UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Rafael de Lucena Valle

IMPLEMENTAÇÃO DO PROTOCOLO COAP PARA SERVIÇOS DE MONITORAMENTO EM REDES DE SENSORES SEM FIO

Florianópolis

2014

RESUMO

Redes de sensores são utilizadas para a captura e processamento de informação para atuação sobre um ambiente, tornando-as importantes em aplicações de controle, telemetria e rastreamento.
Estas redes são compostas por nós sensores que possuem processadores, transmissores e receptores simplificados, restrições de memória e energia, que trabalham em conjunto afim obter dados de um ambiente. Contudo possuem um custo baixo de equipamentos, tornando interessante a implantação destes sistemas.
O protocolo HTTP, um protocolo de aplicação muito utilizado na atualidade, foi desenvolvido para computadores de propósito geral, onde essas restrições não existem. Um protocolo leve como CoAP pode tornar viável o desenvolvimento de aplicações web em redes de sensores sem fio.
Este trabalho propõe uma infraestrutura de comunicação entre redes de sensores sem fio e a Internet, utilizando protocolos leves entre os nós sensores e um gateway GPRS para aproveitar a cobertura da tecnologia GPRS.
Com a implementação do CoAP é esperado uma redução de consumo de energia e memória, em relação a outros protocolos de aplicação existentes.

Palavras-chave: internetworking WSN IPv6 GPRS CoAP IoT

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CoAP	Constrained Aplication Protocol
EPOS	Embedded Parallel Operating System
HTTP	Hipertext Transfer protocol
IETF	Internet Engineering Task Force
M2M	Machine-to-Machine
REST	Representatational State Transfer
UDP	User Datagram Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
XML	Extensible Markup Language
ADESD	Application Driven Embedded System Design
SOC	Service-Oriented Computing

SUMÁRIO

1 INTRODUÇÃO

Redes de sensores e atuadores são utilizadas para a captação, processamento de informação e atuação sobre um ambiente, tornando-as importantes em aplicações de controle, telemetria e rastreamento de sistemas.

Nós que participam destas redes geralmente são computadores e rádios simplificados, que possuem restrições de memória, processamento, energia e capacidade de comunicação, mas um custo relativamente baixo de equipamentos.

O maior consumo de energia neste tipo de aplicação é do rádio, portanto os desafios dos algoritmos de comunicação nesta área são manter os rádios ligados o mínimo de tempo possível sem comprometer a conectividade do nó.

1.1 OBJETIVOS

O objetivo geral desse trabalho é descrever e implementar webservices em uma rede sensores sem fio, que farão a aquisição dos dados do ambiente e disponibilizarão as informações captadas na Internet.

1.1.1 Objetivos Específicos

Este trabalho realizará o desenvolvimento de aplicações web e software de sistema para fazer a ponte entre a rede de sensores e a Internet. O Sistema operacional utilizado será o EPOS, Embedded Paralell Operating System. A aplicação integradora GPRS/802.15.4 irá executar na plataforma EposMoteII utilizando uma extensão GPRS, desenvolvida por uma colega de laboratório.

A comunicação entre os nós sensores e atuadores será feita através do protocolo de aplicação CoAP - Constrained Application Protocol. Um protocolo específico para redes de sensores sem fio. Será utilizado um porte de uma implementação livre do protocolo CoAP.

Sendo assim, os objetivos específicos são: 1. Portar o protocolo CoAP para o EPOS; 2. Implementar uma aplicação de redes de sensores sem fio; 3. Desenvolver a aplicação gateway GPRS/802.15.4.

1.2 JUSTIFICATIVA

Os mecanismos de confiabilidade na transmissão de dados, técnicas para se manter uma conexão do TCP e rearranjos que são feitos para garantir a ordem das mensagens recebidas não são adequados para um dispositivos com suprimento limitado de energia, como uma bateria ou uma placa fotovoltáica. Estas técnicas fazem que os transmissores fiquem ligados por mais tempo, para manter a conexão ou até mesmo para reenvio de mensagens. O maior consumo de energia de um nó sensor é no envio e recebimento de dados, quando mantem seu transmissor ligado. Além quem recebe a mensagem precisa montá-la e tratar as partes corrompidas.

Por sua vez o protocolo do UDP, não mantém conexão, dados são recebidos fora de ordem e o envio é feito de uma mensagem por vez. Isto implica também na redução do tamanho do cabeçalho do pacote.

Estas características demostram uma alternativa interessante para estes equipamentos limitados. Testes feitos em implementações de sistemas operacionais similares ao EPOS, como Contiki e TinyOS, utilizando o protocolo CoAP demonstram redução no consumo de energia e memória em relação ao HTTP.(KULADINITHI K., BERGMANN O., PÖTSCH T., BECKER M., GORG C., 2011)

A falta de padronização dos protocolos afeta o desenvolvimento de uma rede pública ubíqua de uma cidade inteligente por exemplo. Grande parte das soluções utiliza protocolos proprietários, que se comunicacam apenas com os produtos de um mesmo fabricante.

O protocolo HTTP foi desenvolvido para comunicação de computadores de propósito geral, onde as restrições citadas não são comuns. Em relação ao tamanho, o pacote HTTP é um problema para redes ZigBEE, já que estas redes que possuem uma restrição de 128 bytes. O protocolo TCP precisa transmitir mensagens adicionais para manter uma conexão, outra característica que não é interessante para RSSF.

Um protocolo leve como CoAP pode tornar viável a criação de aplicações web em redes de sensores sem fio por um baixo custo. Neste trabalho é proposto uma infraestrutura de comunicação entre redes de sensores sem fio e a Internet, utilizando protocolos leves entre os nós sensores e um gateway GPRS para áreas sem acesso à WIFI, aproveitando a vasta abrangência da tecnologia de telefonia. Com a utilização do CoAP é esperado uma redução de consumo de energia e memória, em relação a outros protocolos de aplicação existentes.

