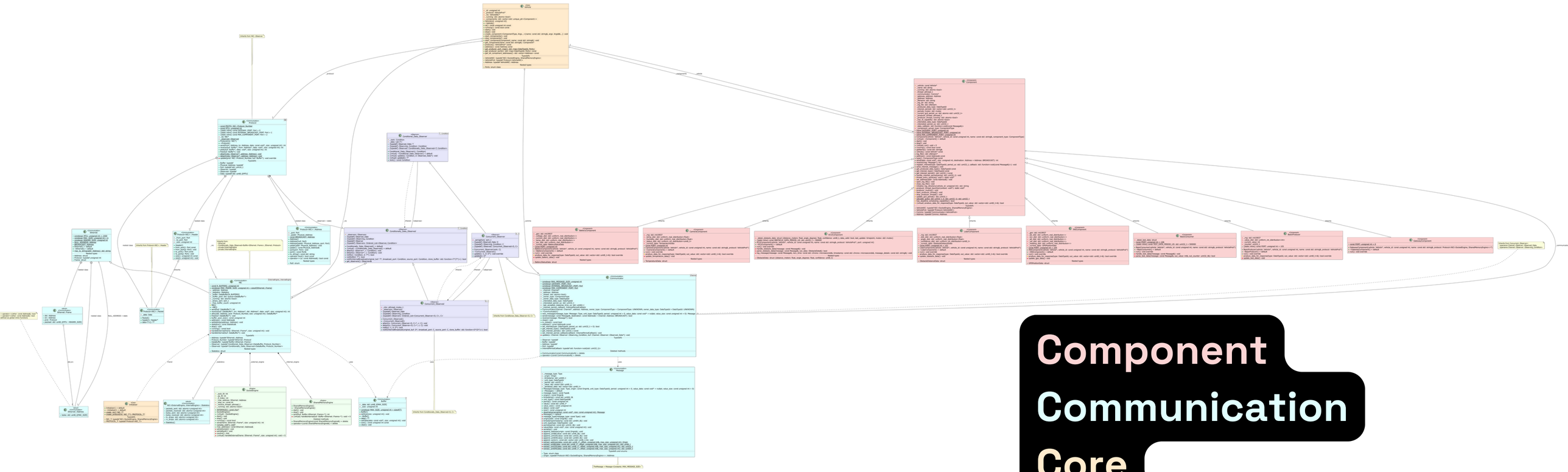
Reliable and secure communication library for critical autonomous systems

CÉSAR AUGUSTO PEREIRA DE SOUZA
ENZO NICOLÁS SPOTORNO BIEGER
JOÃO PEDRO PEREZ RESMER
JOÃO PEDRO SCHMIDT CORDEIRO

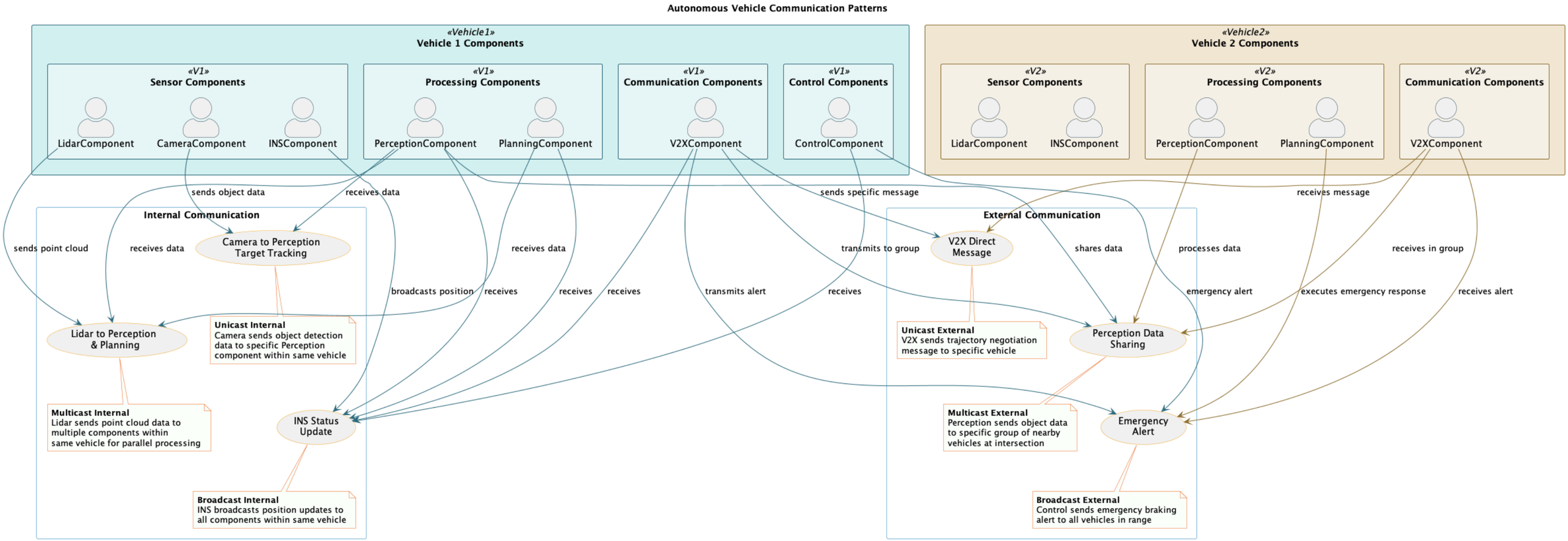
P3 19/05/2025

Class Diagram



Component
Communication
Core
Observer
Engine

Use Case


Autonomous Vehicle Communication Patterns
«Vehicle1»
Vehicle 1 Components
«V1»
Sensor Components
LidarComponent
CameraComponent
INSComponent
«V1»
Processing Components
PerceptionComponent
PlanningComponent
«V1»
Communication Components
V2XComponent
«V1»
Control Components
ControlComponent
«Vehicle2»
Vehicle 2 Components
«V2»
Sensor Components
LidarComponent
INSComponent
«V2»
Processing Components
PerceptionComponent
PlanningComponent
«V2»
Communication Components
V2XComponent
sends object data
receives data
sends specific message
receives message
Internal Communication
External Communication
sends point cloud
receives data
Camera to Perception Target Tracking
receives data
transmits to group
V2X Direct Message
shares data
processes data
receives in group
broadcasts position
receives
receives
receives
transmits alert
receives
emergency alert
executes emergency response
receives alert
Lidar to Perception & Planning
Unicast Internal
Camera sends object detection data to specific Perception component within same vehicle
Unicast External
V2X sends trajectory negotiation message to specific vehicle
Perception Data Sharing
Multicast Internal
Lidar sends point cloud data to multiple components within same vehicle for parallel processing
INS Status Update
Multicast External
Perception sends object data to specific group of nearby vehicles at intersection
Emergency Alert
Broadcast Internal
INS broadcasts position updates to all components within same vehicle
Broadcast External
Control sends emergency braking alert to all vehicles in range

include > C message.h > Message

```
1   #ifndef MESSAGE_H
27  class Message {
31      enum class Type : std::uint8_t {
32          UNKNOWN = 0,
33          INTEREST = 1,
34          RESPONSE = 2
35      };
36
37      typedef Protocol<NIC<SocketEngine, SharedMemoryEngine>>::Address Origin;
38      // static const unsigned int MAC_SIZE;
39
40      /* Methods */
41      ~Message() = default;
42
43      // Getters
44      const Type& message_type() const;
45      const Origin& origin() const;
46      const std::uint64_t& timestamp() const;
47      const DataTypeId& unit_type() const; // Changed return type
48      const unsigned int period() const;
49      const std::uint8_t* value() const; // Returns pointer to internal vector data
50      const unsigned int value_size() const; // Returns size of the value vector
51      const void* data() const; // Serialized data
52      const unsigned int size() const; // Serialized data size
53      static Message deserialize(const void* serialized, const unsigned int size);
54  ⌘L to chat, ⌘K to generate
55      // Public constructor for P3 message types, useful for Communicator and tests
56      Message(Type message_type, const Origin& origin, DataTypeId unit_type, unsigned int period =
57
58  private:
59      friend class Communicator<Protocol<NIC<SocketEngine, SharedMemoryEngine>>>;
60
61      // Private default constructor, used by deserialize
62      Message() = default;
63
64
65      /* Setters
66          (idea is that only Message::deserealize method can set message attributes,
67          i.e., once a message is created, it will be constant through the whole execution) */
68      void message_type(const Type message_type);
69      void origin(const Origin& addr);
70      void timestamp(const std::uint64_t& timestamp);
71      void unit_type(DataTypeId type); // Changed parameter type
72      void period(const std::uint32_t& period);
73      void value(const void* data, const unsigned int size);
74      void serialize();
```

