Gerenciamento Hierárquico de Ambientes Inteligentes Utilizando SNMP

Denilson Rosa da Conceição, Jonas Ayres da Silva, Mathias Gheno Azzolini, Roben Castagna Lunardi, Rafael Pereira Esteves

Instituto Federal do Rio Grande do Sul (IFRS) – Campus Restinga
Porto Alegre – RS – Brazil

{drconceicao,jayres,mgheno}@restinga.ifrs.edu.br

{roben.lunardi, rafael.esteves}@restinga.ifrs.edu.br

Abstract. *Smart environments are designed to improve our daily lives by providing means to monitor a physical environment and react to the context in automatic and proactive ways. These environments are typically composed by a variety of sensors, actuators, and controllers interconnected by a network infrastructure. In order to allow interoperability with current Internet, such devices need to support a TCP/IP stack relying on IPv6 for addressing. Proper management of smart devices is required to allow a human operator to gather updated statistics and perform remote management over smart devices. In this paper, we investigate the use of SNMP, the de facto Internet network management protocol, in the management of smart environments. Since smart devices are typically resource constrained, we propose and develop an SNMP-based hierarchical management architecture to manage an environment composed of Arduino and Raspberry Pi boards. MIB extensions are developed and instrumented to support the proposed management architecture. Results demonstrate the feasibility of the SNMP protocol in managing smart devices without adding significant overheads.*

Resumo. *Ambientes inteligentes são projetados para melhorar nossa vida diária oferecendo meios para monitorar um ambiente físico e reagir ao contexto de forma automática e pró-ativa. Estes ambientes são tipicamente compostos de uma variedade de sensores, atuadores e controladores interconectados por uma infraestrutura de rede. Para permitir a interoperabilidade com a Internet atual, estes dispositivos precisam suportar uma pilha TCP/IP e utilizar IPv6 para endereçamento. O gerenciamento adequado de dispositivos inteligentes é necessário para permitir que um operador humano obtenha estatísticas atualizadas e realize o gerenciamento remoto desses dispositivos. Neste artigo, se investiga o uso do SNMP, o protocolo de gerenciamento de redes padrão da Internet, no gerenciamento de ambientes inteligentes. Uma vez que dispositivos inteligentes normalmente possuem recursos limitados, uma arquitetura de gerenciamento hierárquica baseada em SNMP é proposta e desenvolvida com o objetivo de gerenciar um ambiente composto de placas Arduino e Raspberry Pi. Extensões de MIB são desenvolvidas e instrumentadas para viabilizar a arquitetura de gerenciamento proposta. Resultados demonstram a viabilidade do protocolo SNMP no gerenciamento de dispositivos inteligentes sem adicionar sobrecarga significativa.*

1. Introdução

A Internet das Coisas (IoT) viabilizou uma série de aplicações para facilitar diversos aspectos do cotidiano [Atzori et al. 2010]. Exemplos de aplicações IoT incluem *e-health*, domótica, logística, transporte, entre outras. Tecnologias IoT podem ser aplicadas na implementação dos chamados *ambientes inteligentes*. Ambientes inteligentes são capazes de obter informações sobre o ambiente e de seus usuários de forma a proporcionar uma melhor experiência [Cook and Das 2007]. Uma casa inteligente, por exemplo, pode contribuir para a redução no consumo de energia desligando automaticamente luzes e equipamentos ociosos enquanto que uma fábrica inteligente pode monitorar em tempo real a produtividade de forma a identificar gargalos na produção.

Ambientes inteligentes são materializados através de dispositivos como sensores, atuadores e controladores interconectados por uma infraestrutura de comunicação. Sistemas de identificação por radiofrequência *(RFID - Radio Frequency Identification)* e redes de sensores sem fio *(WSN - Wireless Sensor Networks)* são exemplos de tecnologias IoT que são tipicamente utilizadas na implementação de ambientes inteligentes [Atzori et al. 2010] [Gubbi et al. 2013]. A comunicação dos dispositivos em um ambiente inteligente pode ser feita através de tecnologias como ZigBee, Bluetooth, WiFi, LoraWan, 3G/4G/LTE e, em casos específicos, Ethernet [Santos et al. 2016] [Gubbi et al. 2013].

Um aspecto importante nesse contexto diz respeito ao gerenciamento de ambientes inteligentes e dos dispositivos que fazem parte dos mesmos. O gerenciamento de ambientes inteligentes envolve principalmente a monitoração dos dispositivos, incluindo o estado de operação bem como os valores obtidos pelos sensores, e a configuração remota de dispositivos para permitir, por exemplo, que um operador humano ligue um equipamento (ex: ar-condicionado) antes de chegar na sua residência. Atualmente esse gerenciamento é feito através de soluções proprietárias e/ou protocolos específicos.

Neste artigo, se investiga o uso do protocolo de gerenciamento padrão da Internet, o SNMP *(Simple Network Management Protocol)* [Case et al. 1990] no gerenciamento dos dispositivos de um ambiente inteligente. A motivação para essa abordagem é permitir que plataformas de gerenciamento baseadas em SNMP possam ser rapidamente adaptadas para gerenciar um ambiente inteligente. Além disso, diferentes ambientes (ex: uma casa, uma indústria, uma fazenda) poderiam ser operados através de uma mesma interface de gerenciamento.

O principal desafio em adequar o SNMP como protocolo de gerenciamento para ambientes inteligentes é lidar com as restrições de capacidade dos dispositivos tipicamente utilizados nesse tipo de ambiente. Embora existam propostas que realizam uma série de adaptações e otimizações para permitir que um agente SNMP rode em um dispositivo IoT [Sehgal et al. 2012] elas normalmente dependem de *software* ou bibliotecas específicas que nem sempre estão disponíveis ou podem ser adaptadas para qualquer dispositivo IoT. Neste artigo, adota-se uma abordagem alternativa através de um *gateway* IoT consistindo de um dispositivo de maior capacidade que recebe e envia mensagens SNMP de/para uma aplicação de gerenciamento e interage diretamente com os dispositivos de menor capacidade. Dessa forma, os dispositivos finais (sensores e atuadores) não ficam sobrecarregados e a arquitetura padrão SNMP [Harrington et al. 2002] é mantida. Adicionalmente, com o objetivo de demonstrar a viabilidade da proposta, foi desenvolvido um sistema de gerenciamento de objetos inteligentes baseado na arquitetura proposta.