Em lugares aonde não existe o acesso a rede cabeada ou sem fio, como lugares afastados, na área rural, por exemplo a distribuição da informação para Internet será feita através de um gateway.

O gateway será composto por um EposMoteII e um módulo GPRS, responsável por fazer a ponte entre a rede de sensores e a Internet. Atualmente o padrão GPRS oferece a maior cobertura dentre as tecnologias de transmissão de telefonia no Brasil, atingindo cerca de 5477 municípios.(**??**)

1.3 METODOLOGIA

Será feito um levantamento dos componentes de software e hardware necessários para o desenvolvimento do gateway 802.15.4/GPRS. Neste caso utilizando o mote EPOSMote II e um módulo GPRS, que será desenvolvido em paralelo a implementação do protocolo de aplicação CoAP no sistema operacional EPOS.

Durante o desenvolvimento do protocolo testes serão executados para verificar o correto comportamento e diminuir a depuração em hardware, que geralmente leva mais tempo. Citation Needed

Nos testes de integração do gateway, será utilizada uma placa de desenvolvimento em conjunto com um módulo M95 da Quectel disponibilizada pelo Laboratório. Os testes de envio de mensagens em diversos protocolos, inclusive testes com comandos proprietários adicionais do modem.

Para testes de integração, as aplicações serão executados na plataforma de sensores sem fio EPOS Mote II utilizando o EPOS com o CoAP desenvolvido.

2 REVISÃO BIBLIOGRAFICA

Ambientes inteligentes estão tornando-se realidade, este conjunto de tecnologias que estão sendo desenvolvidas serão um grande passo para as áreas como: construção civil, industria, transporte e automação residencial.

Para tomar decisões e agirem de forma inteligente, estes ambientes necessitam de informacção sobre o contexto aonde serão implantados. As redes de sensores sem fio são um componente chave para estes ambientes, adquirem este contexto captando e comunicando os dados do ambiente.(LEWIS F., 2004)

Este capítulo apresenta uma visão geral sobre redes de sensores sem fio, arquitetura orientada a serviços e os protocolos de aplicaçao existentes.

2.1 REDES DE SENSORES SEM FIO

Avanços recentes nas tecnologias de sistemas eletrônicos, semicondutores, sensores, microcontroladores e rádios tornaram possível o desenvolvimento de redes de sensores de baixo custo e baixo consumo uma realidade.

Sensores ligam o físico com o mundo digital, capturando e revelando fenômenos do mundo real e convertendo em uma forma que pode ser processada, armazenada e utilizada para tomada de decisão. A escolha entre qual sensor a ser utilizado depende muito da aplicação e da propriedade fsica a ser monitorada, alguns exemplos: temperatura, pressão, humidade, entre outros.

Um nó sensor de uma rede de sensores sem fio não é apenas responsável por coletar dados, mas análise da rede, correlação entre dados de outros sensores e comunicação com uma estação base para centralizar a informação. Isto é importante pois geralmente tais redes são compostas por muitos nós. (DARGIE, POELLABAUER, 2010)

Geralmente tais redes possuem centenas ou milhares de nós sensores e possuem as seguintes características: pouca memória, pouco alcance do rádio, baixa capacidade de processamento e bateria, e custo reduzido. Comunicam-se entre si e com estações base utilizando seus rádios sem flo, permitindo disseminação da informação para processamento remoto, visualização, análise e armazenamento.

Os requisitos para uma rede de sensores distribuída são: reconfiguração com estação base, controle autonômo de operação e gerência de energia, auto-monitoramento, eficiência energética para longo tempo de operação e apta a incorporar diversos sensores.(BULT, CHANG, DONG, FIELDING, 1996)

Um nó pertencente a esta rede geralmente é um dispositivo especificamente desenvolvido para um próposito, que possui poucos recursos computacionais e energéticos e se comunicam entre seus semelhantes, como apresentado na figura 2.1.

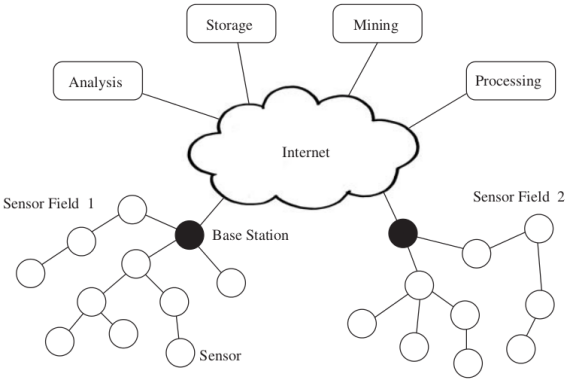

Figura 1 – Overview de uma arquitetura de redes de sensores sem fio retirado de (DARGIE, POELLABAUER, 2010).

Muitas redes de sensores também possuem atuadores, que permitem controle direto ao mundo real. Uma rede de sensores e atuadores recebe comandos da unidade de processamento (microcontrolador) e transforma estes comandos em sinais de entrada para um atuador, que interage com processos fśicos. Estas redes são chamadas de redes de sensores e atuadores sem fio, ou WSAN.

A conservação de energia é um dos objetivos das redes de sensores sem fio, pois não estão ligados diretamente a fonte de energia. Deve-se minimizar o consumo em todos os níveis do sistema, da aplicação até o meio físico, iniciando com o projeto de rádio. (YICK, MUKHERJEE, GHOSAL, 2008)

2.1.1 Plataforma de Hardware

2.1.2 Sistema Operacional

2.1.3 Rede de interconexão

2.1.4 Aplicações

Existe um enorme número de aplicações que pode ser desenvolvido graças as tecnologias de WSN.

