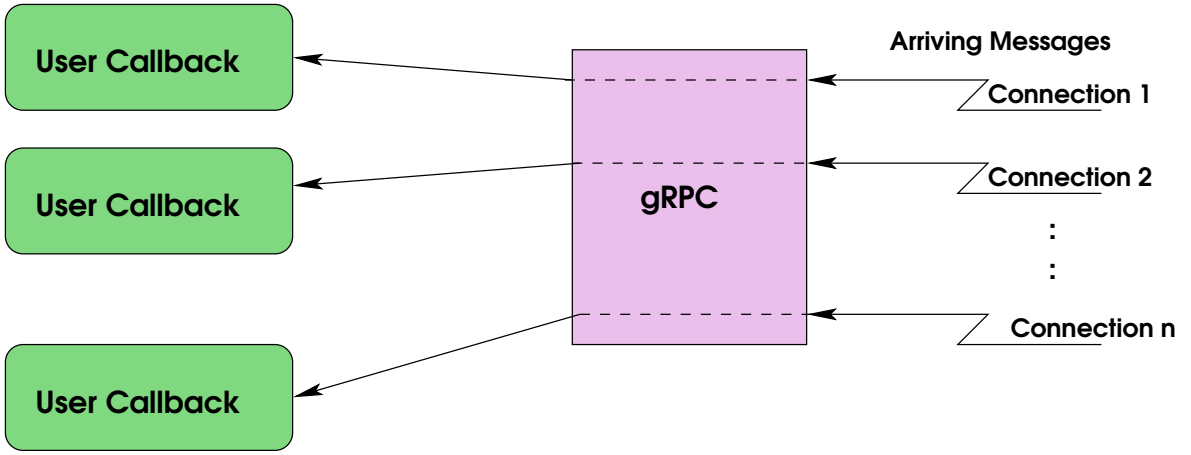
Suitability of gRPC for RMR Communications

24 June 2010

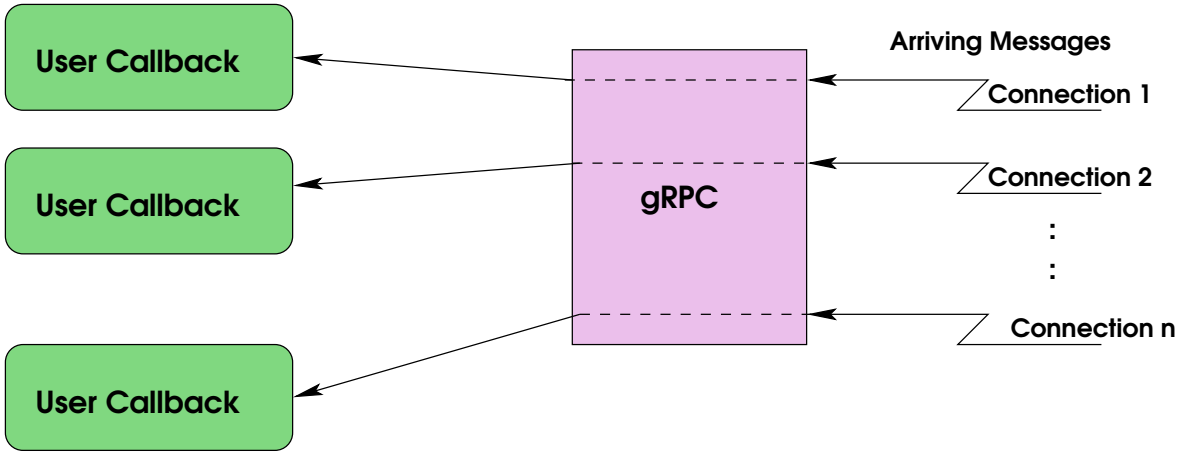
gRPC Functional Overview



Features:

- Callback oriented
- One callback function invoked per connection
- Callbacks can execute concurrently
- Complete messages are passed; no construction at the receiver

gRPC Functional Overview (continued)