Além desta seção introdutória, o artigo é composto de mais 5 seções. Na seção 2 são discutidos os principais trabalhos relacionados ao gerenciamento de ambientes IoT. A arquitetura de gerenciamento para ambientes inteligentes proposta neste trabalho é apresentada e descrita na seção 3. Na seção 4 são discutidos aspectos de implementação do sistema de monitoramento de ambientes IoT. A arquitetura de gerenciamento proposta é avaliada na seção 5, através de um estudo de caso. Finalmente, na seção 6 são apresentadas as conclusões e os próximos passos do trabalho.

2. Trabalhos relacionados

Nos últimos anos, diversas pesquisas foram realizadas no contexto de Internet das Coisas. Grande parte da pesquisa em IoT está relacionada à definição de novas arquiteturas, soluções para comunicação entre dispositivos e estratégias de segurança para prevenir ataques. A seguir, nesta seção, serão apresentados os trabalhos mais relevantes relacionadas à nossa proposta.

Para resolver o problema de mobilidade em redes, Savolainen *et al.* [Savolainen et al. 2013] propõem a utilização do protocolo IPv6 em dispositivos de redes IoT. Devido ao fato do IPv6 provocar um consumo muito alto de memória para a sua utilização, o artigo propõe a utilização do IPv6 apenas nos *gateways* da rede de sensores.

Com o objetivo de integrar diferentes tecnologias de comunicação sem fio, tais como, WiFi, ZigBee, Bluetooth, Qin *et al.* [Qin et al. 2014] propõem uma arquitetura SDN (Software Defined Network) para redes IoT. Para alcançar este objetivo, os autores desenvolvem uma extensão para a arquitetura existente chamada de MINA (*Multinetwork INformation Architecture*), com um controlador SDN para atuar como *middleware* na camada de rede.

No contexto de comunicação e gerenciamento de dispositivos em redes IoT, Benamar *et. al.* [Benamar et al. 2014] realizam uma comparação de diferentes protocolos para serem utilizados em redes IoT. Por exemplo, no contexto de endereçamento IP, o trabalho propõe alternativas com menor consumo de memória e processamento através da utilização do padrão 6LoWPAN [Montenegro et al. 2007]. No contexto de gerenciamento dos dispositivos IoT, o artigo apresenta uma comparação entre os protocolos SNMP, LNMP (*Lightweight Network Management Protocol*), e as soluções propostas pelo grupo da IETF através da iniciativa COMAN (*Constrained networks and devices MANagement*), como o CoAP (*Constrained Application Protocol*) [Shelby et al. 2014].

Mais especificamente relacionado com a análise de tráfego IoT, Chen *et al.* [Chen et al. 2013] propõem um sistema de monitoramento e análise de trafego de dispositivos IoT. Neste trabalho, o sistema apresenta dados como, monitoramento de sensores e o estado dos dispositivos. Apesar de o sistema ser acessível pela Web ou por *smartphone*, não são utilizados protocolos padrão de gerência de redes. Sendo a obtenção dos dados realizadas diretamente através dos pinos de entrada e saída do *hardware* específico. Desta forma, a solução fica limitada ao hardware utilizado.

No trabalho de Chang *et al.* [Chang et al. 2012] apresentam uma plataforma para simulação de sistemas de IoT no contexto da Internet do Futuro (do inglês, *Future Internet*). Nesta simulação, o autor faz uma análise do roteamento das mensagens, que devido à arquitetura da rede, acaba por gerar um grande tráfego de informações para o coordenador de tráfego. Ainda utilizando esta simulação, é feita uma medição do tempo de atraso

na entrega das mensagens entre os componentes. Nesta medição pode ser percebido que à medida que o tempo e tráfego da simulação avançam, este tempo cresce rapidamente.

Um dos usos mais comuns de redes IoT é na implementação de redes de sensores inteligentes. Neste contexto, Ma *et al.* [Ma et al. 2016] propõem uma arquitetura de IoT separadas em quatro camadas: (i) camada de sensoriamento e controle; (ii) camada de troca de dados; (iii) camada de integração de informação; e (iv) camada de aplicação. Neste modelo, as informações só são trocadas entre as camadas subsequentes, sem haver acesso direto entre as camadas que não estão diretamente ligadas. A camada de sensoriamento e controle é a camada responsável por interagir com o mundo físico, transformando em sinais digitais informações analógicas e/ou interagindo com objetos físicos. A camada de troca de dados, por outro lado, é responsável apenas pela comunicação, utilizando diferentes tipos de meios e protocolos. A camada de integração da informação é responsável pelo processamento da informação e garantia de aspectos básicos de segurança. Por fim, a camada de aplicação combina dados coletados de diversos sensores que serão apresentados para o usuário final.

Para permitir o uso dos protocolos de gerenciamento SNMP e NETCONF em dispositivos de baixa capacidade, Sehgal *et al.* [Sehgal et al. 2012] desenvolveram versões simplificadas desse protocolos em dispositivos Atmel AVR Raven executando sobre o sistema operacional Contiki. Embora os autores demonstrem que as otimizações feitas permitem que agentes SNMP e NETCONF sejam executados em dispositivos IoT sem prejuízo significativo de desempenho, não é possível generalizar esse comportamento para outros dispositivos IoT com menor capacidade (ex: Arduino) e/ou que não tenham suporte adequado ao Contiki.

Analisando os trabalhos abordados até então, é possível notar que a maioria deles não exploram adequadamente o uso de protocolos padronizados no gerenciamento de ambientes IoT, ficando restritos a protocolos proprietários e/ou hardwares/softwares específicos, reduzindo a interoperabilidade que é desejável em ambientes inteligentes. Na próxima seção, é apresentada uma proposta de arquitetura baseada no protocolo padrão de gerenciamento da Internet, o SNMP, para contornar as limitações das soluções atuais de gerenciamento de ambientes IoT.

3. Arquitetura de gerenciamento de ambientes inteligentes baseada em SNMP

Nesta seção é apresentada a arquitetura de gerenciamento baseada no protocolo SNMP para dispositivos IoT que compõem um ambiente inteligente. Inicialmente, extensões feitas à base de informações de gerenciamento para viabilizar a arquitetura propostas são apresentadas. Em seguida, os componentes da arquitetura de gerenciamento proposta são detalhados.

3.1. Modelo de Dados

No protocolo SNMP, as informações referentes aos dispositivos gerenciados são estruturadas em uma ou mais bases de informações de gerenciamento (*MIB - Management Information Base*). No início da década passada o IETF definiu uma MIB para o gerenciamento de sensores físicos, a ENTITY-SENSOR-MIB [Bierman et al. 2002] que permite

monitorar os valores de diferentes tipos de sensores como corrente elétrica, potência, temperatura, umidade, entre outros.