2.2 ARQUITETURA ORIENTADA A SERVIÇOS

Arquitetura orientada a serviços é uma forma de organizar infraestrutura e aplicações de software em um conjunto de serviços. Estes são oferecidos por prestadores de serviço, servidores, organizações que implementam os serviços, fornecem descrição dos servicços oferecidos, suporte técnico e de negócio.

O modelo de computação utilizando esta afirmativa é conhecido como Computação Orientada a serviços (SOC). (PAPAZOGLOU M., 2003)

Clientes destes serviços podem ser outras soluções, aplicações, processos ou usuários. Para satisfazer estes requisítos serviços devem:

- Tecnologicamente neutros: utilizar-se de padrões reconhecidos e bem aceitos para comunicação, descrição e mecanismos de descoberta;

- Baixo acoplamento: detalhes desnecessários (o quão desnecessário precisa ser discutido) devem ser escondidos do cliente, que não precisa ter conhecimento sobre o funcionamento interno para utilizar o serviço;

- Localidade transparente: clientes devem ser atendidos independentemente da localidade do serviço disponível.

SOA não é apenas uma arquitetura sobre serviços, mas um relacionamento entre três entidades: o provedor de serviç (service provider), descoberta de serviço (service discovery agency) e o client (service requestor). Abaixo a figura 2.2 demonstra este relacionamento e suas interações: publicar (publish), encontrar (find) e vincular (bind).

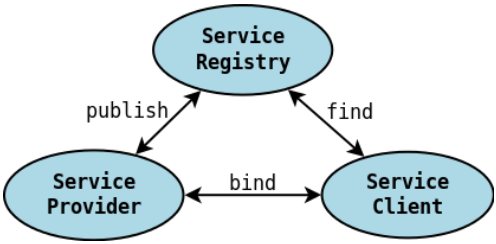

Figura 2 – Arquitetura Orientada a Serviços

2.3 WEBSERVICES

Um serviço web é um sistema de software projetado para suportar interoperabilidade em interações máquina-a-máquina na rede. Possui uma interface descritiva das funcionalidades num formato padronizado (especificamente WSDL). Esta notação abstrata que deve ser implementada por um agente.(W3C Working Group, 2004).

O agente é algo concreto, pode ser um pedaço de hardware ou software que recebe e envia mensagens. Um exemplo é implementar o mesmo serviço web, utilizando agentes em diferentes linguagens. Embora o agente seja diferente, o serviço Web continua o mesmo.

Serviços Web provêem um padrão para a interoperabilidade entre diferentes aplicações de software, que executam em diferentes plataformas de software e hardware.

2.4 RESOURCE ORIENTED ARCHITECTURE

ROA é uma arquitetura boa para RESTful webservices. (RICHARDSON L., RUBY S., 2008)

2.5 RESTFUL

Uma requisicção a um webservice RESTful, utiliza a informação do método como um verbo HTTP e a informacção do escopo ao qual o verbo será utilizado na URI. Ao contrário um estilo RPC tende a ignorar o método HTTP, procurando pelo método a ser utilizado e o escopo na própria URI.

Um serviço web que utiliza HTTP e os princípios REST possui recursos e ações genéricas bem definidas.(FIELDING R., 2000)

Para transfeência de dados utiliza-se formatos genéricos que enfatizam simplicidade e usabilidade pela internet, como XML e JSON.

Os Recursos são usam um identificador único e persistente, as URIs. A URIs possuem estruturas de diretórios, uma URI é uma árvore com ramos subordinados e superordinados conectando os nós. As operações suportas são métodos HTTP explícitos que não salvam estado das aplicações clientes e são idempotentes, são eles: GET: solicita ao webserver a representação de uma informação de um determinado recurso. POST: criar um recurso no webserver. PUT: mudar o estado de um recurso do webserver. DELETE: remover o recurso ou alterar para um estado vazio.

Uma abordagem utilizando SOAP RPC em HTTP não é interessante para uma aplicação de RSSF, já que a quantidade de informação a ser transmitida é consideravelmente maior. Além disso, a aplicação teria que conhecer lógica interna do serviço istrumentando o recurso utilizando chamadas de funções remotas. A figura 2.5 exemplifica e demonstra a diferença de uma aplicação que faz uso de SOAP RPC e outra RESTful.(RICHARDSON L., RUBY S., 2008)

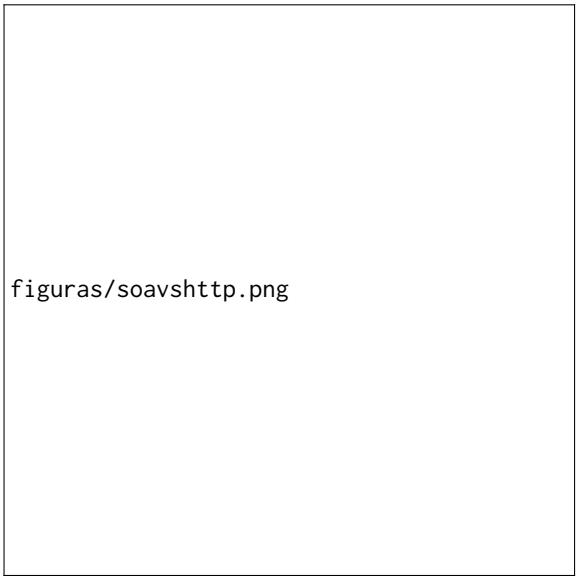

Figura 3 – Imagem retirada de (RICHARDSON L., RUBY S., 2008)

A figura 2.5 faz um comparativo entre o número de bytes transmitidos de diversos servidores web e seus protocolos.

