

INSTITUTO FEDERAL  
SANTA CATARINA  
Campus São José

INSTITUTO FEDERAL  
DE SANTA CATARINA

Curso de Engenharia de Telecomunicações

DISCIPLINA DE PROJETO DE PROTOCOLOS

---

## TAREFA: Modelar um protocolo de aplicação

**Alunos:**

Mario Allan LEHMKUHL DE ABREU

Mario Andre LEHMKUHL DE ABREU

Roicenir GIRARDI ROSTIROLLA

**Prof.:**

Marcelo MAIA SOBRAL

18 de Dezembro de 2018

Trabalho realizado para a disciplina de Projeto  
de Protocolos.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Especificações</b>	<b>1</b>
2.1	Serviços Providos pelo Protocolo	1
2.2	Vocabulário de Mensagens	2
2.3	Comportamento do Protocolo	2
2.3.1	Comportamento do Servidor	2
2.3.2	Comportamento do Cliente	3
<b>3</b>	<b>Captura de Pacotes com Wireshark</b>	<b>3</b>
<b>4</b>	<b>Interface de Acesso</b>	<b>7</b>
<b>5</b>	<b>Utilização da API</b>	<b>8</b>
<b>6</b>	<b>Considerações Finais</b>	<b>10</b>

# Lista de Figuras

1	Cabeçalho da Mensagem	2
2	Iniciando o servidor COAP na interface de Loopback no terminal do Linux.	3
3	Configurando o Wireshark para capturar pacotes na interface de Loopback.	4
4	Captura de pacotes do protocolo COAP no Wireshark	5
5	Identificando o formato da mensagem de envio - CONFIRMABLE (CON).	6
6	Identificando o formato da mensagem de resposta - ACKNOWLEDGEMENT (ACK).	7
7	Utilizando a API coapPdu.	8
8	Criando pdu coap.	9
9	Enviando pdu coap para o servidor.	10

## 1 Introdução

O segundo projeto proposto para a disciplina de Projeto de Protocolos, do curso de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina, Campus São José, ministrada pelo professor Marcelo Maia Sobra, é que os alunos apresentem uma solução de um protocolo de aplicação, bem como realizar a descrição formal do protocolo proposto. Uma lista de protocolos já existentes foi apresentada para que cada grupo tome como base para sua execução. Entre os exemplos apresentados estão:

- STOMP
- TFTP
- MQTT
- COAP

Após discussão em aula, ficou decidido que nosso grupo utilizaria como base para desenvolver este projeto o COAP. A linguagem de programação escolhida pelo grupo para a elaboração do protocolo é C++.

## 2 Especificações

As especificações do protocolo são as seguintes:

- É uma aplicação do tipo Cliente/Servidor;
- A aplicação cliente foi desenvolvida em linguagem C++;
- Utiliza as bibliotecas simplificadas de abertura de Web Socket UDP (User Datagram Protocol) para linguagem C++;
- Possibilitam envio e recebimento de mensagens;
- Para o correto funcionamento do protocolo, deve existir conectividade entre cliente e servidor;

### 2.1 Serviços Providos pelo Protocolo

O protocolo desenvolvido fornece acesso a recursos de comunicação de mensagens através de um modelo REST (Representational State Transfer). O servidor é responsável por disponibilizar os recursos e o cliente poderá então utilizá-los através dos métodos GET, PUT, POST e DELETE.

O projeto utiliza biblioteca Socket++(API simplificada) para fazer a comunicação entre cliente e servidor, então a aplicação desenvolvida monta uma PDU (Protocol Data Unit). Essa PDU contém as informações necessárias para haver a comunicação entre o cliente e o servidor,

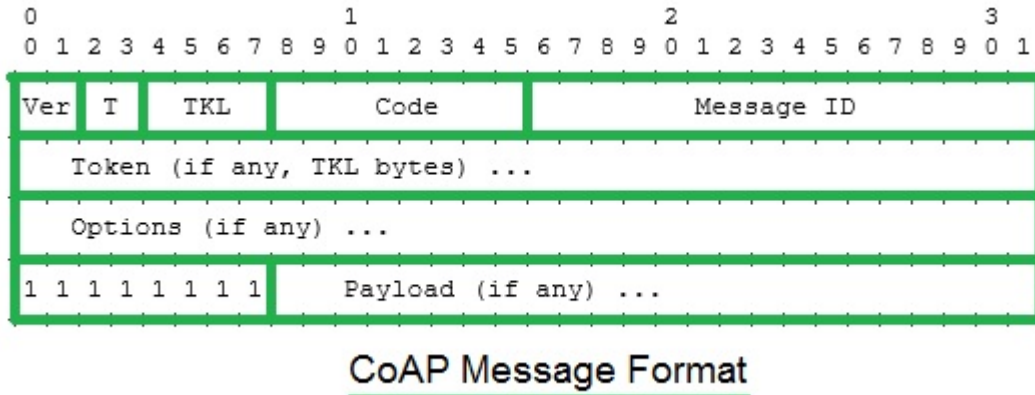


Figura 1 Cabeçalho da Mensagem

como número de versão (Ver), tipo de mensagem (T), código da mensagem (Code), ID da mensagem (Message ID), Token, Options e Payload. A figura 1 mostra o formato do cabeçalho de uma mensagem da aplicação.