Embora a ENTITY-SENSOR-MIB permita a monitoração de sensores de forma relativamente simples, ela apresenta algumas limitações para o gerenciamento de ambientes inteligentes. Por exemplo, existe a necessidade de uma melhor identificação dos sensores, uma vez que diferentes sensores do mesmo tipo podem estar presentes em um ambiente inteligente. A localização física de um determinado sensor também é importante pois permite que o gerente execute ações de gerenciamento localizadas, por exemplo, para desligar todos os equipamentos de uma sala. Ainda, distinguir se um determinado dispositivo é um sensor ou atuador permite determinar que ações podem ser executadas sobre o mesmo.

Para contornar as limitações atuais de gerenciamento foram feitas extensões à ENTITY-SENSOR-MIB com a inclusão de novos objetos para possibilitar o gerenciamento pleno de ambientes inteligentes. A Figura 1 mostra uma representação gráfica da ENTITY-SENSOR-MIB com os novos objetos incluídos. Em seguida, é apresentada uma breve descrição de cada novo objeto.

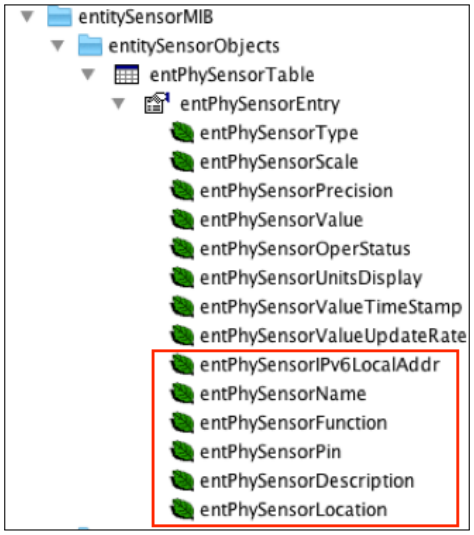

Figura 1. Extensões feitas à ENTITY-SENSOR-MIB

- *entPhySensorIPv6LocalAddr*: armazena o endereço IPv6 (link-local) do dispositivo;
- *entPhySensorName*: nome do dispositivo;
- *entPhySensorFunction*: função do dispositivo, sensor ou atuador:
- *entPhySensorPin*: pino onde o sensor está localizado. Útil para gerenciar dispositivos do tipo Arduino, que pode possuir vários sensores em pinos diferentes;
- *entPhySensorDescription*: breve descrição sobre a função do sensor (ex: luminosidade, temperatura);
- *entPhySensorLocation*: localização física do dispositivo.

3.2. Arquitetura de Gerenciamento Proposta

A arquitetura proposta neste trabalho baseia-se no conceito de arquitetura em camadas, conforme taxonomia definida por Ma *et al.* [Ma et al. 2016]. Para atender as necessidades do nosso trabalho, definiu-se *Camada de Sensoriamento e Controle* como a camada onde os dispositivos obtêm dados de sensores e configuram atuadores, apresentada na Figura 2 como *Dispositivos*. Entende-se por dispositivo o objeto microcontrolado, usualmente com restrições de processamento, capaz de utilizar um ou mais sensores (ex: luminosidade, temperatura, presença de algum tipo de gás, dentre outros) e/ou controladores (acionadores elétricos, emissor de sinais eletromagnéticos, motores de posicionamento, dentre outros). A *Camada de Troca de Dados* é responsável pela comunicação entre os dispositivos e os *Gateways*. Nesta arquitetura, os *Gateways* representam dispositivos com maior capacidade de *hardware*, capazes de realizar tarefas que demandam maior poder de processamento. A *Camada de Integração de Informação* é responsável pela consolidação das informações disponíveis nos *Gateways*. Por fim, a *Camada de Aplicação* é definida pela *Interface Web*, utilizada pelo administrador do sistema para visualizar as informações coletadas do ambiente.

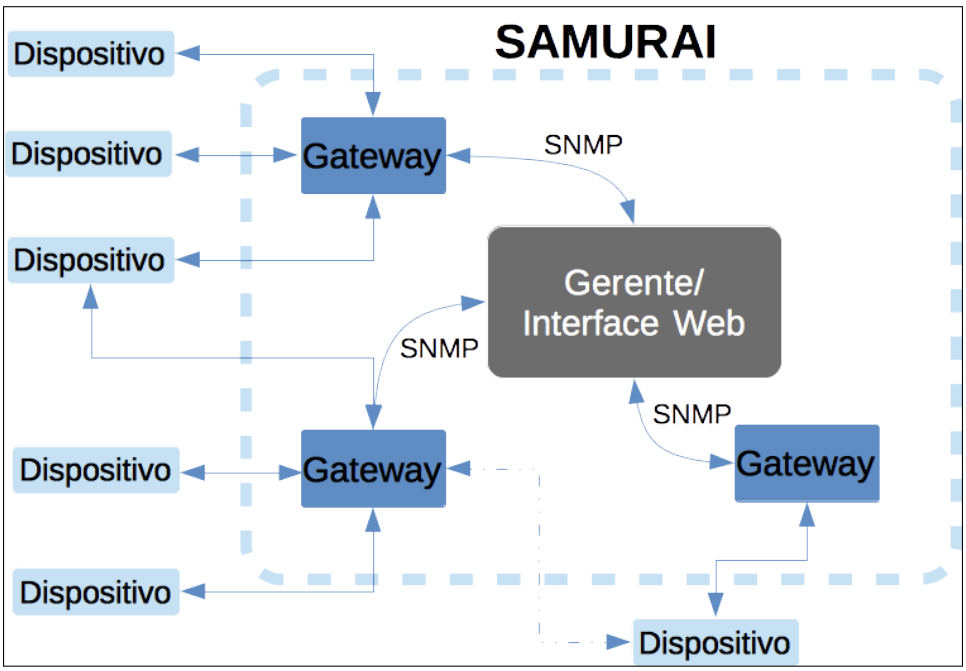

Figura 2. Arquitetura do SAMURAI

Devido às restrições de hardware (processamento, memória, energia, etc) dos dispositivos e pela possibilidade do uso de dispositivos heterogêneos, com diferentes restrições de *hardware*, optou-se pela utilização de *Gateways*. Desta forma, o processamento das informações, bem como a comunicação com os demais dispositivos de um mesmo domínio e com o Gerente ficam a cargo dos *Gateways*, diminuindo a carga de processamento nos dispositivos. Ainda, com a utilização de *Gateways*, permite-se a adoção de diferentes protocolos de comunicação, variando conforme as restrições de cada dispositivo. Portanto, a comunicação entre dispositivos e o *Gateway*, quando necessária, pode ser realizada utilizando protocolos com pouca exigência de processamento e memória (ex: Telnet e HTTP). Neste caso, utilizamos a premissa que os dispositivos estão configurados