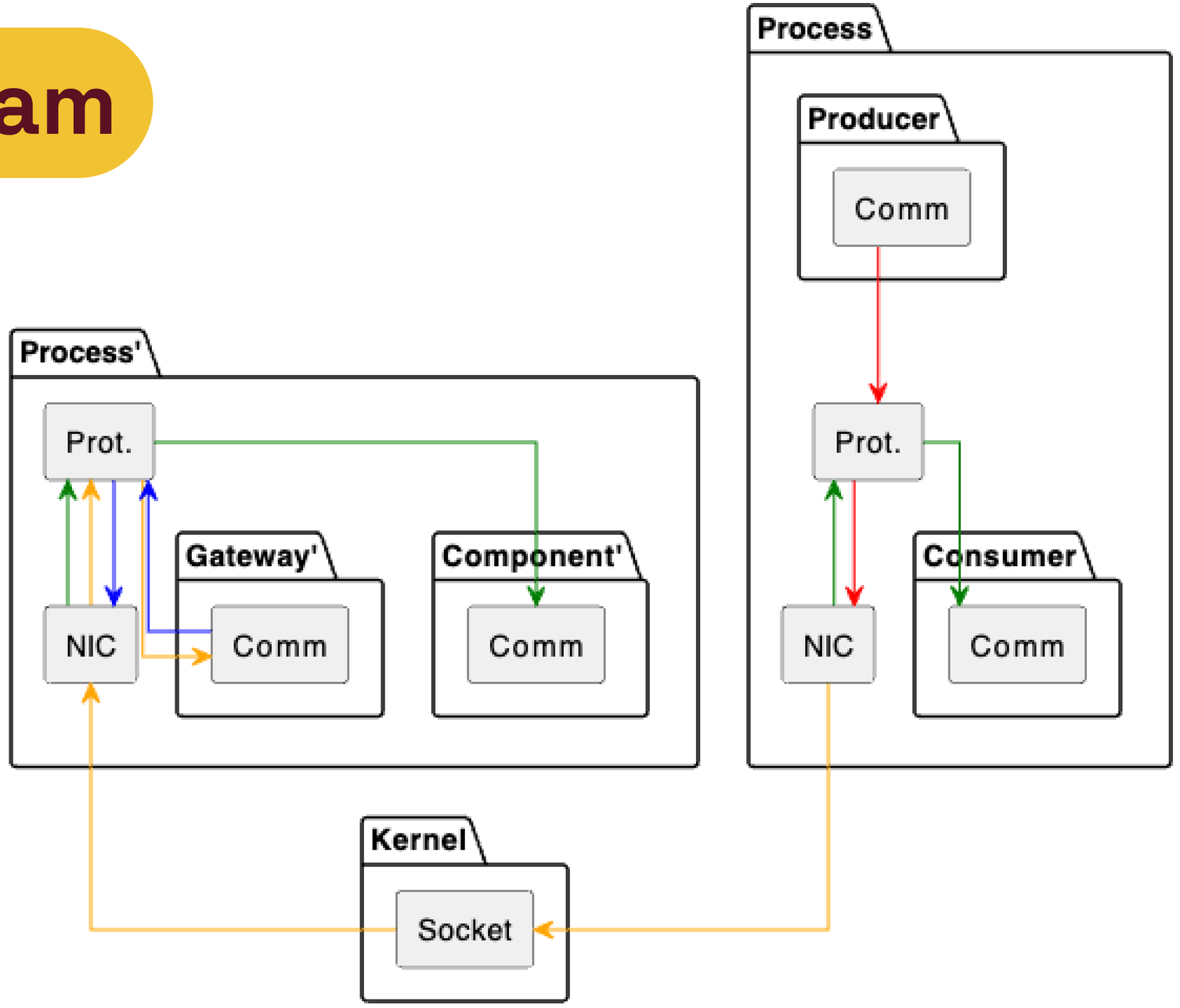
Messsage

include > C teds.h > operator==(const DataTypeId &, const int)

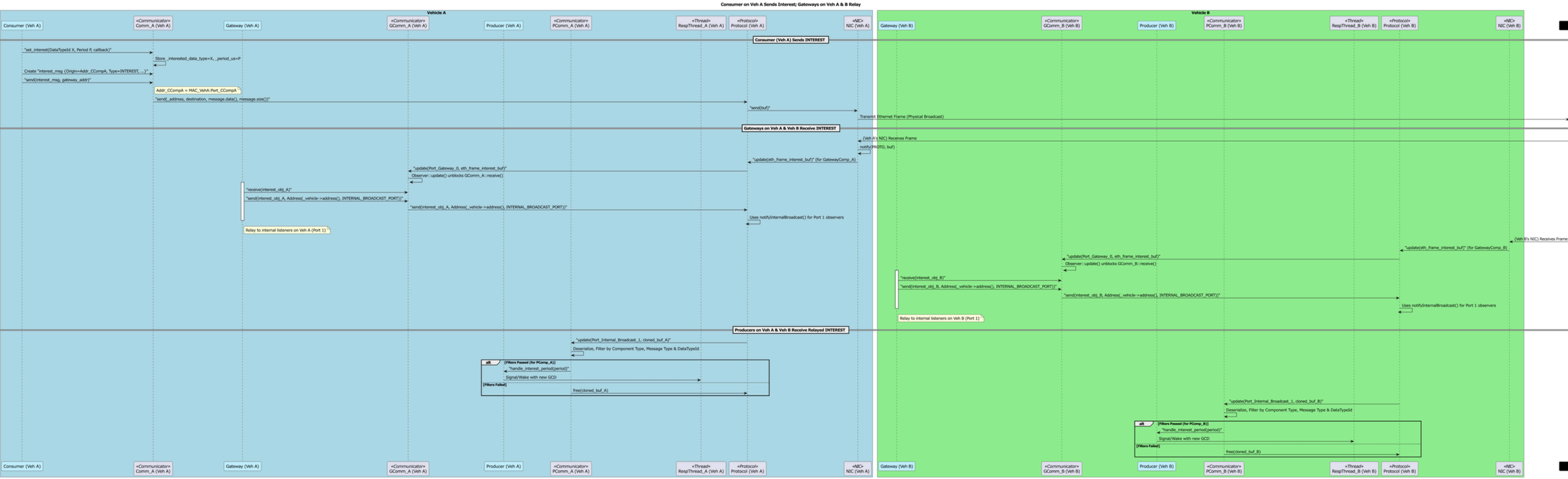
You, last week | 2 authors (enzoniko and one other)

```
1   #ifndef TEDS_H
2   #define TEDS_H
3
4   #include <cstdint>
5
6   enum class DataTypeId : std::uint32_t {
7       UNKNOWN = 0,
8       VEHICLE_SPEED,
9       ENGINE_RPM,
10      OBSTACLE_DISTANCE,
11      // Add other domain-specific data types here
12      SYSTEM_INTERNAL_REG_PRODUCER, // For Gateway registration
13      // Example additional types
14      TEMPERATURE_SENSOR,
15      GPS_POSITION,
16      CUSTOM_SENSOR_DATA_A,
17      CUSTOM_SENSOR_DATA_B
18  };
19
```