## 2.2 Vocabulário de Mensagens

Originalmente, o COAP possui um extenso vocabulário, contando com mais de 30 tipos de mensagens. Entretanto, em nosso projeto implementamos apenas o GET devido a alguns problemas encontrados durante o desenvolvimento. Contudo, seria possível realizar a expansão do protocolo desenvolvido para outros tipos de mensagem, pois é necessário apenas interpretar o conteúdo do campo "CODE" na PDU da mensagem enviada/recebida.

## 2.3 Comportamento do Protocolo

### 2.3.1 Comportamento do Servidor

Para a realização de testes com a aplicação cliente, foi utilizada a biblioteca libcoap como aplicação servidora.

Primeiramente, o servidor abre um WebSocket na porta 5683 e fica aguardando conexões através dele como mostra a Figura 2. Se a porta não for informada, por padrão é utilizado a porta 5683. Quando chegar uma mensagem por ele, o servidor interpreta o conteúdo da mensagem e confirma seu recebimento. Informações como código da mensagem, campos Options e Token, ID da mensagem, tamanho e conteúdo do payload podem ser obtidas através da interpretação da mensagem recebida.

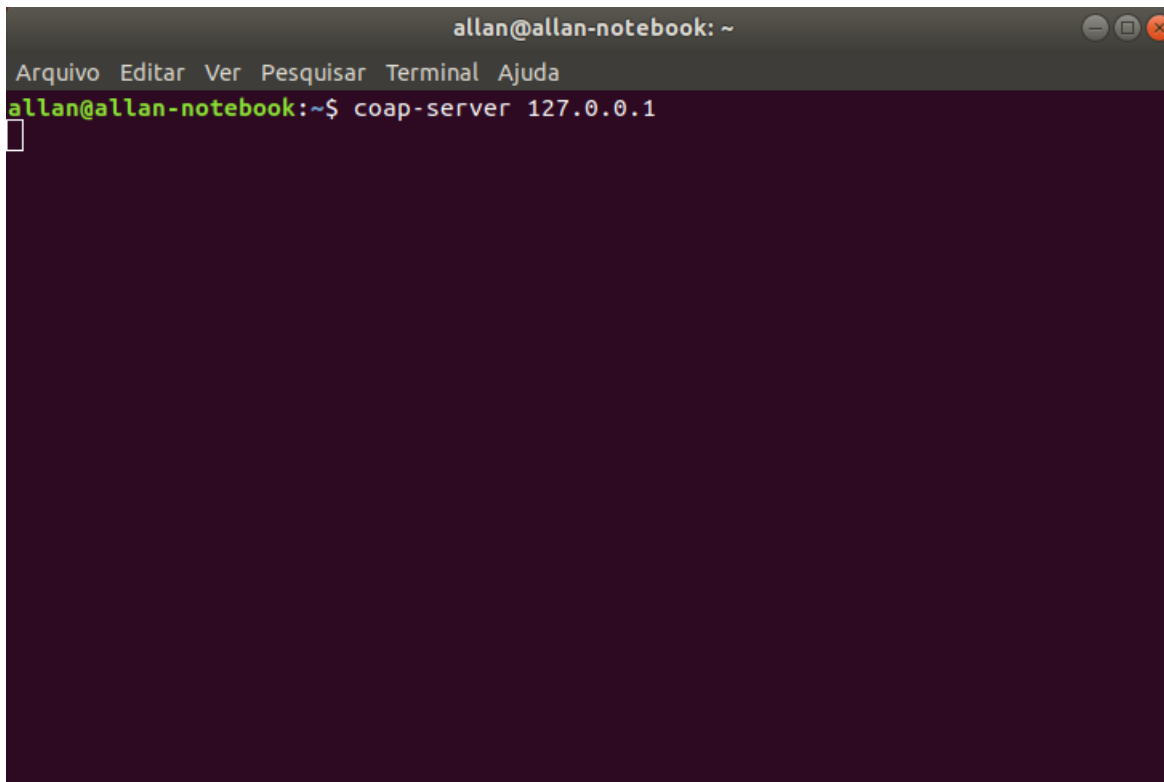


Figura 2 Iniciando o servidor COAP na interface de Loopback no terminal do Linux.

### 2.3.2 Comportamento do Cliente

O primeiro passo é montar a PDU a ser enviada a um servidor. Para isso, é necessário fazer uso da classe "CoapPdu". Com ela, é possível montar uma PDU com as informações relevantes a serem transmitidas, como o código da mensagem a ser enviada (CODE), o ID da mensagem, Payload, tamanho da PDU, campos Options e Token.

