



Reflective Servers:

Seamless Offloading of Resource Intensive Data Delivery

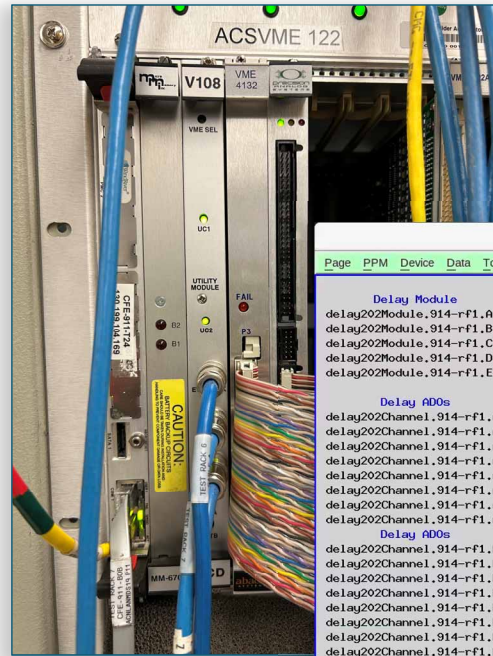
Samuel Clark, Brookhaven National Laboratory

October 12th, 2023



Purpose

- Data updates from some 77,000 devices with >1M control points
- Many devices with very limited resources... how to do so efficiently?
- Proxy clients to devices through Reflective Servers
 - Offload asynchronous data delivery workload
 - Robust connection management
 - Easy deployment & configuration
 - Seamless integration into controls ecosystem



The screenshot shows a software interface with a table of Delay Modules and Delay ADOs. The table has columns for Delay Module, V202, ctrlInterrupt, parityErrorCount, and eventLinkErrorCount. Below this, there are sections for Delay ADOs, each with a table of PPMUser Ch#, History, Delay, and Width.

Delay Module	V202	ctrlInterrupt	parityErrorCount	eventLinkErrorCount
delay202Module.914-rf1.A	On	On	2	4
delay202Module.914-rf1.B	On	On	2	2
delay202Module.914-rf1.C	On	On	2	4
delay202Module.914-rf1.D	On	On	2	4
delay202Module.914-rf1.E	On	On	2	4

Delay ADOs	PPMUser Ch#	History	Delay	Width
delay202Channel.914-rf1.A0	U:1 C:1	Yes	100000	1
delay202Channel.914-rf1.A1	U:1 C:2	No	10	100
delay202Channel.914-rf1.A2	U:1 C:3	No	10000	100
delay202Channel.914-rf1.A3	U:1 C:4	No	0	1
delay202Channel.914-rf1.A4	U:1 C:5	No	0	1
delay202Channel.914-rf1.A5	U:1 C:6	No	0	1
delay202Channel.914-rf1.A6	U:1 C:7	No	0	1
delay202Channel.914-rf1.A7	U:1 C:8	Yes	1	100000