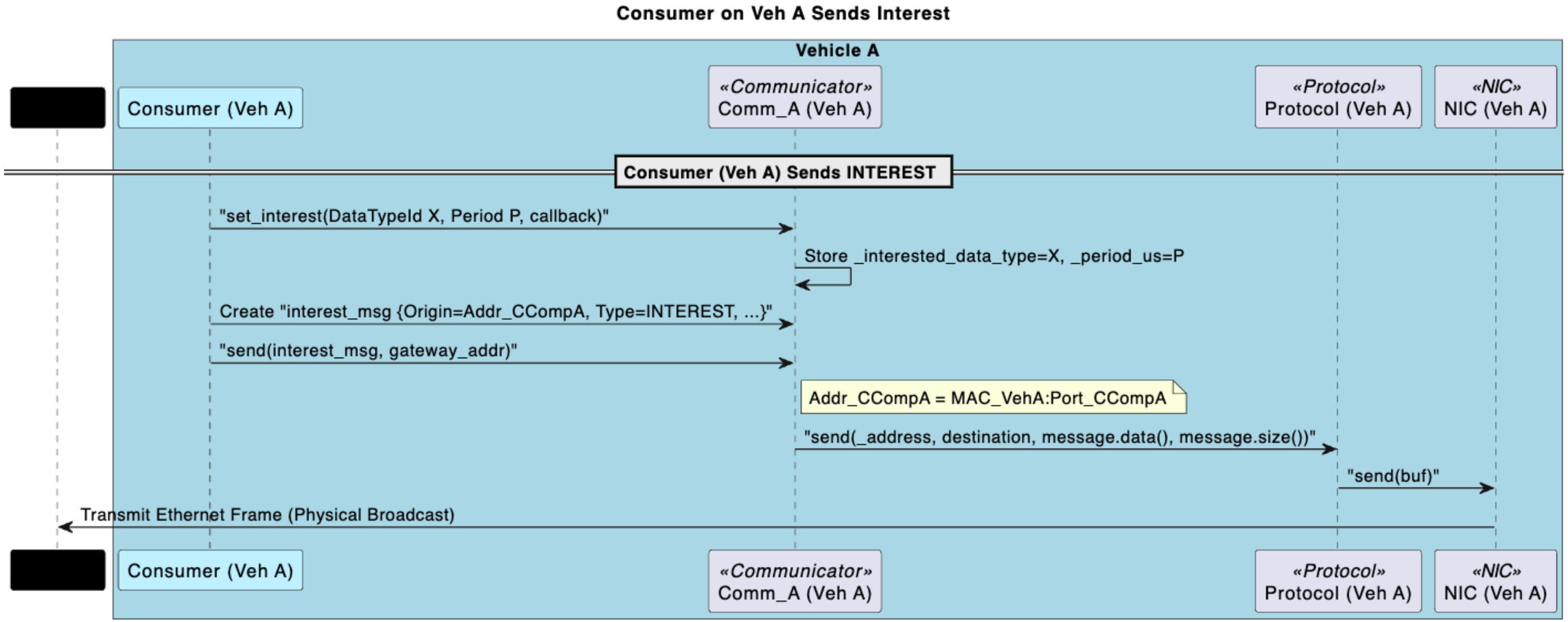
Process Diagram



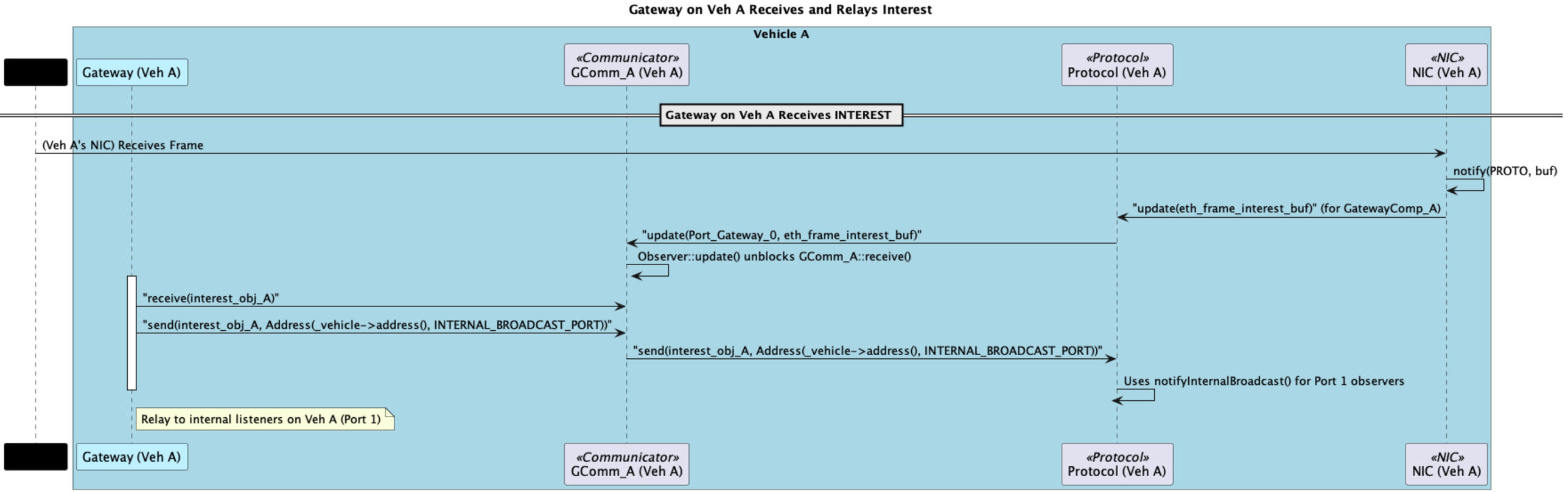
Sequence Diagram 1



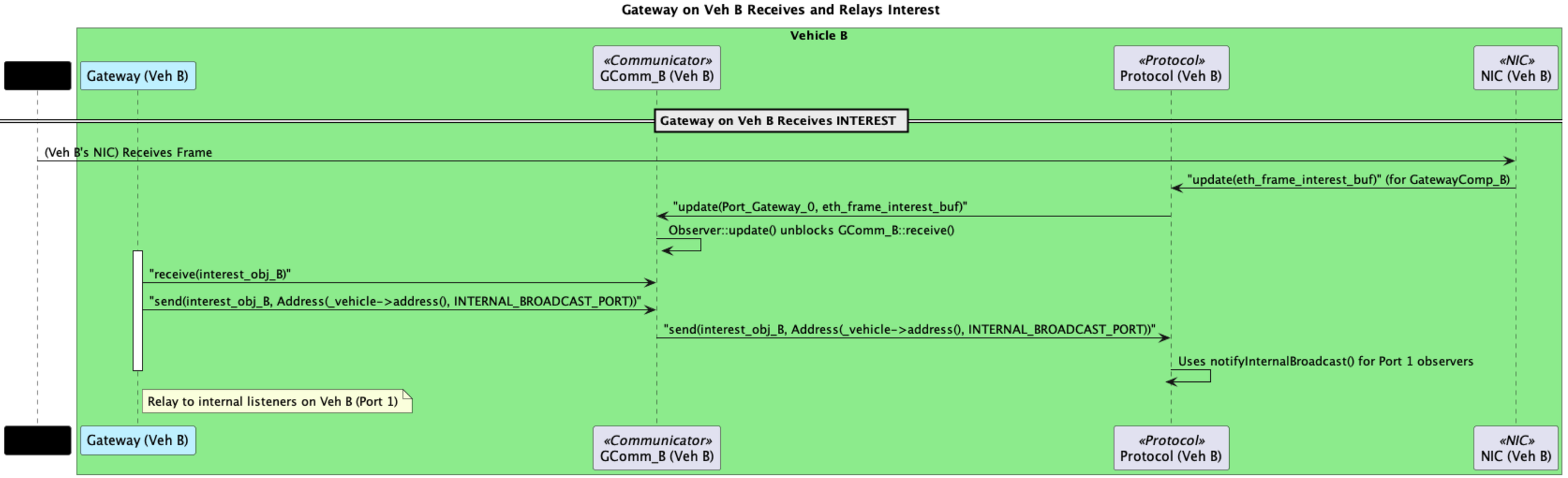
Sequence Diagram 1



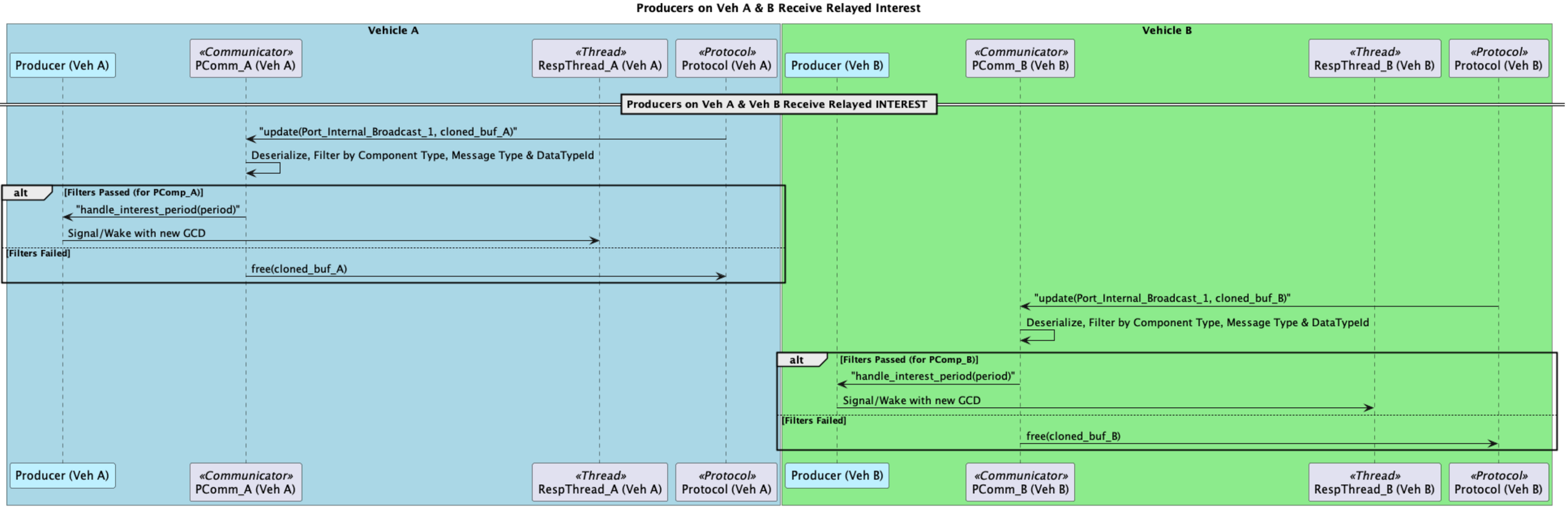
