UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Luan Carlos de Oliveira Moraes

Framework de Comunicação seguro e confiável para Internet das Coisas usando o protocolo XMPP

São Luís - MA

2016

Luan Carlos de Oliveira Moraes

Framework de Comunicação seguro e confiável para Internet das Coisas usando o protocolo XMPP

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito para a obtenção do grau de MESTRE em Engenharia de Eletricidade.

Orientador: Denivaldo Cicero Pavão Lopes

Doutor em Informática – UFMA

São Luís - MA

2016

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Moraes, Luan Carlos de Oliveira.
Framework de Comunicação seguro e confiável para Internet das Coisas usando o protocolo XMPP / Luan Carlos de Oliveira Moraes. - 2016.
97 f.

Orientador(a): Denivaldo Cicero Pavão Lopes.
Dissertação (Mestrado) - Programa de Pós-graduação em Engenharia de Eletricidade/ccet, Universidade Federal do Maranhão, São Luís, 2016.

1. Confiabilidade em Redes. 2. Framework. 3. Internet das Coisas. 4. Segurança em Redes. 5. XMPP. I. Lopes, Denivaldo Cicero Pavão. II. Título.

Luan Carlos de Oliveira Moraes

Framework de Comunicação seguro e confiável para Internet das Coisas usando o protocolo XMPP

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Luan Carlos de Oliveira Moraes e aprovada pela comissão examinadora.

Aprovada em ___ de _____ de 2016

BANCA EXAMINADORA

Denivaldo Cicero Pavão Lopes (orientador)

Doutor em Informática – UFMA

Francisco José da Silva e Silva

Doutor em Ciências da Computação – UFMA

Eveline de Jesus Viana Sá

Doutora em Engenharia Eletrônica e Computação – IFMA

A Deus, por sempre me abençoar. A meus pais, meus irmãos e amigos por estarem sempre ao meu lado, me incentivando e ajudando.

Resumo

A Internet das Coisas (IoT) é um paradigma no qual objetos inteligentes colaboram de forma ativa com outros objetos físicos e virtuais disponíveis na Internet. Ambientes de IoT são caracterizados por um alto grau de heterogeneidade de dispositivos e protocolos de rede. Entretanto, muitas questões tecnológicas desafiadoras e sociais ainda precisam ser abordadas, incluindo a interoperabilidade de dispositivos, autonomia de sistemas, privacidade e questões de segurança, o quê poderia ter um impacto significativo em vários aspectos da vida cotidiana do usuário potencial final. Para lidar com essas questões algum tipo de camada de middleware ou *frameworks* se mostram fundamentais para fazer cumprir a integração de dispositivos e funcionalidades dentro da mesma rede de informação, provendo segurança e confiabilidade. Desta forma, o objetivo deste trabalho é projetar e implementar um *Framework* seguro e confiável neste cenário usando o protocolo *XMPP (eXtensible Messaging and Presence Protocol)*. Baseado no modelo *publish/subscribe*, o *framework* proposto possui mecanismo de confiabilidade na comunicação em tempo real, recursos de segurança fornecidos pelo *XMPP* baseado no protocolo *Transport Layer Security* (TLS) e autenticação *Simple Authentication and Security Layer* (SASL). Com base nos estudos de caso, demonstramos a capacidade do framework na confiabilidade e os resultados obtidos demonstram a viabilidade do mesmo.

Palavras-chaves: Internet das Coisas, Segurança em Redes, Confiabilidade em Redes, *XMPP*, *Framework*.

Abstract

The Internet of Things (IoT) is a paradigm in which smart objects collaborate actively with other physical and virtual objects available in the Internet. IoT environments are characterized by a high degree of heterogeneity of devices and network protocols. However, many challenging social and technological issues still need to be addressed, including the interoperability of devices, autonomous systems, privacy and security issues, which could have a significant impact on many aspects of everyday life of potential end user. To deal with these issues some type of middleware layer or frameworks to show fundamental enforce the seamless integration of devices and functionality within the same network information, providing security and reliability. Therefore, the objective of this work is to design and implement a secure and reliable framework in this scenario using the XMPP protocol (eXtensible Messaging and Presence Protocol). Based on the model publish / subscribe, the proposed framework has reliability mechanism for real-time communication, security features provided by the XMPP based on TLS and SASL authentication. Based on case studies demonstrate the framework's ability reliability and the results demonstrate the feasibility of the model.

Keywords: Internet of Things. Network Security. Reliability Networks. *XMPP*. *Framework*.

Agradecimentos

Agradeço primeiramente a Deus por me conceder saúde e sabedoria, para que fosse possível a conclusão deste trabalho.

Aos meus pais, João da Costa Moraes e Maria Raimunda de Oliveira Moraes e aos meus irmãos por todo o amor, carinho, amizade e paciência demonstrada ao longo da minha vida, bem como a educação e os valores que me transmitiram, tornando - me na pessoa que sou hoje.

Ao meu orientador Prof. PhD. Zair Abdelouahab *(in memorian)*, um sábio ser humano e educador exemplar, uma pessoa que tive um enorme prazer em conhecer e estar durante o mestrado, levo para minha vida seus ensinamentos, sua força de vontade, sua dedicação ao ensino, e acima de tudo a felicidade de viver. Agradeço pelo seu apoio, compreensão e sempre esteve disposto a ajudar e me mostrar sua amizade, respeito e confiança. A este, expresso minha sincera gratidão.

Ao meu coorientador Prof. Dr. Denivaldo Cícero Pavão Lopes, pelo acompanhamento durante os seminários e sua grande ajuda e sugestões na pesquisa.

Aos amigos de laboratório, Mário Braga, Cláudio Aroucha, Willian, Higo Felipe, Renato Ubaldo, Jhonatan Yuri, Dhully, Natália Soeiro, Luiz Aurélio, pela amizade, pelo apoio e pelo conhecimento que dividiram comigo, em especial ao Mario Braga por me ajudar diretamente na codificação do mecanismo proposto.

