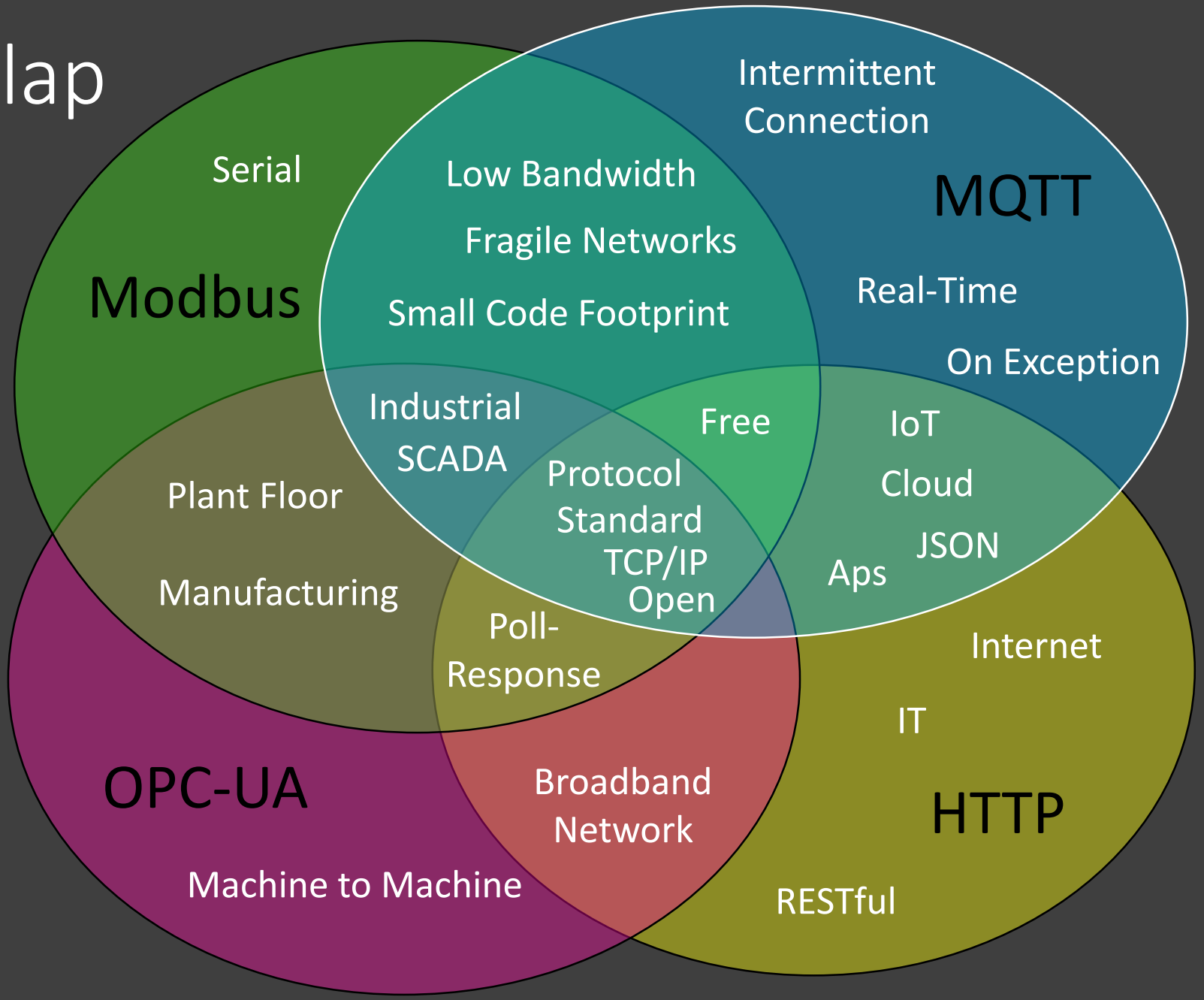
Start by framing the application more clearly

- 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

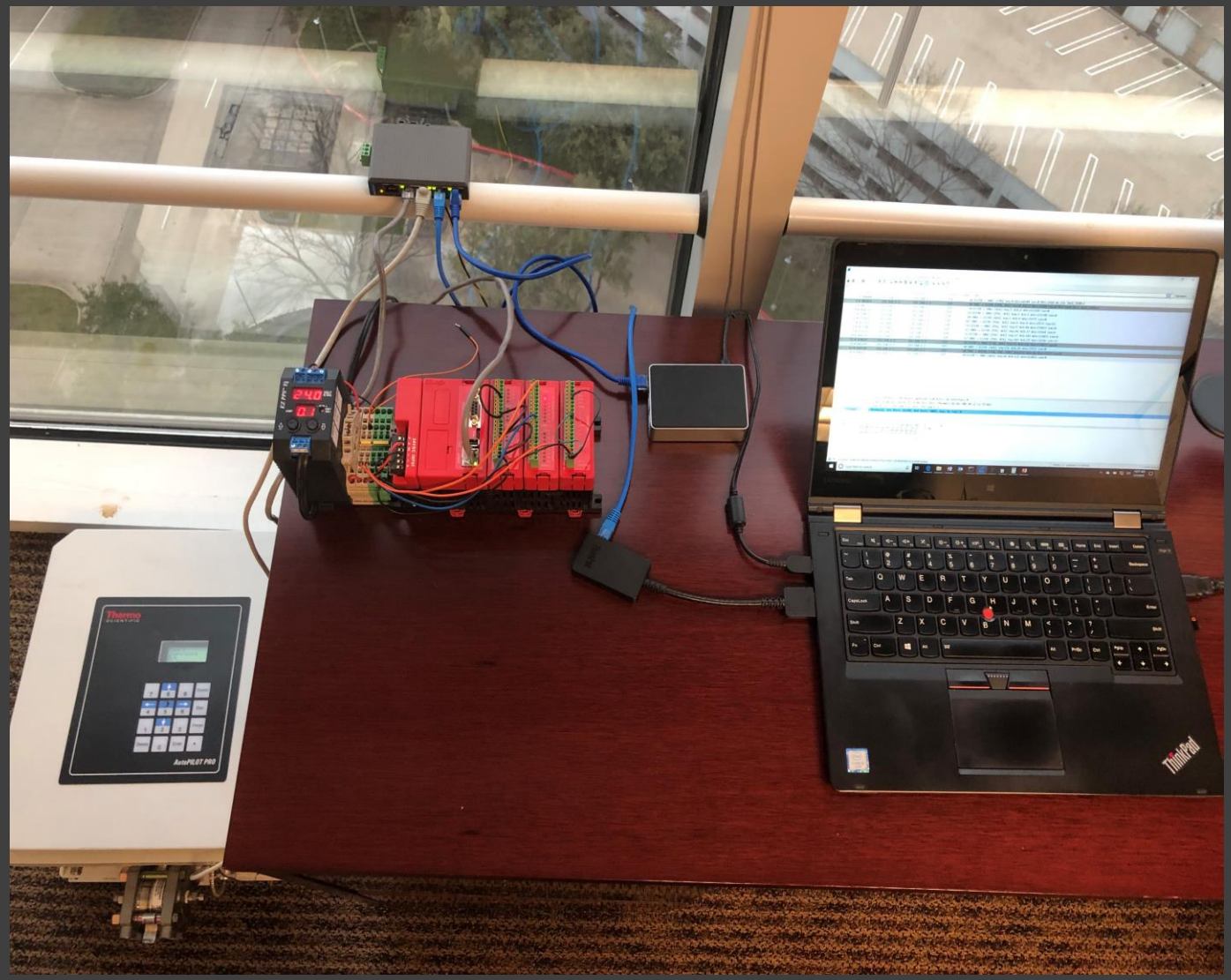
- Many protocols can be used in similar scenarios – that doesn't mean they're a perfect fit for them all
- What are your applications most important requirements?



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

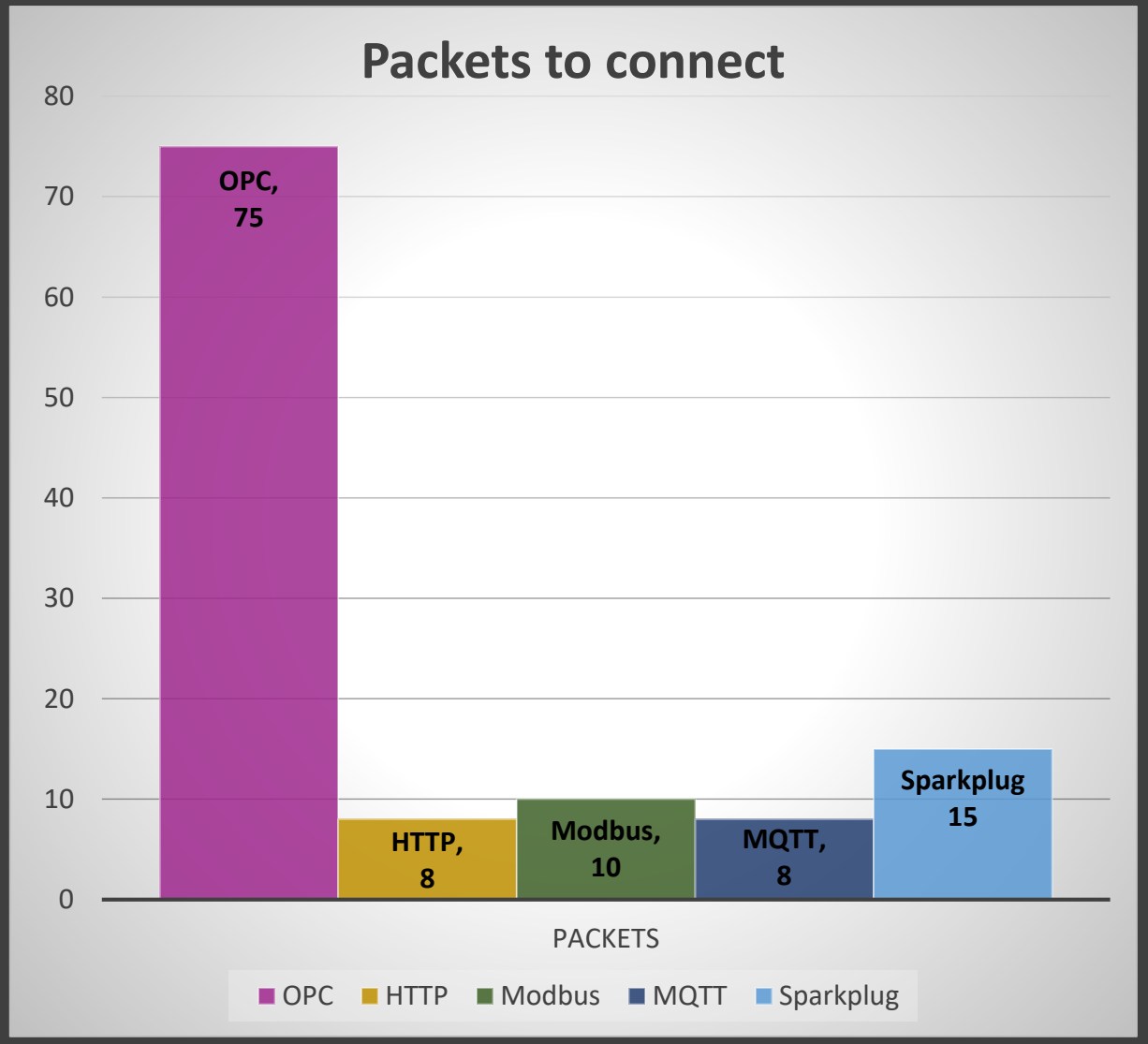
Communication Events That Consume Bandwith

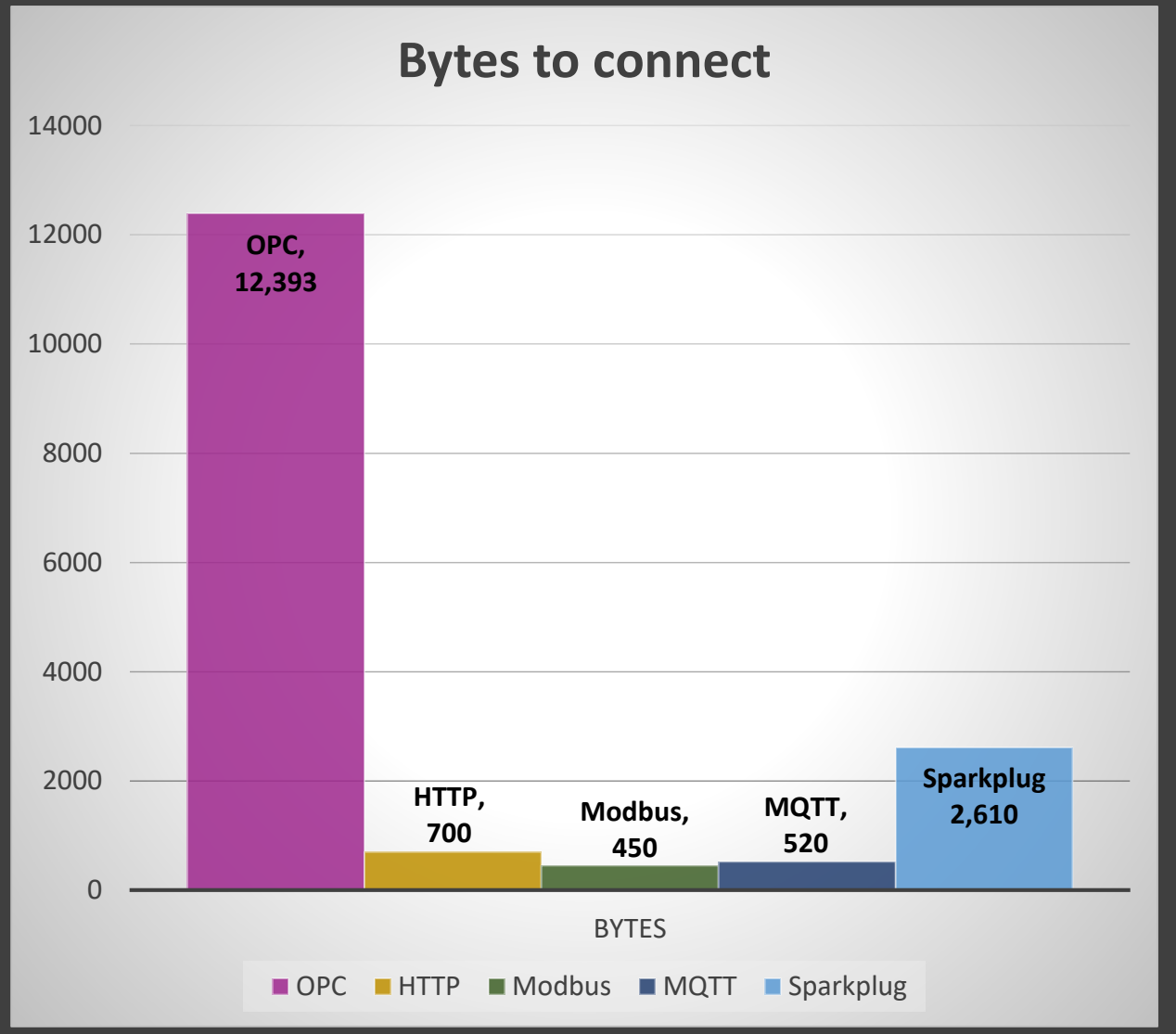
- Making the connection
- Transferring data
- Keeping the connection alive
- Tearing down the connection

