



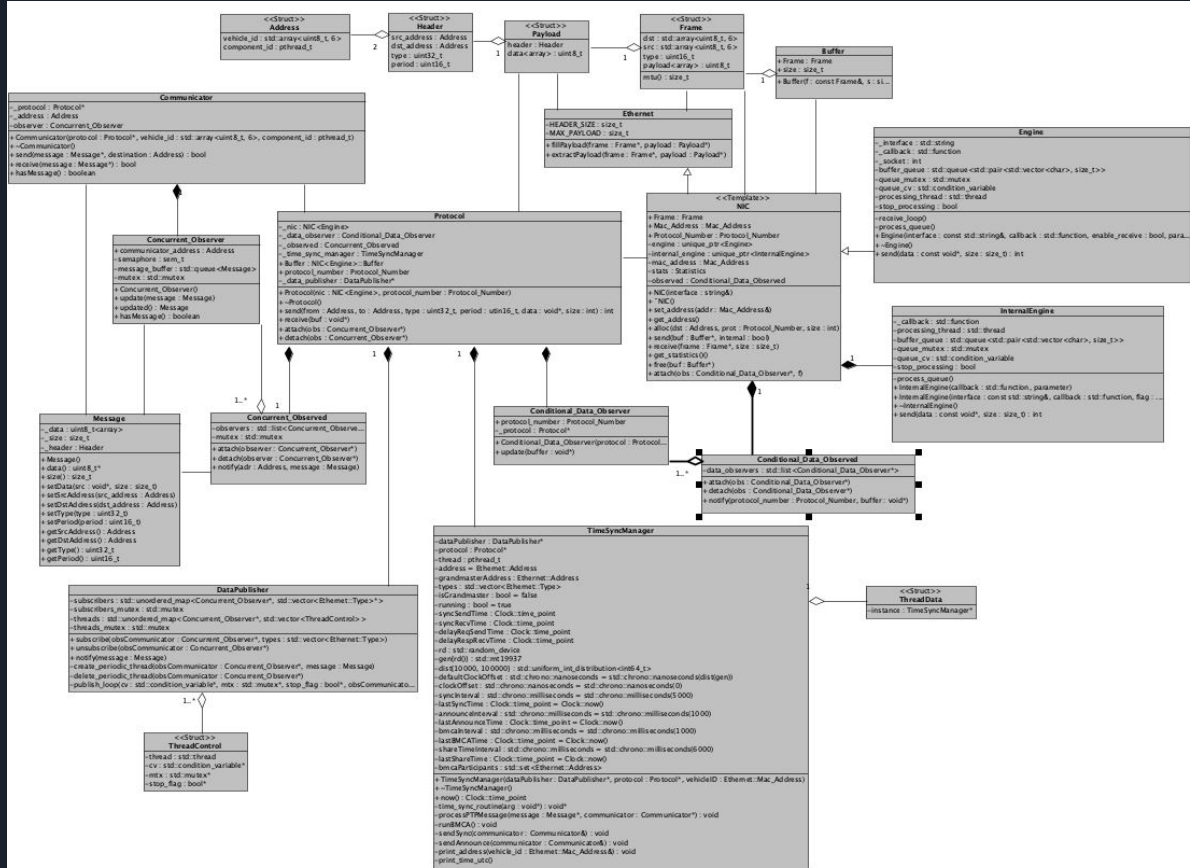
Entrega 4 INE-5424



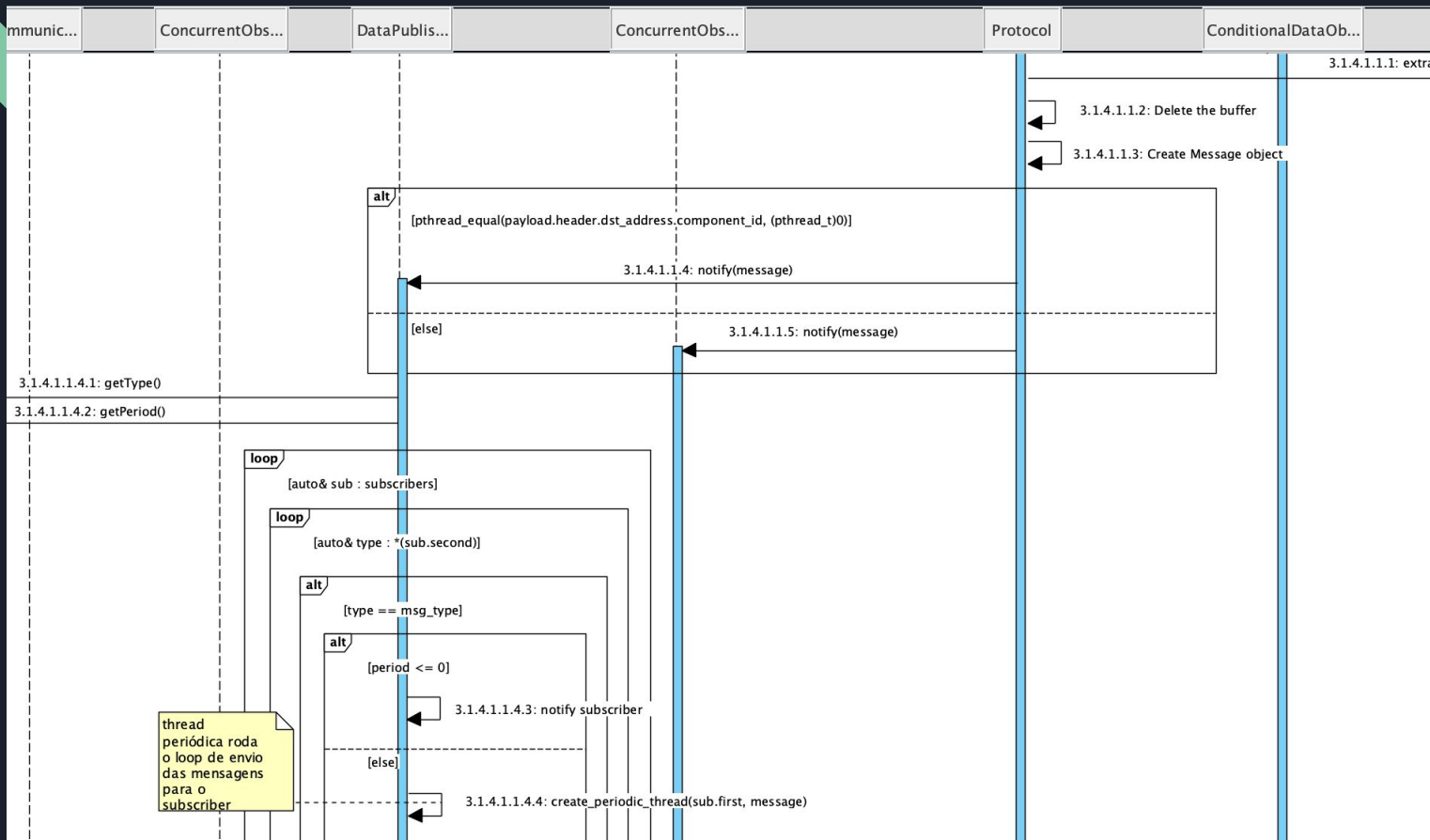
Mudanças em Relação ao P3

- Não utilizamos mais um agendador externo responsável pelos períodos do publish-subscribe
- Toda a lógica foi trazida para dentro da API, na classe DataPublisher

Mudanças em Relação ao P3




Mudanças em Relação ao P3





Mudanças em Relação ao P3

- Agora, cada vez que houver um novo interesse, uma nova thread periódica será criada, responsável por monitorar o período

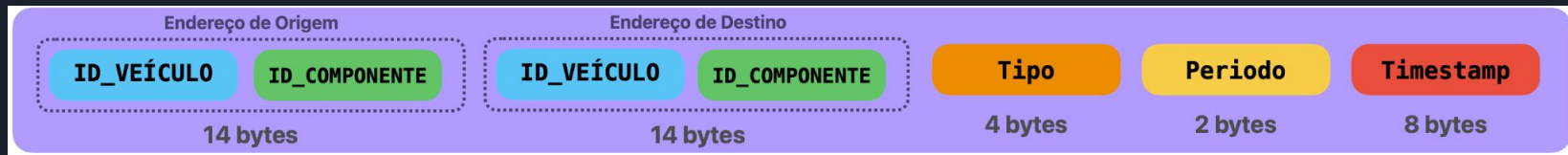


```
// Verifica quais observadores estão interessados na mensagem e os notifica.
void DataPublisher::notify(Message message) {
    Ethernet::Type msg_type = message.getType();
    Ethernet::Period period = message.getPeriod();

    std::lock_guard<std::mutex> lock(subscribers_mutex);
    for (auto& sub : subscribers) {
        for (auto& type : *(sub.second)) {
            if (type == msg_type) {
                // Envia diretamente se não for periódico
                if (period <= 0) {
                    sub.first->update(message);
                } else {
                    // Cria thread periódica para mensagens com período > 0
                    create_periodic_thread(sub.first, message);
                }
                break; // Já encontrou o tipo desejado, pula para o próximo inscrito
            }
        }
    }
}
```