A Coordenação e aos professores do Programa de Pós-Graduação em Engenharia de Eletricidade, pela oportunidade da realização do curso e que nos forneceram conhecimento suficiente para a elaboração deste trabalho contribuindo com seus ensinamentos.

A CAPES pelo financiamento da bolsa de estudo.

Portanto agradeço a todos aqueles que direta ou indiretamente, contribuíram para o desenvolvimento deste trabalho. Meus sinceros agradecimentos.

"A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro."

Albert Einstein

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AMQP *Advanced Message Queuing Protocol*.

CoAP *Constrained Application Protocol*.
CoRE *Constrained RESTful Environments*.

DTLS *Datagram Transport Layer Security*.

EMF *Eclipse Modeling Framework*.

IEEE *Institute of Electrical and Electronics Engineers*.
IETF *Internet Engineering Task Force*.

M2M *Machine-to-Machine*.
MQTT *Message Queue Telemetry Transport*.

NIST *National Institute of Standards and Technology*.

OSI *Open System Interconnection*.
OWL *Web Ontology Language*.

RDF *Resource Description Framework*.
RFC *Request for Comments*.

SASL *Simple Authentication and Security Layer*.
SOAP *Simple Object Access Protocol* .
SSL *Secure Sockets Layer*.

TLS *Transport Layer Security*.

UDP *User Datagram Protocol*.

URI *Uniform Resource Identifier*.

W3C *World Wide Web Consortium*.

XML *Extensible Markup Language*.

XMPP *Extensible Messaging and Presence Protoco* .

XSF *XMPP Standards Foundation*.

Sumário

1 Introdução

Esta dissertação apresenta um *Framework* seguro e confiável para Internet das Coisas usando o protocolo XMPP, bem como um estudo de caso no qual avalia-se o framework de acordo com requisitos propostos. Nesta avaliação, utiliza-se um ambiente para caracterizar o cenário da Internet das Coisas.

Este Capítulo apresenta uma visão geral do trabalho. A seção 1.1 caracteriza a motivação e a justificativa para o desenvolvimento desta dissertação. A seção 1.2 expõe a problemática do trabalho. Descrevem-se os objetivos gerais e específicos na seção 1.3, a metodologia da pesquisa na seção 1.4 e, por último, a seção 1.5 apresenta a estrutura na qual a dissertação está organizada.

1.1 Motivação e Justificativa

Diante dos vários tipos de tecnologias criadas para a Internet a fim de conectar cada vez mais pessoas e variados dispositivos, tem-se o desafio de facilitar a interação com o mundo real o que envolve os objetos que estão ao nosso redor e desconectados da rede. Esta nova fase da Internet visando a conexão de múltiplos dispositivos promete ser ainda mais interativa e dinâmica.

Essa característica de conectar e representar os objetos físicos, reais, através de uma conexão de Internet, representa a chamada Internet das Coisas (inglês: *Internet of Things* - IoT). Gartner [27] define a Internet das Coisas como "uma rede de objetos físicos que contém tecnologia embarcada para se comunicar e interagir com os dispositivos internos ou com o ambiente externo". Com essa tecnologia temos uma série de dispositivos e sensores conectados compartilhando dados.

Uma das propostas da IoT é permitir, com o uso de tecnologias de rastreamento, identificação e troca de informações, que numerosos objetos comuniquem-se automaticamente e à distância. A IoT oferece novas oportunidades de projetos para aplicações interativas as quais, além de conter documentos estáticos, conterão informação em tempo real referentes a lugares e objetos do mundo físico [19].

A mobilidade e a conectividade ilimitada das pessoas com o mundo ao redor, por meio do uso dos dispositivos móveis, já é uma realidade e o crescimento dessa tendência é uma das principais apostas de empresas focadas no futuro.

De acordo com a previsão do Gartner [26], a IoT, o que exclui PCs, *smartphones*, e *tablets*, vai crescer para mais de 26 bilhões de unidades instaladas até 2020, permitindo assim que cada único objeto físico se conecte à Internet e compartilhe informações. Com esse crescimento várias questões desafiadoras poderão surgir.

Os desafios tecnológicos e sociais que essa conectividade impõe ainda precisam ser abordadas, incluindo a interoperabilidade entre dispositivos, escasso limite de armazenamento, a autonomia dos sistemas, questões de segurança, privacidade e confiança. Isto porque a conexão das "Coisas" (ou objetos) na Internet pode ser realizada com a ajuda de vários protocolos e padrões, seja ela adotada a partir da Internet tradicional e de outras tecnologias (ex: Wi-Fi, *Bluetooth*, *Ethernet*, 3G e 4G-LTE, HTTP), ou de outras formas especificamente adaptadas para atender às limitações das "coisas" conectadas.

Para contornar esses desafios na IoT, um conceito comumente associado está o uso de *Middleware* [48] e *Frameworks* [15] de modo a permitir a interoperabilidade e integração dos diversos componentes que fazem parte desses ambientes, tendo em vista que há vários mecanismos (ou protocolos) e não há ainda um protocolo padrão para tal função.

A utilização de protocolos como MQTT, XMPP, CoAP e AMQP [32] são alternativas para prover as funcionalidades essenciais da IoT, incluindo a mobilidade, intermitência de comunicação, extensibilidade, escalabilidade e capacidade de autoconfiguração exigidas por diferentes objetos inteligentes.

Diante disso, tem sido aplicados no mercado o uso de diversos *frameworks*, capazes de minimizar os problemas relacionados a segurança, privacidade, confiabilidade, e ainda questões de interconexão de dispositivos digitais, de carros aos eletrodomésticos, passando pelos celulares e computadores. No entanto, várias abordagens focadas na Internet das Coisas destacam a confiabilidade através da utilização do protocolo TCP *(Transmission Control Protocol)* em português Protocolo de Controle de Transmissão, para realizar a transmissão confiável entre os diversos dispositivos na Internet das Coisas.