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

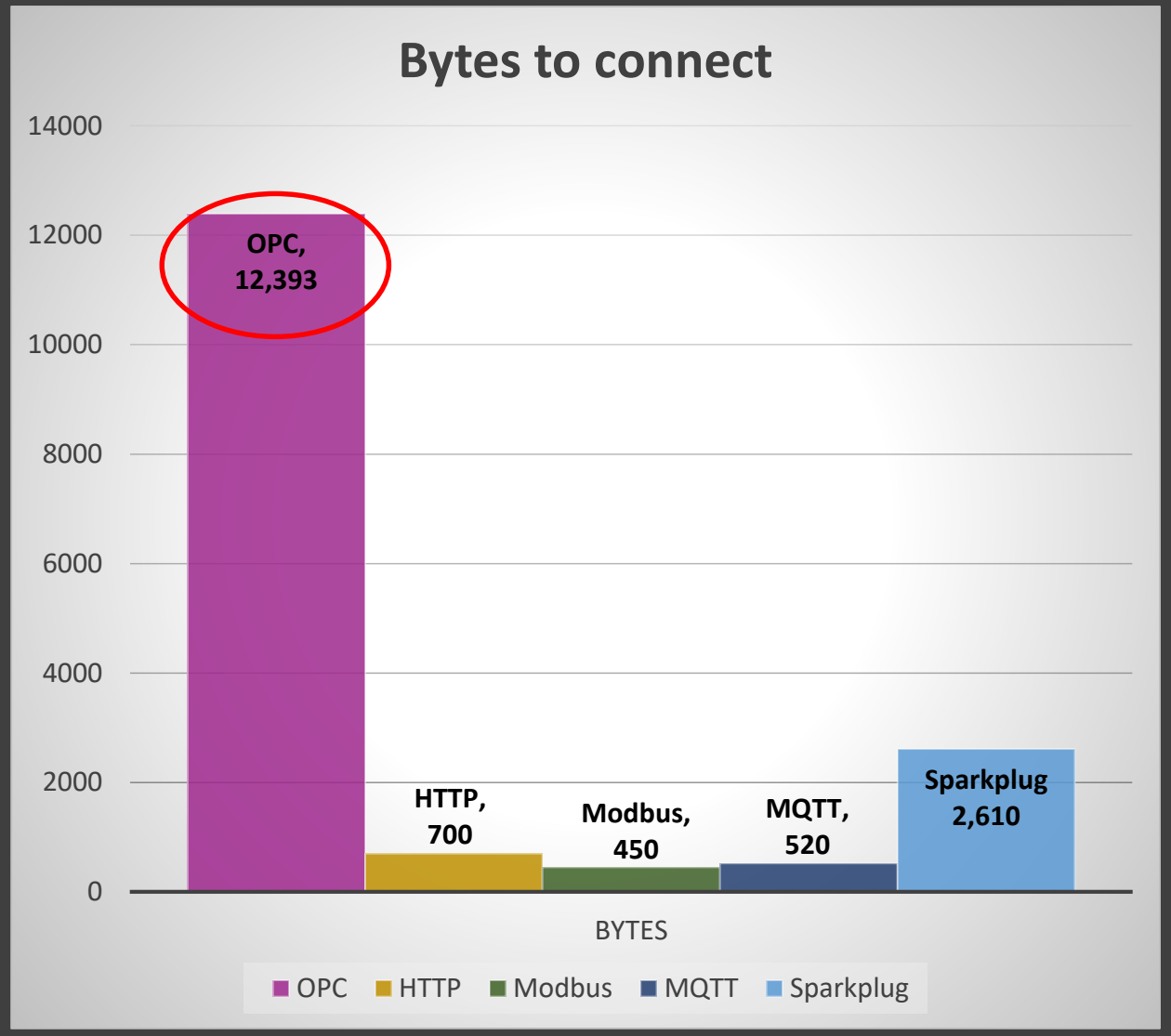
Making the Connection





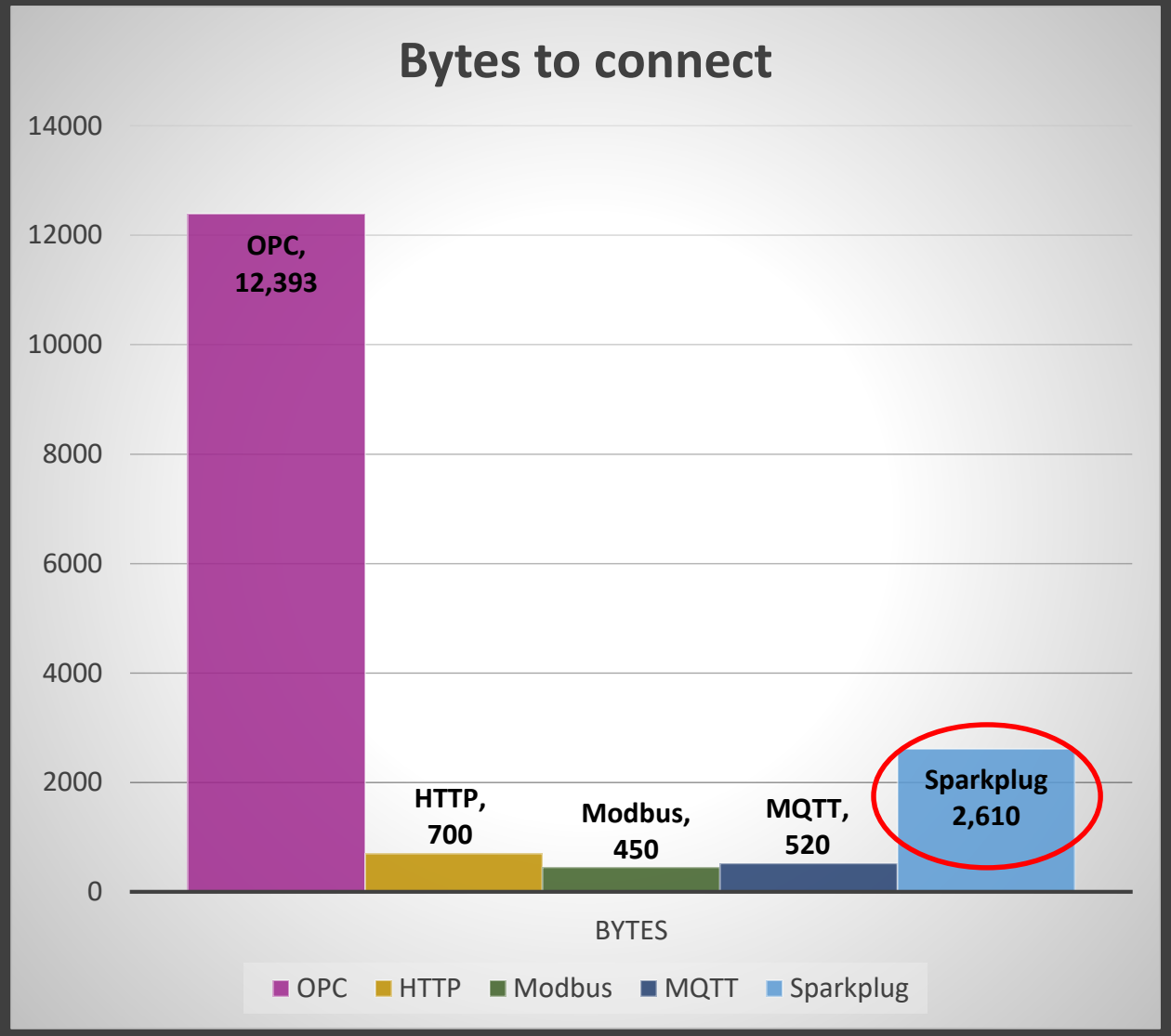
Cost to Connect

- 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

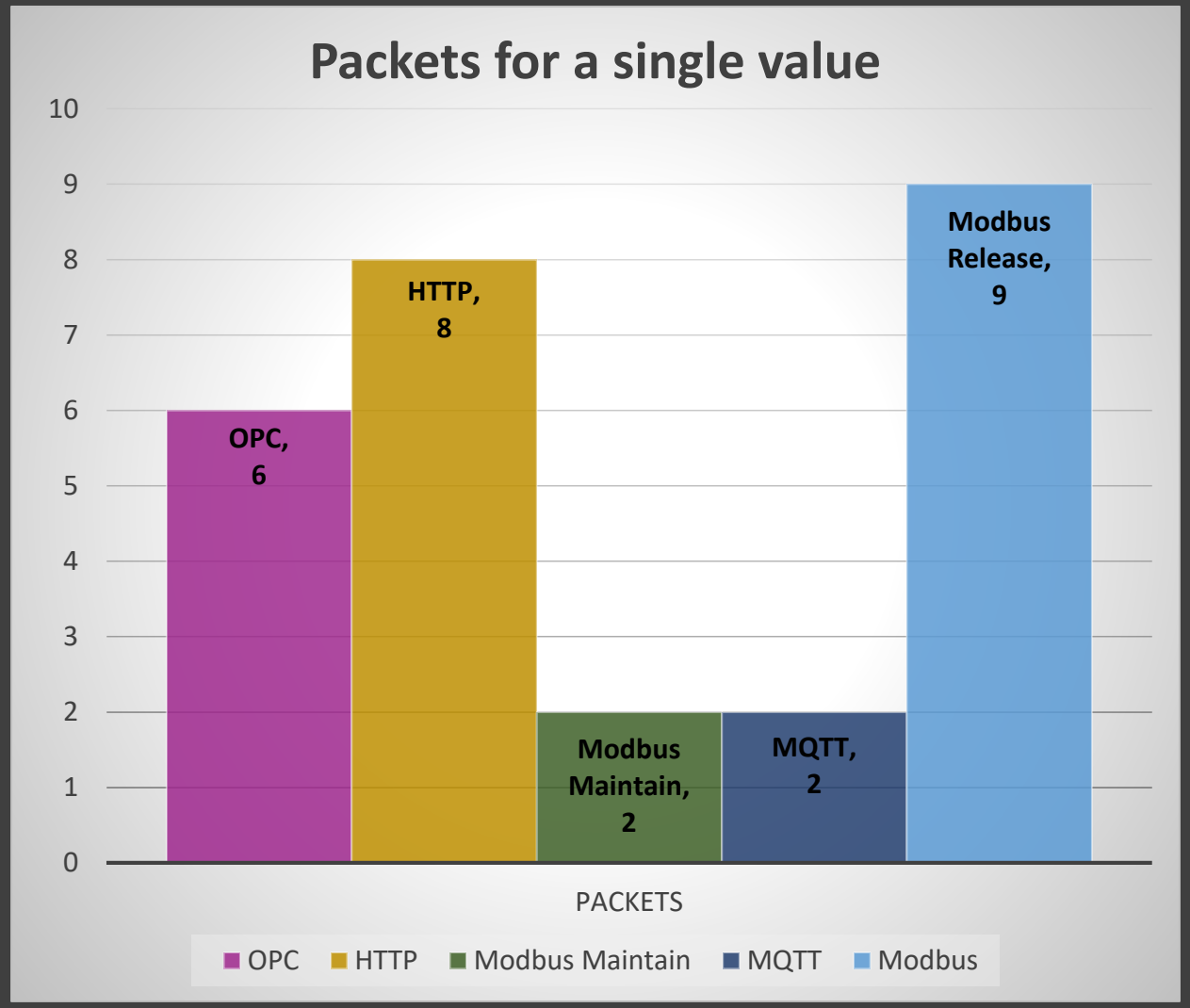
- Sparkplug B sends a bit more data on connection than some of the other protocols, this is because of the “Birth” message