Todos os webservices utilizam o conceito de URI, porém de maneiras

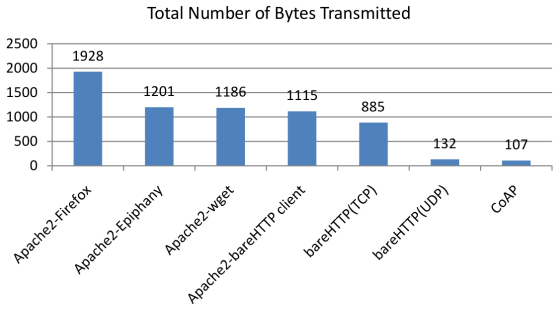

Figura 4 – Imagem retirada de (KULADINITHI K., BERGMANN O., PÖTSCH T., BECKER M., GORG C., 2011)

diferentes. Mas um serviço RESTful sempre expõe a URI para cada pedaço de dado que o cliente quer operar sobre.

Os principais competidores aos serviços web RESTful são os RPC-styles.

2.6 COAP

Um dos principais objetivos do CoAP é ser uma alternativa protocolo web genérico para redes com dispositivos com restrição de energia e memória.

As vantagens de utilizar um protocolo compatível com o HTTP são: a facilidade de integração e o reuso de aplicações. CoAP é um conjunto REST otimizado para M2M, com suporte a descoberta de recursos, multicast e troca de mensagens assíncronas com simplicidade e baixo overhead.

A IETF estabelece as condições mínimas para o desenvolvimento de um protocolo de aplicação compatível com HTTP, mas focado em aplicações aonde energia e memória são escassas. O protocolo CoAP foi projetado levando em consideração as restrições energéticas e altas taxas de falha na transmissão dos pacotes em RSSF.

A comunicação entre os pontos no CoAP é de forma assíncrona usando o UDP. A confiabilidade é um parâmetro opcional e funciona através de um mecanismo de retransmissão exponencial. Possui 4 tipos de mensagem: Confirmável, Não-Confirmável, Confirmação (ACK) e Reset. A figura 2.6.1 mostra o formato do pacote.

2.6.1 Formato das mensagens

Uma mensagem CoAP deve caber num único pacote IP, para que seja transmitida numa camada de enlace limitada.

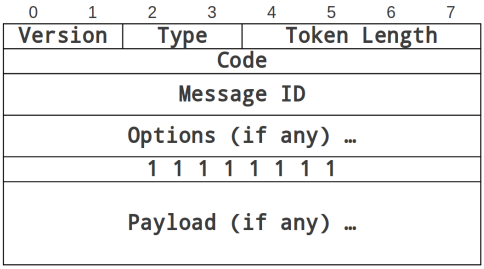

Figura 5 – formato do pacote CoAP em bits. (SHELBY Z., HARTKE K., BORMANN C., 2013)

Os campos do pacote CoAP são: a versão do CoAP, implementações devem utilizar este campo com o valor 1. O tipo: campo para definir o tipo da mensagem: Confirmável (0), Não-Confirmável (1) , de Confirmação (2) ou Reset (3).

O tamanho do Token: utilizado para controle de requisições e repostas. O tamanho do Token pode variar entre 0 e 8 bytes. Tamanhos entre 9 a 15 são reservados e não devem ser usados. É um campo sempre gerado pelo cliente CoAP.

O Código: separados em 3-bit mais significativos para classes e 5-bits menos significativos para detalhe. As classes podem indicar uma requisição (0), uma resposta de sucesso (2), e uma resposta de erro do cliente (4), ou uma resposta de erro do servidor (5), as outras classes são reservadas. Em um caso especial o código 0.00 indica uma mensagem vazia.

O ID da mensagem: usada para deduplicação de mensagens e confirmação ou reset de mensagens. É gerado por quem envia a mensagem, no caso de uma mensagem confirmável ou reset, a resposta deve possuir o ID da mensagem enviada. A implemetação da geração dos IDs está aberta, depende da aplicação que o CoAP será usado, porém é recomendado que o valor inicial seja randômico.

2.6.2 Transmissão de Mensagens

A transmissão de mensagems é controlada basicamente pelos parâmetros: ACK TIMEOUT, ACK RANDOM FACTOR, MAX RETRANSMIT, NSTART, Leisure e PROBING RATE.

Estes parâmetros são respectivamente: o tempo que uma mensagem confirmável aguarda o ACK; fator de randomicidade para gerar os ACK TIMEOUTs subsequentes; contador para o número máximo de tentativas de retransmissão; número limite de interações simultâneas mantidas por um servidor.

A Leisure é o tempo que o servidor aguarda para responder uma requisição multicast, é calculada: $Leisure = S * G/R$. Aonde S é o tamanho estimado da reposta, G é uma estimativa do tamanho do grupo e R é a taxa de transmissão. PROBING RATE: é a taxa média para transmissão de dados.

Estes parâmetros definem a temporização do sistema. Os valores padrões são mostrados na Tabela 2.6.2.

Nome	Valor padrão
ACK timeout	2 segundos
ACK random factor	1.5
NStart	1
Default Leisure	5 segundos
Probing rate	1 Byte/segundo
Max retransmit	4

Tabela 1 – Valores padrão do CoAP.

A retransmissão é controlada por um timeout e um contador. Quando este timeout é atigido e o contador é menor que valor máximo de retransmissão a mensagem é transmitida, o contador incrementado e timeout duplicado. O modelo de retransmissão usa um contador de timeouts e uma função que varia de acordo com o número de tentativas.

Uma falha na transmissão ocorre quando atingir o número máximo de tentavivas ou receber uma mensagem de RESET. Quando receber um ACK a transmissão da mensagem confirmável é completa.

O servidor irá ignorar mensagens que chegam por multicast quando não puder responder nada de útil.

