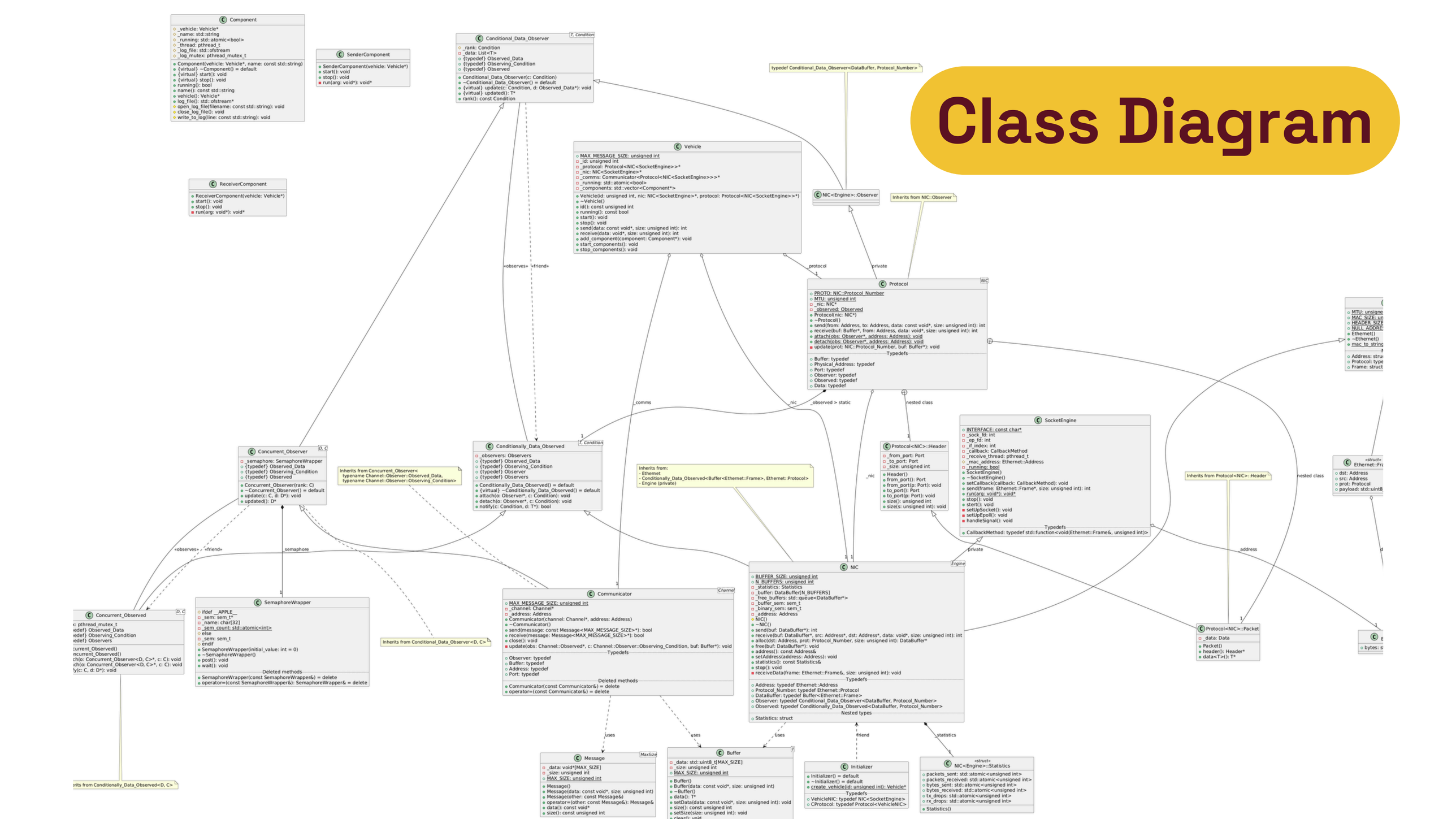
Reliable and secure communication library for critical autonomous systems

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

P1 09/04/2025


Class Diagram
Component
SenderComponent
Conditional_Data_Observer
ReceiverComponent
Vehicle
NIC<Engine>::Observer
Protocol
Concurrent_Observer
Conditionally_Data_Observed
Protocol<NIC>::Header
SocketEngine
Concurrent_Observed
SemaphoreWrapper
Communicator
NIC
Message
Buffer
Initializer
NIC<Engine>::Statistics
Protocol<NIC>::Packet

Buffer Class

Generic Data Buffering

Purpose

Template class providing type-safe data buffering for any data type

Key Features

- Fixed-size memory allocation based on template type
- Memory safety with automatic truncation of oversized data
- Clear data access methods with proper encapsulation