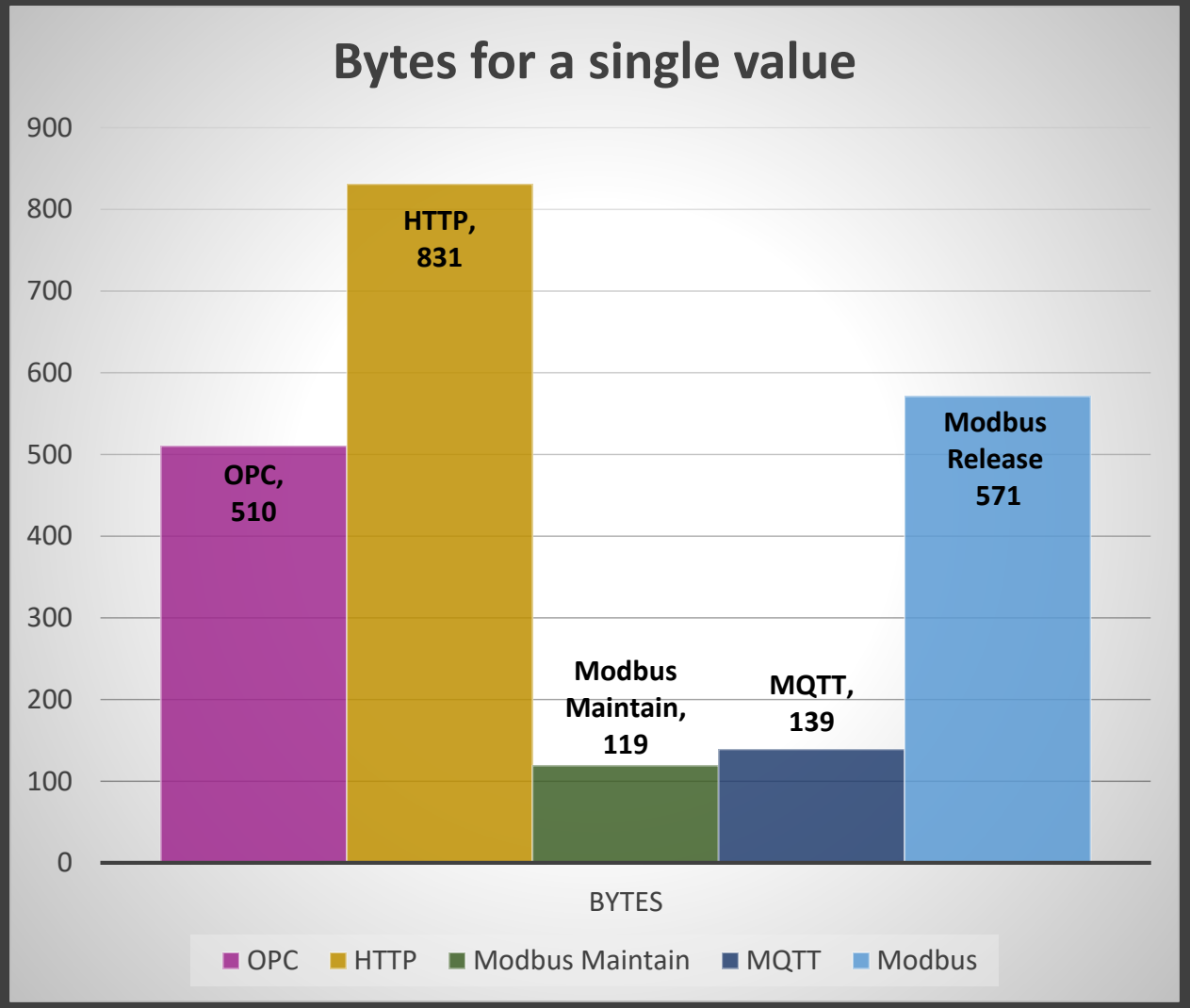


Payload

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

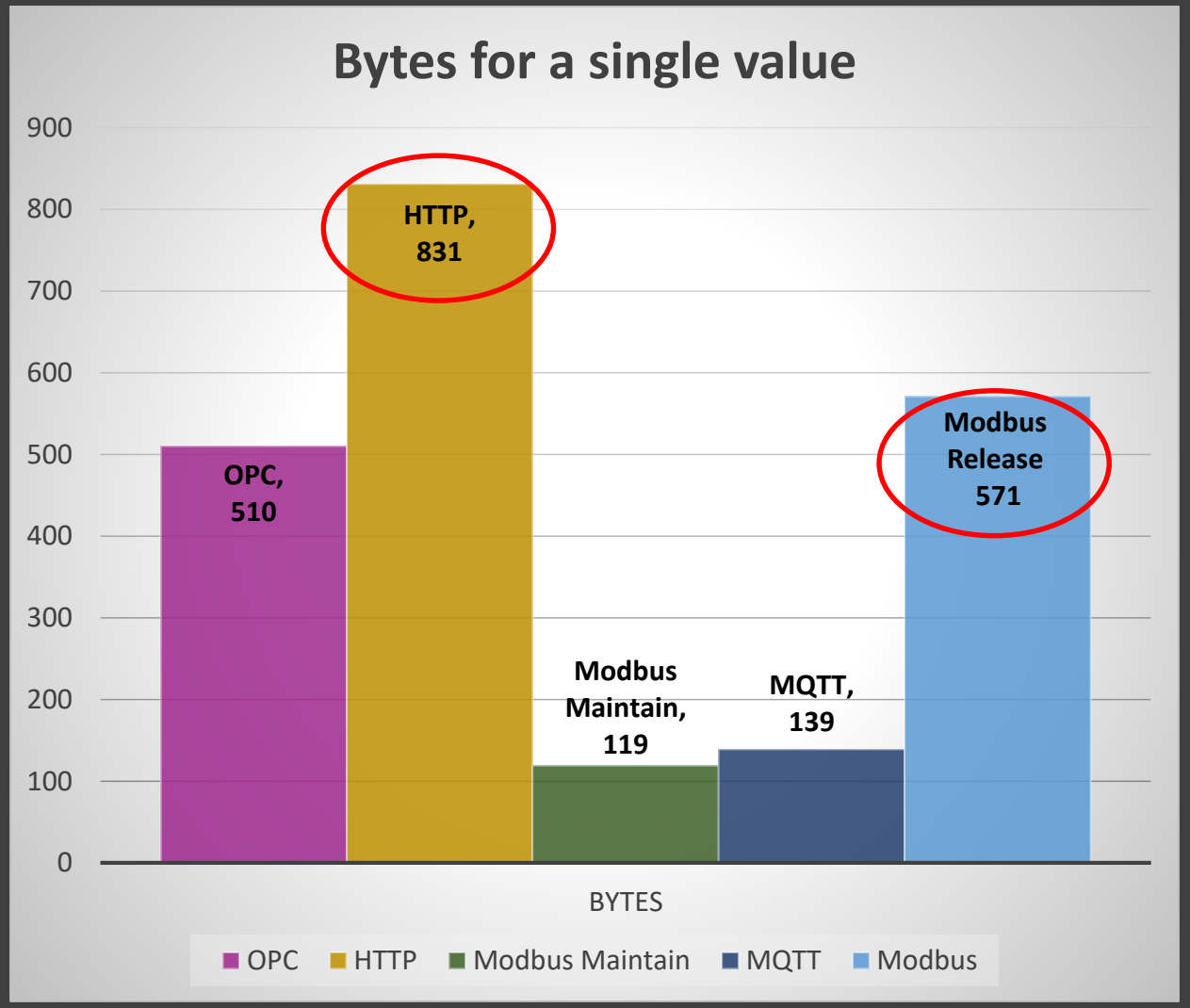
Data used to send a single value





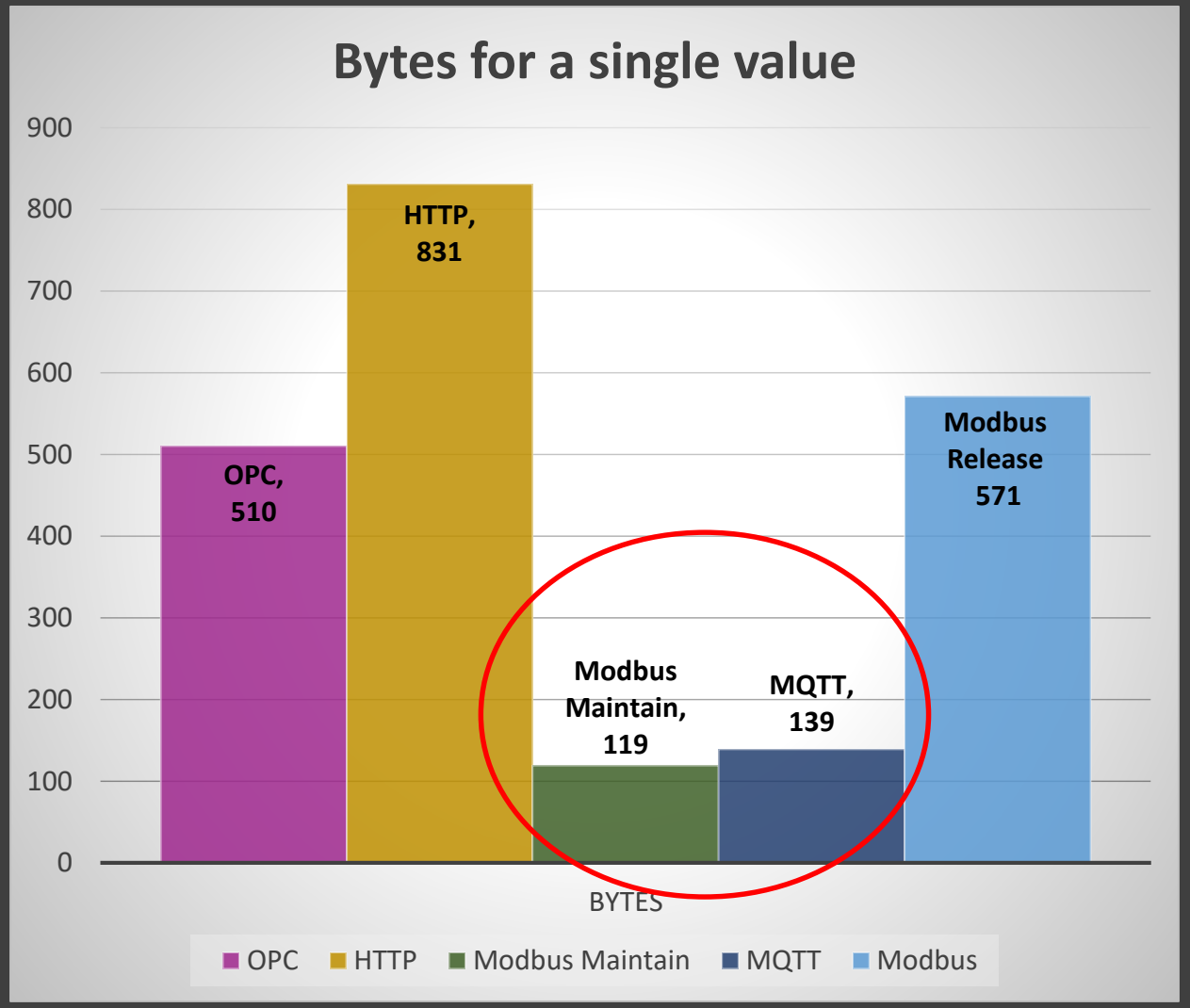
Maintaining a Connection vs Disconnecting When Done

- 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

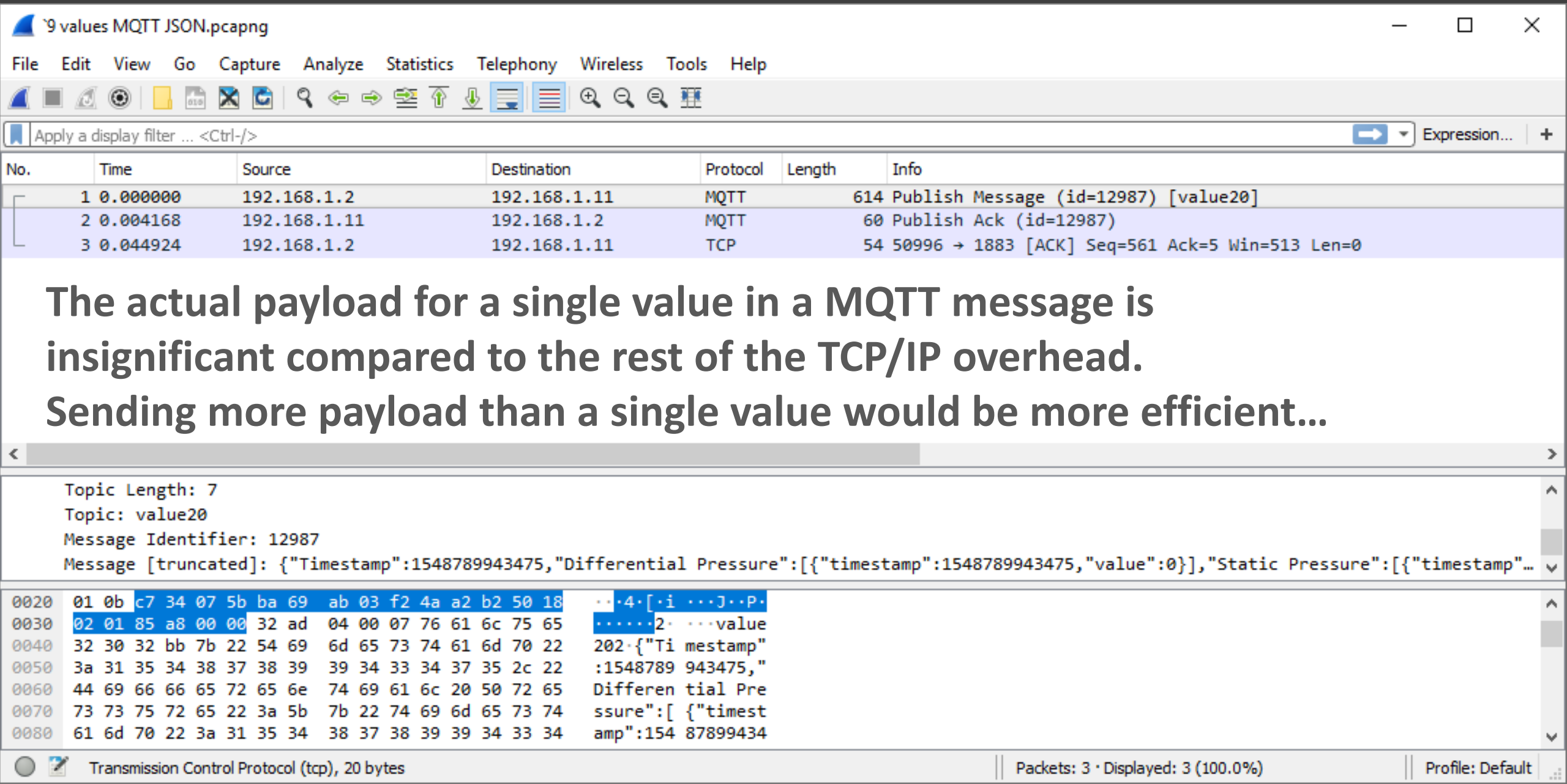


Maintaining a Connection vs Disconnecting When Done

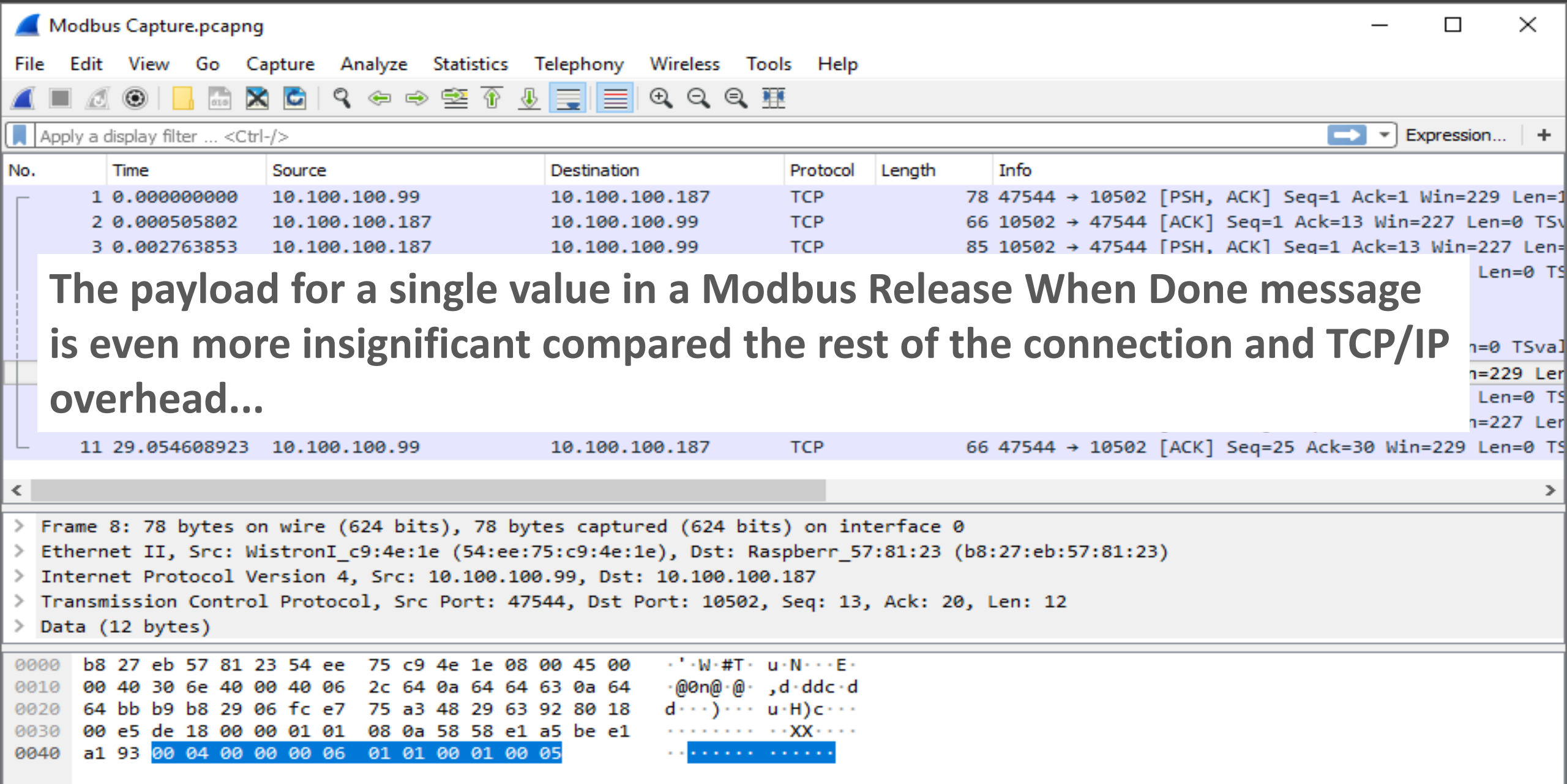
- MQTT is intended to connect then maintain the connection
- Modbus and MQTT are both fairly efficient when you hold the connection open