Em seguida, a aplicação cliente encaminha a PDU através de um WebSocket UDP na porta 5683. O servidor deve estar rodando, com um socket UDP aberto e aguardando conexões nessa mesma porta. O servidor deve confirmar ou não o recebimento da mensagem ao cliente, de acordo com o conteúdo do campo "T" da mensagem, o qual indica se a mensagem é confirmável ou não confirmável.

## 3 Captura de Pacotes com Wireshark

Para executar a captura de pacotes entre o cliente e o servidor COAP, é utilizado o software wireshark, que analisa o tráfego de rede e o organiza por protocolos. Assim através do wireshark é analisado a interface de Loopback (127.0.0.1), no qual é a interface utilizada neste exemplo pelo cliente e o servidor COAP.

Para analisar a interface desejada, ao abrir o wireshark, se executa a opção "capture options" (1), depois se escolhe a interface desejada que nesse exemplo é a de Loopback (2) e por fim se

inicia a análise dessa interface pelo botão "Start" (3) Esses passos são mostrados através da Figura 3.

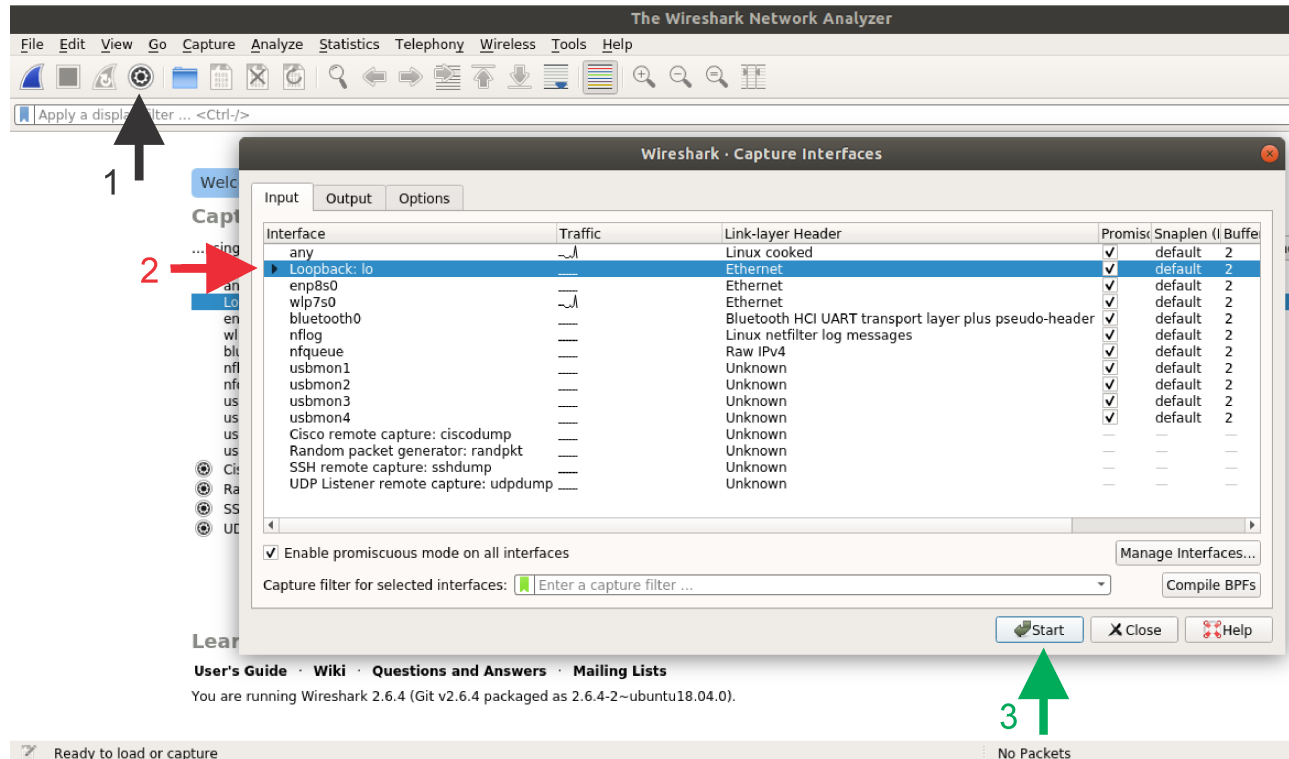
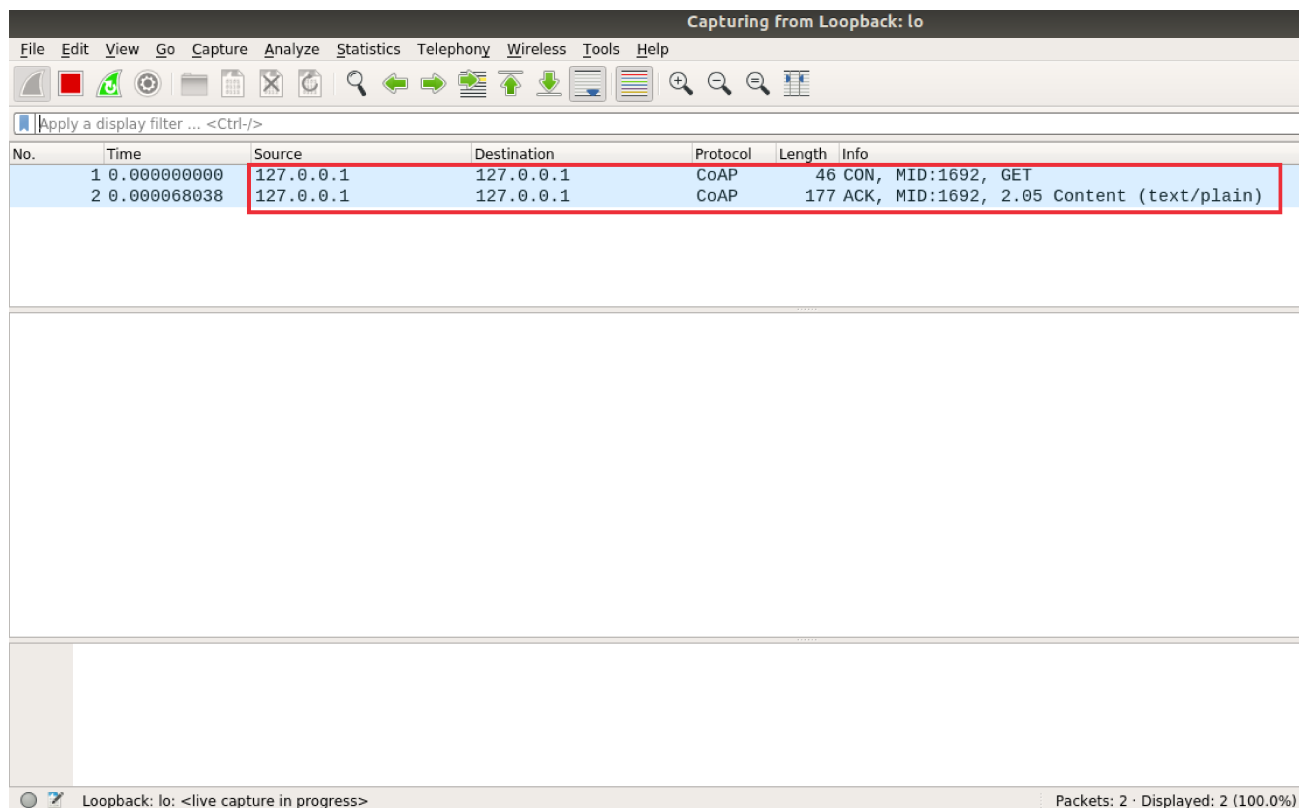