Essa é uma abordagem muito utilizada, no entanto, no ambiente da Internet das Coisas, faz-se necessário utilizar de outros mecanismos ou protocolos para perfeita garantia de entrega de mensagem. Tendo em vista que nesse ambiente há diversos fatores que influenciam na garantia de entrega, tais como análise da disponibilidade dos dispositivos, disponibilidade da rede, entre outros.

Visando abordar essas questões relacionadas a confiabilidade é que se destaca modelos baseado em *framework* onde utiliza informações contextuais para uma melhor análise do estado do dispositivo e da rede que ocorrem durante a transmissão das informações e assim garantir a entrega da mensagem.

Em adição, utilizara-se-á o servidor no ambiente de Computação em Nuvem para resolver o limite de armazenamento que possui os dispositivos de Internet das Coisas, e, para garantir que os dados trafegados entre a aplicação e o servidor sejam realizados de forma segura, utiliza-se a criptografia e a autenticação. A criptografia será utilizada para codificar as mensagens enviadas dos dispositivos ao servidor da Nuvem de forma a garantir confidencialidade e a autenticação na sua comunicação.

1.2 Problemática

Ao desenvolver aplicações no ambiente da IoT, é preciso considerar algumas questões de conectividade, segurança e privacidade entre outros. Aplicações para IoT estão cada vez mais dispostas na Internet, coletando, enviando dados e informações para variados locais ou dispositivos. Como exemplo, temos os dispositivos de saúde que enviam informações para os médicos, relógios de pulso que permitem enviar nosso batimento cardíaco para outra pessoa, aplicações de localização geográfica precisam enviar dados, entre outras aplicações. Ou seja, requisitos como, segurança, privacidade e confiabilidade estão se tornando mais evidentes neste cenário diante dos modelos de aplicações [44].

Hoje, milhões de dispositivos expõem o que eles vêem, ouvem ou de alguma maneira sentem para a Internet. E graças a sistemas baratos embutidos, não precisam mais de antigo sistemas para realizar tal função. Bilhões de outros dispositivos que desafiam a definição de “computador” estão se comunicando através

de rede, quase completamente, com outra máquinas. Muitos dispositivos da Internet das Coisas enviam, recebem instruções, dados ou informações de softwares hospedados em servidores em qualquer lugar do mundo ou mesmo no ambiente de Nuvem [40]. Comunicações entre dispositivos na Internet das Coisas e a nuvem estão se tornando mais evidentes, visto que é necessário muitas vezes transmitir dados significativos para serviços na Nuvem.

A confiabilidade e a segurança nas comunicações se tornam cruciais neste ambiente. A verificação de dados de contexto no qual o dispositivo está inserido é muito importante para provê o perfeito envio e recebimento de dados (confiabilidade) nas comunicações sobre o protocolo TCP. A adição do uso de protocolo de criptografia (TLS) para codificar as mensagens enviadas dos dispositivos ao servidor da Nuvem garantem a confidencialidade e a autenticação na comunicação.

Desta forma, surgiu a necessidade de criar um *framework* seguro e confiável para Internet das Coisas. A proposta consiste em criar uma solução no qual implementa os mecanismo de confiabilidade de comunicação e segurança na comunicação com servidores na Nuvem.

1.3 Objetivos Gerais e Específicos

Esta dissertação propromos o *Framework* - *Framework Reliable for Internet of Things* - FRI com mecanismos de comunicação seguro e confiável no ambiente IoT e a Nuvem usando o protocolo XMPP. Visando avaliar o *framework* proposto, é verificado sua aplicabilidade e realizado testes segundo casos de uso propostos neste trabalho.

A partir das técnicas que serão utilizadas no decorrer da proposta, este *framework* será capaz de realizar a comunicação confiável em tempo real, sobretudo quando houver falha de comunicação entre a aplicação e o servidor em nuvem, houver desconexão, provendo tratamento de erros nas comunicações.

O mecanismo é composto pelo protocolo de comunicação XMPP no qual realizará a interoperabilidade no ambiente IoT, um mecanismo confiável de comunicação, mecanismo de adaptação para diferentes arquiteturas.

Esta dissertação tem por objetivos específicos:

1. O FRI deve permitir um desenvolvimento simplificado e eficiente em termos de custos e implantação de serviços de comunicação entre objetos na Internet das Coisas;

2. Realizar toda comunicação entre objetos de forma segura e confiável em termos de conexão;

3. Deve apoiar a integração de aplicativos entre dispositivos móveis e a Nuvem;

4. Deve verificar a aplicabilidade do framework a uma variedade de aplicações, de forma que o mesmo possa ser utilizado em diversos cenário de aplicações.

1.4 Metodologia de pesquisa

Esta dissertação objetiva desenvolver um estudo e uma proposta para a solução do problema de confiabilidade na comunicação no ambiente IoT. O primeiro passo consiste no levantamento bibliográfico em livros, teses, artigos científicos, anais de congressos, dissertações, periódicos, websites e biblioteca desta instituição de ensino, visando à aquisição da fundamentação teórica e à produção de relatórios das áreas de Internet das Coisas, Segurança da Informação e Computação em Nuvem. No segundo passo, a definição do escopo do problema será realizada, delimitando até que ponto será a contribuição da pesquisa. Espera-se que nesta fase sejam conhecidos os requisitos de segurança, protocolo de comunicação, implantação na IoT, e a implementação do *Framework*. Em seguida, a implementação do ambiente de testes será feita, que tem objetivo de testar o funcionamento do *Framework* desenvolvido, que utilizará como dispositivo de teste um Raspberry Pi 3.

1.5 Estrutura da Dissertação

Esta dissertação se encontra organizada em seis capítulos, descritos a seguir.

O Capítulo 2 apresenta uma visão geral sobre Internet das Coisas, protocolos de comunicação para IoT, Computação em Nuvem, Segurança em redes e

Confiabilidade em rede, destacando suas características, e em seguida, dando enfoque à confiabilidade nas comunicações, considerada neste trabalho.