MQTT Wireshark



Modbus TCP Release When Done Wireshark



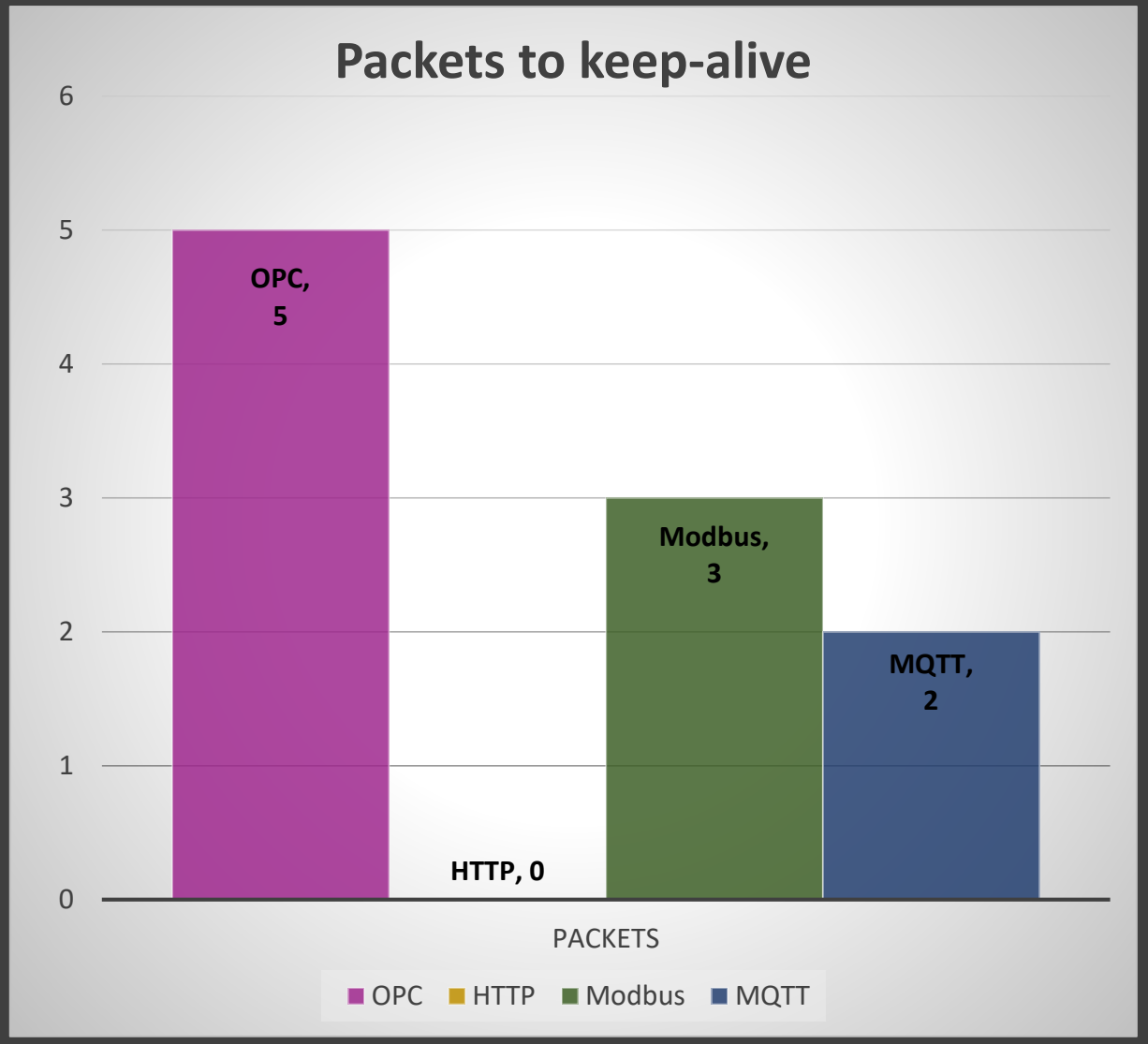
The payload for a single value in a Modbus Release When Done message is even more insignificant compared the rest of the connection and TCP/IP overhead...

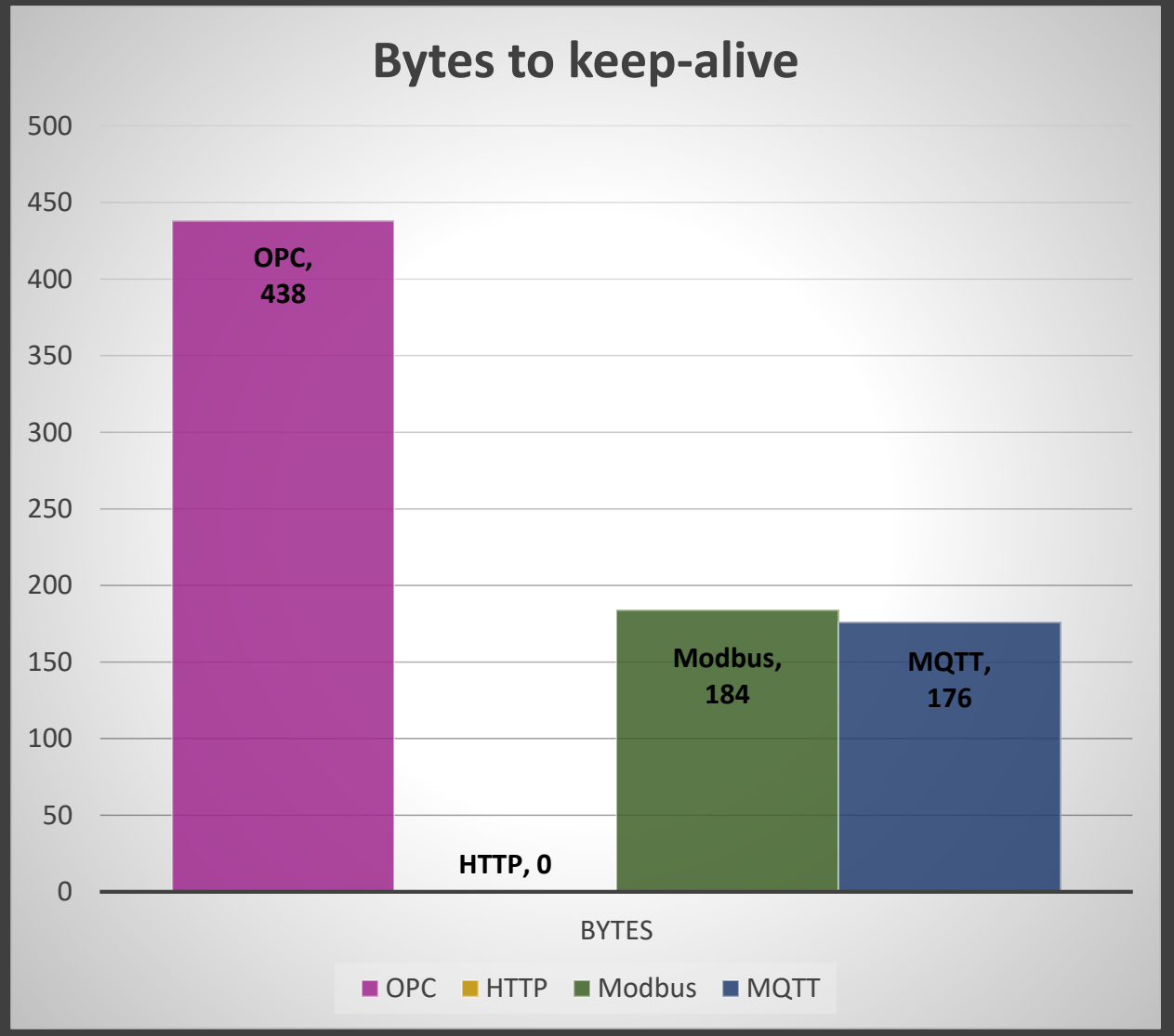
Maintaining the connection

- 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.



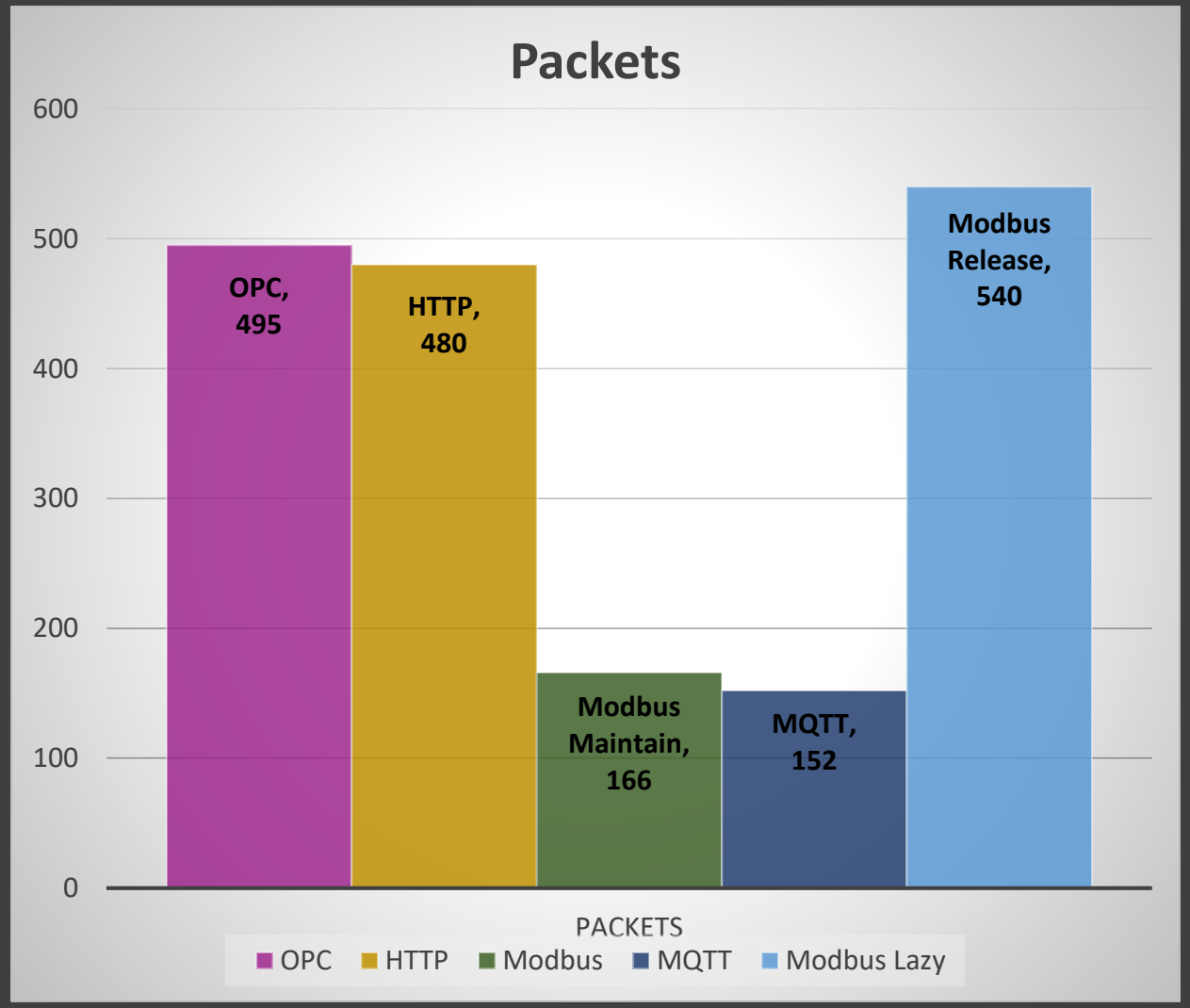


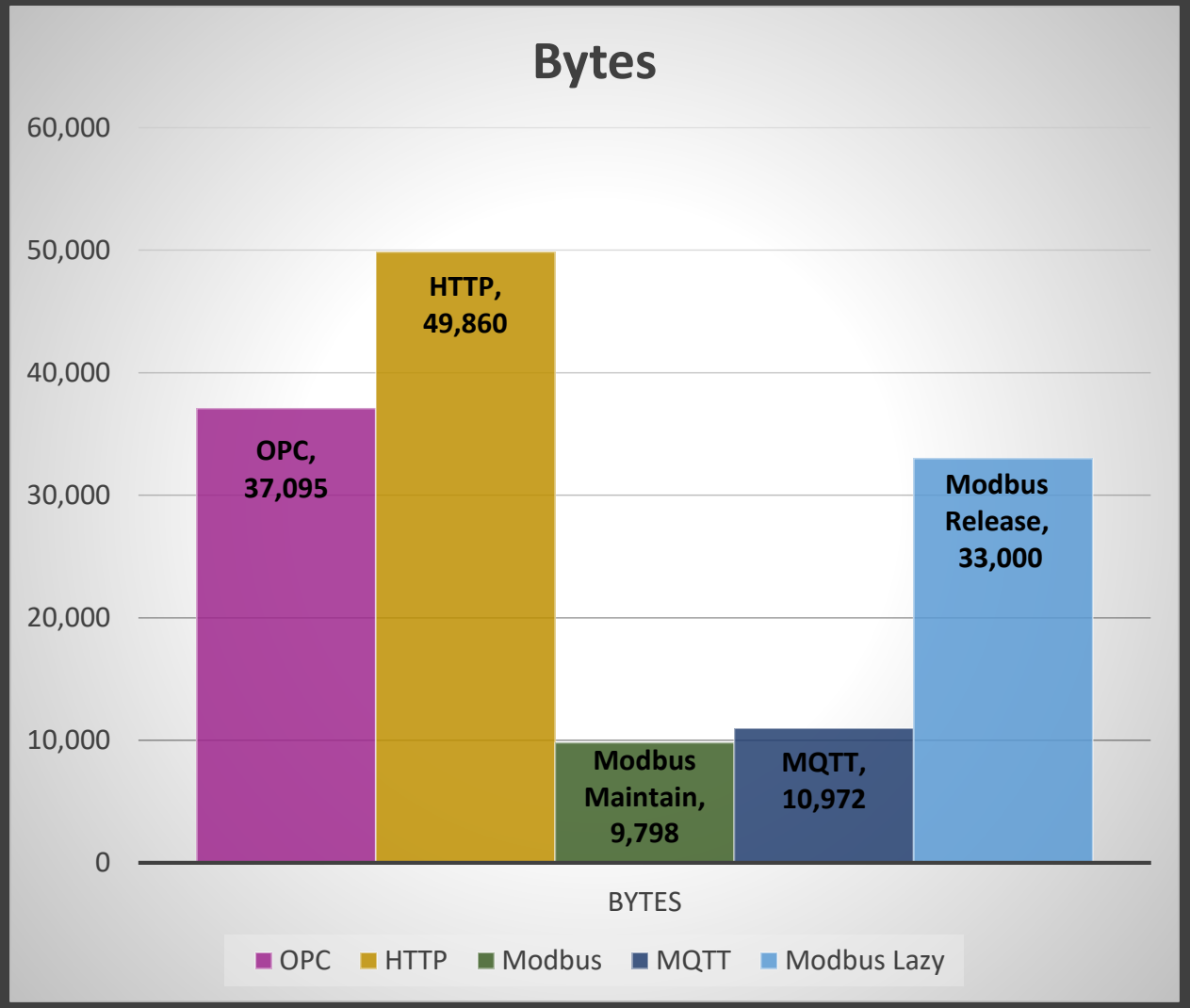
So keeping the things we have learned in mind; What would the “Total Cost of Ownership”(TCO) be to send one value every minute for an hour (60 records), with connection and keepalive cost included?