Pattern

Used in Observer pattern to pass data between components

Integration

Foundation for Ethernet frames, Protocol packets, and Messages

Message Class

Data Transport Container

Purpose

Generic container for data transmission in the communication system

Key Features

- Template-based with size constraints via MaxSize parameter
- Safe memory management with proper initialization and cleanup
- Consistent interface with data() and size() methods

Pattern

- Size validation prevents buffer overflows
- Deep copying ensures data integrity
- Proper initialization prevents memory leaks

Integration

Works with Communicator for message passing

Observer Pattern

Asynchronous Communication Foundation

Purpose Enables asynchronous notification between components

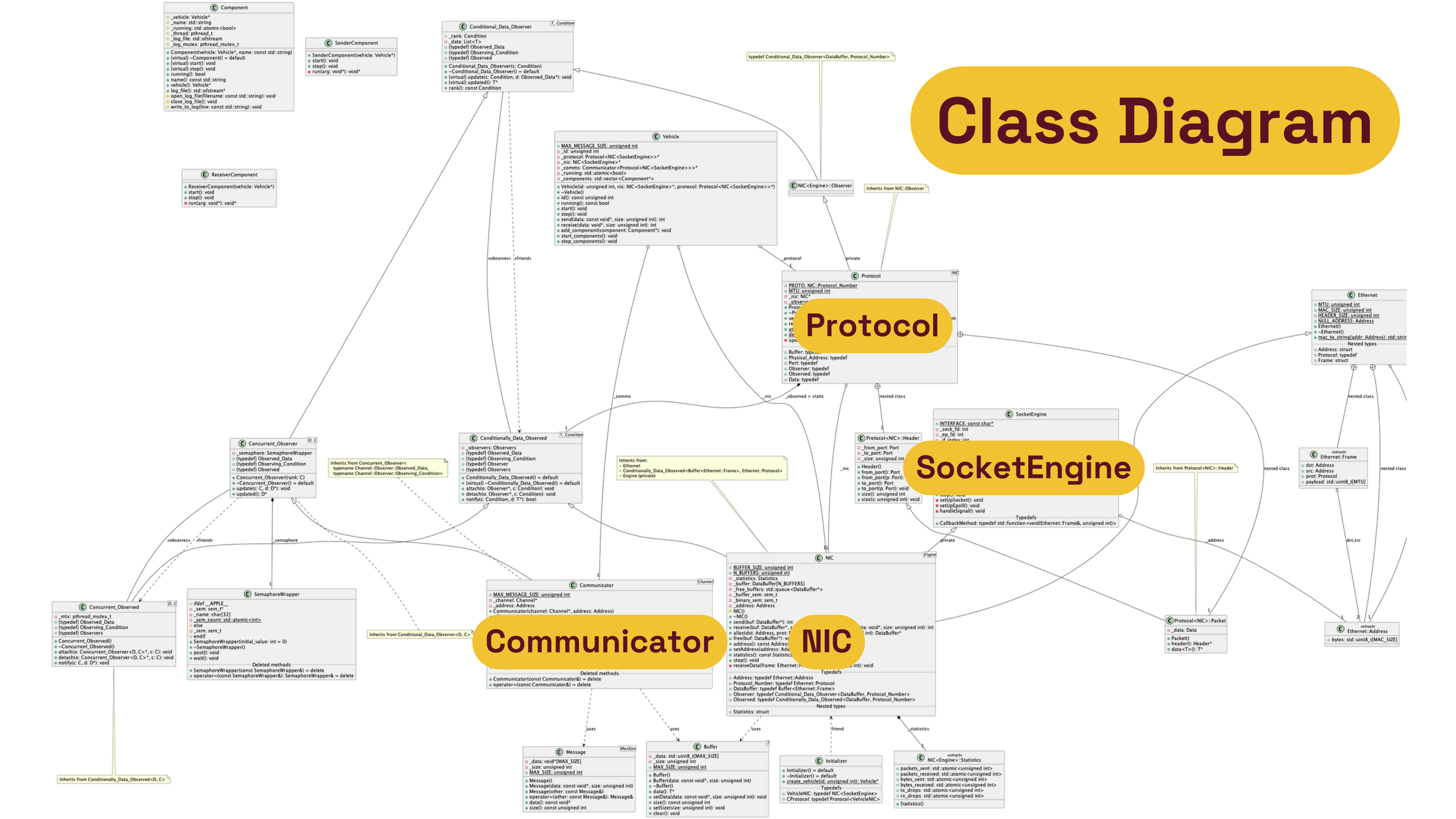
Key Classes

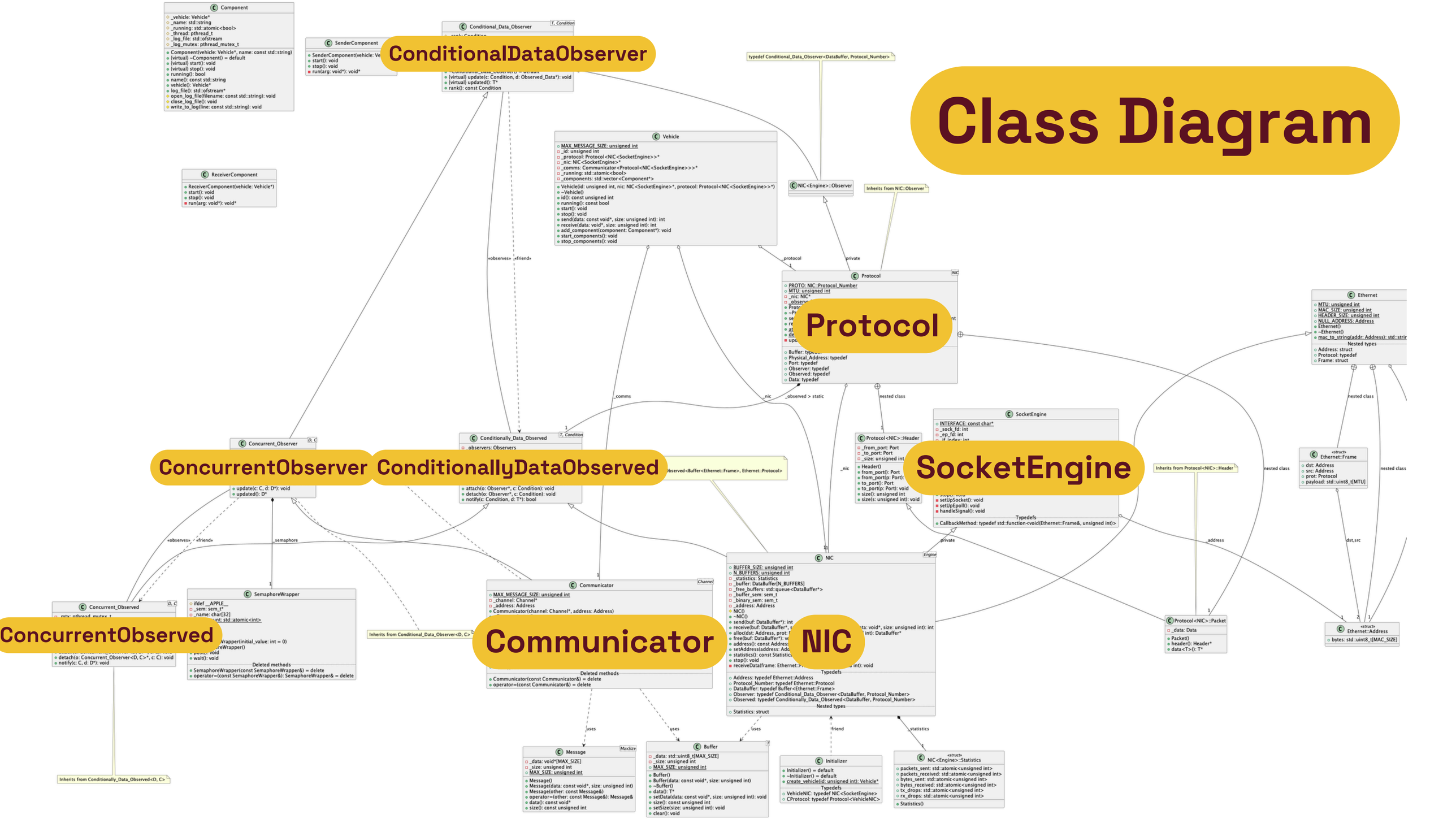
- Conditional_Data_Observer: Base observer with filtering
- Conditionally_Data_Observed: Manages observers by condition
- Concurrent_Observer: Thread-safe implementation with semaphores
- Concurrent_Observed: Thread-safe notification manager

Key Features

- Reference counting for memory safety
- Thread synchronization with semaphores
- Dual pattern implementation: Conditional and Concurrent