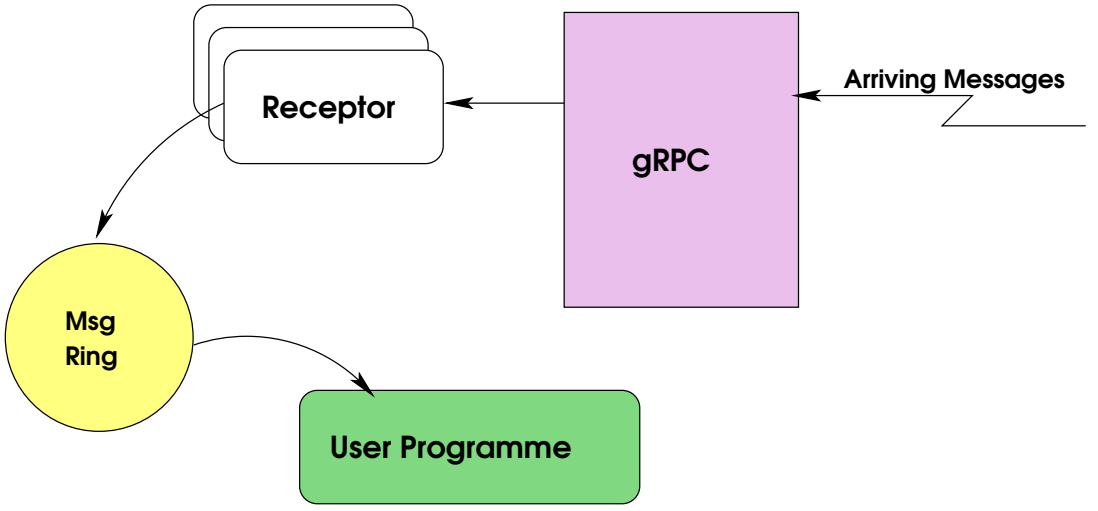


There are 4 *modes* of operation:

- Unary RPC
- Server Streaming
- Client Streaming
- Bidirectional Streaming

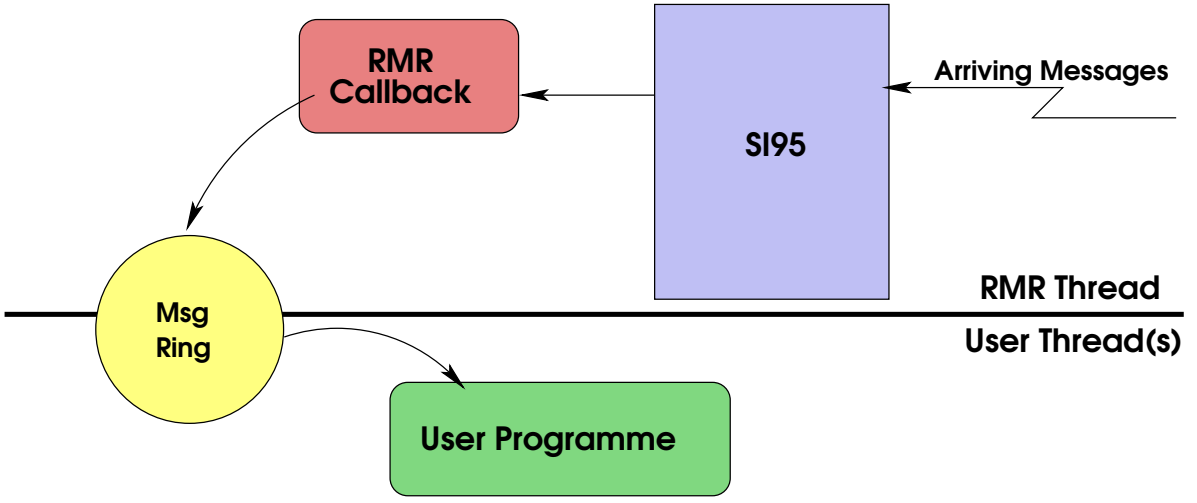
Only **bidirectional streaming** will work.