AKA “Minute Data”

TCO of an hours worth of minute data

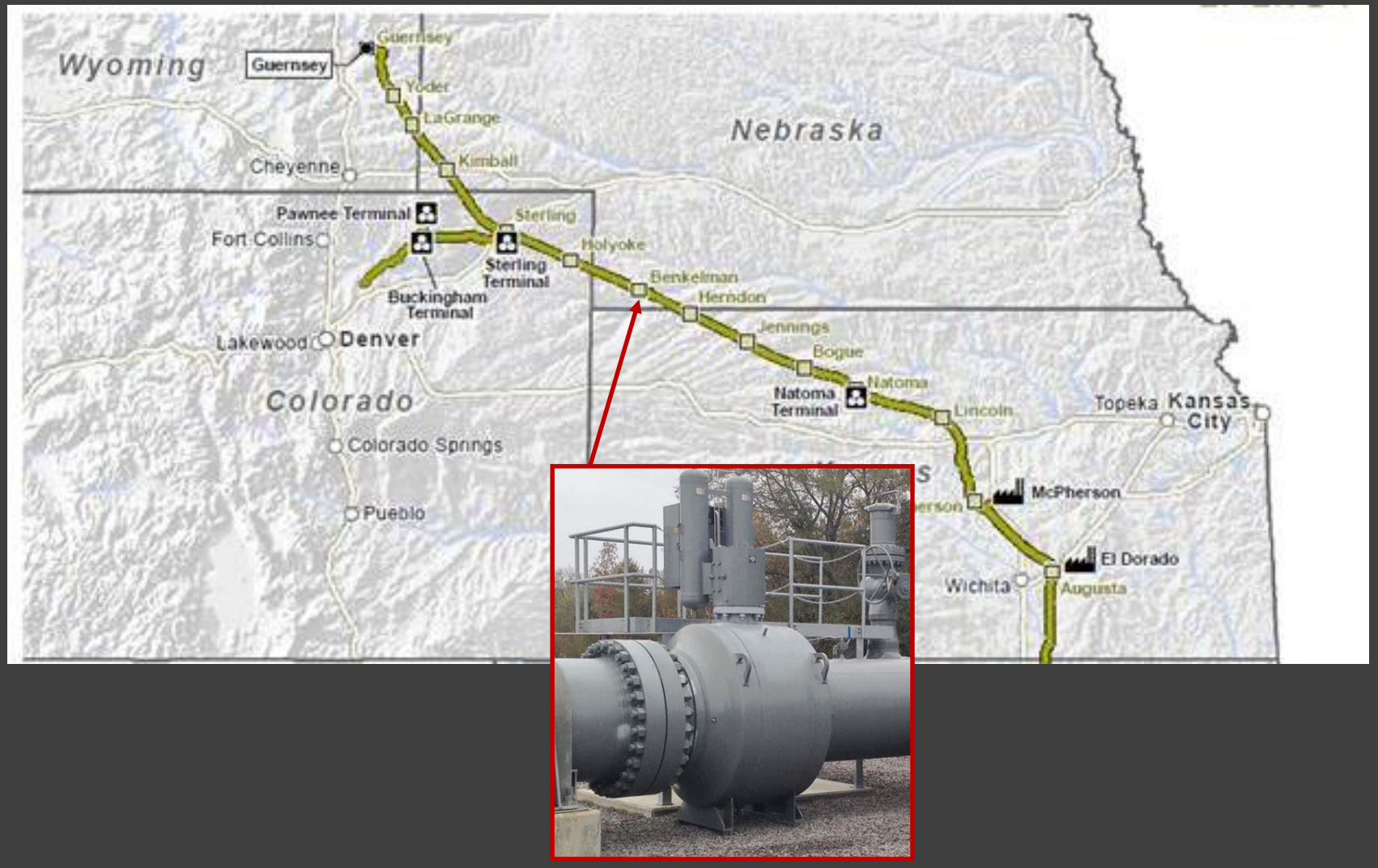
Assuming keep-alive can be set to 5 minutes on OPC, Modbus, and MQTT





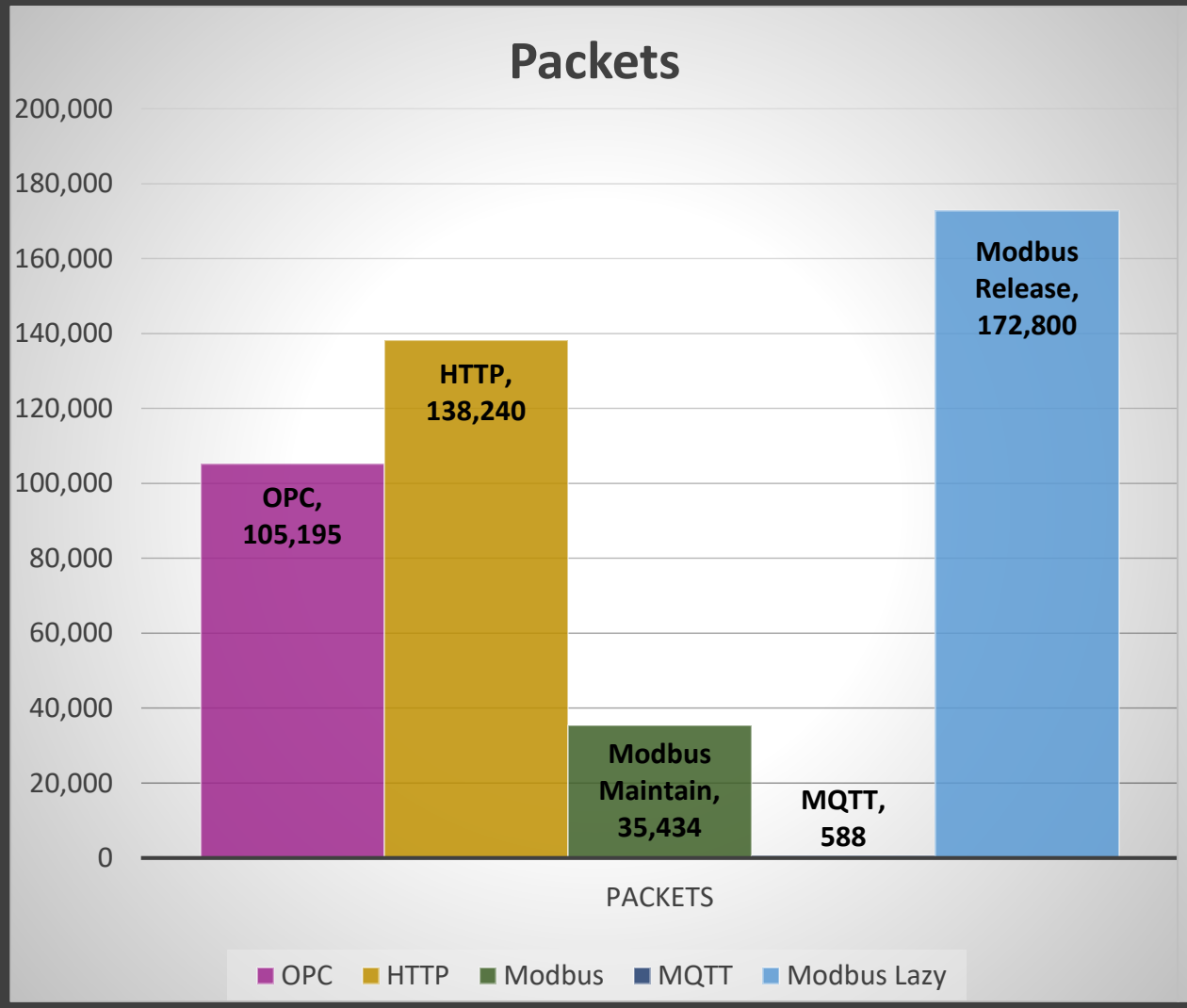
Data On Change Example For a Valve Station

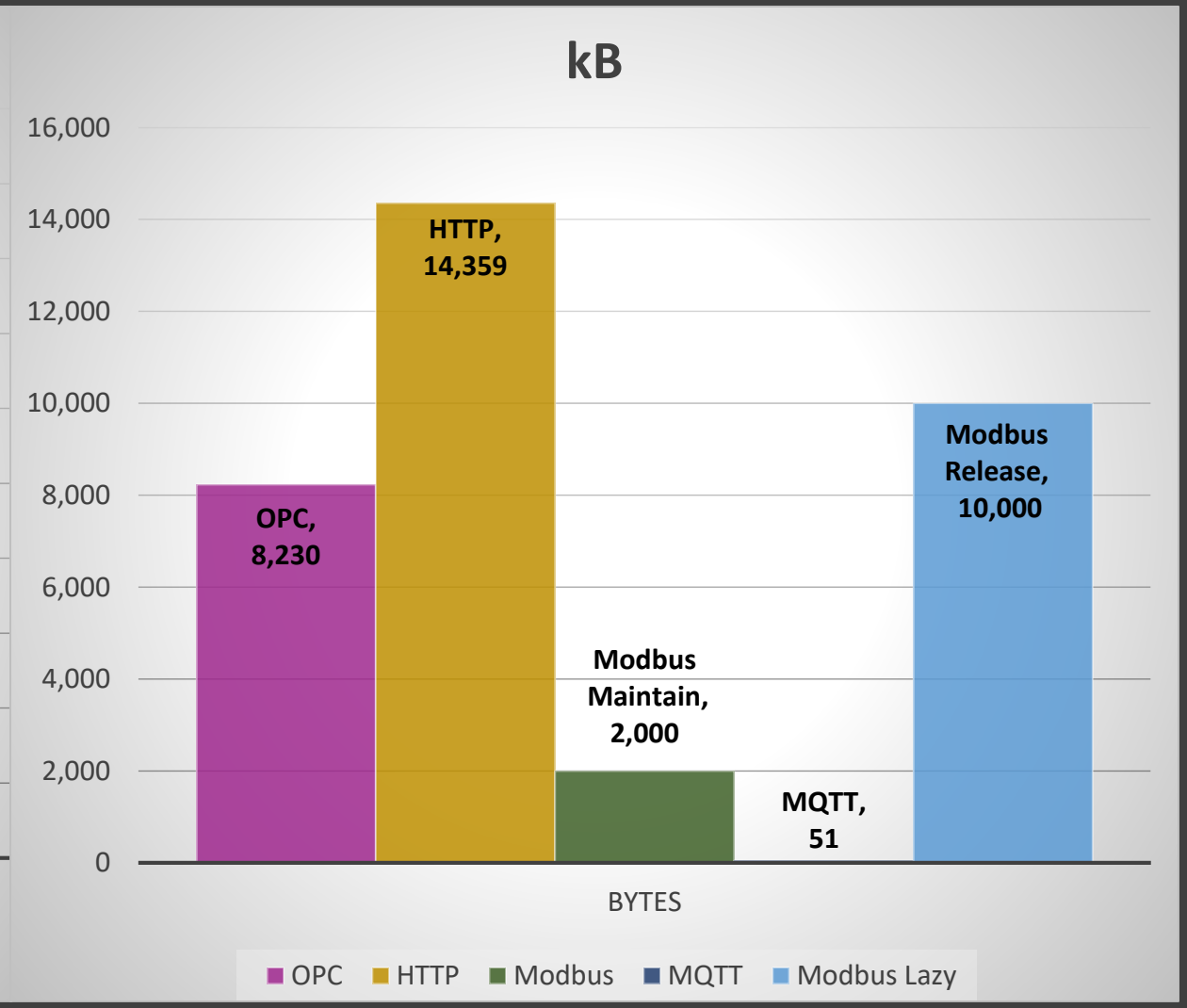
OK, so now what if we needed to know within five seconds when a valve started moving?
(Valve moves twice per day)
(Single tag – not a perfect example)



TCO of a days worth of 5 second data

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

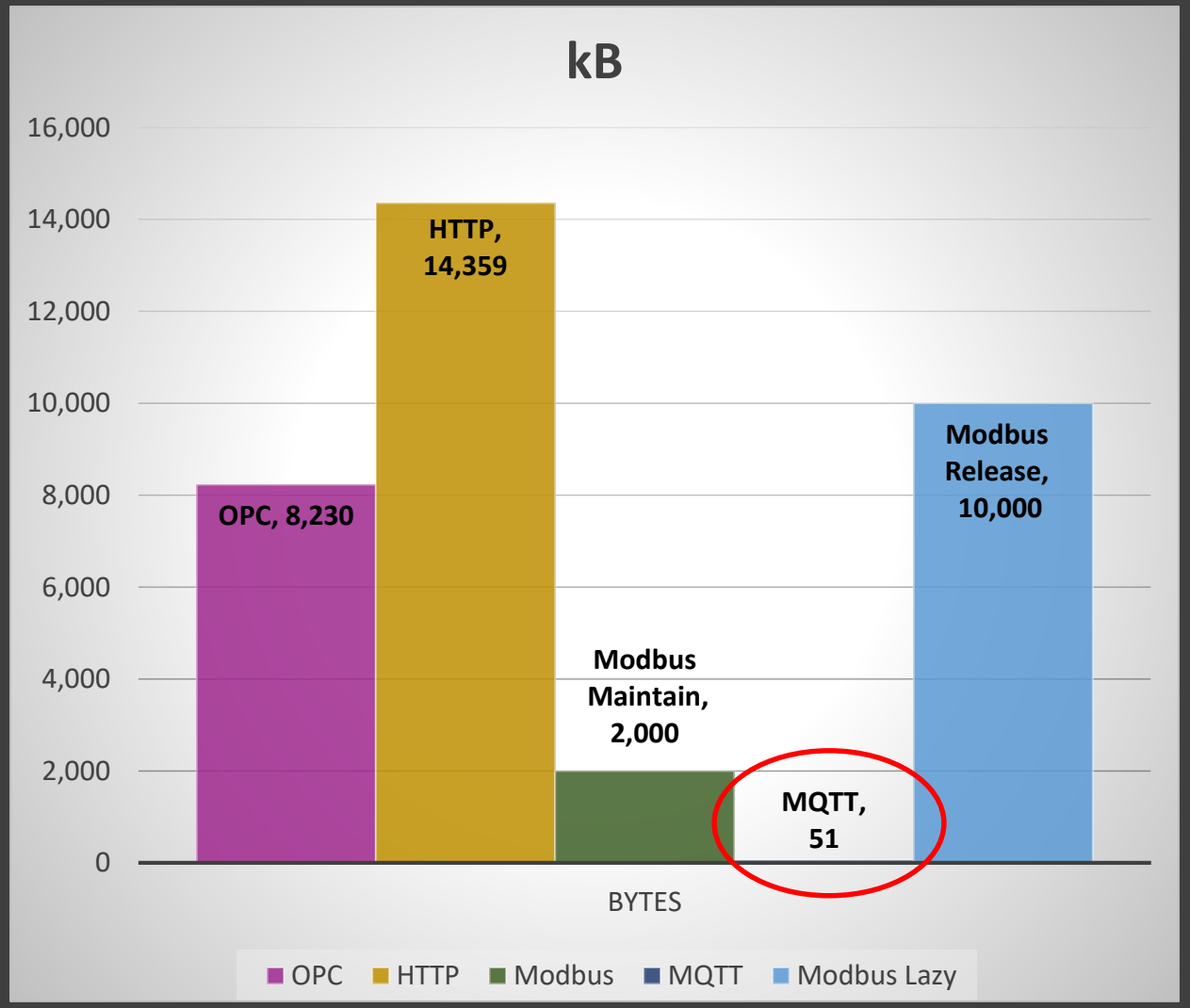




TCO of a days worth of 5 second data

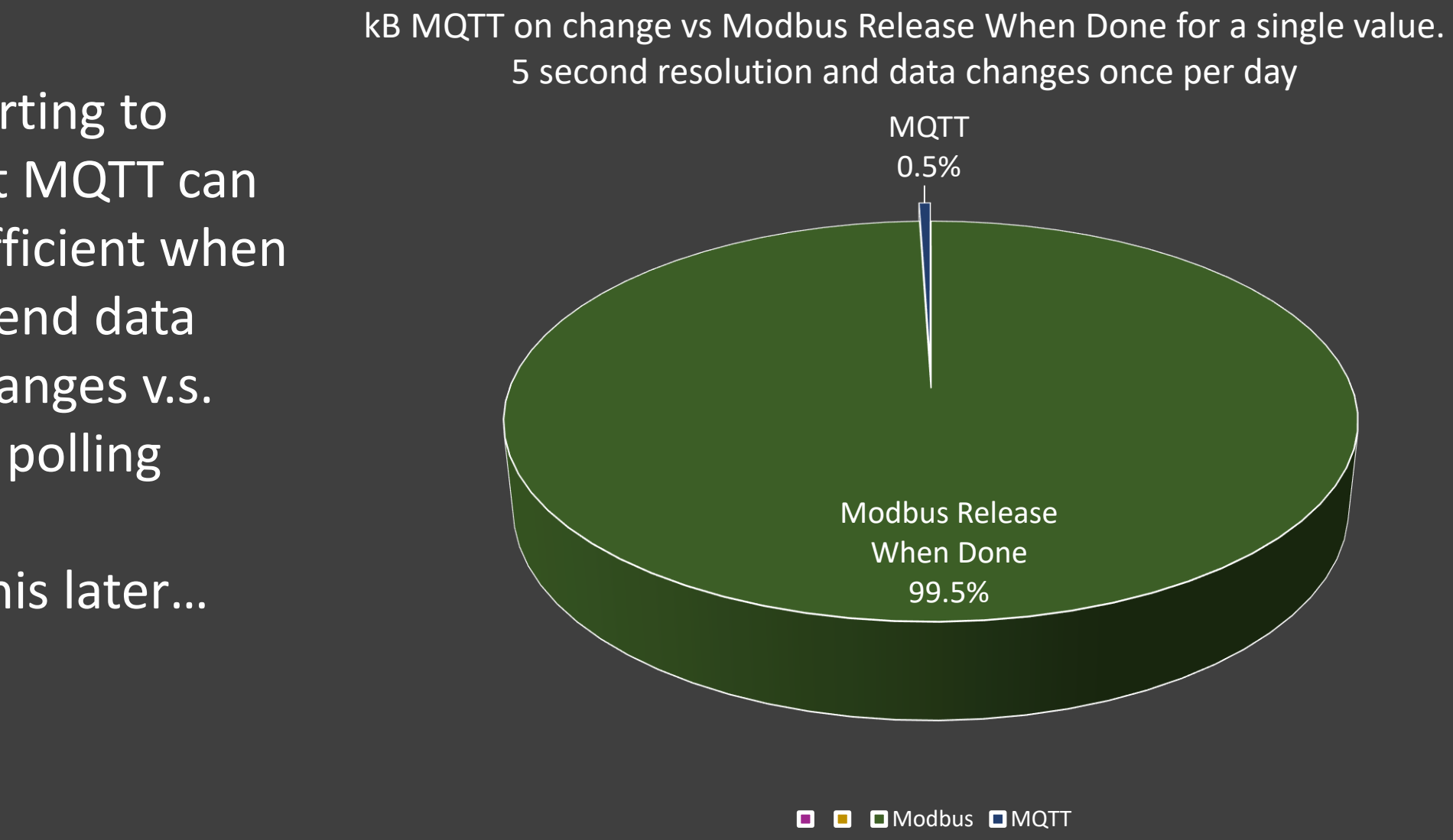
Assuming keep-alive can be set to 5 minutes, data changes twice per day