Integration Used throughout communication stack for asynchronous messaging


Class Diagram
Component
SenderComponent
Conditional_Data_Observer
ReceiverComponent
Vehicle
NIC<Engine>::Observer
Protocol
Ethernet
SocketEngine
Concurrent_Observer
Conditionally_Data_Observed
Protocol<NIC>::Header
Communicator
NIC
SemaphoreWrapper
Protocol<NIC>::Packet
Ethernet::Address
Ethernet::Frame
Message
Buffer
Initializer
NIC<Engine>::Statistics


Class Diagram
ConditionalDataObserver
Protocol
SocketEngine
ConcurrentObserver
ConditionallyDataObserved
ConcurrentObserved
Communicator
NIC
Component
SenderComponent
ReceiverComponent
Conditional_Data_Observer
Vehicle
NIC<Engine>::Observer
Inherits from NIC::Observer
typedef Conditional_Data_Observer<DataBuffer, Protocol_Number>
Protocol
Ethernet
Concurrent_Observer
Conditionally_Data_Observed
Protocol<NIC>::Header
SocketEngine
Inherits from Protocol<NIC>::Header
Ethernet::Frame
Concurrent_Observed
SemaphoreWrapper
Communicator
NIC
Protocol<NIC>::Packet
Ethernet::Address
Inherits from Conditional_Data_Observer<D, C>
Inherits from Conditionally_Data_Observed<D, C>
Message
Buffer
Initializer
NIC<Engine>::Statistics

Ethernet Class

Network Frame Management

Purpose Provides data structures and utilities for Ethernet frames

Key Features

- MAC address handling and string formatting
- Complete Ethernet frame structure with proper memory alignment
- Standard-compliant constants (MTU, MAC_SIZE)

Structures

- Address: MAC address representation
- Frame: Complete frame with source, destination, protocol, and payload

Integration Used by NIC as the foundation for network communication

SocketEngine Class

Network I/O Engine

Purpose

Provides low-level network communication capabilities

Key Features

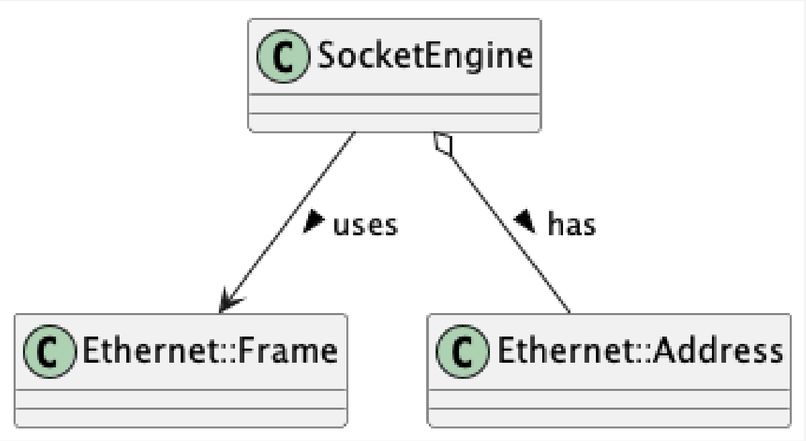
- Raw socket operations for Ethernet frames
- Asynchronous I/O using epoll for efficiency
- Dedicated thread for receiving frames

Technical Implementation

- Thread-safe event loop for frame reception
- Callback mechanism for processing received frames
- Non-blocking I/O for real-time requirements

Integration

Used by NIC as underlying engine for network operations



NIC Class

Network Interface Abstraction

Key Features

- Template-based design parameterized by Engine type
- Thread-safe buffer management with semaphores
- Statistics tracking for network operations

Observer Pattern

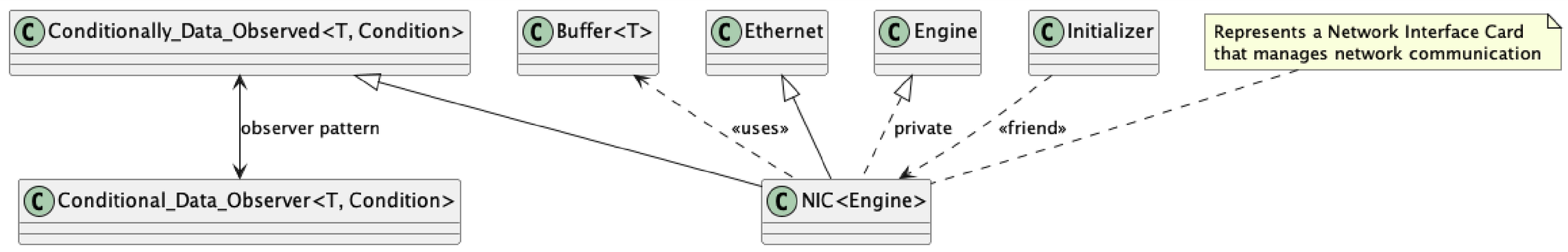
- Extends Conditionally_Data_Observed
- Notifies Protocol based on protocol numbers