para se comunicar em um VLAN dedicada e segura a transmissão dos dados. Sabe-se da importância dos requisitos de segurança, tais como autenticação, autorização, confidencialidade, e integridade durante a comunicação dos dispositivos. Todavia, estes fatores serão analisados em trabalhos futuros. Por outro lado, permite-se que dispositivos com maior capacidade de *hardware*, principalmente em canais de comunicação não confiáveis, realizem a transmissão utilizando protocolos que fazem uso de criptografia (ex: SSH e HTTPS). Adicionalmente, com o uso da arquitetura proposta, cada *Gateway* pode ficar responsável por monitorar um ou mais dispositivos. Desta forma, aumenta-se a resiliência do monitoramento dos dispositivos, diminuindo o efeito de falhas de um *Gateway*. Por fim, a arquitetura proposta permite a mobilidade dos dispositivos (por exemplo, sensores embarcados em equipamentos móveis), permitindo o ingresso e saída do dispositivo do domínio do *Gateway* (representado pela linha pontilhada na Figura 2).

O módulo Gerente monitora os *Gateways*, verificando a consistência da informação, visto que o mesmo dispositivo pode ser monitorado por mais de um *Gateway*. O Gerente, na arquitetura proposta, se comunica com os *Gateways* utilizando o protocolo SNMP. O SNMP foi adotado pela ampla popularidade, fácil implementação e possibilidades de adaptação às necessidades, conforme apresentado anteriormente na subseção 3.1 (Modelo de Dados). Pode-se destacar que o Gerente é impossibilitado de acessar diretamente os dispositivos, evitando, assim, sobrecarga destes dispositivos com maiores restrições de *hardware*.

Após a análise das informação obtidas pelo Gerente, o módulo Interface Web é capaz de processar e apresentar os dados dos dispositivos. Essas informações são processadas para a geração de gráficos com informações configuráveis por um operador humano, conforme a necessidade de monitoramento (ex: temperatura e luminosidade do ambiente, sobrecarga da CPU de um *Gateway*, dentre outros). Além disso, a interface permite o acionamento de dispositivos (ex: abertura de uma fechadura, ligar/desligar luzes, dentre outros). Este sistema de gerenciamento proposto no trabalho foi chamado de SAMURAI (Sistema de Monitoramento Unificado de Redes e Aplicações IoT).

Desta forma, o sistema SAMURAI se comunica diretamente com os *Gateways* para receber as informações e monitorar os dispositivos através do protocolo SNMP. Por sua vez, os *Gateways* se comunicam utilizando protocolo a ser definido para cada dispositivo, conforme as restrições de *hardware* do mesmo. Consequentemente, os agentes SNMP ficam nos *Gateways*, reduzindo o processamento, comunicação e, consequentemente, o consumo de energia dos dispositivos.

4. Implementação

Nesta seção, são abordados os aspectos de implementação do sistema, destacando as tecnologias utilizadas, o processo de desenvolvimento e os principais módulos do SAMURAI. Para a implementação do sistema foi utilizada a linguagem de programação Ruby [Ruby 2016], juntamente com os *frameworks* Ruby on Rails [Ruby on Rails 2016] e Daemons [Ruby Daemons 2016]. Adicionalmente, para a implementação da Interface Web, foi utilizada o *framework* bootstrap [Bootstrap 2016]. Ainda, para a geração de gráficos, foi utilizado a API Google Charts [Google 2016]. Por fim, para a comunicação com os dispositivos foi utilizado o protocolo SNMP (versão 2), bem como demais protocolos de rede como, IPv4 e IPv6.

O módulo Gerente/Interface Web, apresentado anteriormente na Figura 2, pode ser visto em detalhes (como submódulos separados) na Figura 3. O módulo da Interface Web (Figura 3) tem como objetivo prover uma interface de interação com o usuário através de um sistema Web. Desta forma, o usuário (Administrador do Sistema) pode cadastrar os dispositivos associados aos *Gateways* e cujos dados serão apresentados através de gráficos, listas e painéis. Vale destacar que todas as informações são coletadas e armazenadas em um banco de dados.

Por outro lado, o módulo Cadastrador (Figura 3) na aplicação tem como objetivo cadastrar os dispositivos que estão associados aos *Gateways*, os quais podem ser solicitados pela aplicação Web. Vale destacar que o mesmo dispositivo pode estar presente em mais de um *Gateway*, sendo o último o responsável por se comunicar com os dispositivos. Desta forma, a aplicação não se comunica diretamente com os dispositivos, pois todas as requisições são realizadas aos *Gateways* responsáveis por cada dispositivo. Esta forma de acesso aos dados reflete a arquitetura IoT proposta, conforme descrito na subseção 3.2 (Arquitetura de Gerenciamento Proposta).

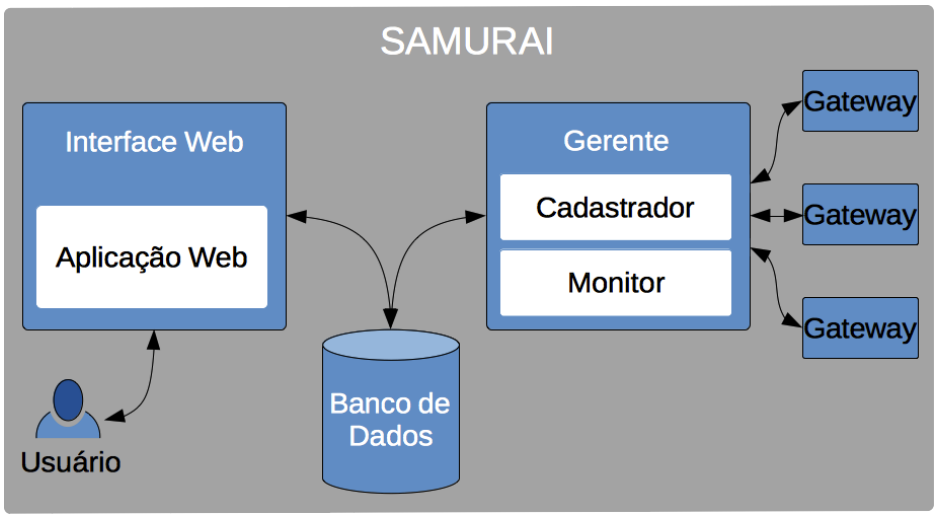

Figura 3. Implementação do Gerente/Interface Web

O módulo Monitor (Figura 3) tem como objetivo coletar dados dos equipamentos do sistema, tanto dos *Gateways* e dos dispositivos que estão a cargo dos mesmos, quanto dos equipamentos diretamente cadastrados pela interface Web do sistema. Ressalta-se que, atualmente, todas as informações são armazenadas em uma base de dados SQLite (em uma próxima versão pretende-se adotar um banco de dados NoSQL).