Na situação aonde possuir uma informação suficientemente nova pode responder na própria mensagem de confirmação (ACK). Essa técnica é chamada de ”Piggy-backed” um mecanismo de transmissão para mensagens con-

firmadas, o cenário é ilustrado na Figura 2.6.2.(SHELBY Z., HARTKE K., BORMANN C., 2013)

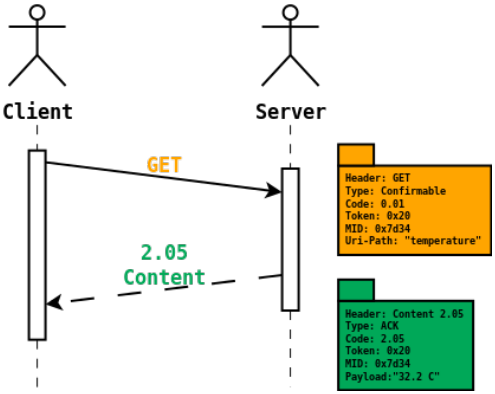

Figura 6 – Resposta na mensagem de confirmação, chamado de piggy-backed.

Fluxo esperado de requisição sem confirmação na figura 2.6.2.

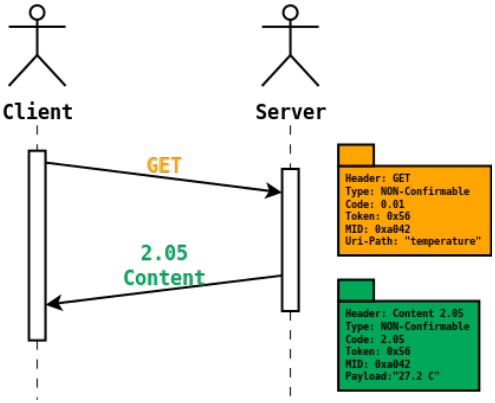

Figura 7 – Fluxo esperado de requisição e resposta sem confirmação

A RFC também prevê fluxo de requisição com confirmação, e resposta separada com confirmação. A figura 2.6.2 exemplifica:

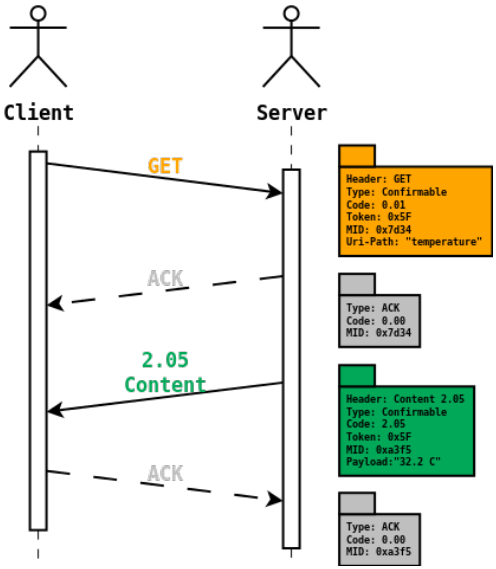

Figura 8 – Fluxo esperado de requisição e resposta com confirmação, com resposta separada

2.6.3 Camada de Reposta e Requisição

Uma requisição é inicializada ao preencher o campo code no cabeçalho do CoAP. Possuem as mesmas propriedades de idempoténcia e only retrieved das requisições HTTP.

2.6.4 Recursos

A descoberta de recursos é feita quando um servidor recebe uma requisição GET para o recurso /well-know/core. O servidor CoAP deve responder no formato CORE link Format.(SHELBY Z., 2012) E a descoberta de serviços no protocolo CoAP é feita através de socket Multicast. Os recursos são identificados por uma URI, e os métodos são implementados de forma similar ao HTTP.

A desenvolver...

2.7 EPOS

O EPOS é um sistema operacional multithread com suporte a preempção, foi desenvolvido em C++ que faz uso intenso de programação orientada a aspectos utilizando templates.

Possui abstrações para entidades temporais como relógio, alarme e cronômetro, biblioteca com estruturas de dados e sequenciadores. Permitindo o uso de ferramentas para geração automatizada de abstrações de sistemas. A portabilidade é atingida utilizando entidades chamados de Mediadores de Hardware que fornecem interfaces simples para acesso as funções específicas de arquitetura. Estas interfaces são utilizadas por entidades abstratas como alarmes e threads periódicas. A figura 5 mostra uma visão abstrata da arquitetura do EPOS.

Foi projetado utilizando ADESD, Application Driven Embedded System Design, um método para projeto de sistemas embarcados orientados à aplicação. Esta metodologia guia o desenvolvimento paralelo de hardware e software além de manter portabilidade. O EPOS possui porte para as seguintes arquiteturas: MIPS, IA32, PowerPC, H8, Sparc, AVR e ARM. (FRÖHLICH A., 1999)

Utiliza um sistema de construção baseado em makefiles e shell scripts.

2.8 TRABALHOS RELACIONADOS

2.8.1 OpenMote

2.8.2 OpenWSN

O projeto OpenWSN é um projeto de visa o código aberto e confia na comunidade para manter atualizado e encontrar erros. Possui uma implementação completa de código aberto, das pilhas de protocolos padrões pra Internet das Coisas. E com a primeira implementação aberta do padrão 802.15.4e, Time Synchronized Channel Hopping.

Por serem sincronizados os motes precisam acordar apenas para transmitir ou receber e periodicamente se comunicar para manter a rede sincronizada quando estiver inativo. Este overhead de temporização é pequeno, cerca de 0.02% de ciclo de trabalho do rádio. (WATTEYNE, VILAJOSANA, KERKEZ, 2012)

O pilha de protocolos é totalmente em C e pode ser compilada em qualquer toolchain que suporte uma plataforma alvo. OpenWSN já foi por-

tado para as seguintes arquiteturas: ARM, AVR, ...