Purpose

Foundation for network communication with Ethernet frames

Buffer-management

Fixed-size buffer pool with thread-safe allocation



Protocol Class

Message Routing Layer

Purpose

Routes messages between NIC and higher-level components

Key Features

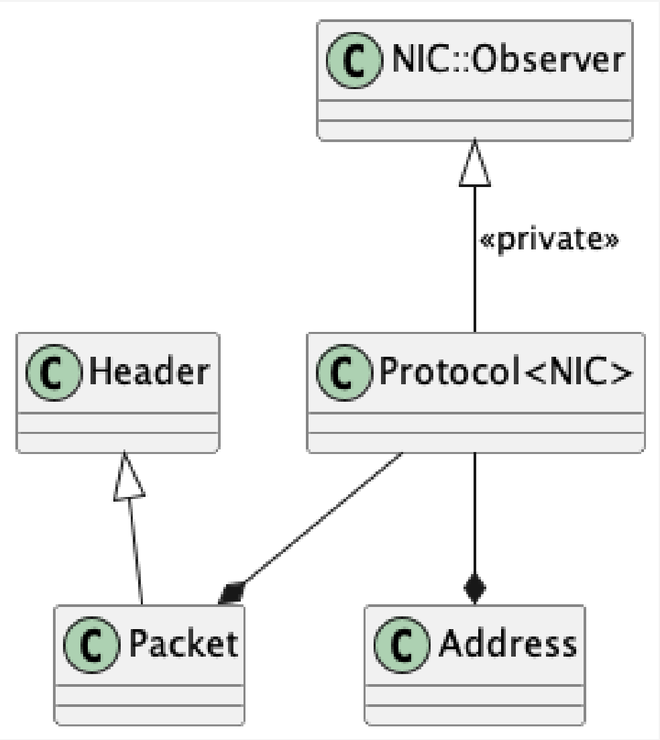
- Port-based addressing for message routing
- Handles protocol headers and packet formatting
- Bridges physical addresses and logical ports

Pattern

- Acts as Observer of NIC (inherits from NIC::Observer)
- Contains Concurrent_Observed to notify Communicators

Key Structures

- Header: Manages ports and message size
- Packet: Combines header with payload
- Address: Combines physical address with port



Communicator Class

High-Level Communication Endpoint

Purpose

Provides clean API for communication between system components

Key Features

- Template-based design works with any Channel implementation
- Implements Observer pattern for asynchronous message reception
- Reference counting for safe memory management

Pattern

Inherits from Concurrent_Observer to receive notifications

Hierarchy

Created by Vehicle, connects to Protocol for message transmission

Initializer Framework

System Orchestration

Purpose

Manages creation and lifecycle of vehicle processes and components

Key Features

- Process-based isolation for each vehicle
- Friend relationships for controlled dependency injection
- Complete communication stack setup