Na Figura 4 pode ser visualizado um dispositivo cadastrado, com IP do(s) *Gateway(s)* correspondente(s), bem como informações do dispositivo monitorado, tais como IP, tipo de sensor, valor correspondente aos dados monitorados, e sala onde o dispositivo foi cadastrado. Além disso, pode ser observado um gráfico da variação da luminosidade medida pelo dispositivo em função do tempo. Os gráficos a serem exibidos podem ser configurados pelo usuário para melhor demonstrar as informações. Por limitações de espaço, não serão apresentadas telas relativas ao cadastro de dispositivos, resumo do sistema (aba *Dashboard*) e demais funcionalidades.

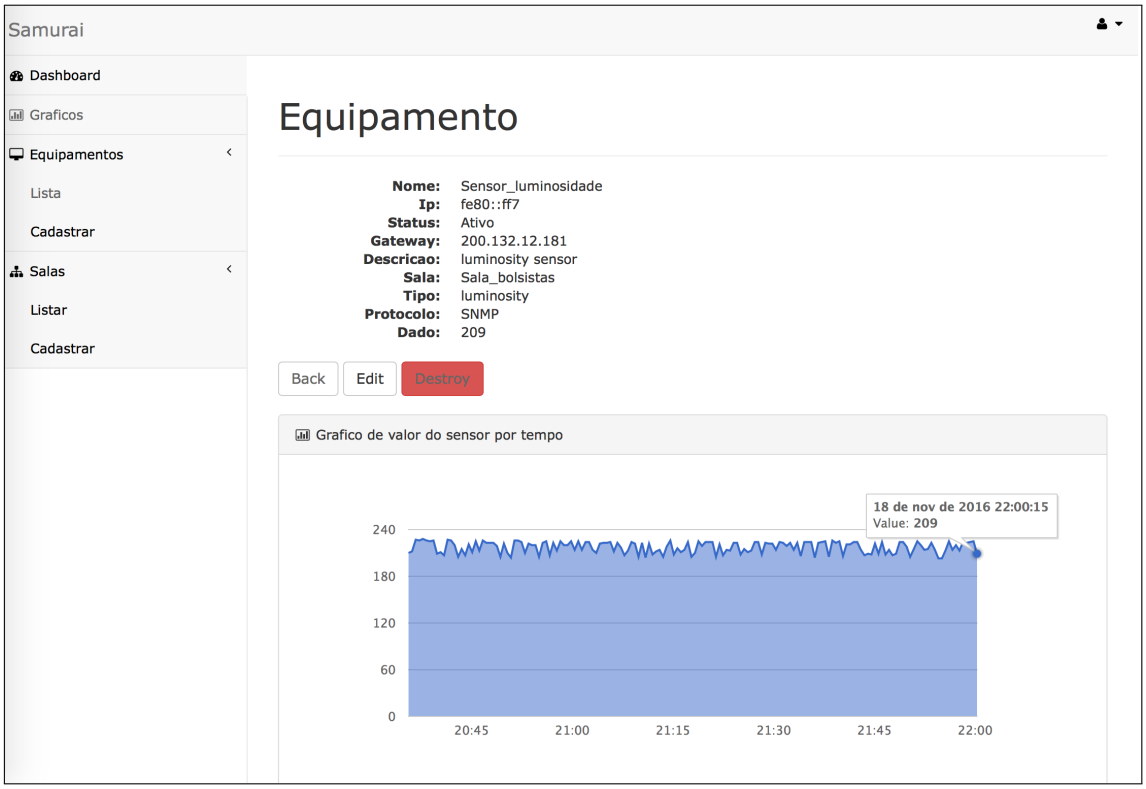

Figura 4. Interface Web do Sistema SAMURAI com dados de Sensor de Luminosidade

5. Estudo de caso

Para validar a arquitetura proposta, um estudo de caso foi conduzido com dispositivos reais em um cenário *indoor*. Os dispositivos são representados por placas controladoras Arduino Uno [Arduino 2016], cada uma equipada com um sensor de luminosidade e uma interface de rede Ethernet para comunicação. O *Gateway* é materializado através de um dispositivo Raspberry Pi 3 Modelo B [Raspberry Pi 2016] executando o sistema operacional Raspbian. O agente SNMP foi instalado no Raspberry Pi utilizando a suíte NET-SNMP [NET-SNMP 2016]. A ENTITY-SENSOR-MIB estendida foi instrumentada através de um módulo de MIB e incorporada ao agente SNMP. A comunicação entre o *Gateway* e os dispositivos é feita através de uma implementação Telnet simples. Dessa forma, as informações obtidas via Telnet são utilizadas para alimentar os objetos da ENTITY-SENSOR-MIB.

Foram utilizados três Arduinos Uno e um Raspberry Pi em nossos experimentos. Para coletar os dados, o Gerente dispara uma requisição *SNMP Walk*[1] para o agente SNMP instalado no *Gateway*. Essa requisição tem por objetivo recuperar as informações de todos os objetos armazenados na ENTITY-SENSOR-MIB para os três Arduinos gerenciados. Após receber receber a requisição SNMP, o *Gateway* recupera o valor dos sensores de luminosidade via Telnet e atualiza as informações da ENTITY-SENSOR-MIB. Por fim, o agente responde à requisição SNMP com as informações atualizadas dos dispositivos.