P4: Sincronização Temporal

- Mudança no Header para conter o timestamp





Sincronização Temporal

Adição de mensagens do tipo PTP

```
// Tipos de dados utilizados pelo Time Synchronization Manager (PTP - IEEE 1588).  
Ethernet::Type constexpr static TYPE_PTP_ANNOUNCE = 0x0B; // (4 bytes) Tipo de dado PTP Announce  
Ethernet::Type constexpr static TYPE_PTP_SYNC = 0x0; // (4 bytes) Tipo de dado PTP Sync  
Ethernet::Type constexpr static TYPE_PTP_DELAY_REQ = 0x01; // (4 bytes) Tipo de dado PTP Delay Request  
Ethernet::Type constexpr static TYPE_PTP_DELAY_RESP = 0x09; // (4 bytes) Tipo de dado PTP Delay Response
```

O Comunicador responsável por enviar/receber mensagens PTP, realizará sua inscrição no DataPublisher com esses tipos.

```
// Tipos de mensagens do PTP que serão processadas  
types.push_back(Ethernet::TYPE_PTP_ANNOUNCE);  
types.push_back(Ethernet::TYPE_PTP_SYNC);  
types.push_back(Ethernet::TYPE_PTP_DELAY_REQ);  
types.push_back(Ethernet::TYPE_PTP_DELAY_RESP);
```

```
Communicator communicator(self->protocol, self->address.vehicle_id, pthread_self());  
self->dataPublisher->subscribe(communicator.getObserver(), &self->types);
```