Possui porte para diversas plataformas de microcontroladores de 16 bits e para as arquiteturas mais novas de 32 bits dos cortex-M. O projeto possui diversas ferramentas para depuração, simulação e ambiente necessário para integrar as aplicações a Internet.(**??**)

Resultados experimentais de uma rede de motes demonstram que os rádios operam num ciclo médio de trabalho de 0.1% e uma média de 0.68 microA em hardware comum. Este consumo baixo permite uma vasta gama de aplicações.

Para utilizar é necessário baixar os seguintes repositórios:

- Software de sistema que irá executar no simulador ou nos motes: https://github.com/openwsn-berkeley/openwsn-fw
- Ferramentas que irão executar no computador de desenvolvimento: https://github.com/openwsn-berkeley/openwsn-sw
- Implementação do CoAP em Python: https://github.com/openwsn-berkeley/coap

2.8.3 Contiki

O Contiki é um sistema operacional criado por Adam Dunkels em 2000, escrito em C, de código aberto para sistemas com restrição de recursos comunicam-se numa rede. Foi desenvolvido para ser um sistema operacional para Internet das coisas. Possui uma camada de abstração RESTful para web services chamada Erbium, que implementa o protocolo CoAP.

Cada processo no Contiki possui bloco de controle, que contém informações de tempo de execução do processo e uma referência para uma protothread, na qual o cógido é armazenado na ROM.

Protothread é uma combinação entre eventos e threads, possuem comportamentos de bloqueio e espera, que permite o intersequenciamento dos eventos, gerando um baixo overhead de memória por não necessitar de salvamento de contexto.

Cada protothread consome 2 bytes de memória, que são utilizados para armazenar a continuidade local, uma referencia utilizada em um pulo condicional durante a execução da thread. É um método similar ao mecanismo de Duffy e Co-rotina em C. (**??**)

O transceiver sem fio é um dos componentes que mais consome energia quando ligado escutando o ambiente, assim uma das estratégias utilizadas é manter o mínimo de tempo possível ligado, mas o suficiente para manter a troca de mensagens na rede. O Contiki propõe uma estratégia de ciclos de

trabalho que consegue manter um nó comunicável em uma rede, porém com seus rádios desligados em aproximadamente 99% do tempo.

2.8.4 LibCoap

LibCoap é uma biblioteca implementada em C do protocolo CoAP. Possui 292K de tamanho compilada estaticamente em sua versão 4.0.1. A licensa da biblioteca é GPL (2 ou maior) ou licensa BSD revisada.

Possui uma suíte de testes para regressão, utilizando o framework de testes CUnit (http://cunit.sourceforge.net/). A documentação pode ser encontrada em: http://libcoap.sourceforge.net/.

É uma biblioteca auto-condida, que possui parser do protocolo e funções básicas de rede utilizando sockets tipo BSD e malloc. Implementação de Hash, String e URI: os headers são hashkey.h, str.h, uri.h utilizados para montar os pacotes CoAP.

É separada num módulo de rede: net.h, aonde é implementado as funções de envio/recebimento de requisições e respostas, com confirmação, sem confirmaao, mensagem de reset e erros.

Para selecionar a camada de transporte é necessário selecionar utilizando flags de preprocessamento. O padrão é socket POSIX. A pilha uIP é selecionada com a flag -DWITH_CONTIKI, ou para selecionar a pilha lwIP -DWITH_LWIP.

2.8.5 TinyOS

O TinyOS é um sistema operacional projetado para sistemas embarcados com comunicação sem fio e restrições energéticas. Foi desenvolvido em nesC, uma linguagem código aberto que é uma extensão do C. É um sistema operacional baseado em eventos desenvolvido para redes de sensores que possuem recursos limitados. Possui uma implementação do CoAP baseada na libCoAP.

2.8.6 CantCoap

É uma implementação em C++ que visa facilitar a criação de pacotes Coap tanto diretamente quanto a partir de uma sequência de characteres, recebida da camada UDP por exemplo.

É possível montar os pacotes CoAP a partir de uma sequência de ca-

racteres recebidos de uma placa de rede. Abaixo exemplos de uso da biblioteca retirados de https://github.com/staropram/cantcoap.

Abaixo um exemplo de uso para montar um pacote e enviar:

```
CoapPDU *pdu = new CoapPDU();
pdu->setType(CoapPDU::COAP_CONFIRMABLE);
pdu->setCode(CoapPDU::COAP_GET);
pdu->setToken((uint8_t*)"\3\2\1\0",4);
pdu->setMessageID(0x0005);
pdu->setURI((char*)"test",4);

// send packet
ret = send(sockfd,pdu->getPDUPointer(),pdu->getPDULength(),0);
```

Quando receber a mensagem a forma de uso é mostrada abaixo:

```
// receive packet
ret = recvfrom(sockfd,&buffer,BUF_LEN,0,recvAddr,recvAddrLen);
CoapPDU *recvPDU = new CoapPDU((uint8_t*)buffer,ret);
if(recvPDU->validate()) {
    recvPDU->getURI(uriBuffer,URI_BUF_LEN,&recvURILen);
    ...
}
```

Por ser uma biblioteca bem simplificada e não possuir dependências diretas com a implementação da camada de transporte e ser código livre, foi escolhida para a implementação teste do trabalho.

3 DESENVOLVIMENTO

Este trabalho implementa uma biblioteca que utiliza a camada UDP do EPOS para dar suporte ao protocolo CoAP, uma aplicação gateway GPRS/802.14.5 utilizando o EPOS e um componente de hardware GPRS que será acoplado ao EposMoteII.

Para isto foram realizados diversos estudos durante o desenvolvimento do protocolo, pesquisas para escolher um módulo GPRS adequado a tarefa. Além disso testes de validação dos sistemas de software e validação do módulo GPRS, levantamento de requisitos para porte de uma biblioteca CoAP para o EPOS (libCoap, libCantCoap, microCoap, entre outras) e EPOSMoteII.