O Capítulo 3 apresenta os trabalhos relacionados sobre *frameworks* para IoT.

O Capítulo 4 apresenta o mecanismo proposto para confiabilidade. A arquitetura, funcionamento e todos os componentes são mostrados.

O Capítulo 5 apresenta o protótipo para comunicação confiável e os resultados da dissertação.

Por fim, o Capítulo 6 apresenta as conclusões obtidas no desenvolvimento deste trabalho e possíveis trabalhos futuros. As considerações finais desta dissertação são apresentadas nesta seção.

Todo este projeto e sua documentação está disponível livremente sob a licença LGPL *(Lesser General Public License)* no endereço eletrônico https://github.com/labsac/FRI-Framework. Para maiores detalhes sobre a licença LGPL, acesse o endereço eletrônico http://www.gnu.org/licenses/lgpl.html.

2 Fundamentação Teórica

Neste capítulo, s conceitos fundamentais das tecnologias que servira de base para o desenvolvimento do projeto são apresentados. O entendimento de cada tecnologia foi essencial para melhor integrá-las e construir o alicerce do desenvolvimento da proposta do *Framework* para provê comunicações confiáveis.

2.1 Internet das Coisas

Atualmente graças ao progresso no campo de dispositivos embarcados, objetos físicos tais como eletrodomésticos, máquinas industriais, sensores sem fio e atuadores podem atualmente se conectar a Internet. A conexão desses objetos do dia a dia com a Internet é denominada Internet das Coisas (do inglês, *Internet of Things* - IoT) [27]. Esses objetos ou "coisas" podem ser quaisquer dispositivos, tais como, pneus, sensores, atuadores, telefones celulares, entre outros, que possam ser identificados e interligados à Internet para trocar informações e tomar decisões para atingir objetivos comuns [8].

Essa rede de objetos interconectados não só colheria informações do ambiente (sensores) e interação com mundo físico (atuação/comando/controle), mas também utiliza os padrões existentes da Internet para fornecer serviços de transferência de informações, análise, aplicativos e comunicações.

Alimentado pelo domínio de dispositivos habilitados pela tecnologia sem fio aberta, como *bluetooth*, identificação por radiofrequência (RFID), Wi-Fi, serviços de dados e telefonia assim como nós sensores e atuadores embutidos, a IoT cresce cada vez mais e está transformando a Internet atual em uma Internet totalmente integrada [30].

Para L. Atzori *et al.* [8] e L. Tan e N.Wang [56], a IoT vem ganhando grande destaque no cenário das telecomunicações e está sendo considerada a revolução tecnológica que representa o futuro da computação e comunicação. Devido a importância da IoT, o Conselho Nacional de Inteligência dos EUA (*National Intelligence*

Council - NIC) a considera como uma das seis tecnologias civis mais promissoras e que mais impactarão a nação no futuro próximo. O NIC prevê que em 2025 todos os objetos do cotidiano (por exemplo, embalagens de alimento, documentos e móveis) poderão estar conectados à Internet [19].

Para Gartner [26], a Internet das Coisas vai chegar a 26 bilhões de unidades em 2020, indo de 0,9 bilhões em 2009, e terá impacto sobre a informação disponível para conectar as empresas como os fornecedores de produtos. A partir da linha de produção e armazenamento da entrega de varejo a prateleiras das lojas, a Internet das Coisas está a transformar os processos de negócios.

Segundo I. Lee e K. Lee [35], a IoT proporciona de forma mais precisa a visibilidade em tempo real do fluxo de materiais e produtos, tendo em vista a utilização da tecnologia para redesenhar os fluxos de trabalho de fábrica, melhorar o rastreamento de materiais, e otimizar os custos de distribuição.

Graças ao paradigma IoT, estima-se que uma grande quantidade de objetos estarão conectados à Internet, se tornando os maiores emissores e receptores de tráfego da rede. A Figura 2.1 mostra as novas dimensões do mundo das tecnologias da comunicação e informação da Internet no futuro. Nessa figura é possível observar "o que" pode ser conectado a Internet, "quando" pode ser conectado e "onde" pode se conectar [19].

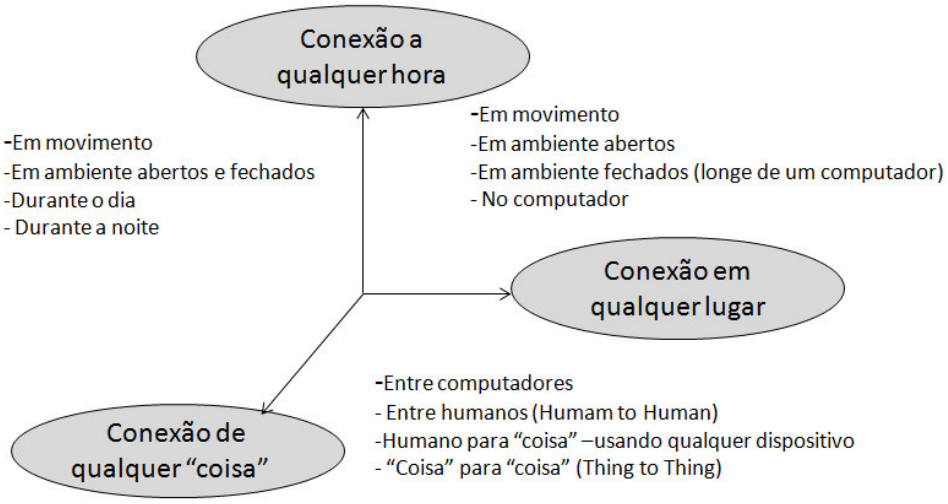


Figura 2.1: Uma Nova Dimensão para IoT [19].

Para A. Gluhak *et al.* [28], as aplicações na IoT podem ser classificados com base em vários fatores, tais como: disponibilidade de rede, cobertura, escala,

heterogeneidade, repetibilidade, participação do usuário e impacto. Já J. Gubbi *et al.* [30], categoriza as aplicações em quatro domínios: *Home*, *Transport*, *Community*, *National*, como está representado na Figura 2.2.