An RMR-like Setup



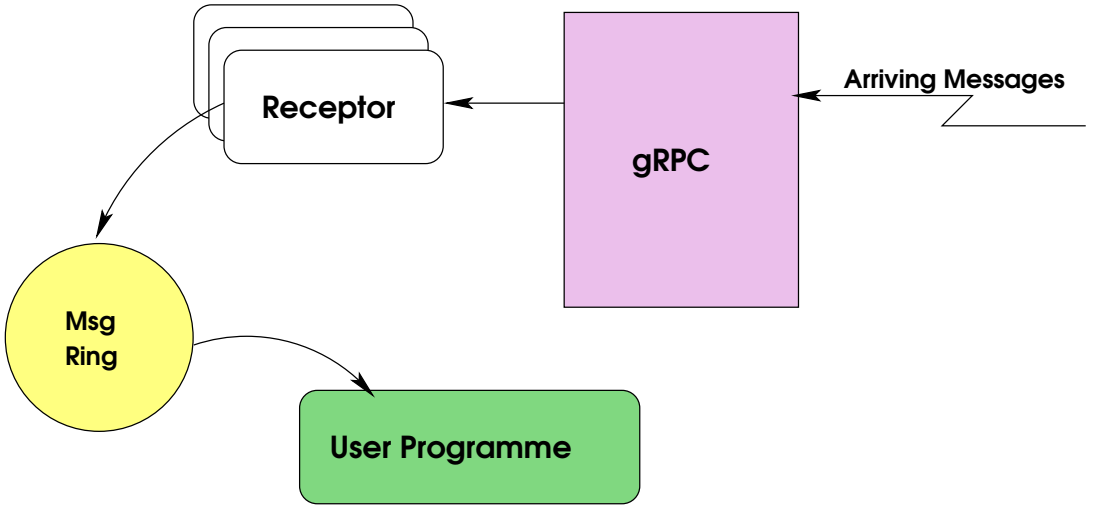
- One service/one callback type: Receptor (white)
- Concurrently executing Receptors; one for each connection
- Single stream for user programme to manage
- Extra buffer copy needed to pass message via ring

RMR/SI95 Comparison



- Single RMR thread; one callback (red)
- Extra copy **NOT** needed to pass message via ring
- Callback must reconstruct large messages

Measurement Point



- Rate and latency measured in the Receptor
- Initial experiments with and without ring insertion
- Single connection/single Receptor

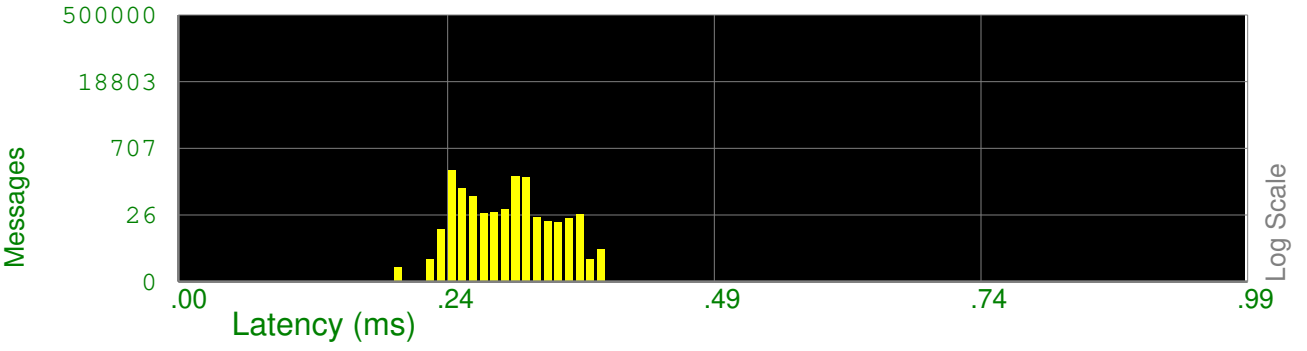
Throughput Results

Transport	Messages	Elapsed	Rate
gRPC	3,000,000	85.4 sec	35K msg/sec
NNG	1,000,000	19.3 sec	52K msg/sec [10]
RMR/NNG	1,000,000	missing	38K msg/sec [10]
RMR/SI95	10,000,000	43.5 sec	230K msg/sec