Porte da biblioteca para o EPOS. Utilizando testes para validar o funcionamento entre diferentes arquiteturas e compiladores. A execução dos testes foi feita no Qemu.

Implementação dos mecanismos de transmissão para mensagens confirmáveis e não-confimáveis, requisição e resposta.

A aplicação que será responsável pelo roteamento de mensagens para Internet utiliza a tecnologia GPRS, provida por um módulo GSM/GPRS da Quectel o M95.

3.1 LEVANTAMENTO DE REQUISITOS

Infraestrutura flexível para a contrução de aplicações embarcadas em modelo de webservices utilizando redes de sensores sem fio.

O usuário irá acessar a rede de sensores sem fio por uma aplicação html5, hospedada em http://minhascoisas.io. Nesta aplicação é possível enviar requisições para a rede de sensores de teste e listar os serviços oferecidos.

3.1.1 Requisitos Funcionais

Coletar informação do ambiente através de sensores e transmití-las através da Internet. Fácil integração com a Internet mesmo em locais sem rede WIFI.

As principais funções deste gateway são receber os dados da rede de sensores e encaminhá-las para um servidor remoto que armazenará essas informações e exibirá de forma conveniente para o usuário final.

Será possível comunicar-se em tempo real com a rede de sensores,

utilizando um módulo GPRS que irá repassar as requisições e respostas alimentadas pelo usuário.

As funções a da aplicação do gateway são:

1. Configuração, envio e recebimento de SMS;

2. Configuração contexto PDP, Configuração GPRS;

3. Configuração TCP/IP e manutenção de conexão TCP/IP.

3.1.2 Requisitos Não Funcionais

Os webservices que vão executar nos motes devem herdar a caracteristica de baixo consumo energético para que possam durar por anos, e serem extensíveis, podendo ser reutilizada em outras arquiteturas.

Além os serviços serão listatos utilizando o padrão (SHELBY Z., 2012) disso os dados captados por sensores serão disponibilizados na forma de webservices CoAP, protocolo específico para este tipo de aplicacção.

A padronização na comunicação visa facilitar a interconexão dos sistemas de diversas plataformas.

Características destes sistemas são eficiente em:

- Armazenamento: deve ser suficientemente pequeno para ser utilizado em microcontroladores.

- Energia: cosumir pouca energia para longa durabilidade com bateria.

- Valor: utilizar uma infraestrutura de hardware simples para realizar as tarefas.

3.2 ESPECIFICAÇÃO

3.2.1 Arquitetura

A figura 3.2.1 ilustra a interconexão entre os nodos da rede.

3.2.2 Componentes

O CoAP implementado é composto pelos seguintes componentes:

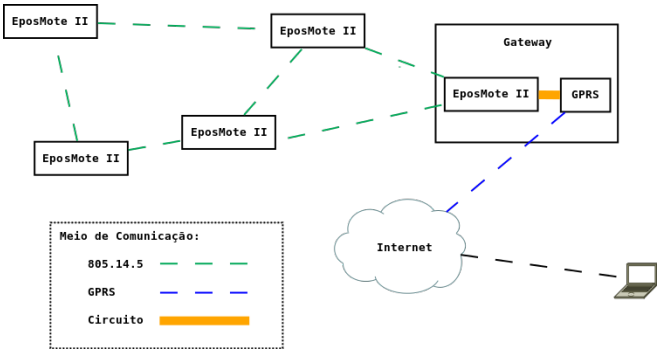

Figura 9 – Visão geral sobre comunicação do sistema.

A implementação consiste num módulo que trata requisições, encapsula em pacotes e transmite por mecanimos de transmissão baseados em (SHELBY Z., HARTKE K., BORMANN C., 2013).

A biblioteca utilizada para montar o pacote CoAP foi: https://github.com/staropram/cantcoap.git. Na qual enviei algumas correções e testes para facilitar a verificação da execução correta dos algoritmos internos durante mudanças no código. As alterações podem ser visualisadas aqui: https://github.com/staropram/cantcoap/commits?author=rafaeldelucena.

Para o funcionamento desta biblioteca no EPOS, e para utilizar uma MTU limitada a 128 bytes utilizo um buffer com um valor máximo e armazeno os dados do pacote no buffer. Foi necessário alterar os tipos das variáveis para se adquerem ao EPOS.

O desenvolvimento de um mecanismo de retransmissão de mensagens não-confirmadas utilizando uma lista ordenada. Mecanismo de requisicçõ e resposta, as requisições pendentes foram armezenadas num Hash com a chave sendo o token gerado pelo cliente.

O protocolo CoAP foi modelado conforme é mostrado na figura 3.2.2 abaixo:

Para validar o comportamento utilizei alguns testes da própria biblioteca CoAP portados para o EPOS. Foi necessário implementar a função assert. Já que seria bem mais trabalhoso adicionar uma ferramenta de testes no sistema de build do EPOS.

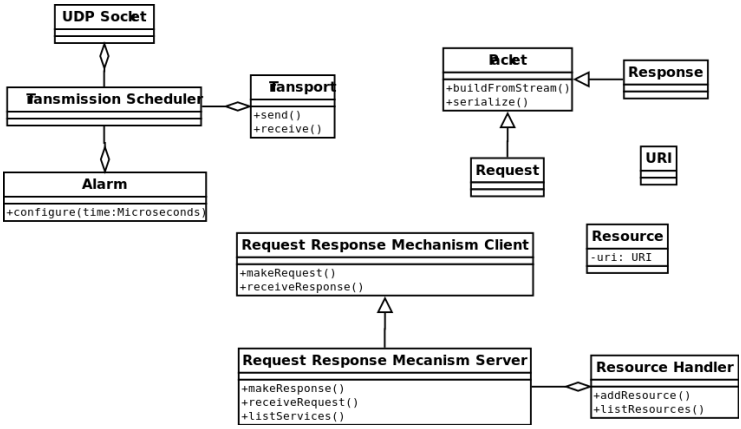

Figura 10 – Diagrama UML das entidades de software implementadas.