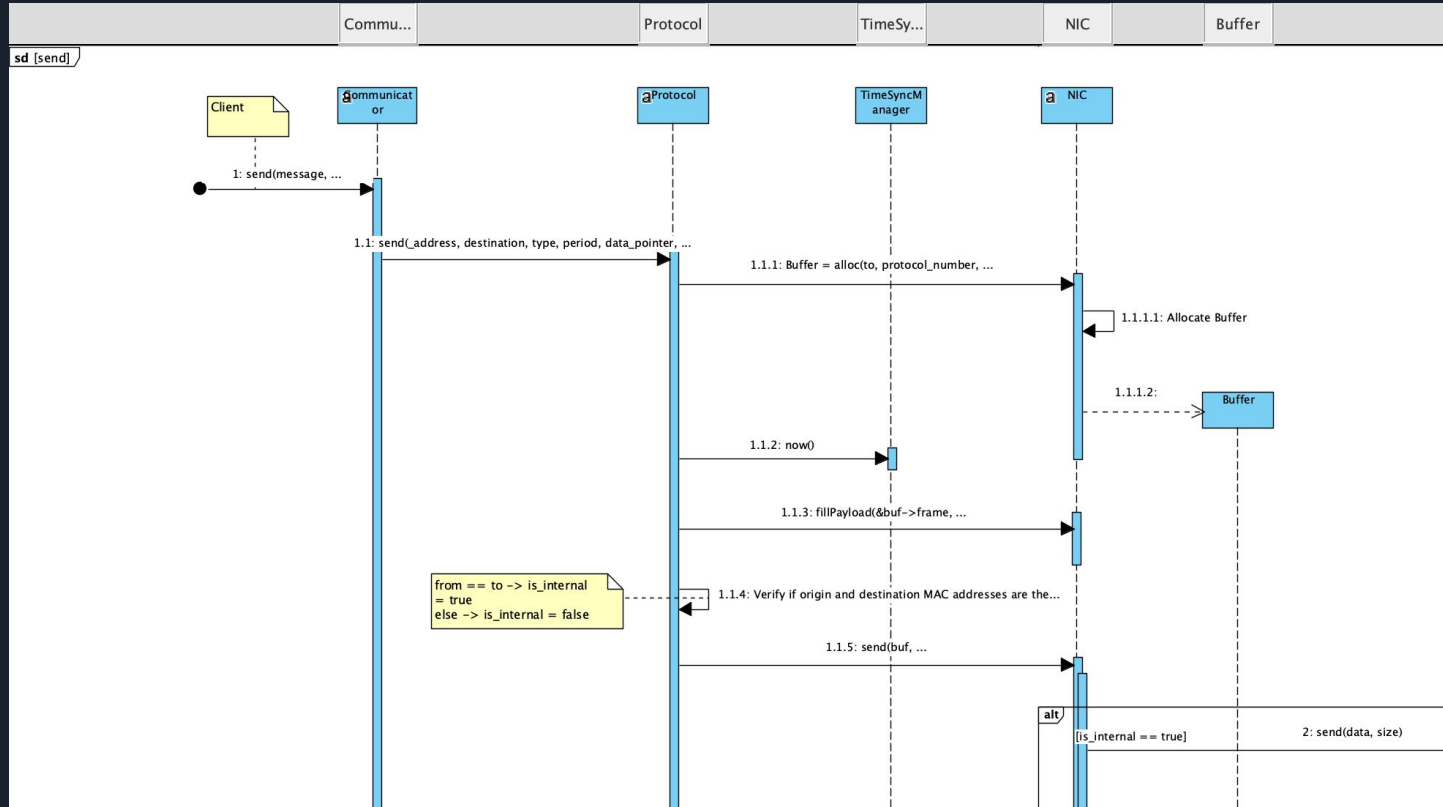

Sincronização Temporal

Criação da classe TimeSyncManager

- Responsável por gerenciar a sincronização dos relógios
- Implementa o algoritmo BMCA para eleger o grandmaster
- Executa em uma thread separada para enviar/receber mensagens de sincronização
- O timestamp das mensagens é preenchido pelo Protocol no momento de envió.

```
// Obtem a hora atual de acordo com o etiquetador (TimeSyncManager).  
auto now = _time_sync_manager.now();  
auto timestamp = std::chrono::duration_cast<std::chrono::nanoseconds>(now.time_since_epoch()).count();  
payload.header.timestamp = timestamp; // Horário de envió
```

Sincronização Temporal





Sincronização Temporal

Responsável por gerenciar a sincronização dos relógios

- Offset fictício (default) utilizado para simular adiantamento do relógio local.
- Offset do GC (clockOffset) atualizado dinamicamente com offset do GC.

```
std::uniform_int_distribution<int64_t> dist{0, 10000};

// Offset default do veículo (utilizado para simular erro de relógio).
std::chrono::microseconds defaultClockOffset{dist(gen)};
// Offset do GM atual.
std::chrono::microseconds clockOffset{0};
```

- Tem a função now() que retorna o horário do nó já corrigido pelo offset calculado

```
// Retorna o horário atual corrigido pelo offset calculado, em microssegundos
Clock::time_point now() {
    return Clock::now() + std::chrono::duration_cast<Clock::duration>(defaultClockOffset) +
        std::chrono::duration_cast<Clock::duration>(clockOffset);
}
```

Sincronização Temporal

Cálculo do Offset do Grandmaster:

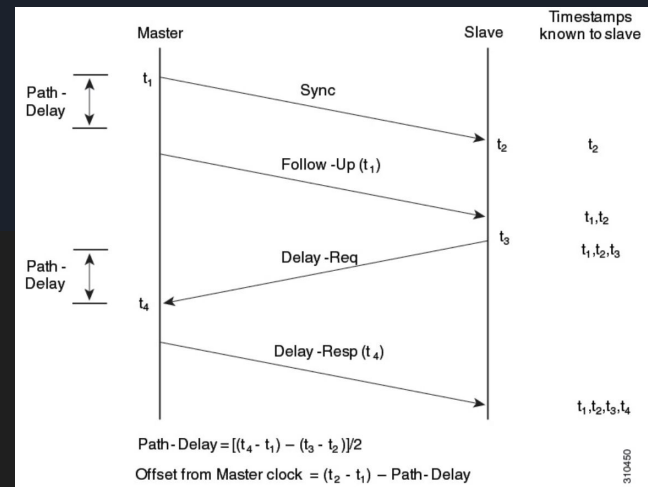
```
// Cálculo do offset baseado nas fórmulas do PTP
// t1: syncSendTime (GM)      t2: syncRecvTime (Slave)
// t3: delayReqSendTime (Slave) t4: delayRespRecvTime (GM)
```

```
auto t1 = syncSendTime;
auto t2 = syncRecvTime;
auto t3 = delayReqSendTime;
auto t4 = delayRespRecvTime;
```

```
// Calcula offset e delay corretamente
```

```
auto offset = std::chrono::duration_cast<std::chrono::microseconds>(((t2 - t1) - (t4 - t3)) / 2);
//auto delay = std::chrono::duration_cast<std::chrono::microseconds>((t2 - t1 + (t4 - t3)) / 2);
```

```
clockOffset = offset;
```