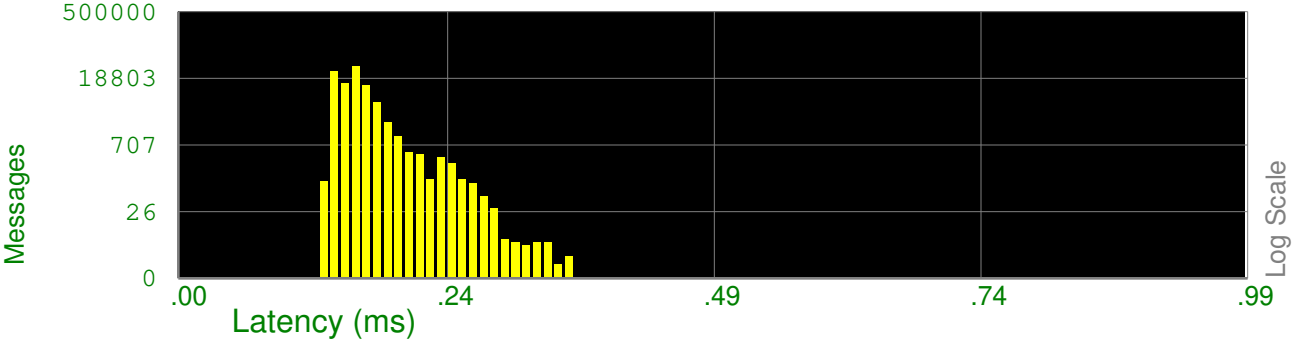
For gRPC:

- Rate for 4K messages averaged about 31K msg/sec
- Rate for 20 byte messages averaged about 41K msg/sec
- MTU varied between 9K and 1500 bytes without affecting the throughput

Latency

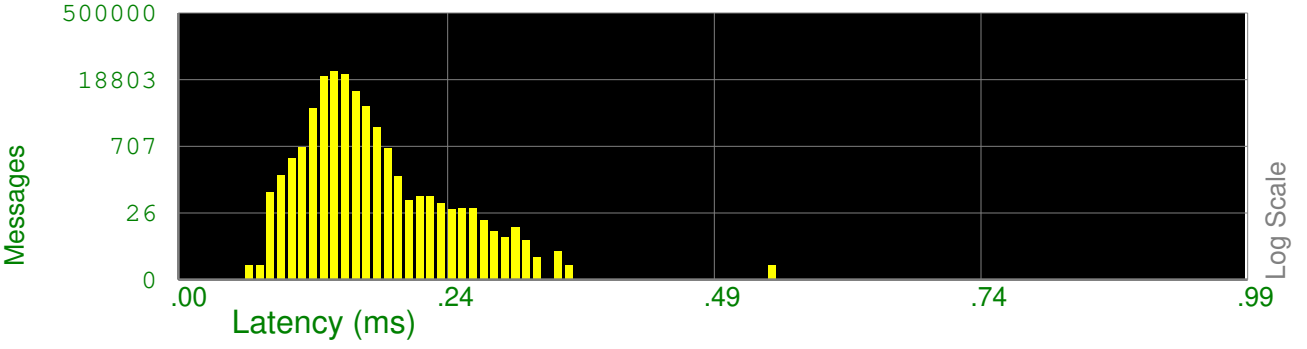


1000 messages at 100 messages/second.

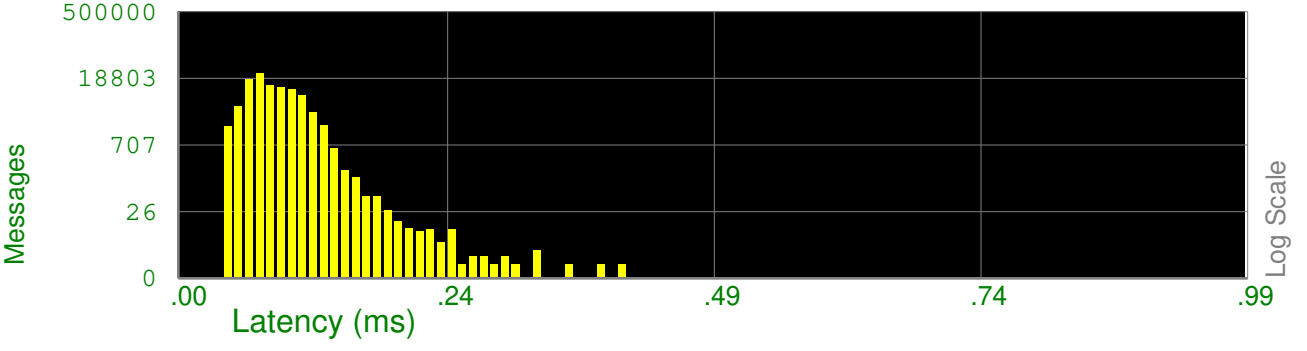


100,000 messages at 850 messages/second.