Figura 3 Configurando o Wireshark para capturar pacotes na interface de Loopback.

Iniciado a análise da interface de Loopback, quando o cliente COAP enviar um pedido de CODE igual a GET para o servidor COAP, se tudo estiver correto, o wireshark captura o pacote de pedido e também o de resposta enviado do servido para o cliente como mostra a Figura 4.



The image shows the Wireshark network protocol analyzer interface. The title bar indicates 'Capturing from Loopback: lo'. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The main packet list table contains two entries, both highlighted with a red border:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	CoAP	46	CON, MID:1692, GET
2	0.000068038	127.0.0.1	127.0.0.1	CoAP	177	ACK, MID:1692, 2.05 Content (text/plain)

Below the packet list is a large pane for packet details and packet bytes. At the bottom, the status bar shows 'Loopback: lo: <live capture in progress>' and 'Packets: 2 · Displayed: 2 (100.0%)'.

Figura 4 Captura de pacotes do protocolo COAP no Wireshark

Com a captura dos pacotes, através do wireshark, é analisado os seus conteúdos. A Figura 5, exibe a análise do pacote enviado do cliente ao servidor COAP. Na análise é possível ver o IP de origem e destino (127.0.0.1), porta de destino (5683) e ver os campos do cabeçalho da PDU do pacote, como Versão (1), Tipo (Confirmable), Token(0), CODE (GET) e Message ID (1692).