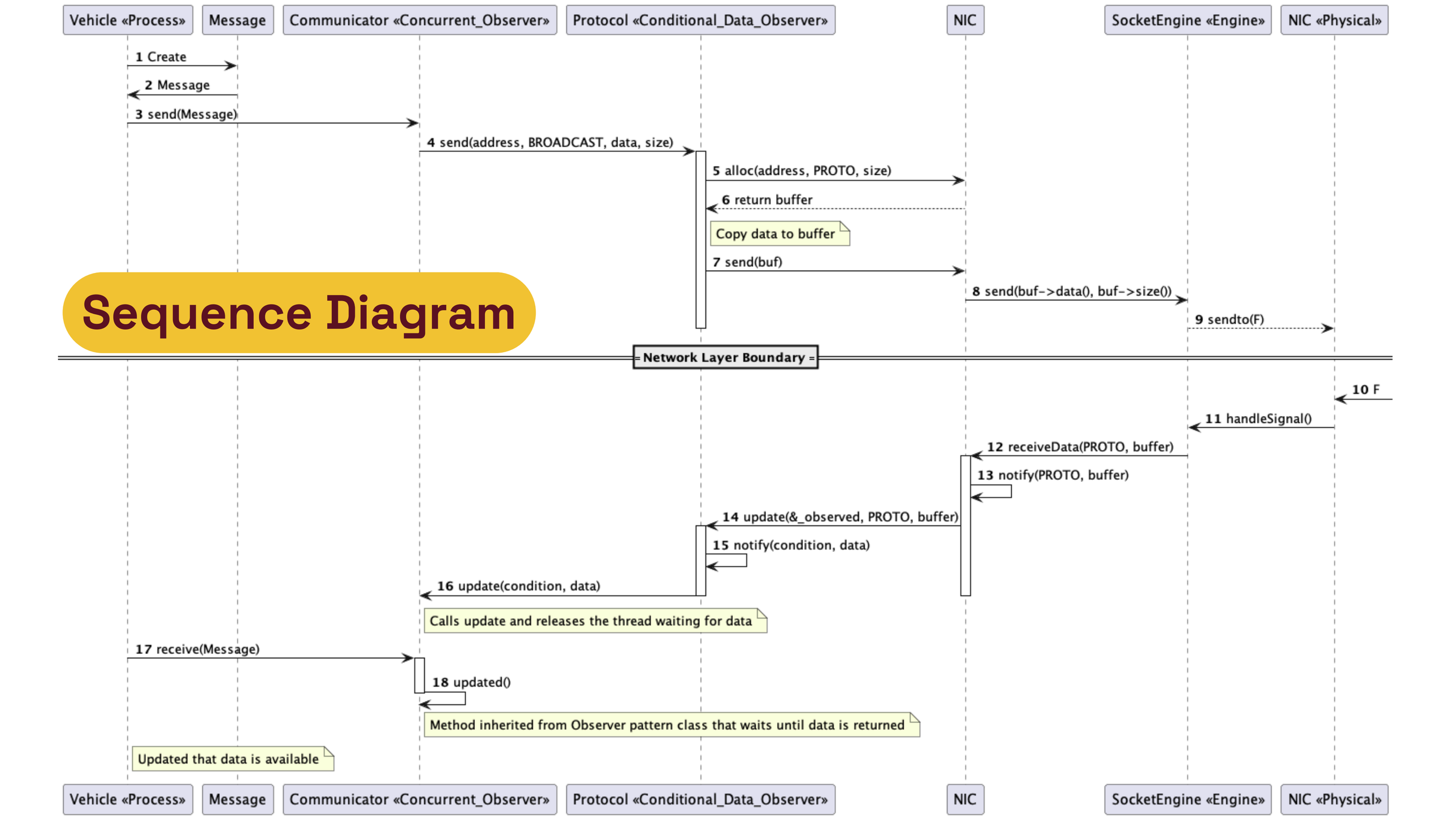
Pattern

Modified Dependency Injection where:

- Initializer creates NIC and Protocol
- Injects them into Vehicle
- Vehicle creates its own Communicator