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…



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

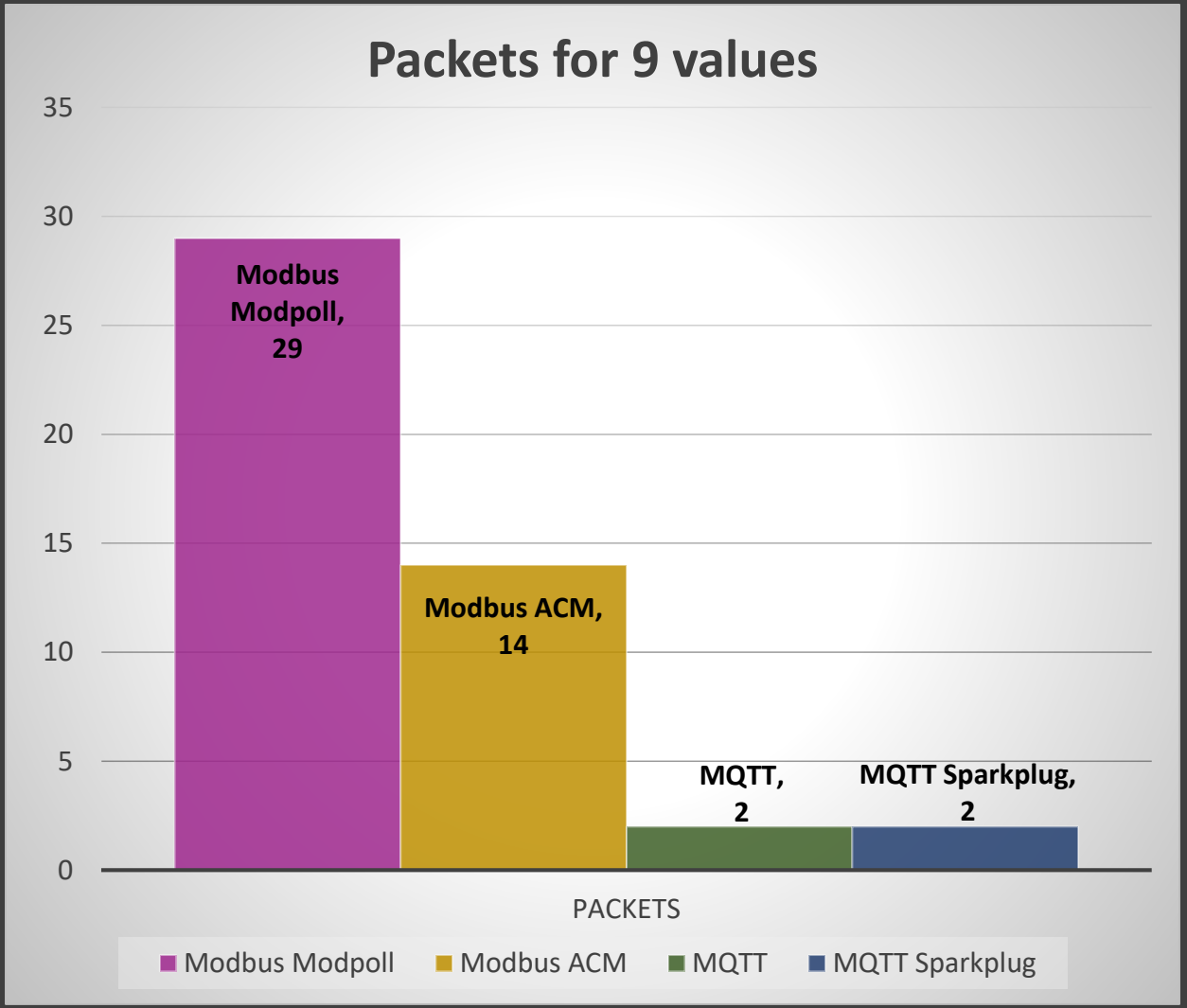
Considerations

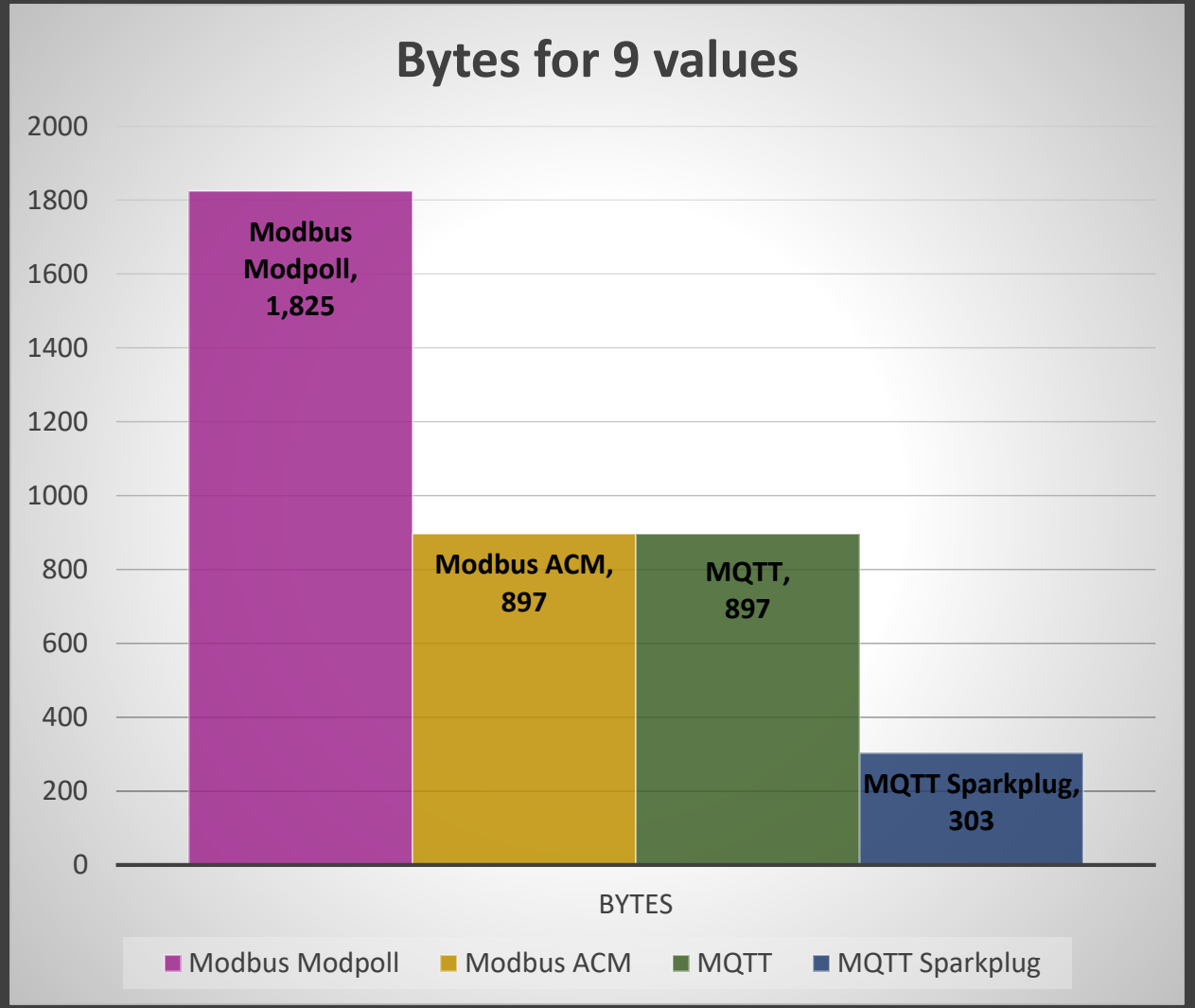
- These values are not in a contiguous Modbus block so Modbus has to poll three different ranges of registers
- Not all drivers for the same protocol will give the same results
- Sparkplug can be more efficient if it is implemented as intended vs sending unnecessary data