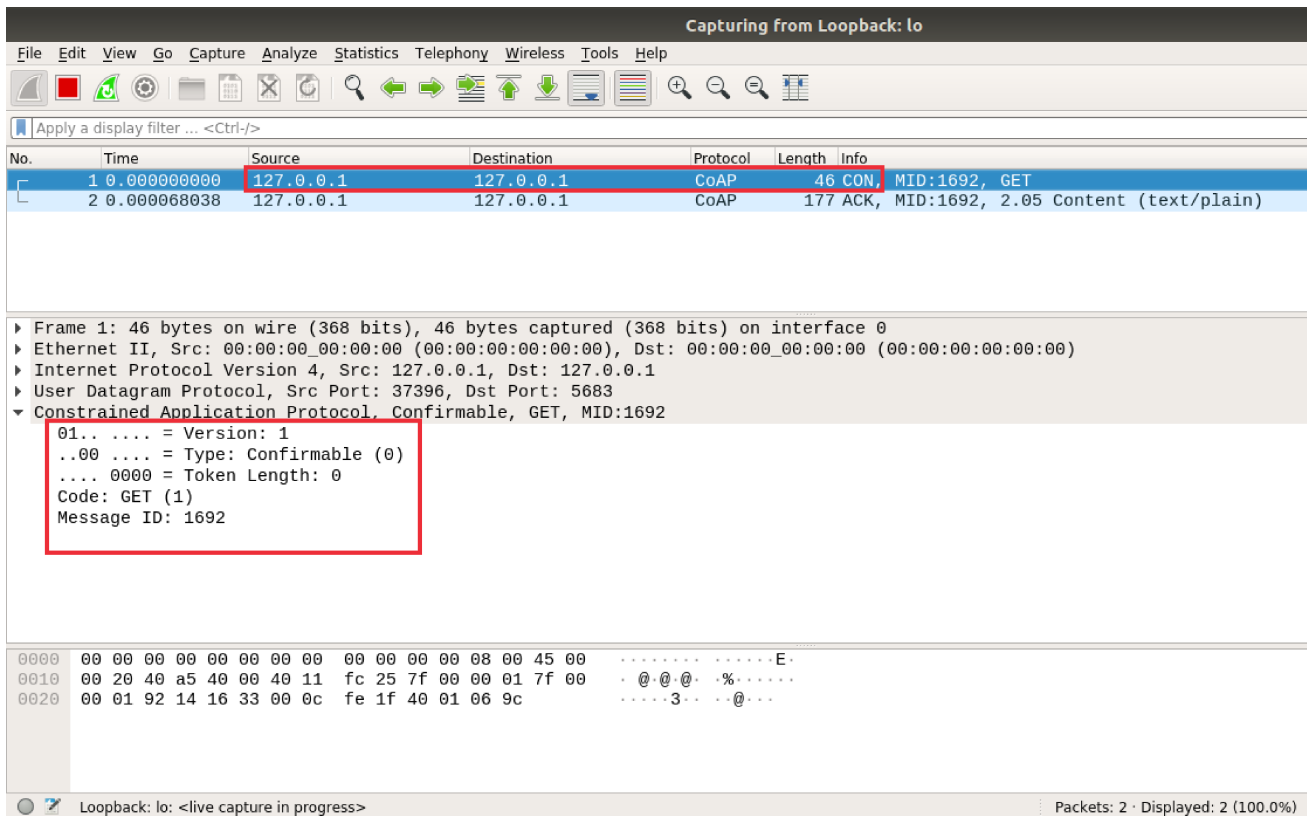


Figura 5 Identificando o formato da mensagem de envio - CONFIRMABLE (CON).

A Figura 6, exibe a análise do pacote de resposta do servidor ao cliente COAP. Na análise também é possível ver o IP de origem e destino (127.0.0.1), porta de destino (nesse exemplo 37396) e ver os campos do cabeçalho da PDU do pacote, como Versão (1), Tipo (Acknowledgement), Token(0), CODE (Content) e Message ID (1692). Mas na resposta, além dos mesmos campos da mensagem do cliente, o servidor também informa os campos de "Option" e "Payload", no qual o "Payload" repassa uma mensagem ao cliente.