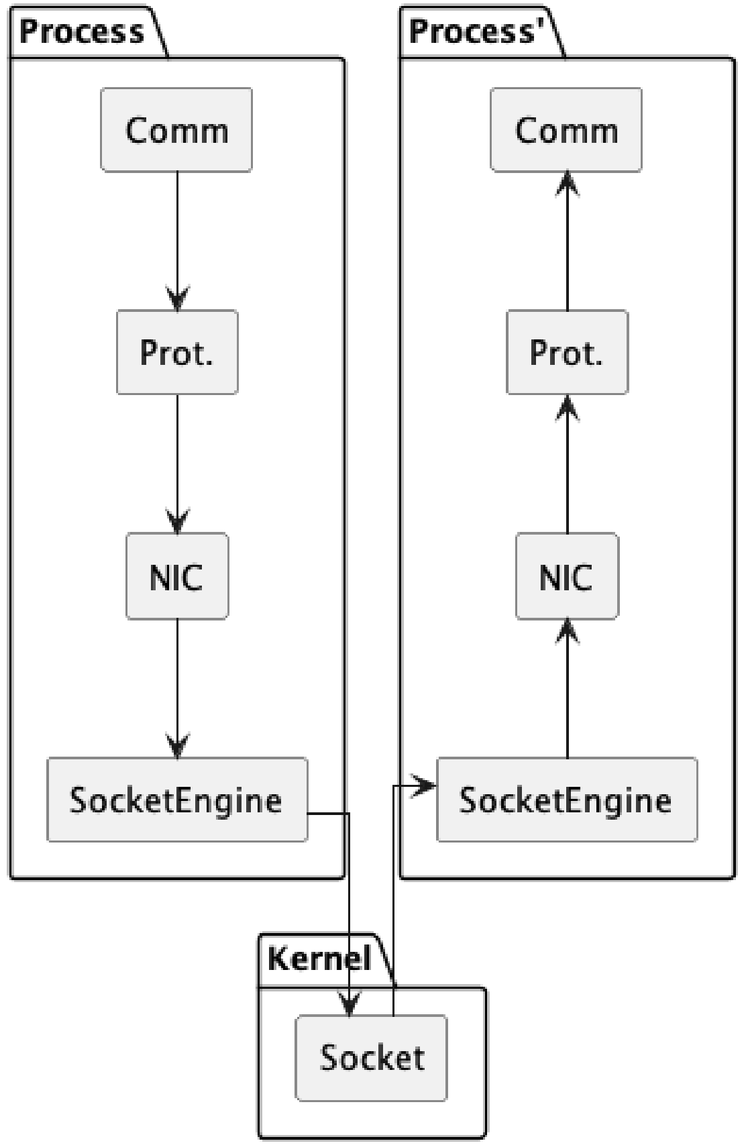
Hierarchy

Initializer → NIC/Protocol → Vehicle → Communicator


Vehicle «Process»
Message
Communicator «Concurrent_Observer»
Protocol «Conditional_Data_Observer»
NIC
SocketEngine «Engine»
NIC «Physical»
1 Create
2 Message
3 send(Message)
4 send(address, BROADCAST, data, size)
5 alloc(address, PROTO, size)
6 return buffer
Copy data to buffer
7 send(buf)
8 send(buf->data(), buf->size())
9 sendto(F)
Sequence Diagram
Network Layer Boundary
10 F
11 handleSignal()
12 receiveData(PROTO, buffer)
13 notify(PROTO, buffer)
14 update(&_observed, PROTO, buffer)
15 notify(condition, data)
16 update(condition, data)
Calls update and releases the thread waiting for data
17 receive(Message)
18 updated()
Method inherited from Observer pattern class that waits until data is returned
Updated that data is available
Vehicle «Process»
Message
Communicator «Concurrent_Observer»
Protocol «Conditional_Data_Observer»
NIC
SocketEngine «Engine»
NIC «Physical»