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}

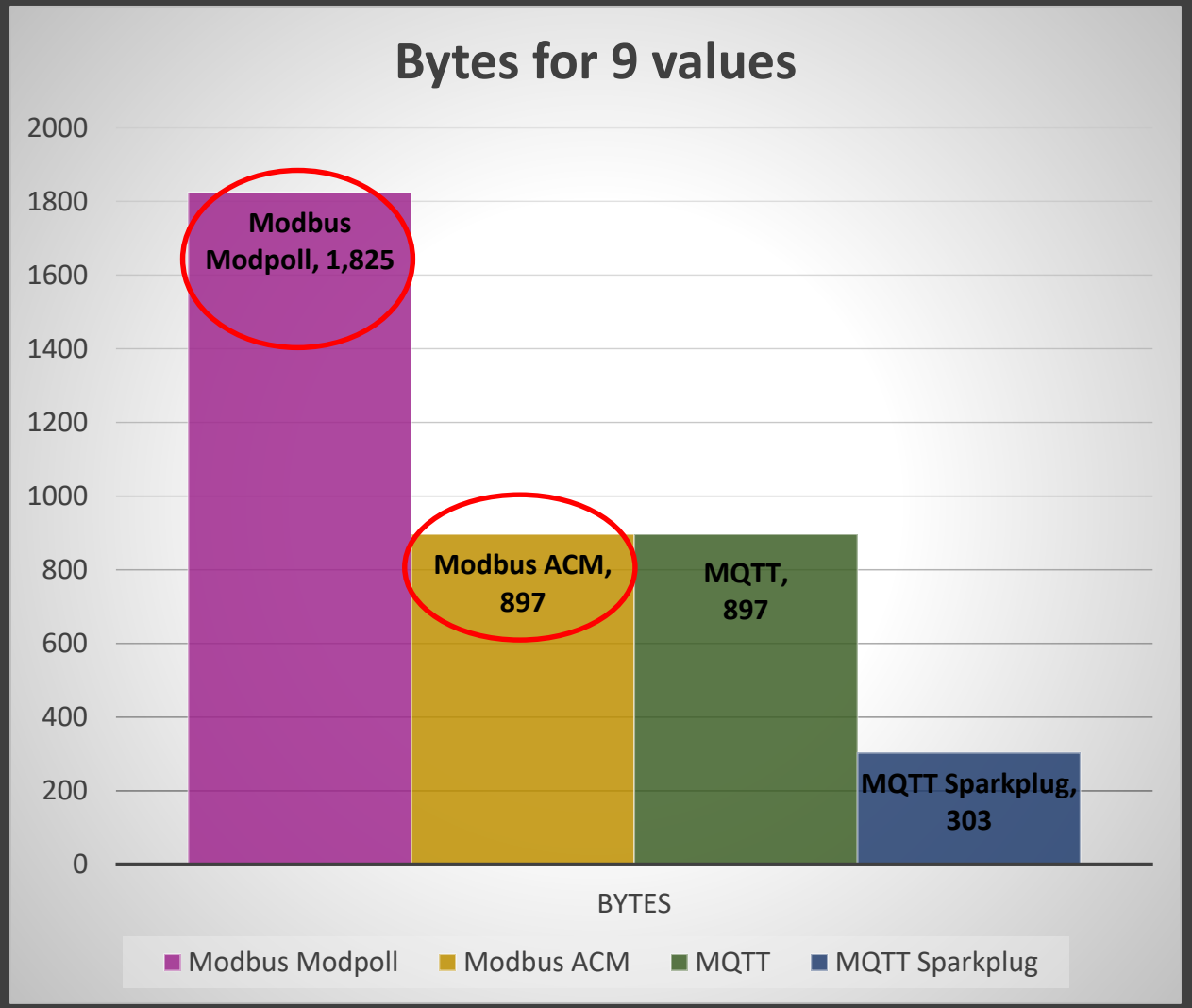
Real Device, 9 values





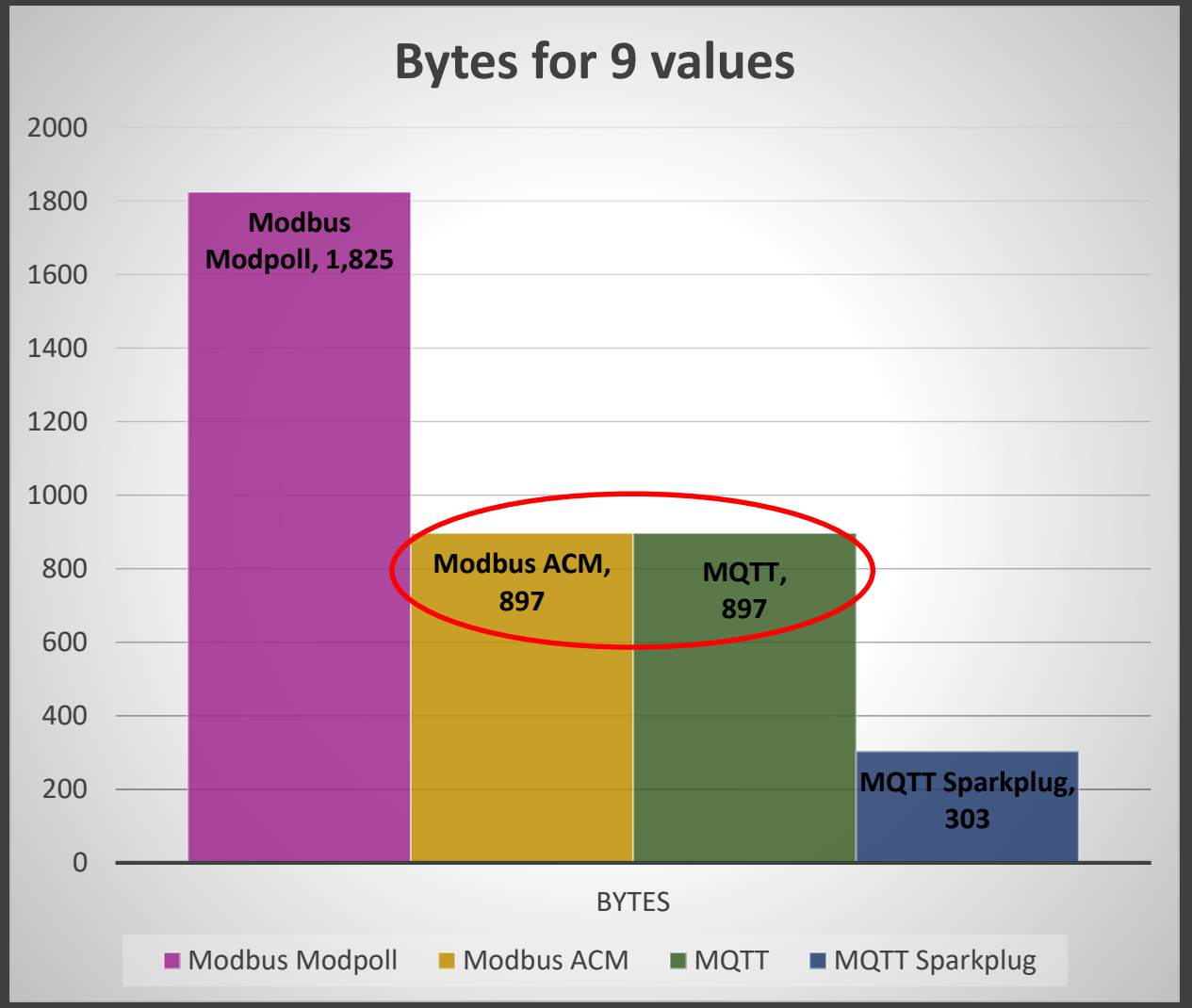
Different Modbus Drivers

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



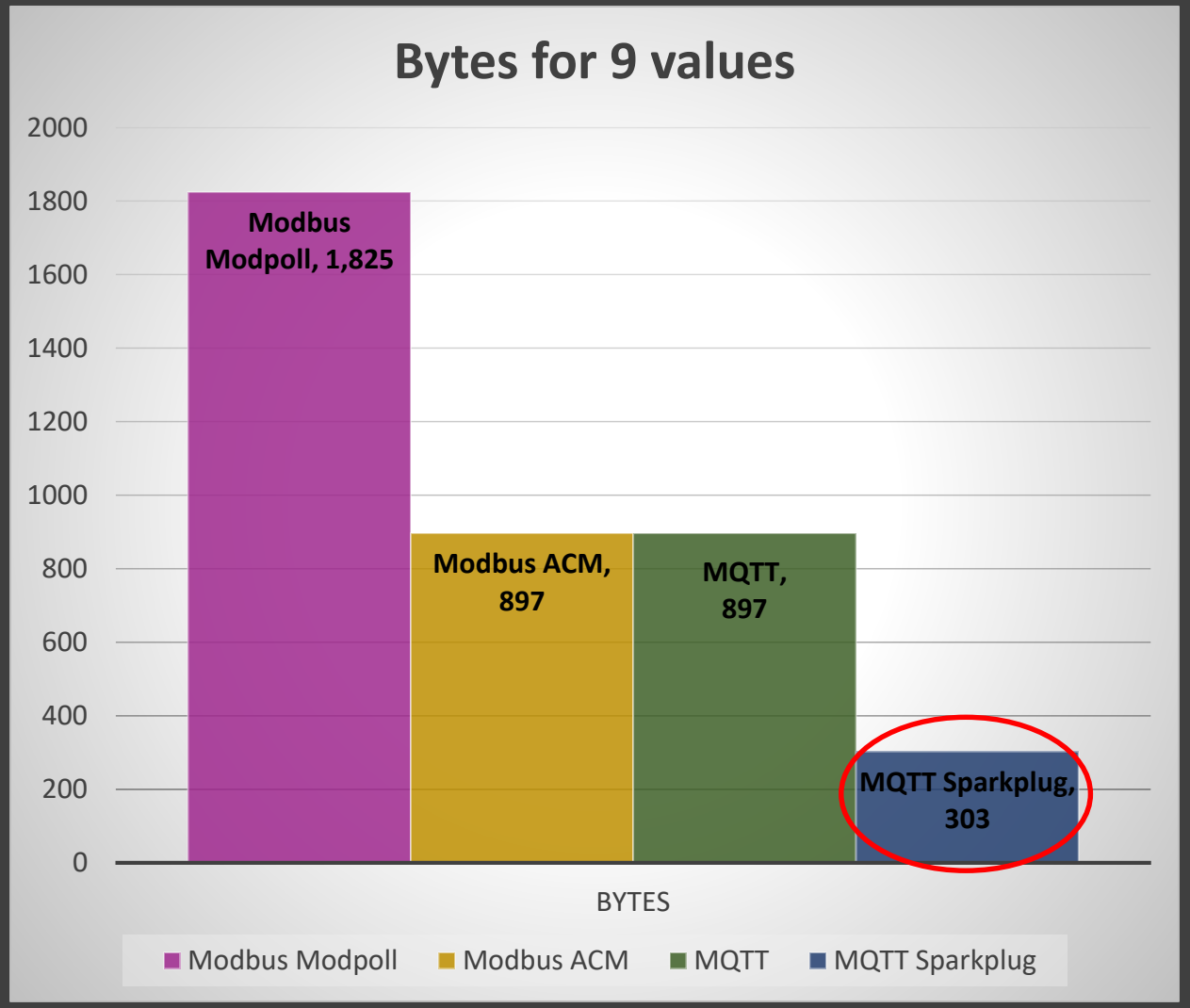
MQTT VS Modbus

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

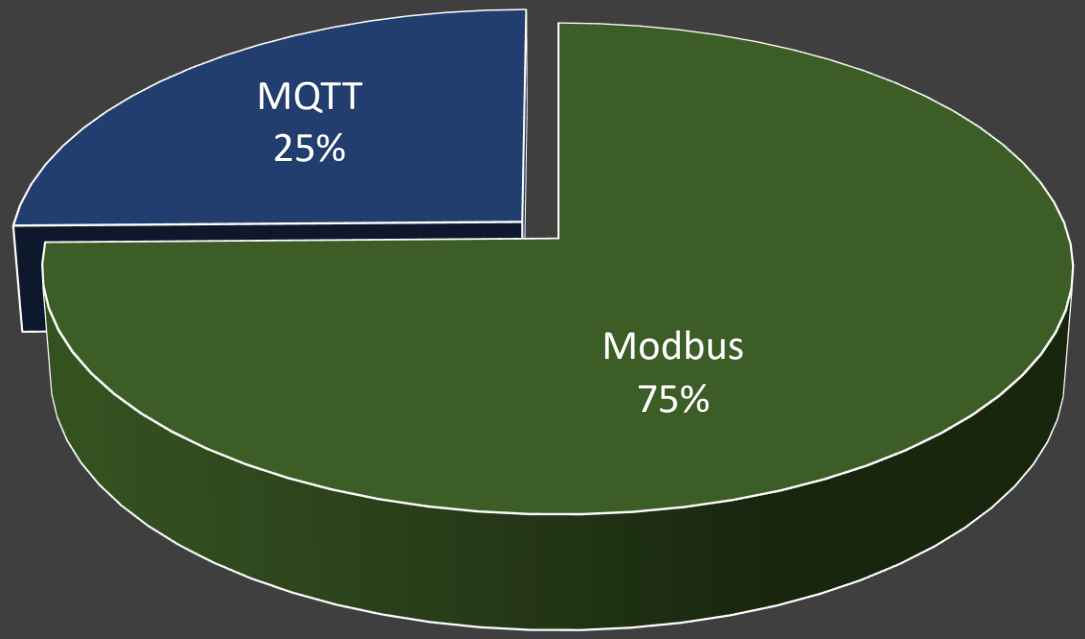


Sparkplug B Compresses Data 3x

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



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

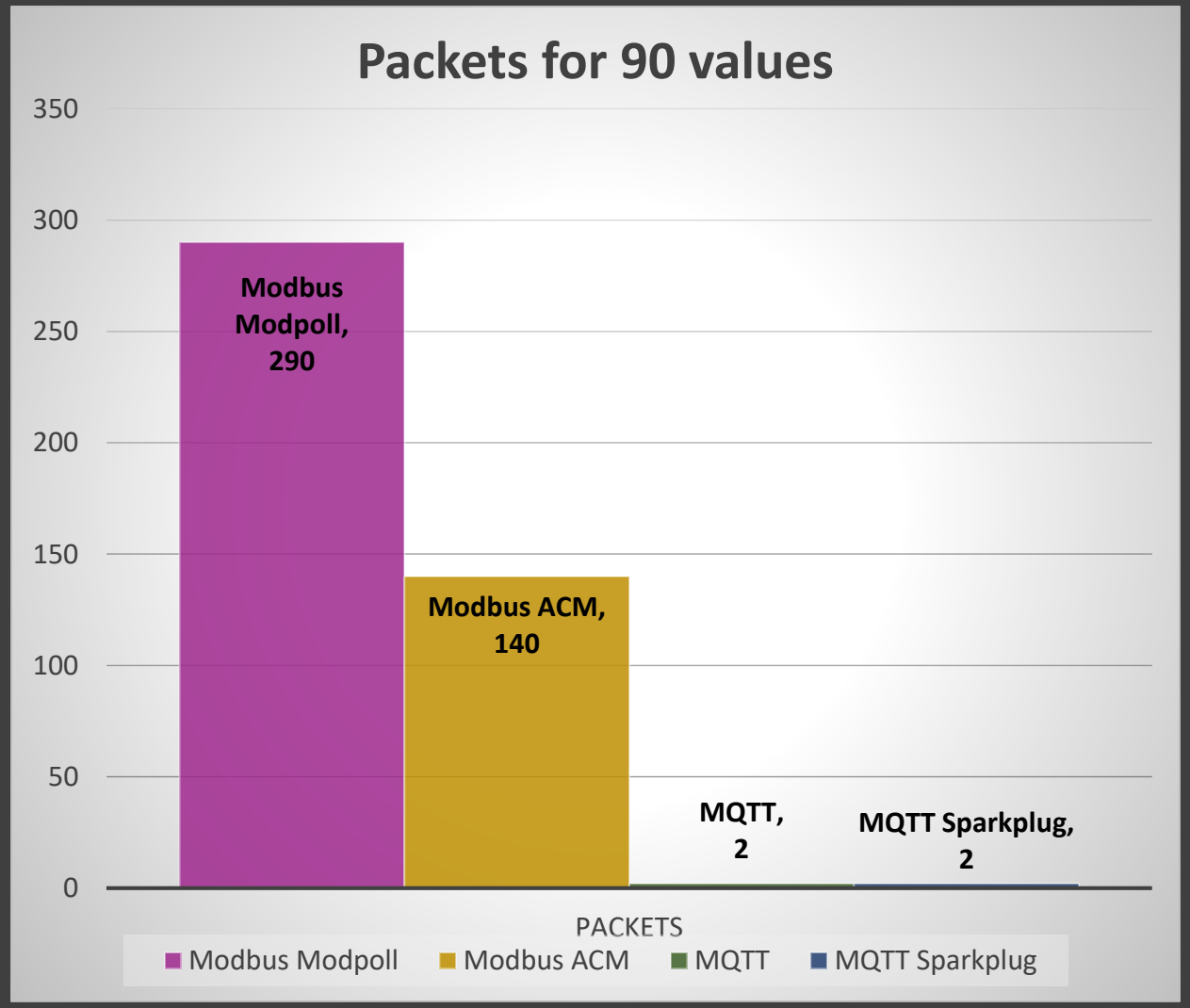
```
{
    "alias":13,
    "timestamp":1520722891000,
    "value":1
}
```

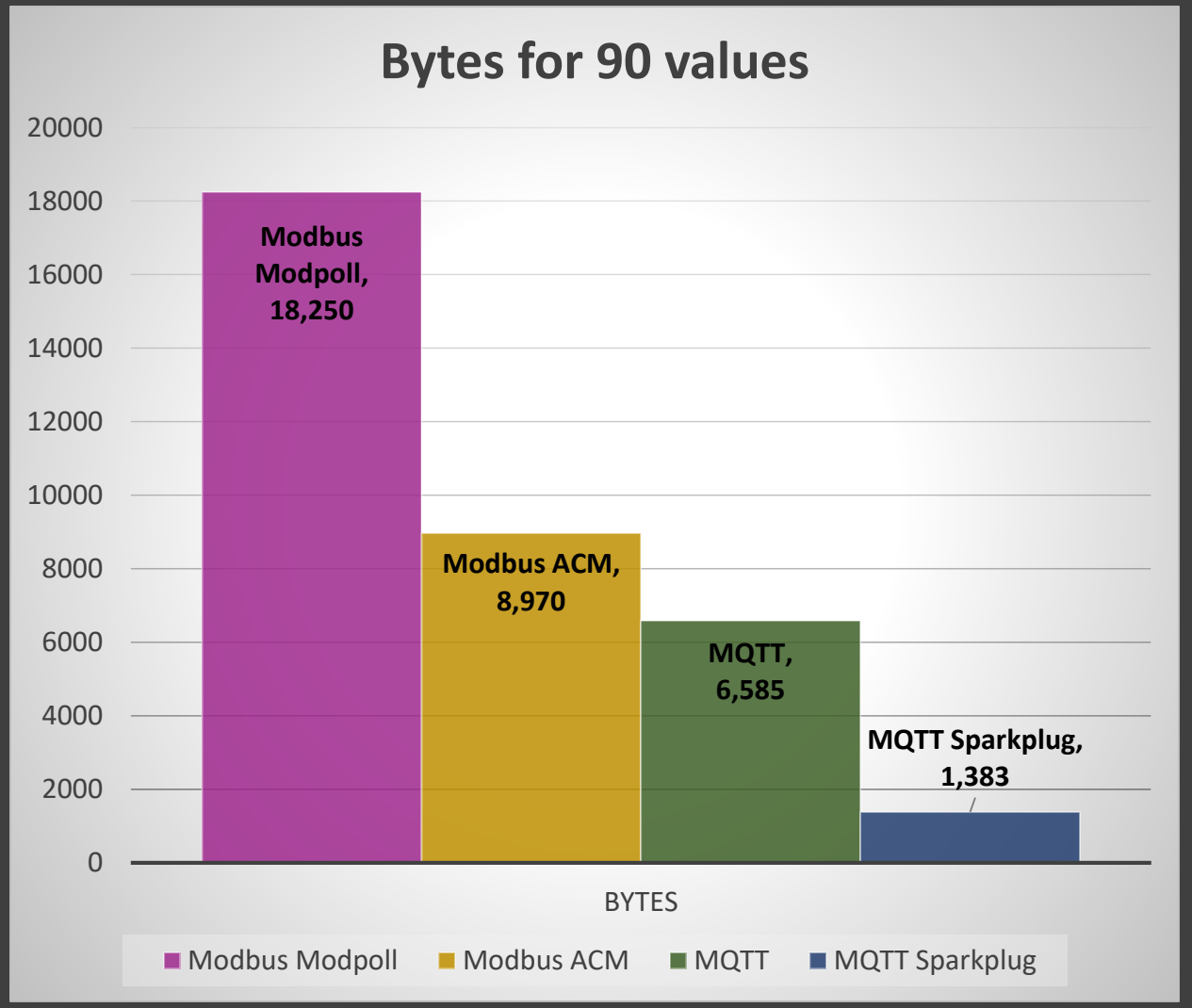
Data Concentration

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

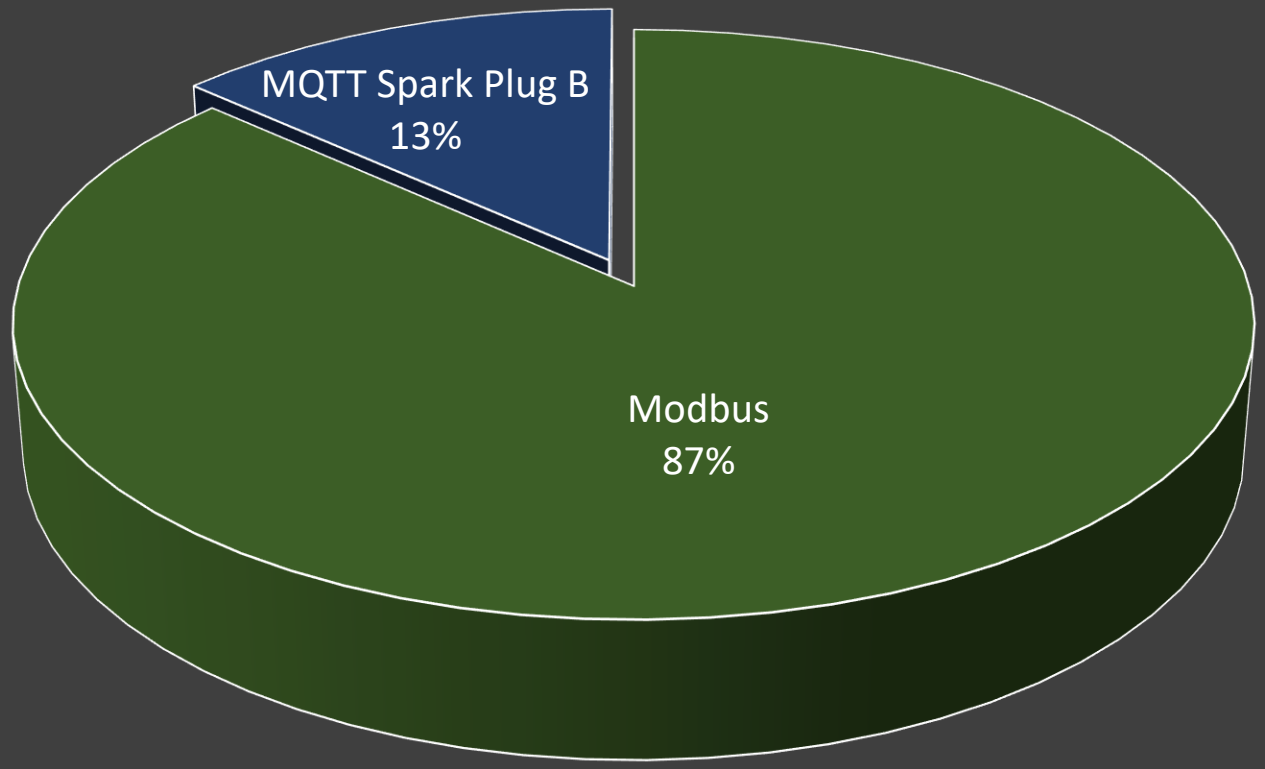
Same Device, 90 values sent every 10 minutes