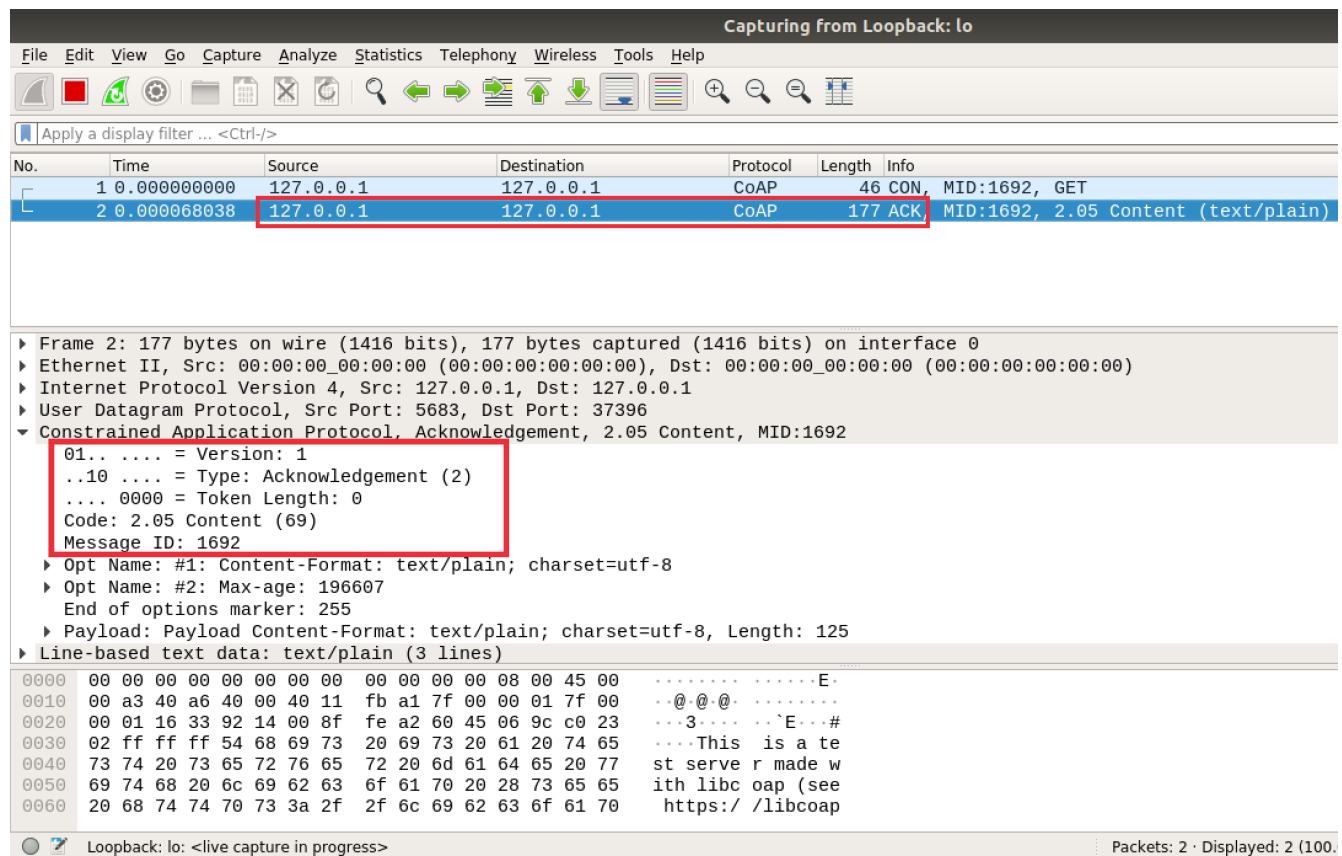


Figura 6 Identificando o formato da mensagem de resposta - ACKNOWLEDGEMENT (ACK).

## 4 Interface de Acesso

Para o usuário ter acesso a interface do protocolo é necessário apenas rodar o servidor CoAP e em seguida executar a aplicação cliente.

Para compilar o projeto basta entrar na pasta do mesmo e executar em linha de comando o seguinte:

- `g++ -o app *.cpp -std=c++11`

Em seguida, deve executar o programa da seguinte maneira:

- `./app 127.0.0.1 5683 recurso-a-ser-utilizado`

Sendo que o parametro "recurso-a-ser-utilizado" deve ser incluído caso o usuário queira solicitar algum recurso do servidor. Nesse caso se testou como exemplo o recurso "time" fornecido pelo servidor CoAP. Uma Confirmação com a resposta da solicitação é apresentada na tela.

## 5 Utilização da API

Para a utilização da API coapPdu deve-se chamar ela em sua aplicação como mostrado na figura 7.