Sequence Diagram 1



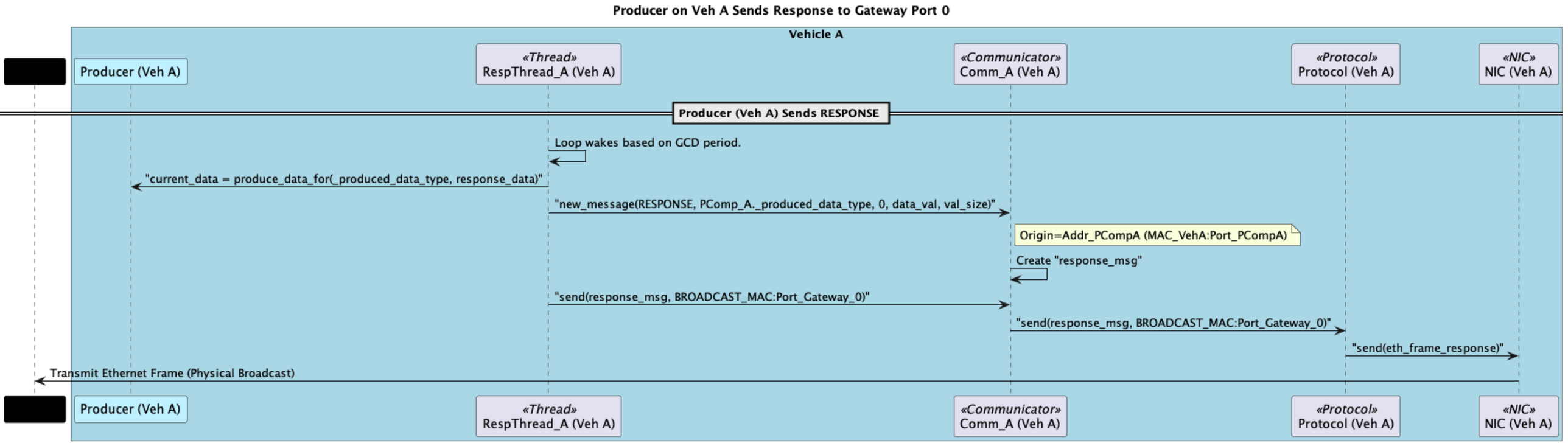
Sequence Diagram 1



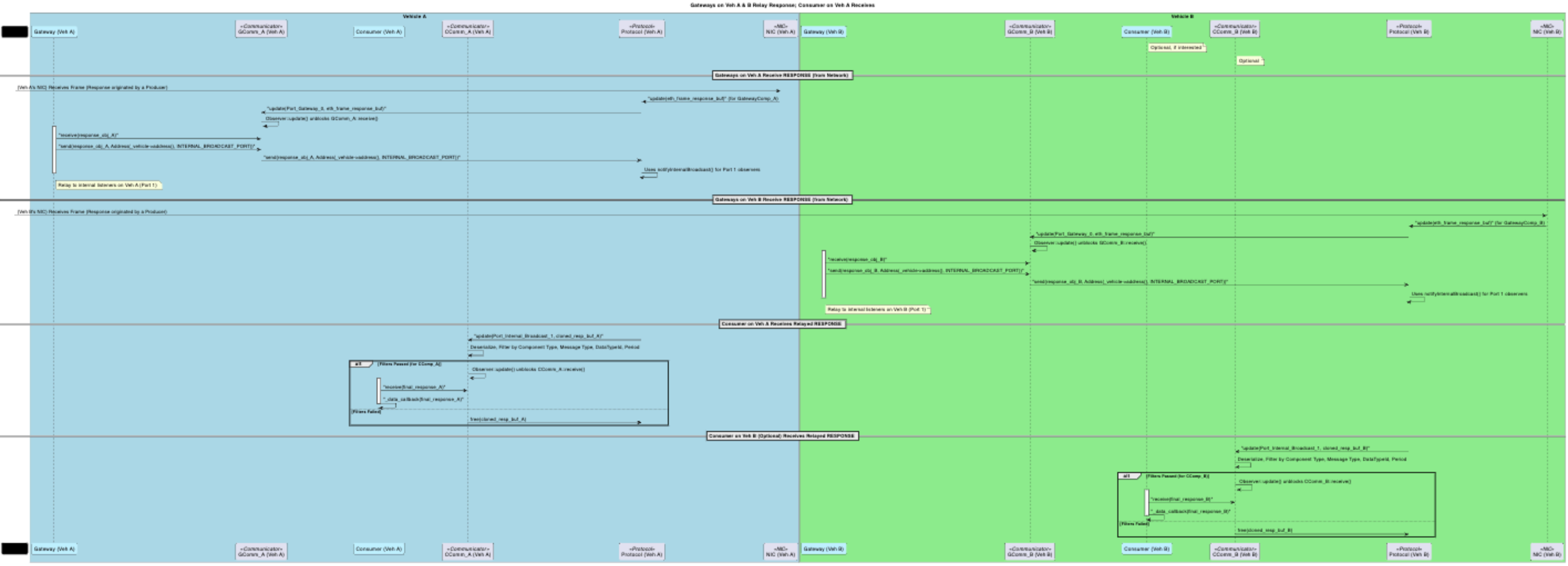
Sequence Diagram 1



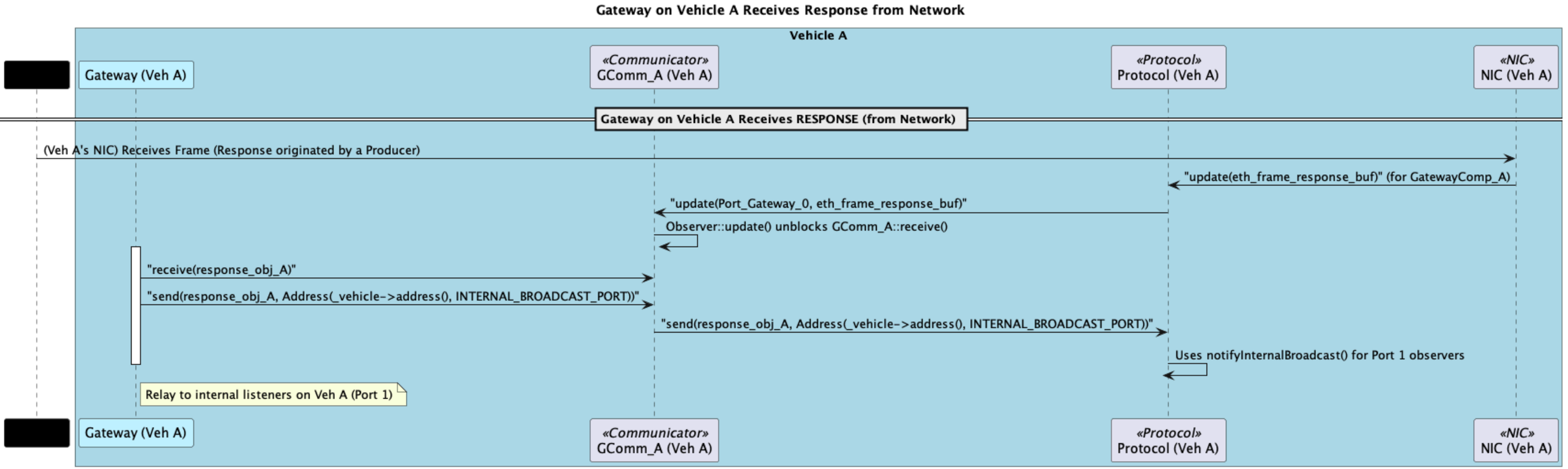
Sequence Diagram 2



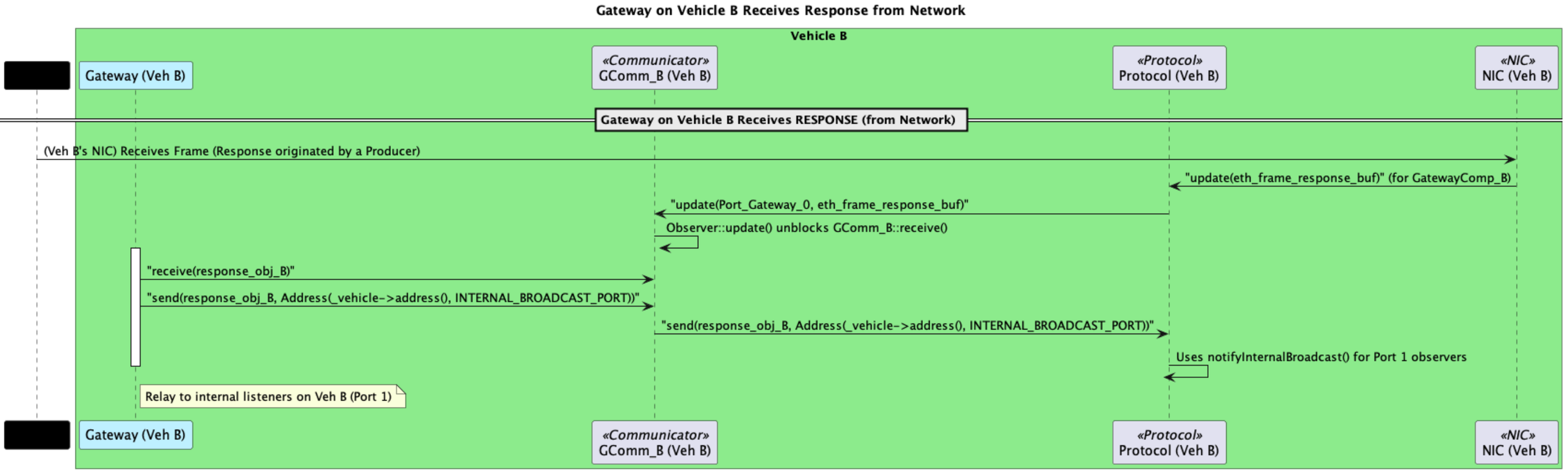