Latency (continued)

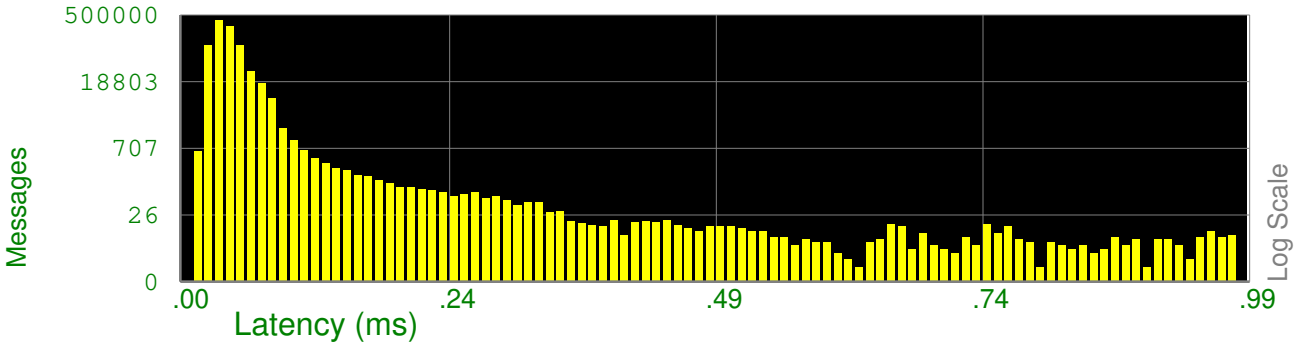


100,000 messages at 4000 messages/second.



100,000 messages at 8500 messages/second.

Latency (continued)

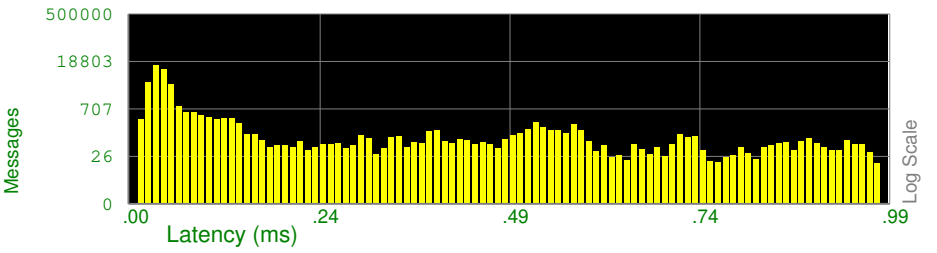


1,000,000 messages at 39K messages/second.

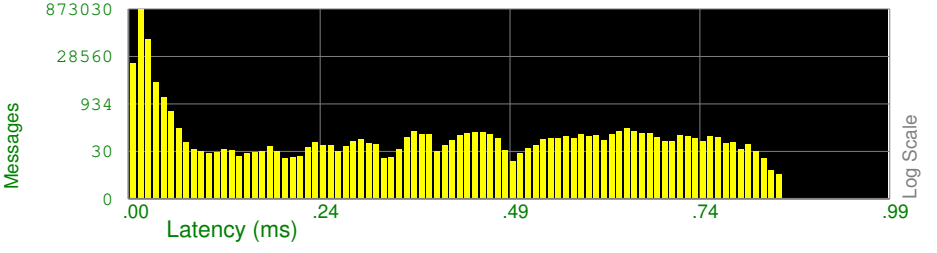
Minimum:	0.03 ms
Maximum:	>1.0 ms
50th percentile:	0.05 ms
95th percentile:	0.09 ms
99th percentile:	>1.0 ms

RMR Latency; A Quick Comparison

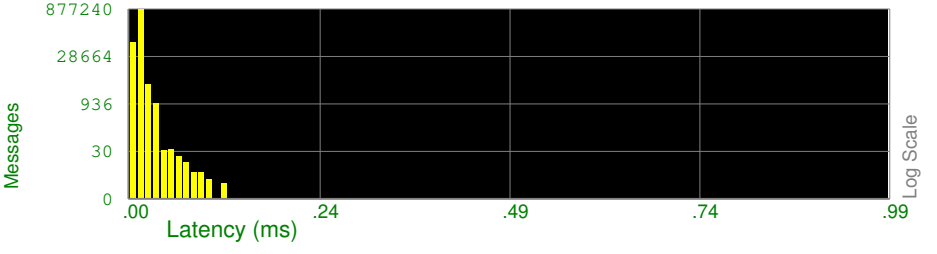
Rcv Rate: ~300K msg/sec
Nagle's: ON
Pinned: Neither
99.5/99.9: >1.0ms



