Sistemas Operacionais II - P6

GRUPO C - TARDE

João Vitor dos Santos
Pablo Lopes Teixeira
Fábio Henrique Antunes Coelho
Tiago Oliveira da Luz

Mudanças da Última Entrega

- Unicast na resposta com chaves MAC
  - Quando a RSU recebe mensagem de um novo Vehicle, além de setar uma TAG que precisa enviar as chaves MAC ela salva o endereço do novo Vehicle.
  - No envio das chaves MAC pela RSU, o endereço do Vehicle é enviado no Frame junto com as chaves MAC.
  - Quando um Vehicle recebe uma mensagem com chaves MAC, ele confere se o endereço recebido no Frame igual ao endereço dele.

- Mudança de Metadata para Attributes

NIC.send()

```
// VERIFY IF THE RSU RECEIVED THE MESSAGE
if(_packet_origin == Ethernet::Attributes::PacketOrigin::RSU) {
    if(_quadrant == sender_quadrant) {
        ConsoleLogger::log("RSU: Message with quadrant " + std::to_string(sender_quadrant) + " is in my quadrant " + std::to_string(_quadrant));
        if(!_vehicle_table.check_vehicle(&sender_address)) {
            ConsoleLogger::log("RSU: New vehicle found with address: " + mac_to_string(sender_address));
            std::array<unsigned char, ETH_ALEN> sender_address_array;
            memcpy(sender_address_array.data(), &sender_address, ETH_ALEN);
            _vehicle_table.set_vehicle(sender_address_array);
            _send_mac_key = true;
            memcpy(&_unicast_addr, &sender_address, ETH_ALEN);
        }
    }
```

```
if (is_local_broadcast) {
    _attribute_map_id++;
    {
        std::lock_guard<std::mutex> lock(_attribute_map_mutex);
        Ethernet::MessageInfo message_info;
        memcpy(&message_info.origin_mac, _address, ETH_ALEN);
        memcpy(&message_info.origin_id,  frame->data() + 6, 2);
        message_info.quadrant = 0;
        message_info.timestamp = 0;
        message_info.mac = 0;

        _attribute_map[_attribute_map_id] = message_info;
    }
    notify(prot, _attribute_map_id.load(), buf);
    //ConsoleLogger::print("NIC: Frame sent BROADCAST LOCAL.");
    return 0;
```

NIC.process_incoming_data()

```
if (attributes.get_packet_origin() == Ethernet::Attributes::PacketOrigin::RSU && sender_quadrant == _quadrant) {
    ConsoleLogger::log("Received RSU message");
    auto system_timestamp = attributes.get_timestamp();
    _time_keeper->update_time_keeper(system_timestamp, t);

    if (attributes.get_has_mac_keys()) {
        Address dest;
        memcpy(&dest, frame->data(), ETH_ALEN);
        if (mac_to_string(dest) == mac_to_string(_address)){
            ConsoleLogger::log("Received RSU message has MAC keys");
```

```
// VERIFY IF THE RSU RECEIVED THE MESSAGE
if(_packet_origin == Ethernet::Attributes::PacketOrigin::RSU) {
    if(_quadrant == sender_quadrant) {
        ConsoleLogger::log("RSU: Message with quadrant " + std::to_string(sender_quadrant) + " is in my quadrant " + std::to_string(_quadrant));
        if(!_vehicle_table.check_vehicle(&sender_address)) {
            ConsoleLogger::log("RSU: New vehicle found with address: " + mac_to_string(sender_address));
            std::array<unsigned char, ETH_ALEN> sender_address_array;
            memcpy(sender_address_array.data(), &sender_address, ETH_ALEN);
            _vehicle_table.set_vehicle(sender_address_array);
            _send_mac_key = true;
            memcpy(&_unicast_addr, &sender_address, ETH_ALEN);
        }
    }
```

NIC.process_incoming_data()

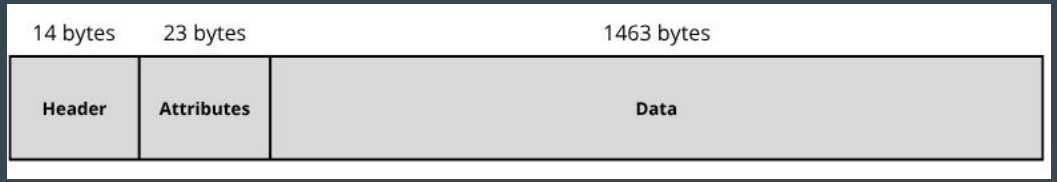
```
if(_mac_handler->verify_mac(frame->data(), payload_size, attributes.get_mac())) {
    ConsoleLogger::log("MAC verification successful");

    _attribute_map_id++;
    Ethernet::MessageInfo message_info;
    memcpy(&message_info.origin_mac, sender_address, ETH_ALEN);
    memcpy(&message_info.origin_id, frame->data() + 6, 2);
    message_info.quadrant = attributes.get_quadrant();
    message_info.timestamp = attributes.get_timestamp();
    message_info.mac = attributes.get_mac();
    {
        std::lock_guard<std::mutex> lock(_attribute_map_mutex);
        _attribute_map[_attribute_map_id] = message_info;
    }

    if (!notify(prot, _attribute_map_id.load(),buf)) {
        _attribute_map.erase(_attribute_map_id);
        free(buf);
```

Para Essa Entrega

- Informações de origem e tempo não fazem parte da mensagem em comunicação interna
  - Quando externo: cria uma struct e salva com os dados ao receber na NIC
  - Quando interno: cria uma struct e salva com zero ao receber na NIC

- É possível solicitar informações de origem e tempo das mensagens(internas e externas)
  - Map entre um ID e Attribute na NIC
  - ID é passado junto no notify() e update()
  - Pair com ID e Data nos Observers

Comunicação: Frame



- Header:
  - _from_paddr, _from_port, _to_paddr, _to_port, _lenght
- Attributes:
  - _timestamp, _sync_state, _mac, _packet_origin, _quadrant, _has_mac_keys
- Data:
  - Array de unsigned char
  - Uso:
    - Chaves MAC e Vehicle de destino das chaves MAC
    - Mensagens de resposta e interesse

Comunicação: Biblioteca

- LRU_Cache:
  - Armazena as chaves MAC
- VehicleTable:
  - Armazena os novos veículos
- Map:
  - Mapeia os Attributes com um ID
- BufferPool:
  - Armazena os frames
- List<pair>:
  - Armazena o ID do Attribute e o Dado