Sequence Diagram 3



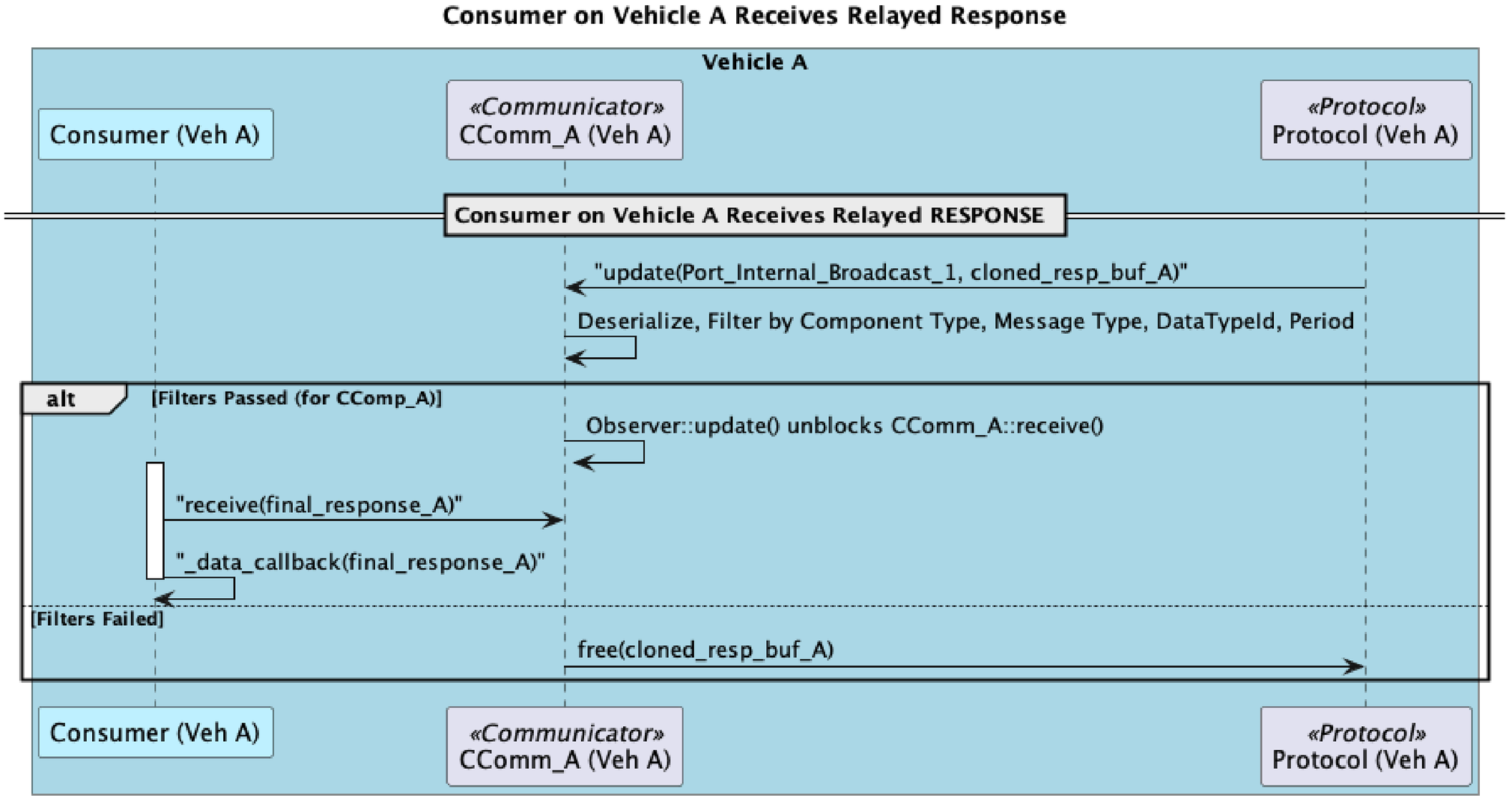
Sequence Diagram 3



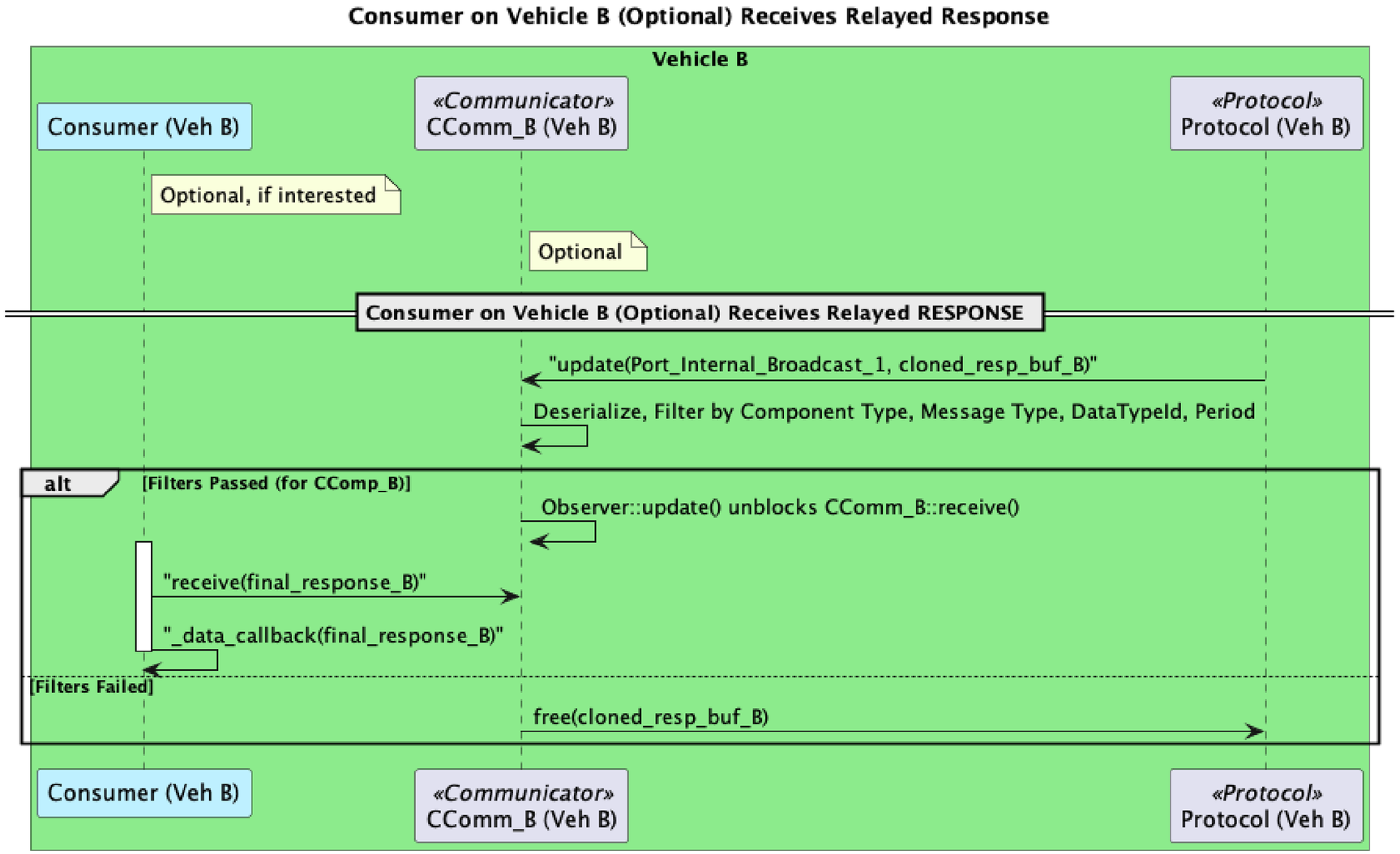
Sequence Diagram 3



Sequence Diagram 3



Sequence Diagram 3

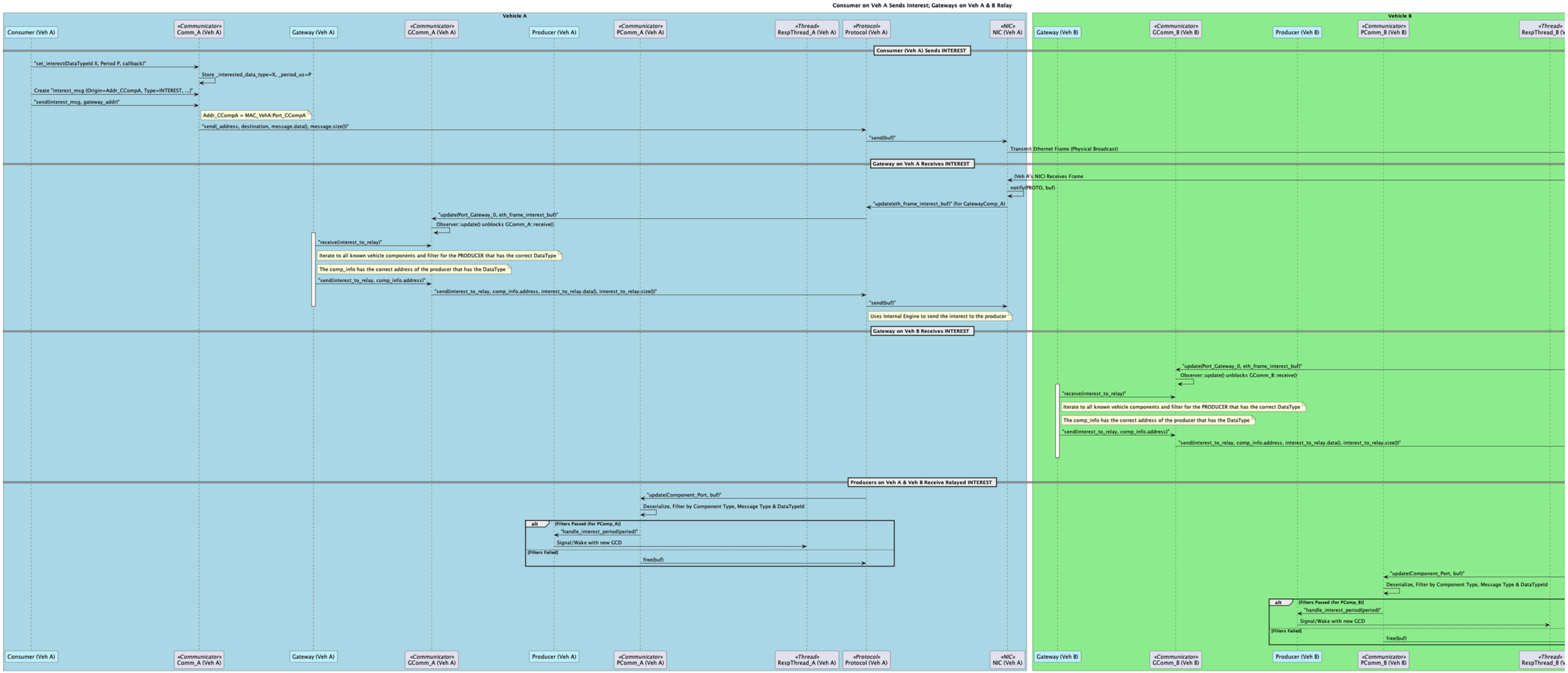