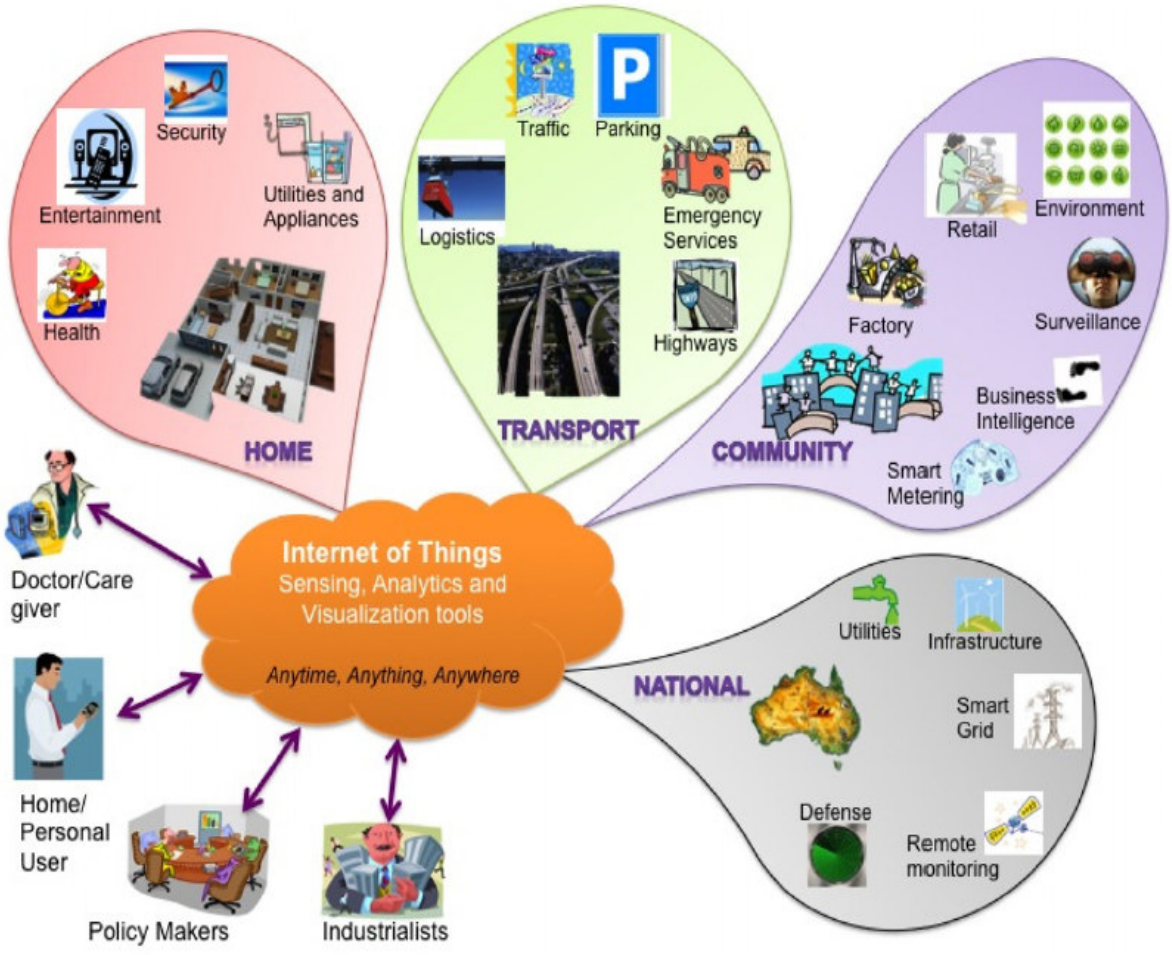


Figura 2.2: Internet das Coisas mostrando os usuários finais e áreas de aplicação com base nos dados. [30].

As aplicações do tipo *Home* representa a IoT na escala de um indivíduo ou de casa, como exemplo temos sistema de monitoramento em casa para o cuidado de idosos, o que permite ao médico acompanhar os doentes e os idosos em suas casas, reduzindo assim os custos de hospitalização por meio de intervenção precoce e tratamento. Também temos aplicações de controle de equipamentos de casa como, ar-condicionado, máquinas de lavar, permitindo uma melhor gestão da casa e energia.

Ainda se tratando de aplicações *Home*, J. Gubbi *et al.* [30] cita a utilização de Rede Social, neste caso o Twitter como conceito onde as 'coisas' poderão periodicamente twittar as leituras que podem ser facilmente seguidos de qualquer lugar com a criação de um *TweetOT*.

As aplicações do tipo *Community* refere-se a "Rede de coisas"dentro de um ambiente de trabalho onde são utilizadas apenas pelos proprietários e os dados podem ser libertados seletivamente, como exemplo temos: o monitoramento do ambiente através de sensores, segurança, iluminação, controle de temperatura, etc.

Na classificação do tipo *National* o domínio de aplicações é geralmente para otimização de serviços em vez de consumo. Este tipo de aplicação está sendo usado por empresas de serviços públicos (medidor inteligente por empresas de fornecimento de eletricidade) para a gestão dos recursos, a fim de otimizar o custo versus lucro, tais como: *Smart grid*, monitoramento de rede água potável, monitoramento de irrigação em terrenos agrícolas, monitoramento de parâmetros do solo que permite tomada de decisão e informação em relação à agricultura, etc.

Nas aplicações do tipo *Transport*, como transporte inteligente e logística inteligente são colocados em um domínio separado, devido à natureza do compartilhamento dos dados e sua infraestrutura necessária. O tráfego urbano é um dos principais contribuintes para a qualidade do ar urbano, emissões de gases poluentes, etc. O congestionamento do trânsito cria custos significativos sobre as atividades econômicas e sociais das cidades, diante dessas questões as aplicações da IoT permitem fornecer informações para um melhor planeamento e uma melhor programação das operações de logísticas através das redes de sensores existentes por exemplo. A IoT permitirá a utilização de redes de sensores em larga escala para o monitoramento on-line de tempos de viagem, origem-destino, comportamento de escolha de rota, monitoramento de poluentes do ar e ruído, entre outros.