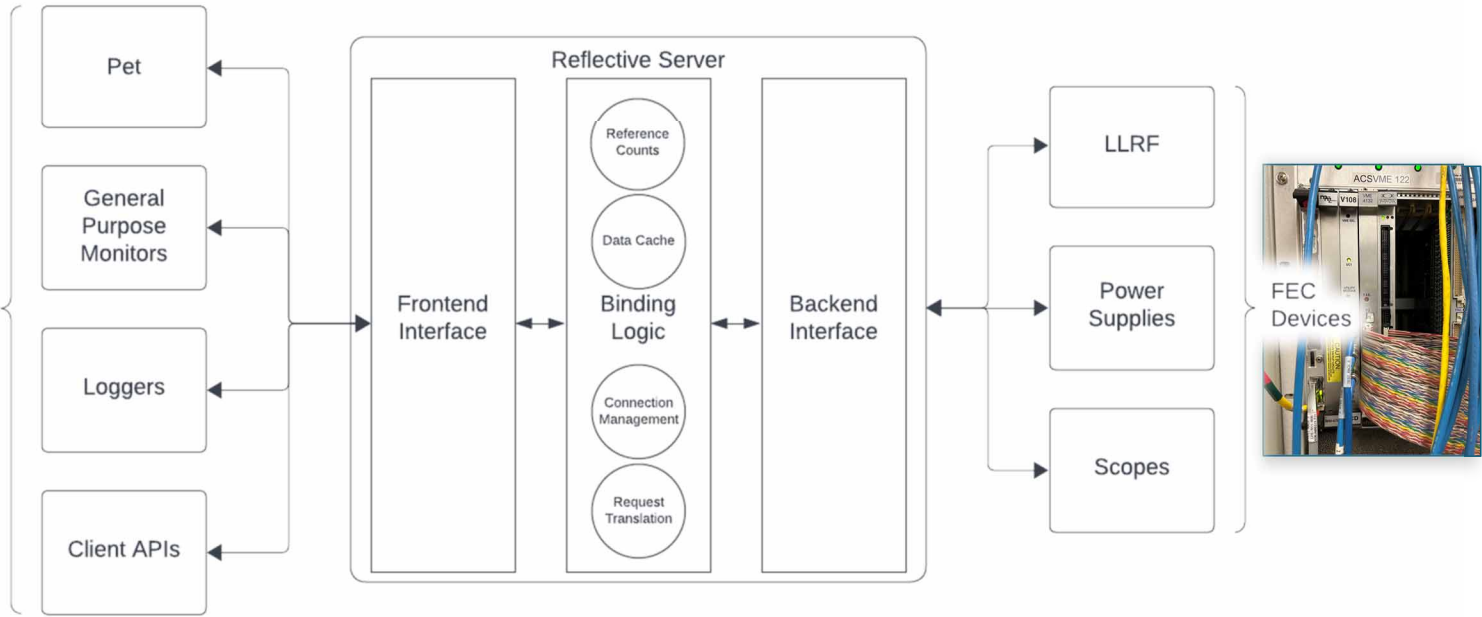
Delay ADOs	PPMUser Ch#	History	Delay	Width
delay202Channel.914-rf1.B0	U:1 C:1	Yes	1	10000
delay202Channel.914-rf1.B1	U:1 C:2	No	0	1
delay202Channel.914-rf1.B2	U:1 C:3	No	0	1
delay202Channel.914-rf1.B3	U:1 C:4	No	0	1
delay202Channel.914-rf1.B4	U:1 C:5	No	0	1
delay202Channel.914-rf1.B5	U:1 C:6	No	0	1
delay202Channel.914-rf1.B6	U:1 C:7	Yes	10	100
delay202Channel.914-rf1.B7	U:1 C:8	Yes	10	100

Delay ADOs	PPMUser Ch#	History	Delay	Width
delay202Channel.914-rf1.C0	U:1 C:1	Yes	2000	1000
delay202Channel.914-rf1.C1	U:1 C:2	Yes	1	1000
delay202Channel.914-rf1.C2	U:1 C:3	No	0	1
delay202Channel.914-rf1.C3	U:1 C:4	No	0	1
delay202Channel.914-rf1.C4	U:1 C:5	No	0	1
delay202Channel.914-rf1.C5	U:1 C:6	No	0	1
delay202Channel.914-rf1.C6	U:1 C:7	No	0	1
delay202Channel.914-rf1.C7	U:1 C:8	No	0	1

Delay ADOs	PPMUser Ch#	History	Delay	Width
delay202Channel.914-rf1.D0	U:1 C:1	No	33000	100
delay202Channel.914-rf1.D1	U:1 C:2	No	0	200000

Client Applications

Client Application	IP	Port	Protocol	Count	Rate	Bytes	Rate
Client A	10.10.10.1	80	TCP	10000	10000	1000000	1000000
Client B	10.10.10.2	80	TCP	5000	5000	500000	500000
Client C	10.10.10.3	80	TCP	2000	2000	200000	200000
Client D	10.10.10.4	80	TCP	1000	1000	100000	100000
Client E	10.10.10.5	80	TCP	500	500	50000	50000



Implementation

- Reflective Server implemented in Python
- Fan-out data updates to 100s of clients
- All protocol features supported

- Efficient connection management
- Modular codebase for protocol flexibility
- Internal optimizations to reduce device traffic

Performance

- Ability to handle much higher client load than FECs directly
- Analysis shows ~2x latency penalty for using Reflection
- Latency measures include...
 - Network round-trip time
 - Data marshalling in Reflective Server
 - Request/Response processing
 - Data fan-out in binding logic

Asynchronous Data Delivery Latency