Reliable and secure communication library for critical autonomous systems

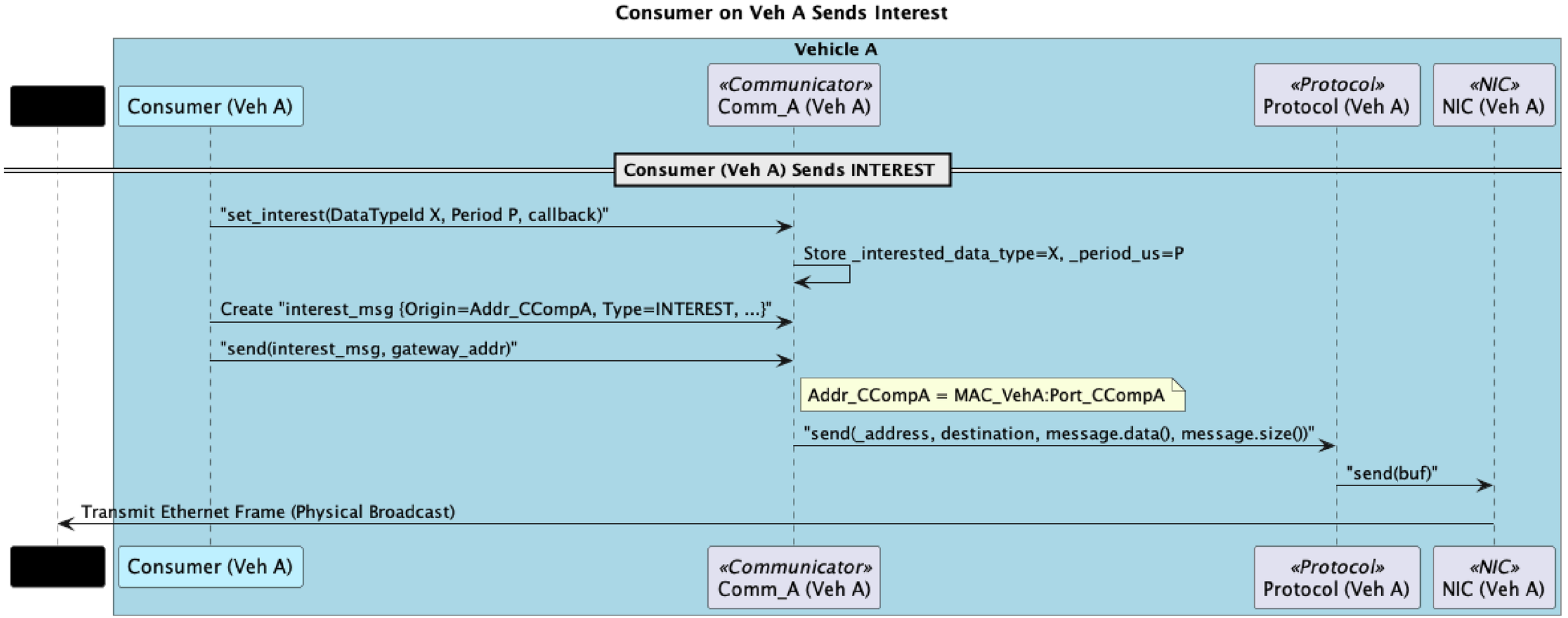
CÉSAR AUGUSTO PEREIRA DE SOUZA
ENZO NICOLÁS SPOTORNO BIEGER
JOÃO PEDRO PEREZ RESMER
JOÃO PEDRO SCHMIDT CORDEIRO