Sincronização Temporal

Função BMCA:

- Utiliza o ID do veículo como critério de escolha.
- Em caso de empate, utiliza o ID do componente como critério de desempate.

```
// Eleição do Grandmaster usando BMCA (Best Master Clock Algorithm)
// critério: Candidato com maior vehicle_id (mac_address fictício da nic)
void runBMCA() {
    clockOffset = std::chrono::microseconds(0);

    if (bmcaParticipants.empty()) {
        std::cout << "Nenhum participante encontrado para BMCA" << std::endl;
        isGrandmaster = true;
        grandmasterAddress = address;
        std::cout << "Grandmaster eleito: ";
        print_address(grandmasterAddress.vehicle_id);
        std::cout << "\n" << std::endl;
        return;
    }

    // Seleciona o maior endereço como Grandmaster (critério simples)
    auto grandmaster = address;
    for (const auto& participant : bmcaParticipants) {
        if (participant > grandmaster) {
            grandmaster = participant;
        }
    }
    grandmasterAddress = grandmaster;

    isGrandmaster = (grandmaster == address);

    std::cout << "Grandmaster eleito: ";
    print_address(grandmasterAddress.vehicle_id);
    std::cout << "\n";

    bmcaParticipants.clear();
}
```



Sincronização Temporal

Parametrização das taxas de envio e processamento do BMCA

- Intervalo de envio das mensagens SYNC
- Intervalo de envio das mensagens Announce
- Intervalo de processamento do BMCA
- Intervalo de compartilhamento do relógio local (apenas para visualização nos testes)

```
// Intervalos de tempo para eventos periódicos
std::chrono::milliseconds syncInterval{100};
Clock::time_point lastSyncTime = Clock::now();

std::chrono::milliseconds announceInterval{500};
Clock::time_point lastAnnounceTime = Clock::now();

std::chrono::milliseconds bmcaInterval{3000};
Clock::time_point lastBMCATime = Clock::now();

std::chrono::milliseconds shareTimeInterval{1000};
Clock::time_point lastShareTime = Clock::now();
```

Sincronização Temporal

Possui uma thread separada responsável por enviar/receber mensagens de sincronização e execução do BMCA.

```
// Loop principal da thread que ficará escutando e enviando mensagens
while (self->running) {
    if (communicator.hasMessage()) {
        Message msg;
        communicator.receive(&msg);           // Recebe mensagem
        self->processPTPMessage(&msg, &communicator); // Processa mensagem recebida
    }

    auto now = Clock::now();

    // Se é grandmaster e passou intervalo para enviar sync, envia
    if (self->isGrandmaster && ((now - self->lastSyncTime) >= self->syncInterval)) {
        self->sendSync(communicator);
        self->lastSyncTime = now;
    }

    // Se passou intervalo para enviar announce, envia
    if ((now - self->lastAnnounceTime) >= self->announceInterval) {
        self->sendAnnounce(communicator);
        self->lastAnnounceTime = now;
        //std::cout << "Enviando mensagem Announce" << std::endl;
    }
}
```

```
// Se passou intervalo para rodar BMCA (eleição do grandmaster), roda
if ((now - self->lastBMCATime) >= self->bmcaInterval) {
    std::cout << "Rodando BMCA" << std::endl;
    // Roda BMCA para eleger o grandmaster
    self->runBMCA();
    self->lastBMCATime = now;
}

if ((now - self->lastShareTime) >= self->shareTimeInterval) {
    self->lastShareTime = now;

    auto time = self->now();

    if (self->isGrandmaster) {
        std::cout << "GM" << std::endl;
        self->print_time_utc();
    } else {
        std::cout << "M" << std::endl;
        self->print_time_utc();
    }
}

// Remove inscrição para mensagens antes de encerrar
self->dataPublisher->unsubscribe(communicator.getObserver());

// Libera memória alocada para passagem de dados da thread
delete data;

pthread_exit(nullptr);
```


Teste de Sincronização Temporal

TESTE: Sincronização de Tempo entre Veículos

Veículos sincronizam seus relógios com o Grandmaster.