[1]Uma requisição SNMP Walk é constituída de uma série de requisições *get* e *get-next* que recuperam todos os objetos associados a um OID (*Object Identifier*) *específico.*

Do ponto de vista do Administrador do Sistema, os dados dos dispositivos e dos *Gateways* são facilmente visualizados através da interface do SAMURAI. Por exemplo, pode-se monitorar o valor de um sensor gerenciado pelo *Gateway*, utilizando o objeto da MIB *entPhySensorValue* (presente na ENTITY-SENSOR-MIB). No caso do "Gráfico de valor do sensor no tempo", como demonstrado na Figura 5, apresenta-se o valor do sensor durante o tempo (tempo de relógio do *Gateway*). No exemplo, pode-se observar que o dado monitorado (item "Dado" na Figura 5) - neste caso, valor correspondente à intensidade da iluminação obtida através de um sensor LDR (*Light Dependent Resistor*) - possui o valor 1012 (dez mil e doze). Pode-se observar, ainda, que o valor permanece praticamente constante durante o período que foi realizada a medição. Além disso, no "Gráfico de tempo de resposta do dispositivo" (Figura 5) pode ser observado o tempo de resposta das requisições do *Gateway* ao dispositivo, medida em milissegundos. Este tempo de resposta é calculado pelo início da conexão com o *Gateway* até a resposta do dispositivo. Neste gráfico, pode-se observar picos nos tempos de resposta, os quais usualmente representam os instantes nos quais o dispositivo está respondendo às requisições de dados solicitadas pelo *Gateway* via protocolos IPv6 e Telnet.

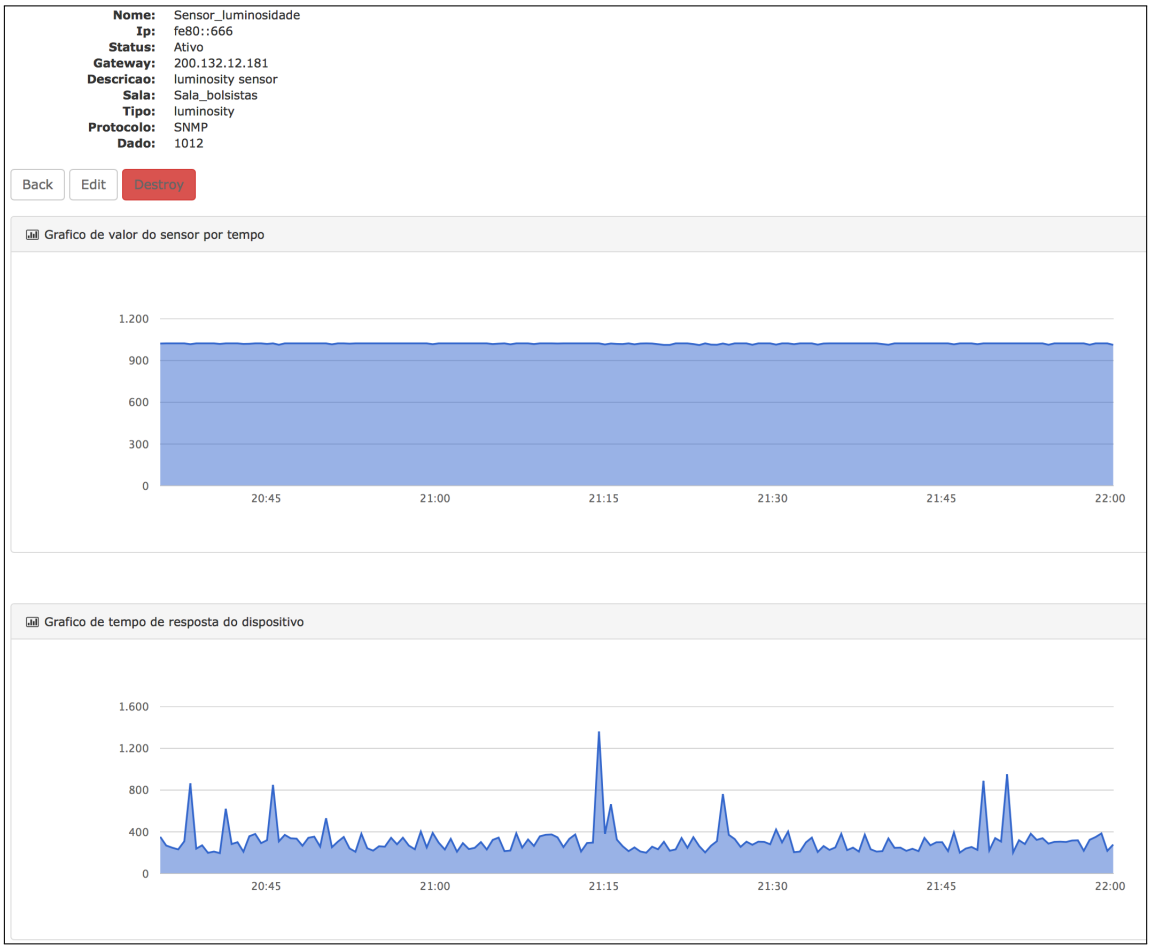

Figura 5. Monitoramento do Sensor pelo SAMURAI

Na Figura 6, pode ser observado o tempo de resposta do *Gateway* monitorado pelo SAMURAI. Este tempo de resposta, equivale ao tempo para que o *Gateway* receba o

retorno de todos os respectivos dispositivos (quando ativos). Consequentemente, o tempo de resposta do *Gateway* (também obtido em milissegundos) é maior que a resposta de um único dispositivo. Além disso, pode ser observado que existem picos de resposta em maior quantidade do que o dispositivo analisado. Isto ocorre pelo fato de que cada dispositivo pode responder às requisições do *Gateway* em tempo distintos. Desta forma, sempre que um dispositivo apresenta aumento no seu tempo de resposta, o tempo de resposta do *Gateway* também é alterado na exibição do sistema.