P3 19/05/2025

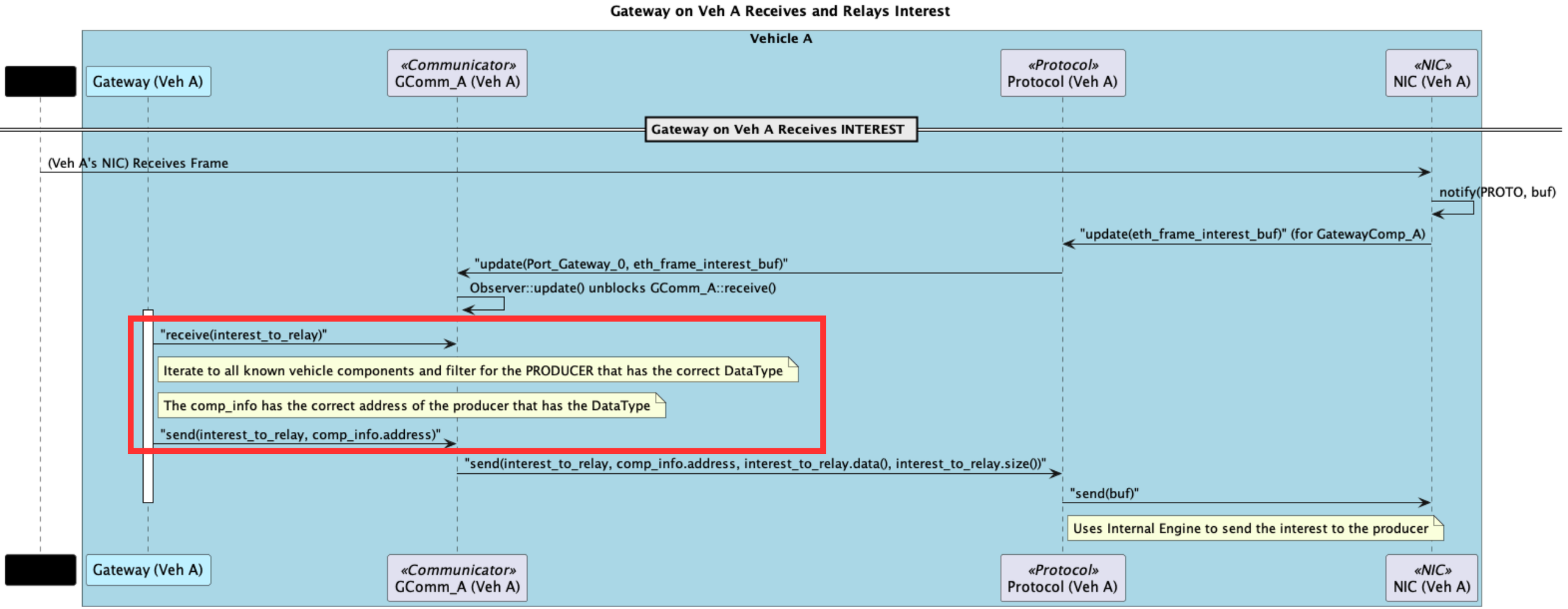
Sequence Diagram 1



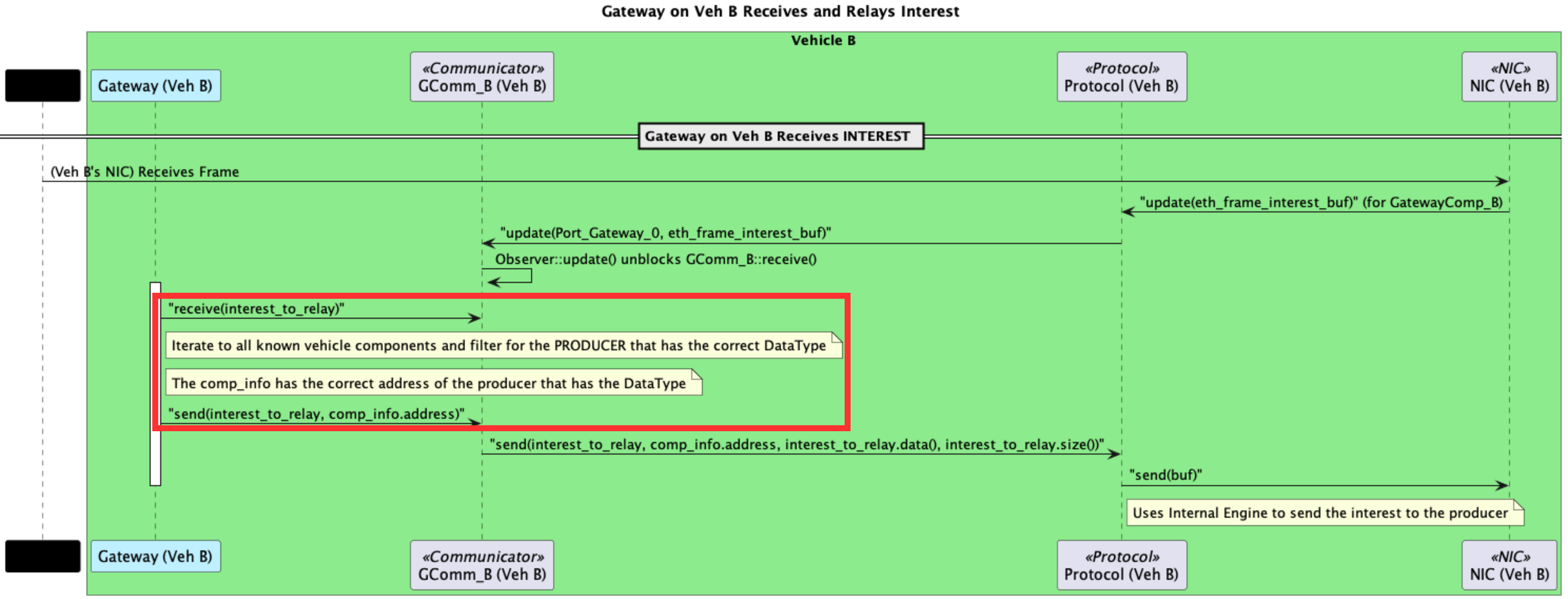
Sequence Diagram 1



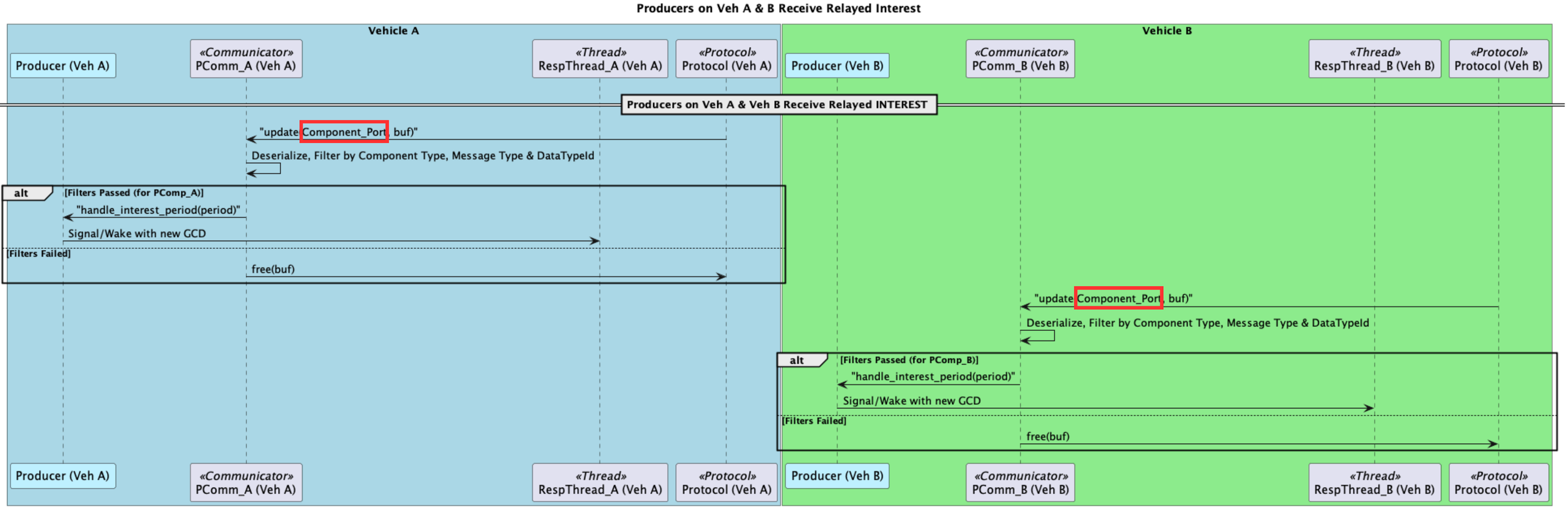
Sequence Diagram 1



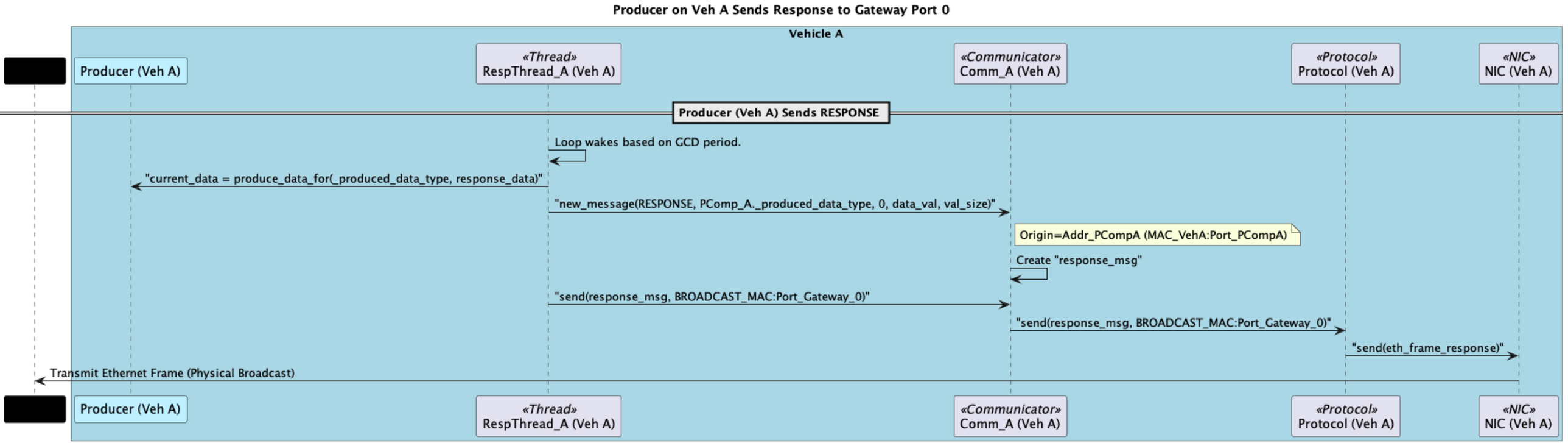
Sequence Diagram 1



Sequence Diagram 1