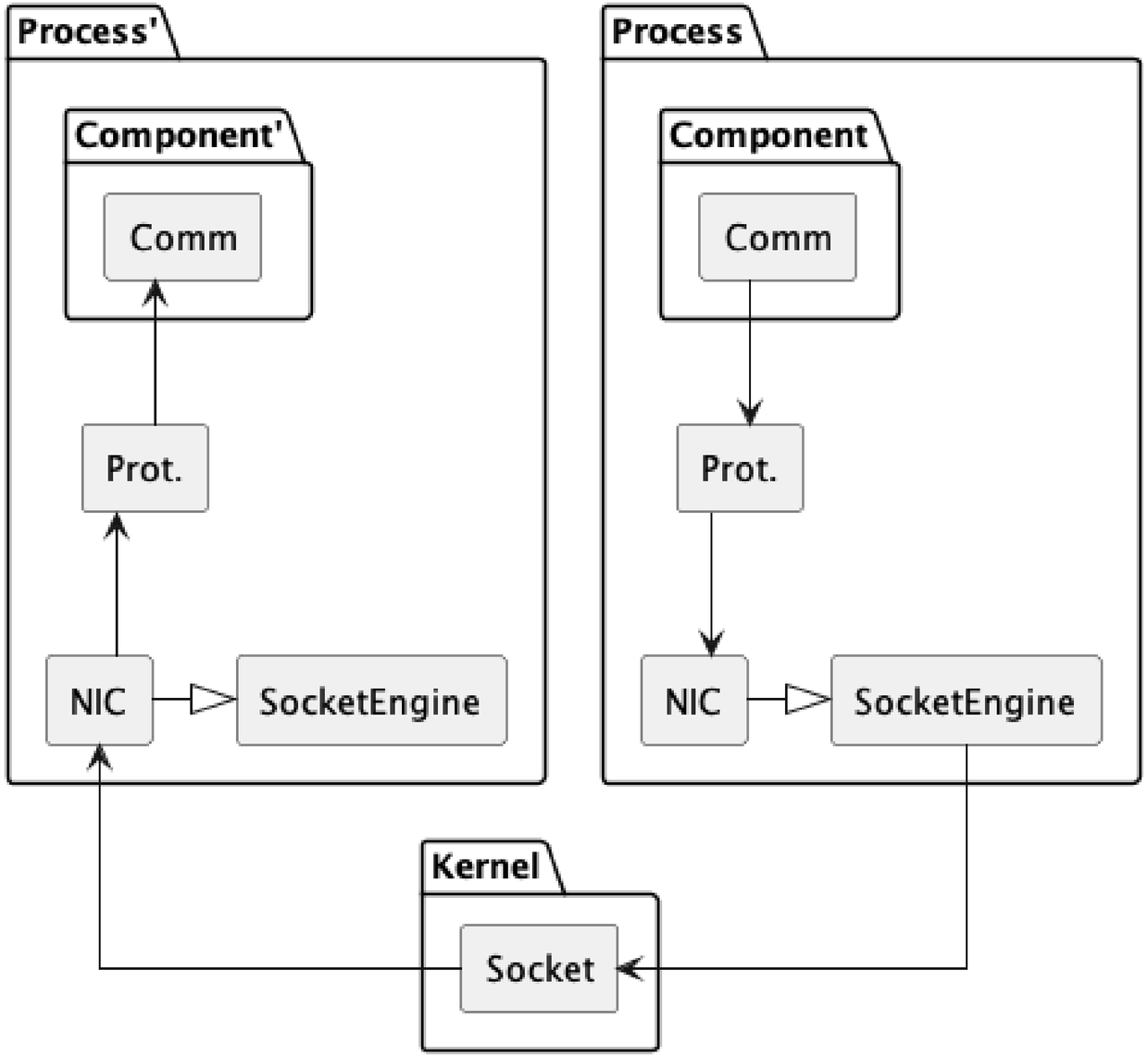
Process Diagram

Before



Process Diagram

After



Performance Evaluation

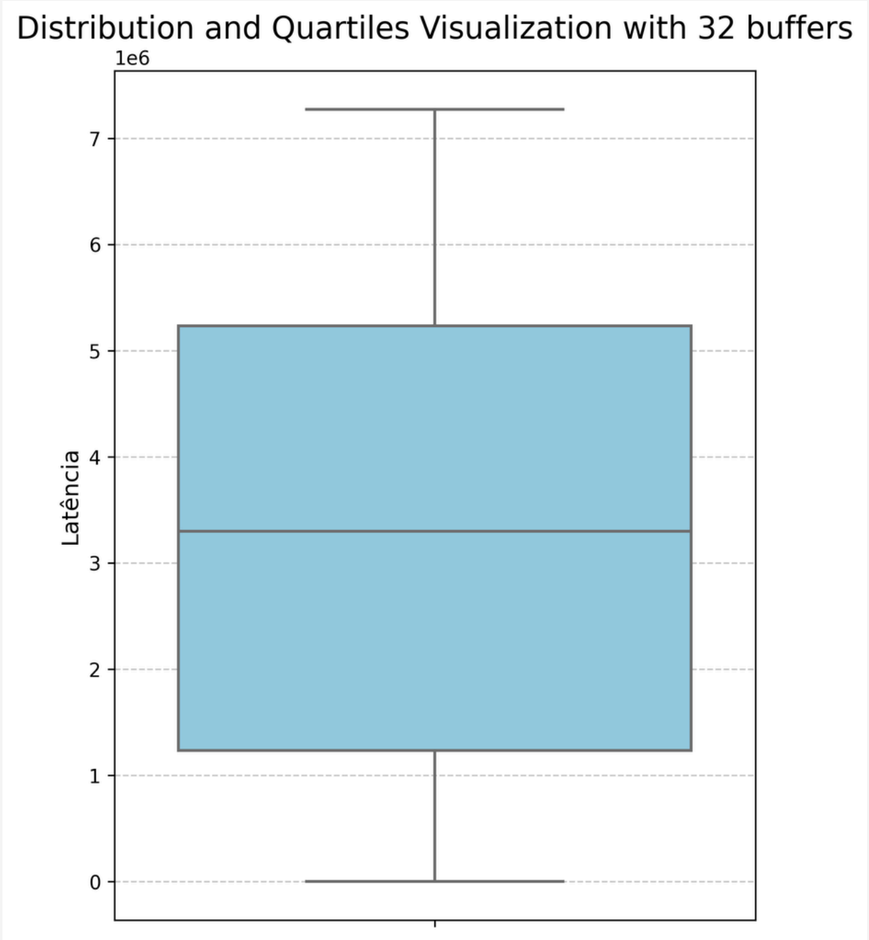
Latency

- Parâmetros:
  Buffer Pool: 32 buffers
  Frequência das mensagens: entre 0 e 50 µs
  Frota: 1000 veículos

- Medições: 489175
  Latência máxima observada: 7271761 µs
  Latência mínima obsevada: 106 µs

- Latência média: 3281330,396 µs
  Desvio Padrão: 2119083,121

- Q1 = 1233384 µs
  Mediana = 3302067 µs
  Q3 = 5233205 µs



Performance Evaluation

Latency