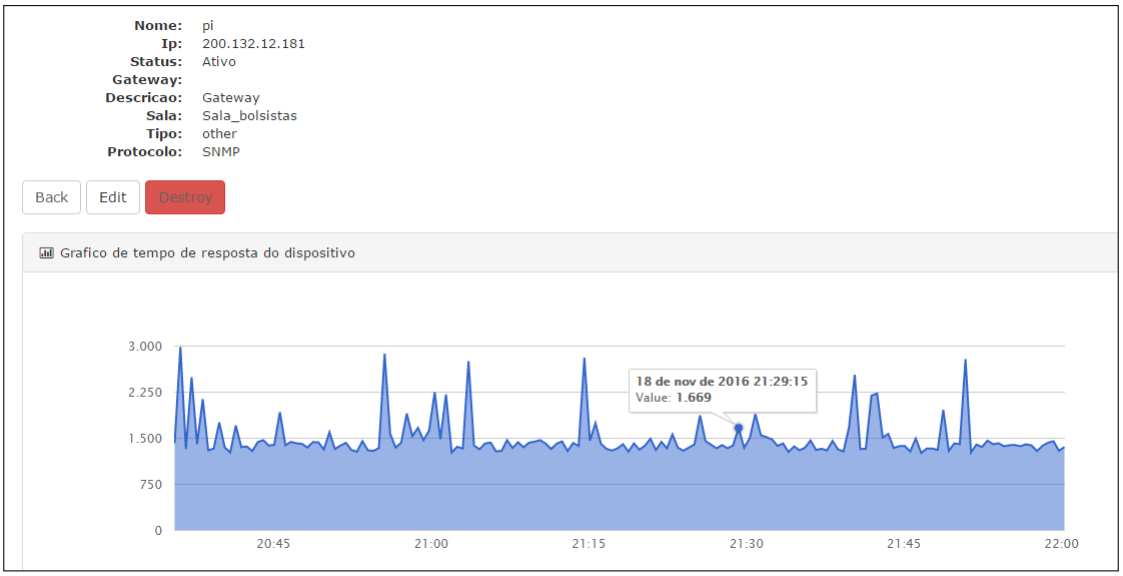

Figura 6. Monitoramento do *Gateway* pelo SAMURAI

Outro objetivo do estudo de caso é avaliar quantitativamente o desempenho da comunicação entre o *Gateway* e os dispositivos em relação ao tempo de resposta, tempo de CPU e utilização de memória, para verificar o impacto desta etapa no desempenho total da solução. O tempo de resposta nesse caso consiste do tempo que o *Gateway* leva para obter as informações dos dispositivos e atualizar os objetos da ENTITY-SENSOR-MIB. As medições referentes à tempo de CPU e utilização de memória são obtidas no *Gateway* através do utilitário *ps*. Os experimentos foram repetidos cerca de 80 vezes e os resultados são apresentados com intervalo de confiança de 95%.

A Figura 7 mostra o tempo de resposta para a comunicação entre o *Gateway* e os dispositivos finais para recuperar as informações de um, dois e três dispositivos, respectivamente. É possível observar que o tempo que o *Gateway* leva para recuperar as informações dos dispositivos finais (Arduino) cresce linearmente com o número de dispositivos gerenciados, o que é esperado uma vez que a quantidade de interações necessárias para recuperar os valores dos sensores e alimentar a ENTITY-SENSOR-MIB cresce com o número de dispositivos monitorados. No entanto, vale ressaltar que no pior caso (três dispositivos) esse tempo fica na ordem de 180 milissegundos por objeto, o que é consistente com o tempo de resposta acumulado no *Gateway* (Figura 6).

A Figura 8 apresenta a utilização de CPU para cerca de 260 requisições SNMP (*get* e *get-next*) de objetos da ENTITY-SENSOR-MIB. Essas requisições menores são derivadas das requisições *SNMP Walk* disparadas pelo módulo Gerente do SAMURAI.

É possível observar que a utilização de CPU se mantém abaixo de 4% para mais de 250 requisições SNMP, o que pode ser considerado baixo.

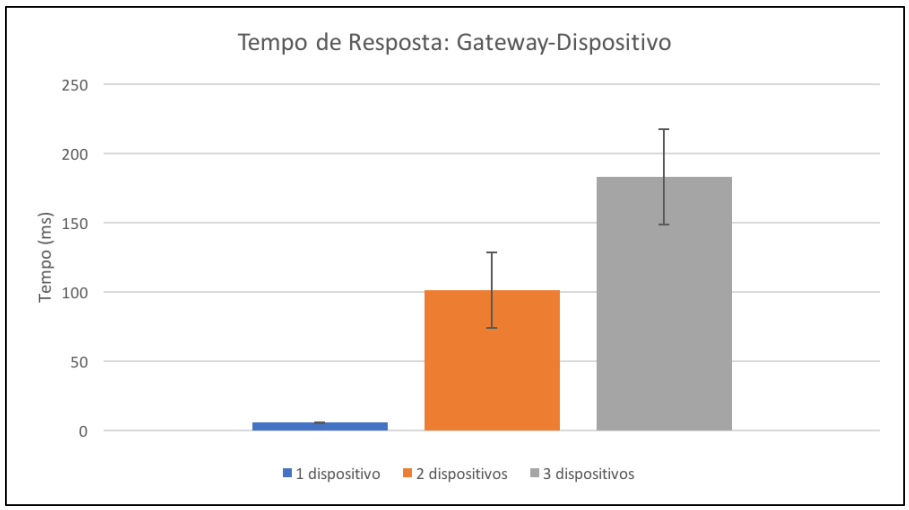

Figura 7. Tempo de resposta na entre o *Gateway* e os dispositivos

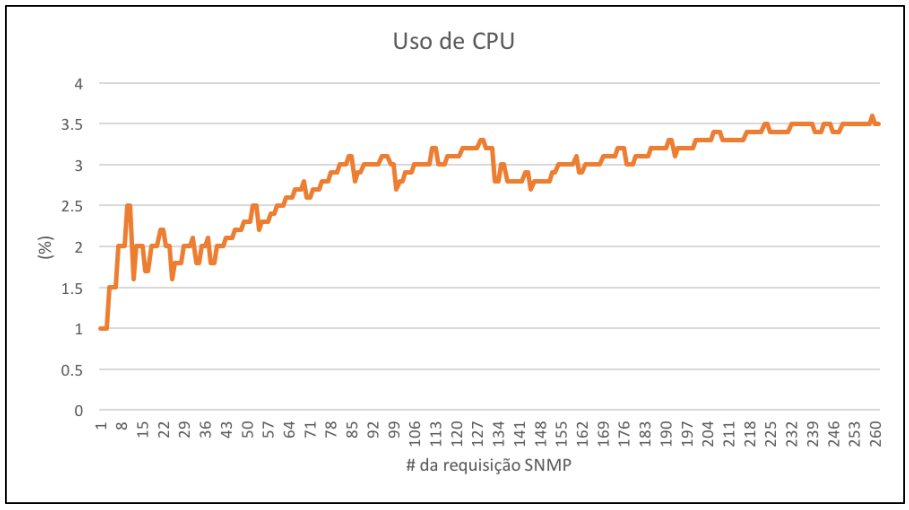

Figura 8. Utilização de CPU no *Gateway*

Com relação ao uso de memória, observou-se que o agente SNMP consome cerca de 4 MB da memória do *Gateway* (Raspberry Pi), o que corresponde a cerca de 0,5% da memória do mesmo sem variação significativa. Dessa forma, pode-se concluir que o *Gateway* é capaz de atender as requisições do módulo Gerente sem prejuízo de desempenho.