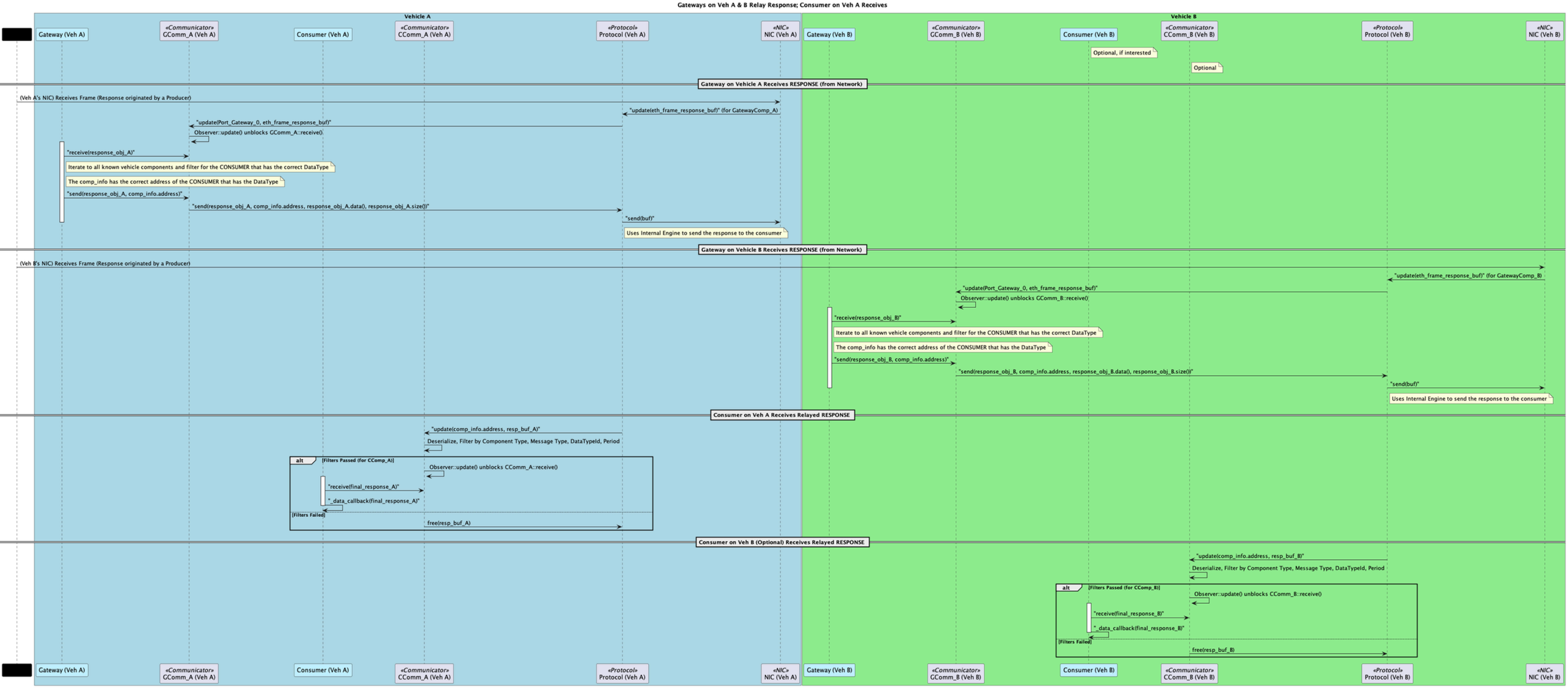


Sequence Diagram 2

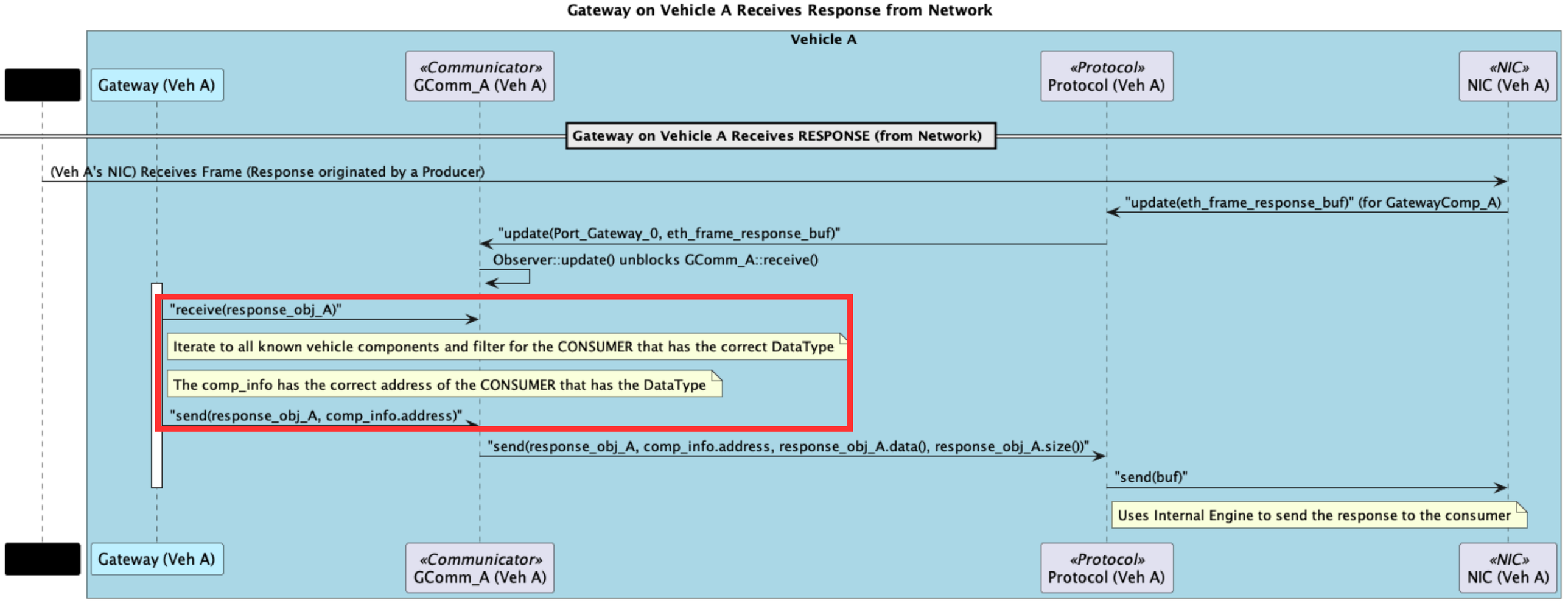


No changes

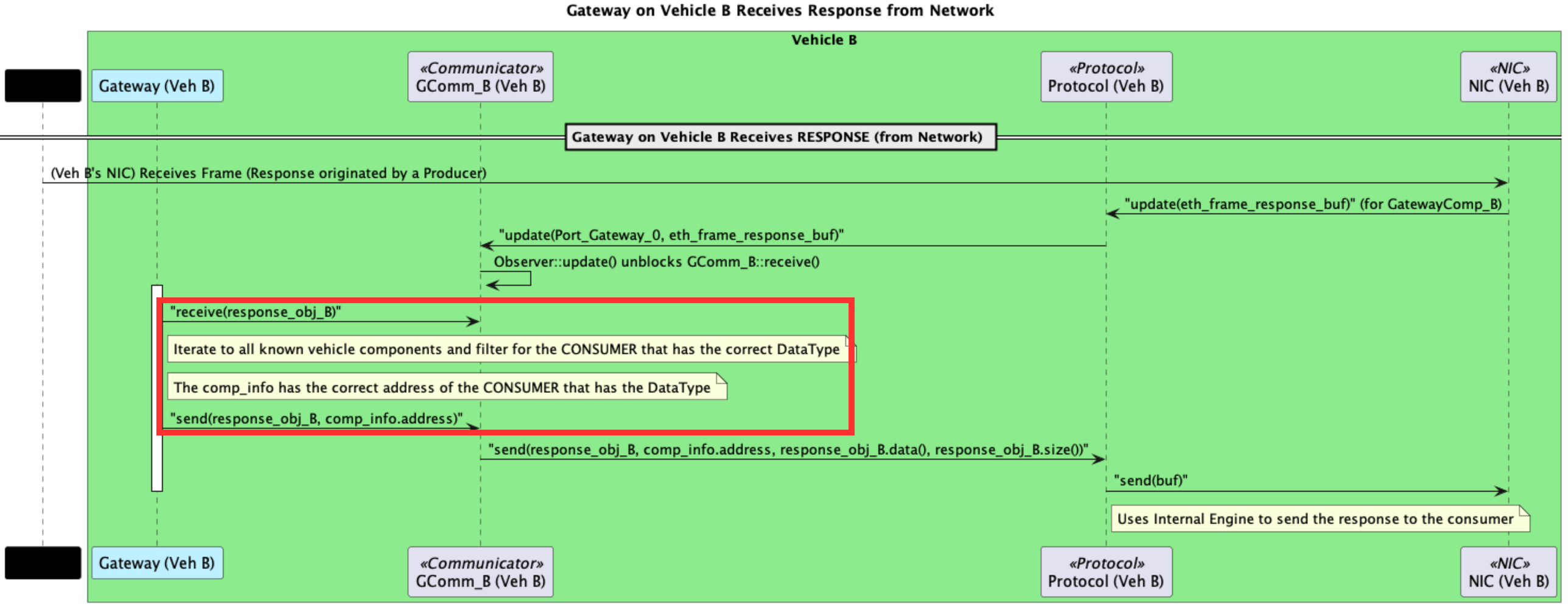
Sequence Diagram 3



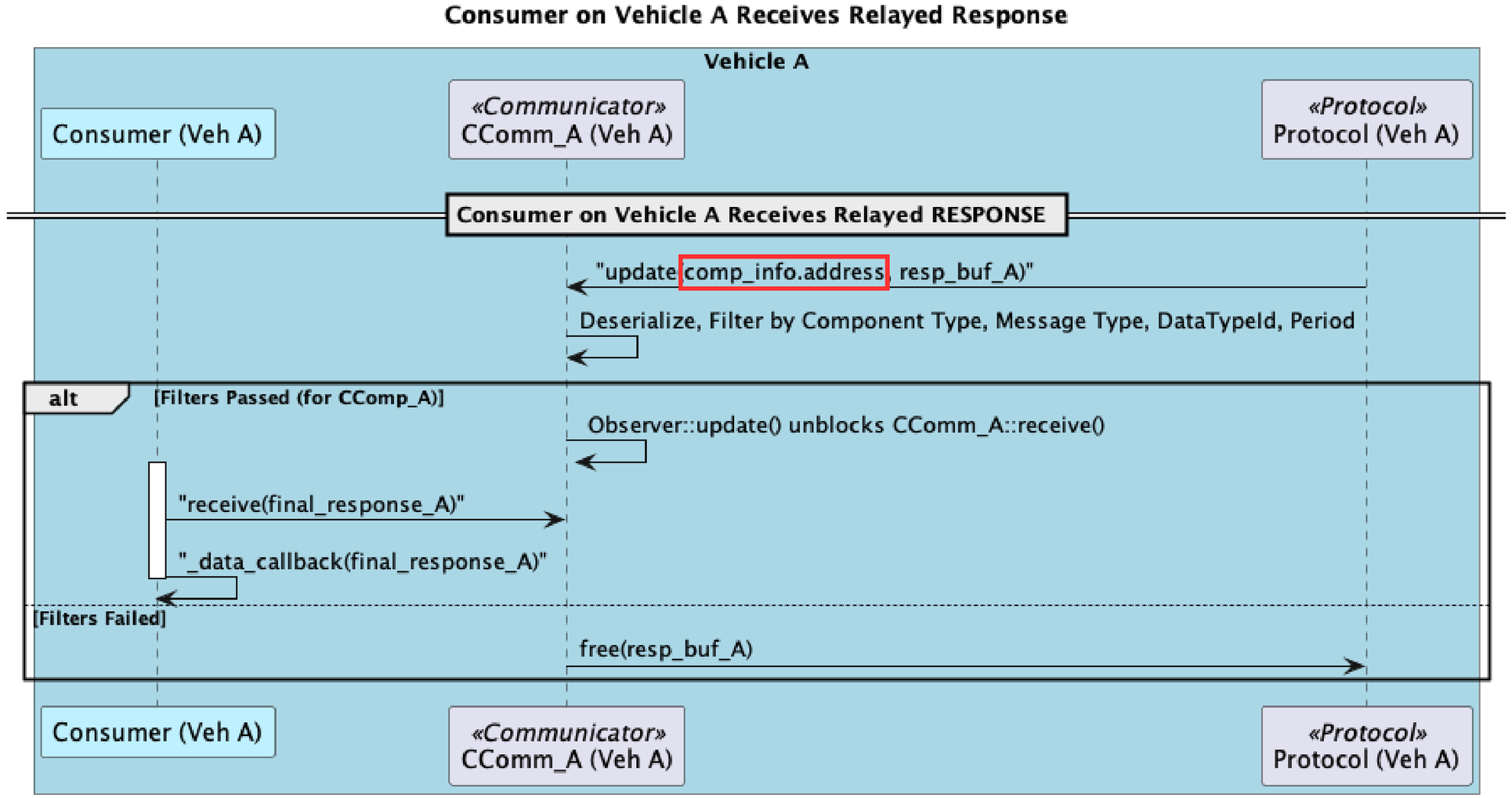
Sequence Diagram 3



Sequence Diagram 3



Sequence Diagram 3



Sequence Diagram 3