A grande variedade de padrões e tecnologias nestes cenários e a forma como eles devem interoperar é o principal desafio que pode impedir o desenvolvimento de aplicações da Internet das Coisas. A heterogeneidade dos elementos da Internet das Coisas precisa de uma solução completa para tornar os serviços de Internet das coisas onipresentes uma realidade [30].

Para A. Al-Fuqaha *et al.* [3], os principais elementos e tecnologias envolvidas para fornecer as funcionalidades na IoT podem ser classificados como: identificação, sensoriamento, comunicação, computação, serviços e semântica. O resumo desses elementos e os exemplos das tecnologias empregadas são mostradas na Figura 2.3.

IoT Elements		Samples
Identification	**Naming**	EPC, uCode
	Addressing	IPv4, IPv6
Sensing		Smart Sensors, Wearable sensing devices, Embedded sensors, Actuators, RFID tag
Communication		RFID, NFC, UWB, Bluetooth, BLE, IEEE 802.15.4, Z-Wave, WiFi, WiFiDirect, , LTE-A
Computation	**Hardware**	SmartThings, Arduino, Phidgets, Intel Galileo, Raspberry Pi, Gadgeteer, BeagleBone, Cubieboard, Smart Phones
	Software	OS (Contiki, TinyOS, LiteOS, Riot OS, Android); Cloud (Nimbits, Hadoop, etc.)
Service		Identity-related (shipping), Information Aggregation (smart grid), Collaborative-Aware (smart home), Ubiquitous (smart city)
Semantic		RDF, OWL, EXI

Figura 2.3: Tecnologias quem envolvem a construção da IoT. [3].

Na Figura 2.3, o elemento de identificação de objetos *(Identification)* é mostrado para nomear e comparar serviços de acordo com sua demanda. Muitos métodos de identificação estão disponíveis para a Internet das Coisas tais como códigos eletrônicos de produtos (EPC) e códigos ubíquos (UCODE). Os métodos de endereçamento de objetos *(Addressing)* da Internet das Coisas inclui o IPv6 e IPv4, utilizados para proporcionar uma identidade clara para cada objeto no interior da rede.

A utilização de Sensoriamento *(Sensing)* para recolher dados a partir de objetos relacionados dentro da rede e enviá-los de volta para um armazenamento de dados, banco de dados, ou em nuvem. Os dados coletados são analisados para tomar ações específicas baseadas em serviços de acordo com a necessidade, para tanto se utiliza das tecnologias RFID, *Smart Sensors*, *Wearable sensing devices*.

Normalmente, os nós da Internet das Coisas deve operar utilizando baixa potência na presença de links de comunicação com perdas e ruidosos. Exemplos de

protocolos de comunicação utilizados são WiFi, Bluetooth, IEEE 802.15.4, Z-wave, e LTE-Advanced.

Várias plataformas de hardware foram desenvolvidos para executar aplicações da Internet das Coisas, tais como: Arduino, UDOO, FriendlyARM, Intel Galileo, Raspberry Pi, Inventor, BeagleBone, Cubieboard, Z1, WiSense, Mulle, Sky T-Mote e Smart Phones. Além disso, muitas plataformas de software são utilizados para fornecer funcionalidades, tais como: Contiki OS com simulador chamado Cooja que permita pesquisadores e desenvolvedores simular e emular rede de sensores sem fio (RSSF) aplicações , TinyOS , LiteOS , Motim OS e Android OS projetado para ambientes da Internet das Coisas [3].

Os serviços de IoT podem ser categorizadas em quatro classes segundo A. Al-Fuqaha *et al.* [3], são eles: Serviços relacionados com a identidade, informações de agregação de serviços, serviços colaborativos-consciente. Podemos citar alguns exemplos de aplicações baseadas nessas categorias: *smart healthcare*, *smart grids*, Casa inteligente, edifícios inteligentes, sistemas inteligentes de transporte (ITS), etc.

A utilização de semântica para IoT se refere à capacidade de extrair conhecimento de forma inteligente por máquinas diferentes para fornecer os serviços necessários. Na extração de conhecimento inclui descobrir e usar recursos e modelagem de informações, apoiada por tecnologias de Web Semântica, como o *Resource Description Framework* (RDF), *Web Ontology Language* (OWL) e XML Interchange (EXI) para dispositivos com recursos limitados.

Para facilitar e simplificar os trabalhos dos programadores de aplicativos e prestadores de serviços, vários grupos diferentes foram criados para fornecer protocolos de apoio à Internet das Coisas, incluindo os esforços liderados pelo *World Wide Web Consortium* (W3C), *Internet Engineering Task Force* (IETF), EPCglobal, e *Institute of Electrical and Electronics Engineers* (IEEE) e as Normas Europeias de Telecomunicações Institute (ETSI). A Figura 2.4, fornece um resumo dos protocolos definidos por esses grupos.

Na Figura 2.4 os principais protocolos estão divididos como: Descoberta de serviços, Protocolo de roteamento, Camada de rede, Camada física/ Camada de Dispositivo e outros protocolos influentes. A descrição de alguns desses protocolos estão detalhados na subseção 2.4.