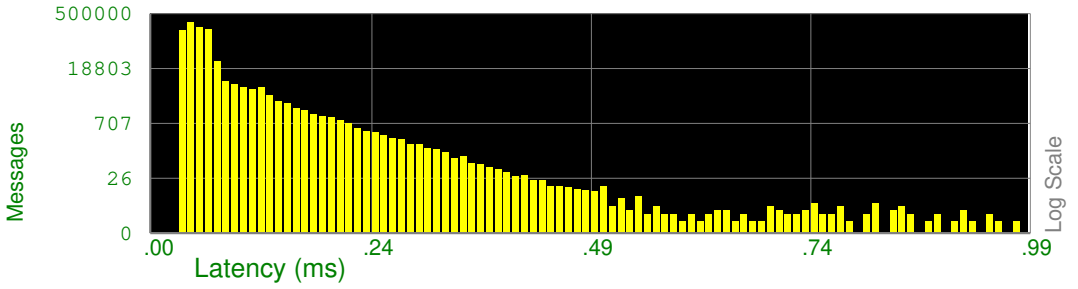
Rcv Rate: ~112K msg/sec
Nagle's: OFF
Pinned: Neither
99.5/99.9: 0.07 / 0.69 ms



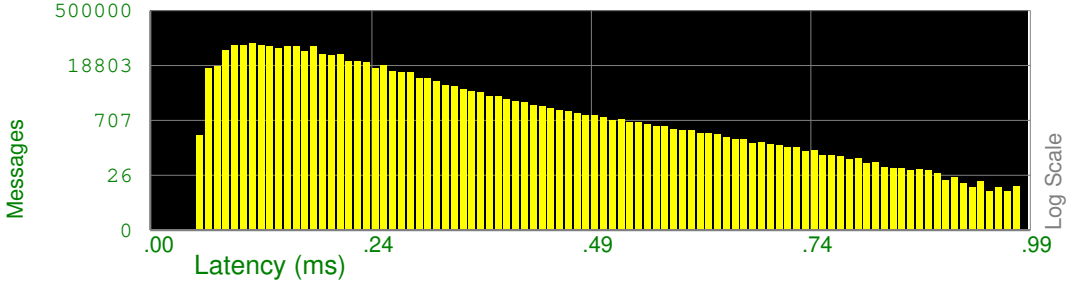
Rcv Rate: ~136K msg/sec
Nagle's: OFF
Pinned: Receiver
99.9: 0.04 ms



Pinning gRPC Processes



Unpinned gRPC Receiver



Pinned gRPC Receiver

- Reduced 99th percentile latency from 0.87ms to 0.52ms
- Did not eliminate the tail

References

[1] **"boost C++ Libraries"**
https://www.boost.org/doc/libs/1_73_0/doc/html/lockfree/reference.html

[2] **"gRPC A high-performance, open source, universal RPC framework"**
https://grpc.io/

[3] **"About gRPC"**
https://grpc.io/about

[4] **"gRPC Frequently Asked Questions - FAQ"**
https://grpc.io/faq

[5] **"gRPC Concepts"**
https://grpc.io/concepts

[6] **"Protocol Buffers C RPC implementation"** source repository
https://github.com/protobuf-c/protobuf-c-rpc

[7] **"Nagle's Algorithm"**
https://en.wikipedia.org/wiki/Nagle%27s_algorithm

[8] Nagle, John; **"Congestion Control in IP/TCP Internetworks"**
RFC 896, January 6, 1984
https://tools.ietf.org/html/rfc896

[9] **"NNG Reference Manual"**
https://nng.nanomsg.org/man/v1.3.0/index.html

[10] **"RMR vs NNG Sending Performance"**
https://wiki.o-ran-sc.org/display/RICP/RMR_nng_perf

[11] **"Google Protocol Buffers"**
https://developers.google.com/protocol-buffers

[12] **"RIC Message Routing"**
https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/index.html#

[13] **"RMR Overview Manual Page"**
https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/rmr.7.html

[14] Trejo, David; **"Nagle's Algorithm"**
http://www.davidromerotrejo.com/2016/09/nagles-algorithm.html?m=1