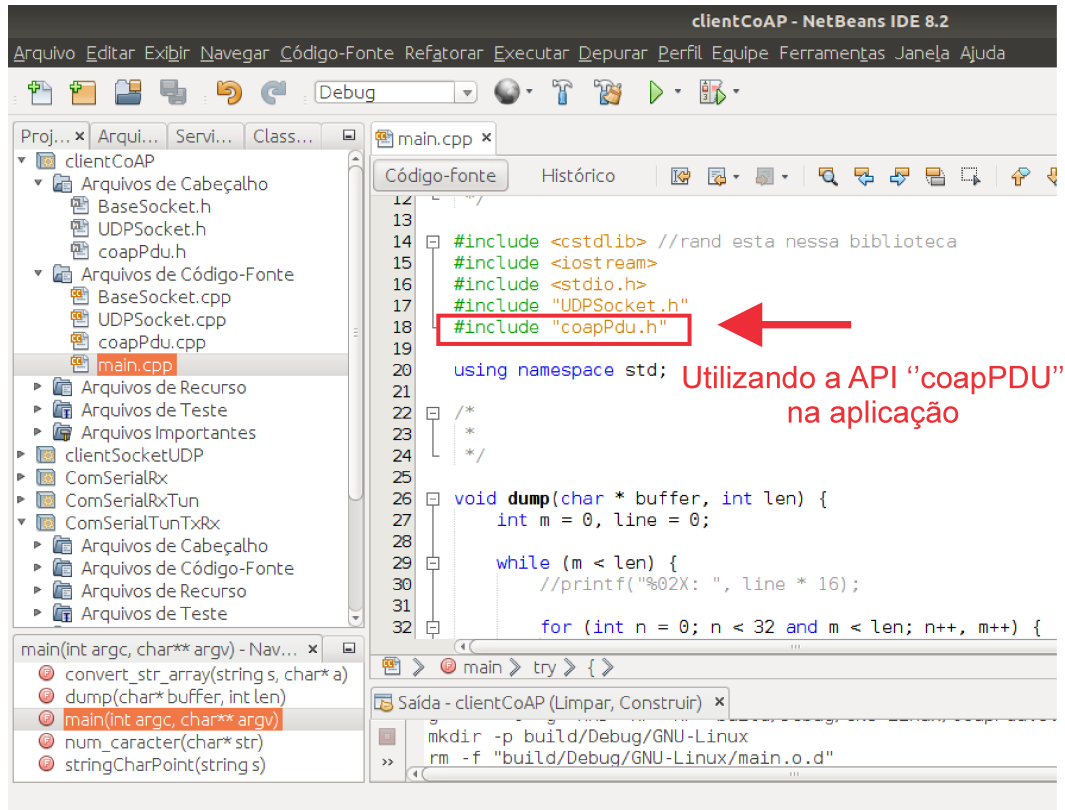


Figura 7 Utilizando a API coapPdu.

Para utilizá-la deve-se instanciar o seu objeto passando como parâmetro o tamanho da pdu coap. Depois de criado a pdu, usa-se os métodos listados abaixo para montar o cabeçalho da pdu:

- `setVersion(versao_t versao)` - Configura o campo versão da pdu. Deve-se passar um argumento do tipo enum já fornecido pela API.
- `setType(type_t type)` - Configura o campo Tipo da pdu. Deve-se passar um argumento do tipo enum já fornecido pela API.
- `setTLK(tlk_t tlk)` - Configura o campo Tamanho do Token da pdu. Deve-se passar um argumento do tipo enum já fornecido pela API.
- `setCode(code_t code)` - Configura o campo Code da pdu. Deve-se passar um argumento do tipo enum já fornecido pela API.
- `setMsgID()` Configura o campo mensagem ID da pdu. Esse método gera um número aleatório para esse campo.

caso se utilize um recurso do servidor usa-se o seguinte método:

- `setURI(char * uri, int tamanhoUri)` - Utilizar um recurso. Nessa implementação só funciona com um único parâmetro. EX: `time`; Ex2: `.well-known/core` (não vai funcionar).

Essas ações são mostradas na figura 8.

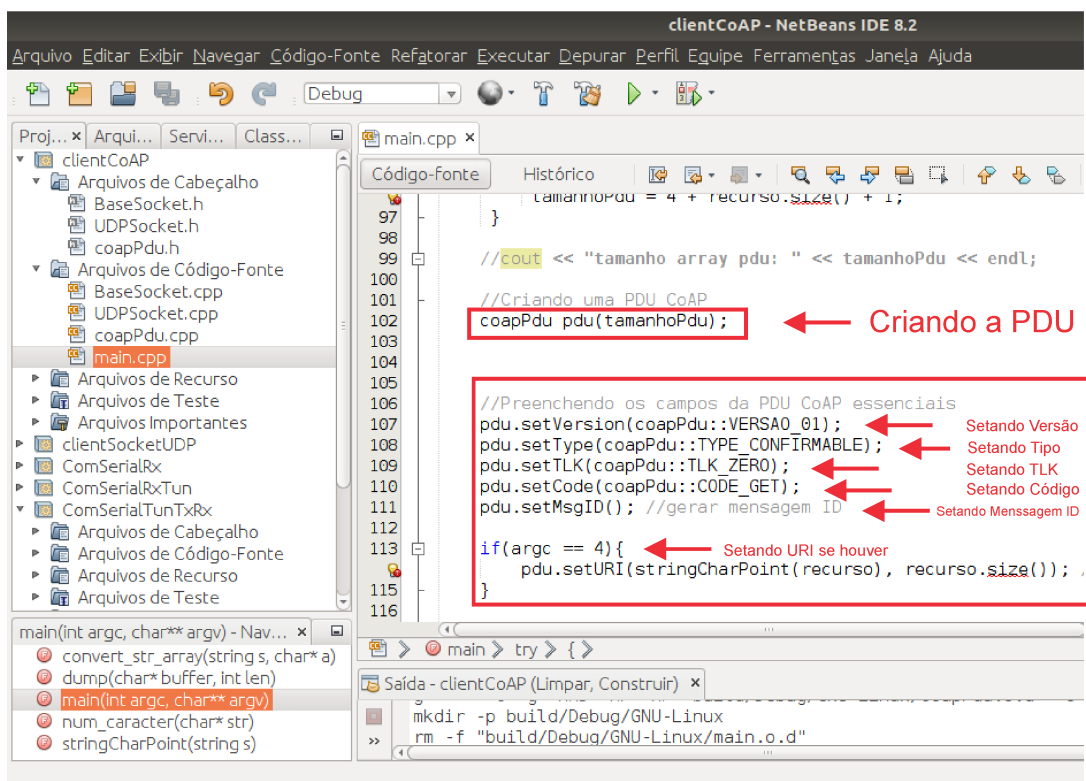


Figura 8 Criando pdu coap.