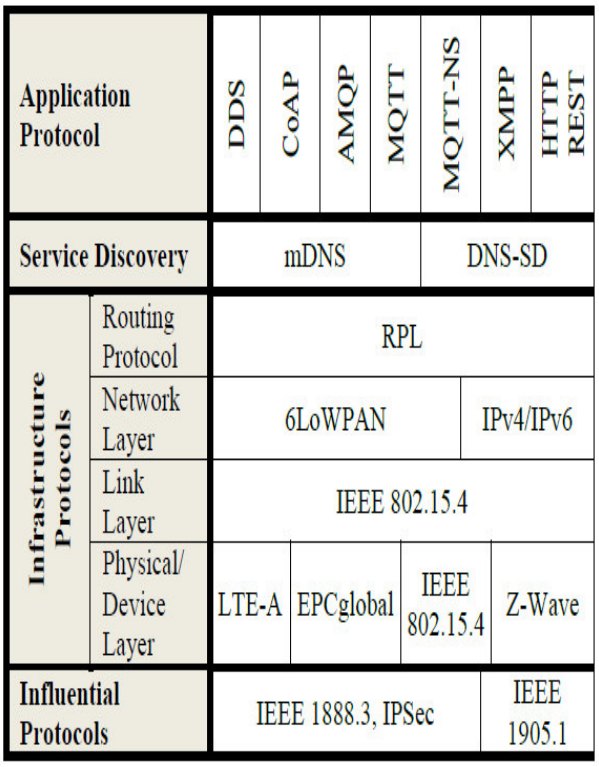


Figura 2.4: Resumo dos protocolos para desenvolvimentos de aplicações para IoT. [3].

2.2 Computação em Nuvem

A computação em nuvem é uma das tecnologias mais utilizadas nos dias de hoje. Os serviços fornecidos pela Nuvem tem uma grande vantagem quando comparados com os serviços tradicionais. As vantagens dessa tecnologia proporciona ao usuário serviços de fácil acesso e de baixo custo e garantindo características como disponibilidade e escalabilidade.

De acordo com a definição do *National Institute of Standards and Technology* (NIST), Computação em Nuvem é um modelo para permitir o acesso à rede conveniente, ubíquo e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor de serviços [39].

A nuvem é uma representação para a Internet ou infraestrutura de comunicação entre os componentes arquiteturais, baseada em uma abstração que oculta à complexidade de infraestrutura. Cada parte desta infraestrutura é provida

como um serviço e, estes serviços são normalmente alocados em data centers, utilizando hardware compartilhado para computação e armazenamento [50].

Segundo N. Fernando *et al.* [23], a virtualização dos recursos é o requisito fundamental para tornar a nuvem escalável, criando a ilusão de recursos computacionais infinitos ao usuário da nuvem. Para NIST a divisão desse modelo se dá em cinco características essenciais, três modelos de serviço e quatro modelos de implantação, apresentado na Figura 2.5 que será descrito nas seções seguintes.

Figura 2.5: Modelo visual da definição de Computação em Nuvem [22]

As principais características definidas pelo NIST como mostra na Figura 2.5 são:

- **Amplo Acesso à Rede:** o compartilhamento de recursos são disponibilizados pela rede e acessados por meio de mecanismos padronizados que promovem o uso de plataformas clientes heterogêneas (por exemplo, celulares, *notebooks*, *tablets*, e estações de trabalho, dispositivos de Internet das Coisas, etc). Desta forma, poderá acessar de qualquer lugar, a qualquer hora, de qualquer dispositivo, desde que tenham conexão (Internet);

- **Elasticidade Rápida:** Ao solicitar os recursos da nuvem o usuário passará a ter a ilusão de recursos ilimitados. Para o consumidor, as capacidades disponíveis para provisionamento frequentemente parecem ser ilimitadas que podem ser apropriadas em qualquer quantidade e a qualquer momento;

- **Serviço Mensurado:** os sistemas em nuvem automaticamente controlam e otimizam o uso dos recursos por meio de uma capacidade de medição em algum nível apropriado de abstração para o tipo de serviço. Com isso, todas as atividades e serviços são transparentes tanto para o provedor como também para o consumidor do serviço utilizado;

- ***Self-Service* sob demanda:** um consumidor pode solicitar unilateralmente recursos computacionais, com a minima interação humana com os provedores de cada serviço;

- **Pooling de Recurso:** o consumidor só terá que se preocupar com um dispositivo que seja capaz de acessar a Internet, pois todos os recursos serão fornecidos (exemplos de recursos incluem o armazenamento, processamento, memória, largura de banda, de rede e máquinas virtuais). Os recursos oferecidos pela nuvem serão liberados quando o usuário fizer uma solicitação, podendo ser em pequena ou larga escala de acordo com a necessidade. Exemplos de recursos incluem o armazenamento, processamento, memória e largura de banda de rede [22].

2.2.1 Modelos de Serviços

O ambiente da Computação em Nuvem é dividido em três modelos de serviços, como mostrado na Figura 2.5. O detalhamento desses modelos segundo o NIST é ilustrado na Figura 2.6 e descrito como segue:

- **IaaS:** Esse modelo fornece a infraestrutura necessária como um serviço. O consumidor não gerencia ou controla a infraestrutura subjacente da nuvem mas tem o controle sobre os sistemas operacionais, armazenamento, e aplicações implantadas; e possivelmente controle limitado a selecionar componentes de rede;

- **PaaS:** Um ambiente de desenvolvimento é fornecido, ou seja, uma plataforma de computação utilizando a infraestrutura da nuvem. A capacidade fornecida ao consumidor é permitir implementar na infraestrutura da nuvem aplicações