3.3 TESTES

Testes do cliente: Fazendo uma requisição confirmáveis e não-confirmáveis do tipo: GET, POST, PUT, DELETE. Recebendo respostas: vÃ¡lidas e invÃ¡lidas.

Testes do Servidor: Recebendo e respondendo requisições: que possui recurso, que não possui, descoberta de recurso.

3.3.1 Testes na placa de desenvolvimento do móuldo GPRS

Os testes feitos foram: Enviar e recebimento de mensagens; criar socket TCP, enviar e receber mensagem via socket, fazer requisição HTTP, foi possível utilizando os comandos proprietários do modem.

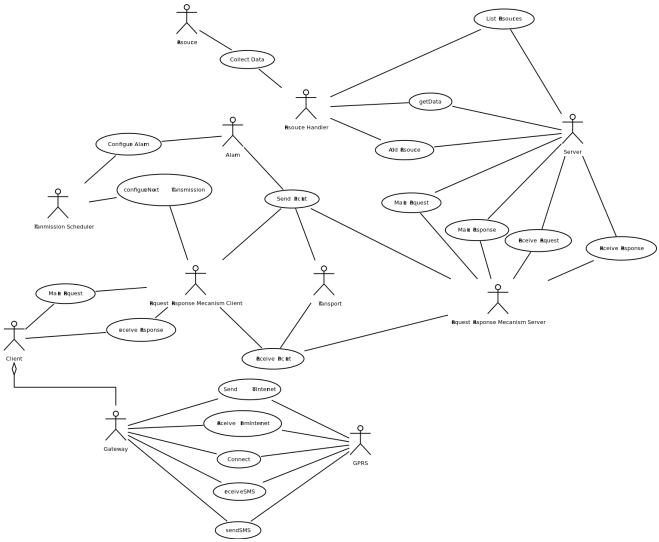

Figura 11 – Diagrama de casos de uso.

4 AVALIAÇÃO

4.1 ANÁLISE FUNCIONAL

4.1.1 Limitações Funcionais

A falta de uma implementacação segura, utilizando os padrões DTLS.

4.2 ANÁLISE ESTRUTURAL

4.3 ANÁLISE DE DESEMPENHO

O tempo de execução das principais funções foi medido. Abaixo a tabela 4.3, que faz um comparativo com as diversas arquituras e operações entres as principais soluções livres.

OS	CoAP	Size	Memory
EPOS	CantCoap		
Contiki	Erbium		
Contiki	libcoap		
TinyOS	libcoap		

Tabela 2 – Comparação das implementações

5 CONSIDERAÇÕES FINAIS

A desenvolver...

5.0.1 Conclusão

5.0.2 Trablhos Futuros

REFERÊNCIAS

Richardson, Leonard and Ruby, Sam. *RESTful web services*. 2008.

Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal. *Wireless Sensor Network survey*. 2008.

Antônio Augusto Fröhlich, Wolfgang Schröer-Preikschat. *Embedded Parallel Operating System*. 1999. <https://epos.lisha.ufsc.br/>.

Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. 2000.
<http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm>.

Z. Shelby, K. Hartke, C. Bormann *Constrained Application Protocol (CoAP)*. 2013. <http://www.ietf.org/id/draft-ietf-core-coap-18.txt>.

Z. Shelby . *Constrained RESTful Environments (CoRE) Link Format*. 2012.
<http://tools.ietf.org/rfc/rfc6690.txt>.

Koojana Kuladinithi, Olaf Bergmann, Thomas Pötsch, Markus Becker, Carmelita Görg *Implementation of CoAP and its Application in Transport Logistics*. 2011.
<http://hinrg.cs.jhu.edu/joomla/images/stories/coap-ipsn.pdf>

Mike P. Papazoglou. *Service-Oriented Computing: Concepts, Characteristics and Directions*. Web Information Systems Engineering - WISE , pp. 3-12, 2003.

Franck L. Lewis. *Wireless Sensor Networks*. Journal Smat environments: technologies, protocols, and applications, pp. 11-46, 2004.

Alexandre Massayuki Okazaki, AntrÃ´nio Augusto FrÃ¶hlich. *ADHOP: an Energy Aware Routing Algorithm for Mobile Wireless Sensor Networks*. 2012.

Booth, David and Haas, Hugo and McCabe, Francis and Newcomer, Eric and Champion, Michael and Ferris, Chris and Orchard, David *Web Services Architecture*. <http://www.w3.org/TR/ws-arch>. 2004.

Watteyne, Thomas and Vilajosana, Xavier and Kerkez, Branko and Chraim, Fabien and Weekly, Kevin and Wang, Qin and Glaser, Steven and Pister, Kris. *OpenWSN: a standards-based low-power wireless development environment*. 2012.

Dargie, Waltenegus and Poellabauer, Christian *Fundamentals of wireless sensor networks: theory and practice*. 2010. John Wiley & Sons.

Bult, K. and Burstein, A. and Chang, D. and Dong, M. and Fielding, M. and Kruglick, E. and Ho, J. and Lin, F. and Lin, T.-H. and Kaiser, W.J. and Marcy, H. and Mukai, R. and Nelson, P. and Newburg, F. L. and Pister, K.S.J. and Pottie, G. and Sanchez, H. and Sohrabi, K. and Stafsudd, O.M. and Tan, K. B. and Yung, G. and Xue, S. and Yao, J. *Low power systems for wireless microsensors*. 1996