Parâmetros do teste:

Interface de rede: enp0s1
Número de veículos: 1
Número de aparições: 3
Intervalo entre aparições: 5s
Tempo de permanência dos veículos: 15s

★ Aparição #1 de 3

Vehicle ID: 1c:73:04:e1:45:bc | Offset inicial do relógio: 7757 µs

(S) Relógio atual: 2025-05-28 04:03:59.604400 UTC
(S) Relógio atual: 2025-05-28 04:04:00.604400 UTC

Rodando BMCA

Nenhum participante encontrado para BMCA
Grandmaster eleito: Vehicle ID: 1c:73:04:e1:45:bc

*(GC) Relógio atual: 2025-05-28 04:04:01.604420 UTC
*(GC) Relógio atual: 2025-05-28 04:04:02.604411 UTC
*(GC) Relógio atual: 2025-05-28 04:04:03.604415 UTC

★ Aparição #2 de 3

Vehicle ID: 6b:cf:3f:da:9a:f4 | Offset inicial do relógio: 5547 µs

(S) Offset ajustado: -1148 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -1691 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -1973 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -2116 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -2181 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -2217 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)
(S) Offset ajustado: -2238 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

(S) Offset ajustado: -2238 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

(S) Offset ajustado: -2248 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

(S) Offset ajustado: -2254 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

Rodando BMCA

Grandmaster eleito: Vehicle ID: 6b:cf:3f:da:9a:f4
(S) Relógio atual: 2025-05-28 04:04:04.604417 UTC
(S) Relógio atual: 2025-05-28 04:04:04.700339 UTC
(S) Relógio atual: 2025-05-28 04:04:05.604428 UTC
(S) Relógio atual: 2025-05-28 04:04:05.700339 UTC
(S) Relógio atual: 2025-05-28 04:04:06.604418 UTC

Rodando BMCA

Grandmaster eleito: Vehicle ID: 6b:cf:3f:da:9a:f4
*(GC) Relógio atual: 2025-05-28 04:04:06.702682 UTC

(S) Offset ajustado: 1100 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 1637 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 1906 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 1906 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2039 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2105 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2144 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2162 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2172 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2179 µs (Vehicle ID: 1c:73:04:e1:45:bc)

★ Aparição #3 de 3

(S) Offset ajustado: 2179 µs (Vehicle ID: 1c:73:04:e1:45:bc)
Vehicle ID: 85:64:d1:18:a9:de | Offset inicial do relógio: 4 µs

(S) Offset ajustado: 2179 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: -2791 µs (Vehicle ID: 85:64:d1:18:a9:de)

(S) Offset ajustado: -6968 µs (Vehicle ID: 85:64:d1:18:a9:de)

(S) Offset ajustado: -1996 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 75 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: -6787 µs (Vehicle ID: 85:64:d1:18:a9:de)

(S) Offset ajustado: -2794 µs (Vehicle ID: 1c:73:04:e1:45:bc)

Rodando BMCA

Grandmaster eleito: Vehicle ID: 85:64:d1:18:a9:de
(S) Relógio atual: 2025-05-28 04:04:09.702772 UTC
(S) Relógio atual: 2025-05-28 04:04:09.799278 UTC

Rodando BMCA

Grandmaster eleito: Vehicle ID: 85:64:d1:18:a9:de
(S) Relógio atual: 2025-05-28 04:04:10.613393 UTC
(S) Relógio atual: 2025-05-28 04:04:10.703163 UTC
(S) Relógio atual: 2025-05-28 04:04:10.802826 UTC
(S) Relógio atual: 2025-05-28 04:04:11.613371 UTC
(S) Relógio atual: 2025-05-28 04:04:11.703164 UTC

Rodando BMCA

Grandmaster eleito: Vehicle ID: 85:64:d1:18:a9:de

(S) Offset ajustado: 3849 µs (Vehicle ID: 1c:73:04:e1:45:bc)
*(GC) Relógio atual: 2025-05-28 04:04:11.809617 UTC

(S) Offset ajustado: 4575 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

(S) Offset ajustado: 1923 µs (Vehicle ID: 1c:73:04:e1:45:bc)

(S) Offset ajustado: 2649 µs (Vehicle ID: 6b:cf:3f:da:9a:f4)

(S) Offset ajustado: 4809 µs (Vehicle ID: 1c:73:04:e1:45:bc)