A partir dos resultados obtidos é possível verificar que a comunicação entre o *Gateway* e os dispositivos não exige grandes quantidades de recursos computacionais, podendo ser feita com dispositivos de baixo custo. O tempo de resposta total superior a 1 segundo para os três dispositivos pode ser explicado pela grande quantidade de variáveis (14) que são recuperadas em cada requisição. É possível otimizar essa comunicação para recuperar um subconjunto dos objetos da ENTITY-SENSOR-MIB, como, por exemplo, os valores dos sensores (*entPhySensorValue*) que mudam com frequência maior. No en-

tanto, a avaliação feita neste trabalho procurou explorar o pior caso, isto é, recuperando todas as variáveis de todos os dispositivos gerenciados.

6. Conclusões e Trabalhos Futuros

O monitoramento de objetos e a gerência de dispositivos representam um passo vital para a manutenção e controle da operação de um sistema em rede. Especificamente no contexto de ambientes IoT, poucas soluções existentes se preocuparam em com a padronização de tal monitoramento, especialmente para utilização em ambientes de *hardware* heterogêneo. Para abordar este problema, propôs-se um conjunto de adaptações na ENTITY-SENSOR-MIB, com o objetivo de padronizar o monitoramento de objetos de ambientes IoT.

Foi proposta uma arquitetura com múltiplos *Gateways*, onde mais de um *Gateway* pode gerenciar o mesmo dispositivo. Desta forma, espera-se que mesmo que um *Gateway* fique indisponível, outro (desde que também esteja monitorando o dispositivo em questão) possa manter a gerência e o controle sobre o dispositivo. Além disso, foi proposto um sistema de monitoramento, intitulado SAMURAI, para viabilizar os experimentos e estudos de caso da arquitetura proposta.

Na avaliação realizada, pôde-se observar que o tempo de resposta dos dispositivos ficou na casa dos 180 milissegundos por objeto consultado, demonstrando que a solução está dentro do esperado. Ainda, pode ser observado que mesmo com mais de 250 requisições SNMP, a utilização da CPU ficou abaixo de 4%, valor que representa pouco impacto para as demais funcionalidade do *Gateway*. Adicionalmente, o agente SNMP utiliza pouca memória do *Gateway*, ficando próximo de 0,5%. Por fim, o Sistema SAMURAI, ao consultar o *Gateway*, necessita aguardar pouco mais de 1 (um) segundo para obter as informações de todos os sensores monitorados, resultado importante, especialmente considerando a grande quantidade de objetos recuperados em mais de 250 requisições.

Como trabalhos futuros, pretende-se (*i*) investigar estratégias para garantir níveis de segurança, especialmente nas Camadas de *Troca de Dados* e *Integração de Informação*, (*ii*) realizar experimentos com diferentes dispositivos, simulando ambientes heterogêneos, (*iii*) avaliar a arquitetura e o sistema proposta em cenário com milhares de dispositivos para avaliar a escalabilidade da solução e (*iv*) comparar a solução proposta com outros protocolos de comunicação projetados para ambientes IoT como o MQTT e o CoAP.

Referências

Arduino (2016). URL: https://www.arduino.cc. Acessado em: abril de 2016.

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Benamar, N., Jara, A., Ladid, L., and Ouadghiri, D. E. (2014). Challenges of the Internet of Things: IPv6 and Network Management. In *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 328–333.

Bierman, A., Romascanu, D., and Norseth, K. (2002). RFC 3433: Entity Sensor Management Information Base.

Bootstrap (2016). URL: http://getbootstrap.com. Acessado em: agosto de 2016.

Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, J. (1990). RFC 1157: Simple network management protocol (SNMP).

Chang, K. D., Chen, J. L., Chen, C. Y., and Chao, H. C. (2012). IoT operations management and traffic analysis for Future Internet. In *2012 Computing, Communications and Applications Conference*, pages 138–142.

Chen, T. Y., Wei, H. W., Hsu, N. I., and Shih, W. K. (2013). A IoT Application of Safe Building in IPv6 Network Environment. In *IEEE COMPSAC 2013*, pages 748–753.

Cook, D. J. and Das, S. K. (2007). How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53 – 73.

Google (2016). URL: https://developers.google.com/chart/. Acessado em: agosto de 2016.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Harrington, D., Wijnen, B., and Presuhn, R. (2002). RFC 3411: An architecture for describing simple network management protocol (SNMP) management frameworks.

Ma, H., Liu, L., Zhou, A., and Zhao, D. (2016). On Networking of Internet of Things: Explorations and Challenges. *IEEE Internet of Things Journal*, 3(4):441–452.

Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. (2007). RFC 4944: Transmission of IPv6 packets over IEEE 802.15. 4 networks.

NET-SNMP (2016). URL: http://net-snmp.sourceforge.net. Acessado em: abril de 2016.

Qin, Z., Denker, G., Giannelli, C., Bellavista, P., and Venkatasubramanian, N. (2014). A Software Defined Networking architecture for the Internet-of-Things. In *IEEE NOMS 2014*, pages 1–9.

Raspberry Pi (2016). URL: https://www.raspberrypi.org. Acessado em: agosto de 2016.

Ruby (2016). URL: https://www.ruby-lang.org. Acessado em: agosto de 2016.

Ruby Daemons (2016). URL: http://daemons.rubyforge.org. Acessado em: agosto de 2016.

Ruby on Rails (2016). URL: http://rubyonrails.org. Acessado em: agosto de 2016.

Santos, B. P., Silva, L. A. M., Celes, C. S. F. S., Neto, J. B. B., Peres, B. S., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. A. F. (2016). *Livro dos Minicursos do SBRC 2016*, pages 1–50. Sociedade Brasileira de Computação.

Savolainen, T., Soininen, J., and Silverajan, B. (2013). IPv6 Addressing Strategies for IoT. *IEEE Sensors Journal*, 13(10):3511–3519.

Sehgal, A., Perelman, V., Kuryla, S., and Schonwalder, J. (2012). Management of resource constrained devices in the internet of things. *IEEE Communications Magazine*, 50(12):144–149.

Shelby, Z., Hartke, K., and Bormann, C. (2014). RFC 7252: The constrained application protocol (CoAP).