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





MQTT Sparkplug B x10 buffered vs Modbus Poll Response



	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
20	60	/Valves/PID01/Ramp	0	0	574.2	831.6	1709.4	2706
21	60	/Valves/PID01/Remote SetPoint	0	0	574.2	831.6	1709.4	2706
22	60	/Valves/SSD/Status	0	0	574.2	831.6	1709.4	2706
23	60	/WaterRun01/Water BBL PD	0	0	574.2	831.6	1709.4	2706
24	60	/APPlunger/A STATE	1	9.57	574.2	831.6	1709.4	2706
25	60	/APPlunger/CONSECUTIVE EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
26	60	/APPlunger/Consecutive Non Arrivals	1	9.57	574.2	831.6	1709.4	2706
27	60	/APPlunger/COUNT FAST ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
28	60	/APPlunger/COUNT LATE ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
29	60	/APPlunger/COUNT NO ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
30	60	/APPlunger/COUNT NORMAL ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
31	60	/APPlunger/COUNT SLOW ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
32	60	/APPlunger/EARLY ARRIVALS	1	9.57	574.2	831.6	1709.4	2706
33	60	/APPlunger/History/ArrvlTime1	1	9.57	574.2	831.6	1709.4	2706
34	60	/APPlunger/History/ArrvlTime10	1	9.57	574.2	831.6	1709.4	2706

Facilities Tags

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

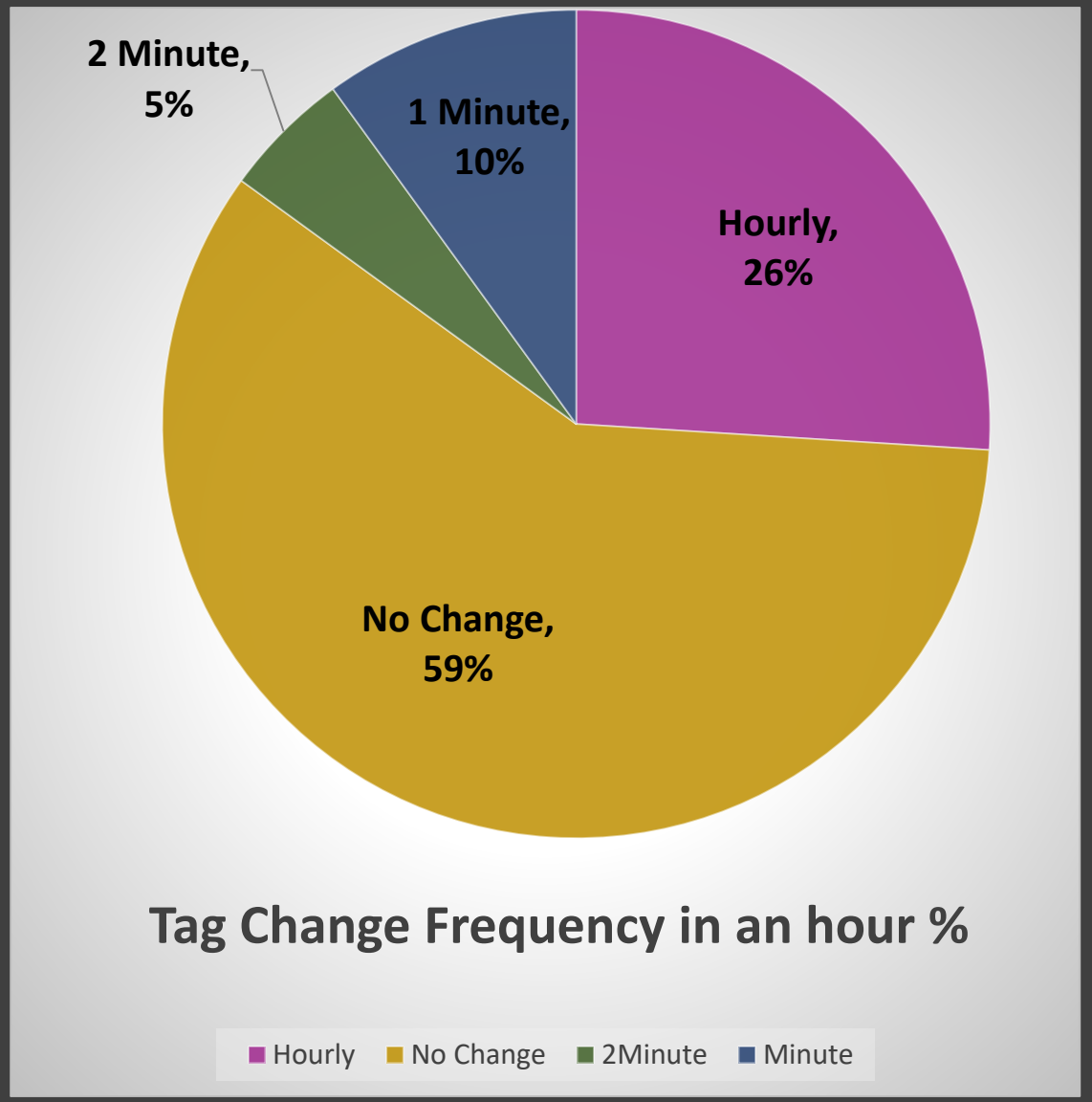
36 analog values would change each minute

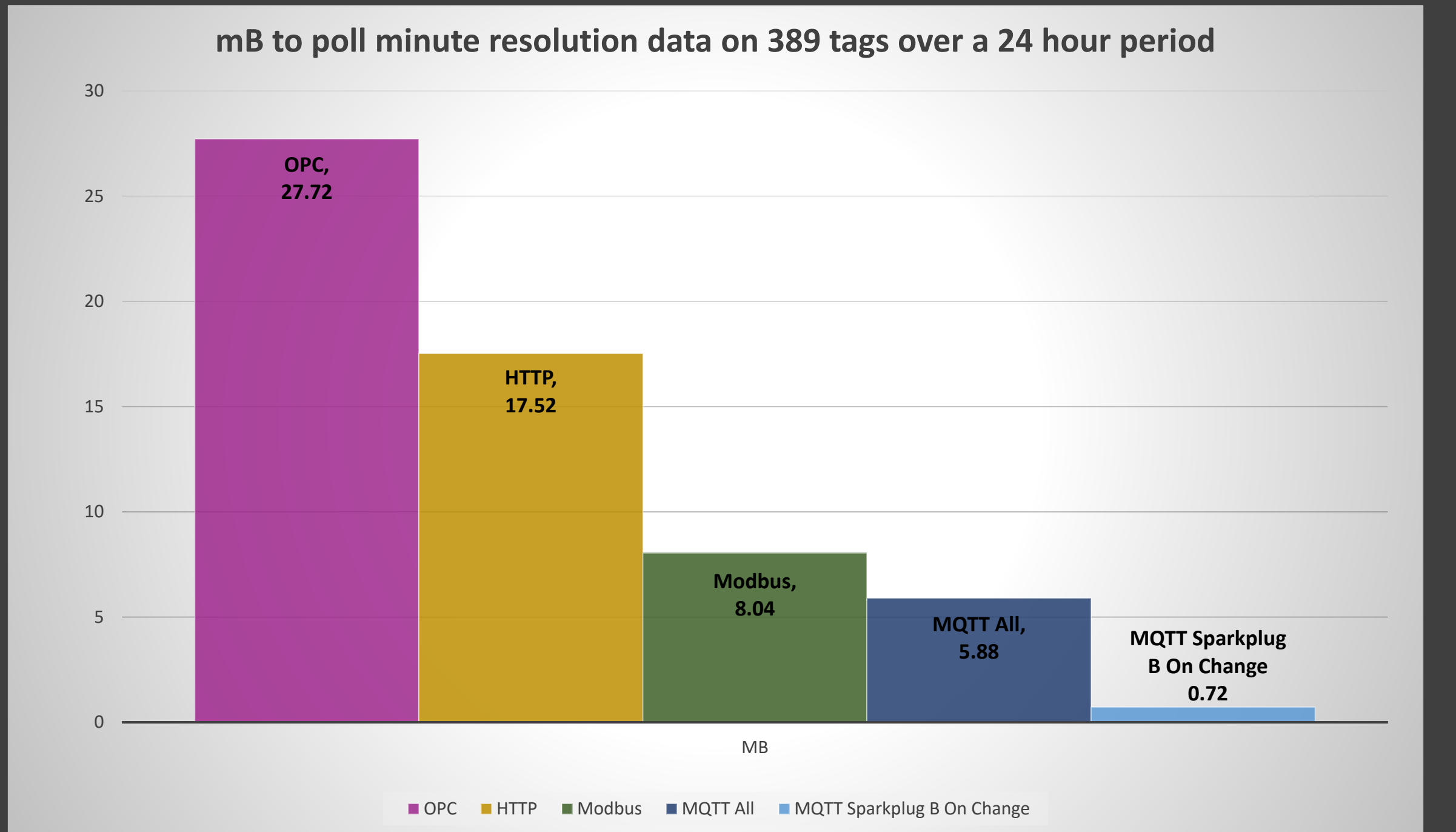
	B	C	D	N	O	P	Q	R
1	item	Tag Path	Changes per hour	MQTT	MQTT All	Modbus	HTTP	OPC
211	60	/Valves/PID01/DeadBand	0	0	574.2	831.6	1709.4	2706
212	60	/Valves/PID01/Derivative	0	0	574.2	831.6	1709.4	2706
213	60	/Valves/PID01/Integral	0	0	574.2	831.6	1709.4	2706
214	60	/Valves/PID01/Local SetPoint	0	0	574.2	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

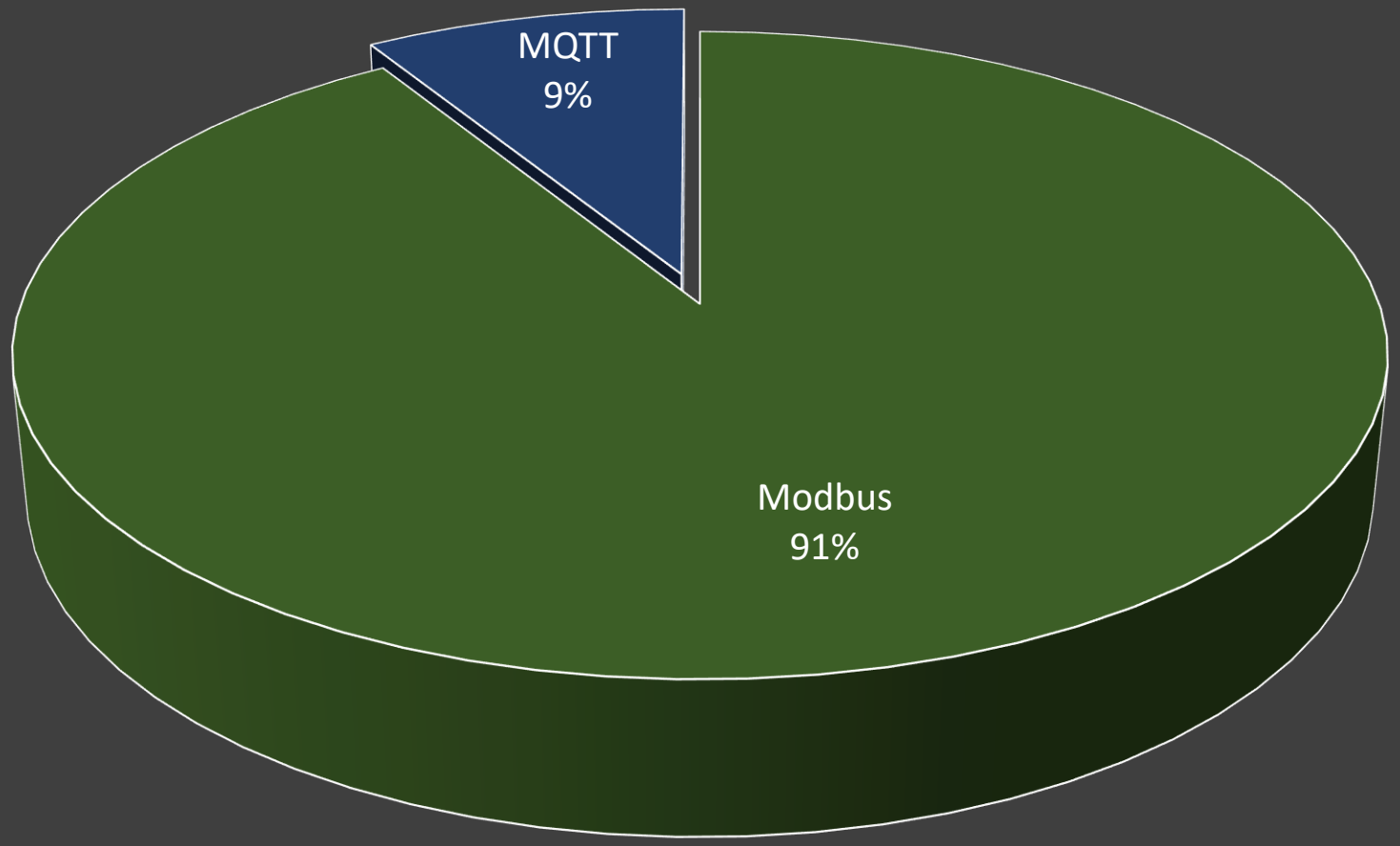
59% of the data doesn’t change even once in a hour

So why poll for it each time…




mB to poll minute resolution data on 389 tags over a 24 hour period
30
25
20
15
10
5
0
OPC, 27.72
HTTP, 17.52
Modbus, 8.04
MQTT All, 5.88
MQTT Sparkplug B On Change 0.72
MB
OPC
HTTP
Modbus
MQTT All
MQTT Sparkplug B On Change

mB MQTT Sparkplug B On Change vs Modbus Poll Response



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 bandwith 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