Com a pdu montada usa-se o método `getPointerCoapPdu()` para pegar o objeto da pduCoap e usar na API `SocketUDP` para enviar pro servidor, como mostrado na figura 9.

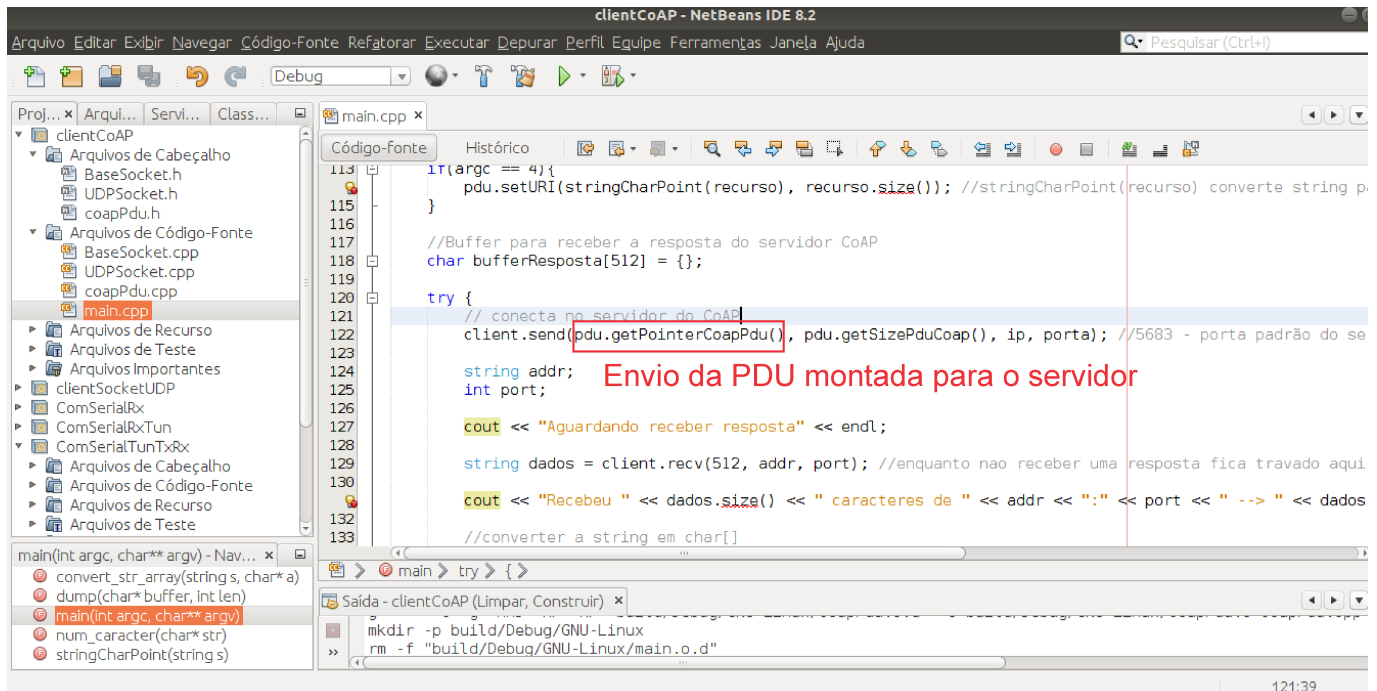


Figura 9 Enviando pdu coap para o servidor.

## 6 Considerações Finais

Conforme já mencionado anteriormente, o CoAP possui um vocabulário de mais de 30 mensagens diferentes, mas para esse projeto se especificou a utilização das mensagens utilizadas pelo cliente COAP, que são GET, PUT, POST e DELETE. No entanto neste trabalho foi implementado apenas a mensagem GET, devido a alguns problemas encontrados durante o desenvolvimento. Entretanto mesmo só conseguindo implementar a mensagem GET, pode-se conseguir compreender o funcionamento básico do protocolo CoAP.

Durante o desenvolvimento se verificou o protocolo, sua estrutura que é composta por modelo de mensagem e modelo de solicitação/resposta e como é feito o formato da mensagem. Além disso também se conseguiu fazer uma demonstração para o entendimento do protocolo e se conseguiu implementar uma API independente da aplicação.

## Referências

- [1] PTC29008: Projeto 2: Protocolo de aplicação, publicado em: = [https://wiki.sj.ifsc.edu.br/wiki/index.php/ptc29008:\\_projeto\\_2:\\_protocolo\\_de\\_aplica%c3%a7%c3%a3o](https://wiki.sj.ifsc.edu.br/wiki/index.php/ptc29008:_projeto_2:_protocolo_de_aplica%c3%a7%c3%a3o), note = Acessado em: 2018-08-07.
- [2] The Constrained Application Protocol (CoAP), publicado em: = <https://tools.ietf.org/html/rfc7252>, note = Acessado em: 2018-12-18.