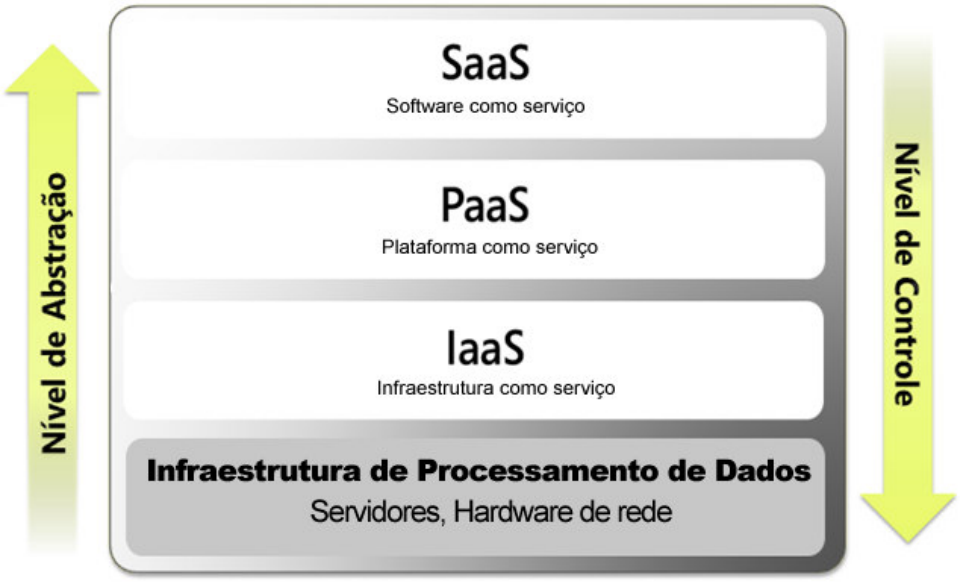


Figura 2.6: Modelos de Serviços, [22]

utilizando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. Exemplo Microsoft Azure;

- **SaaS**: a capacidade fornecida ao consumidor é usar as aplicações do provedor executando na infraestrutura da nuvem. As aplicações são acessíveis a partir de vários dispositivos clientes através de uma interface cliente, como um navegador *web* (por exemplo, email baseado na web), ou uma *interface* de programa, permitindo aos clientes acessar esses aplicativos remotamente através da Internet. Temos como exemplo desse tipo de *softwares* como serviço: Google Apps, SalesForce (SFDC), NetSuite, Oracle, Microsoft.

2.2.2 Modelos de Desenvolvimento

Para R. Buyya, J. Broberg, e A. M. Goscinski [16], modelos de nuvem podem ser adotados de acordo com a necessidades das aplicações, da abertura de acesso, modelo de negócio e do tipo de informação e nível de visão desejado. Ou seja, depende da disponibilização de recursos onde certas organizações desejam que utilizadores possam aceder e utilizar determinados recursos no seu ambiente de computação em nuvem. Desta forma, surge a necessidade de prover a visibilidade em ambientes mais restritos onde somente usuários devidamente autorizados possam utilizar os serviços providos. A nuvem pode ser classificada em:

- **Nuvem Pública:** As nuvens públicas são aquelas que são utilizadas por terceiros, sendo de propriedade de um provedor de serviços e os seus recursos vendidos ao público, [22]. Segundo o NIST, esta pode ser de propriedade, gerenciada, e operada por uma empresa, instituição acadêmica ou organização governamental;

- **Nuvem Privada:** Segundo A. Fox *et al.* [25] as nuvens privadas dizem respeito aos *data centers* de uma empresa ou organização que não está disponível para o público. Para Q. Zhang*et al.* [62] a nuvem privada oferece maior grau de controle sobre confiabilidade, desempenho, e segurança, mas que é frequentemente criticada por ser similar à fazendas de servidores proprietários tradicionais;

- **Nuvem Comunitária:** A nuvem é compartilhada por diferentes organizações que constituem uma comunidade que se dedica a um determinando assunto, interesses, missão, requisitos de segurança, política e sobre considerações de flexibilidade;

- **Nuvem Híbrida:** é definido pelo NIST como uma combinação de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública). Existe uma composição dos modelos que permanecem entidades únicas, mas são unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicativo (por exemplo, uma nuvem estourando para balancear a carga entre as nuvens).

2.3 Computação em Nuvem na IoT

Para A. Botta *et al.* [13], a Internet das Coisas representa uma das tecnologias mais inovadoras atualmente, permite cenários de computação ubíqua e pervasiva com armazenamento limitado e capacidade de processamento, envolve alguns obstáculos como, confiabilidade, desempenho, segurança e privacidade.

Por outro lado, a computação em nuvem tem capacidades virtualmente ilimitadas em termos de armazenamento e poder de processamento, é uma tecnologia muito mais madura, e tem a maioria das questões que envolvem a Internet das Coisas parcialmente resolvidos. Assim, um novo paradigma de TI surge, a computação

em nuvem e Internet das coisas se complementam e se mesclam criando uma nova Internet [63]. Esse novo paradigma é chamado *CloudIoT*.

Essas duas grandes tecnologias (Nuvem e IoT) tem passado por uma rápida e independente evolução. Estes mundos são muito diferentes um do outro e suas características são muitas vezes complementares. Esta complementaridade é a principal razão pela qual muitos pesquisadores têm proposto e estão a propor a sua integração, geralmente para obter benefícios em cenários de aplicação específicos [6] [1] [29].

Várias questões como acessibilidade, tipos de componentes, capacidades computacionais, armazenamento, *big data*, etc, são aspectos complementares da IoT e *cloud*, que quando juntas uma pode se beneficiar da capacidade da outra, ou mesmo compensar suas limitações tecnológicas. Como exemplo, a Nuvem pode oferecer uma solução eficaz para o gerenciamento de serviço de Internet das Coisas e sua composição, bem como para a implementação de aplicações e serviços que exploram as coisas ou os dados produzidos por eles. Por outro lado, a Nuvem pode se beneficiar da IoT aumentando o seu alcance para lidar com as coisas do mundo real de uma forma mais distribuída e dinâmica, e para a entrega de novos serviços em um grande número de cenários da vida real [13].

As principais motivações que conduzem a integração da Nuvem com a Internet das Coisas podem ser relacionados com: comunicação, armazenamento e processamento [13]. A maioria dos trabalhos científicos está realmente vendo a Nuvem como uma peça que faltava no cenário integrado, isto é, eles acreditam que a Nuvem preencherá algumas lacunas da IoT. Outros acreditam no contrário, que a IoT preencherá algumas lacunas da Nuvem. A seguir, vemos mais detalhadamente essas questões:

- **Comunicação:** Graças ao paradigma *CloudIoT*, as aplicações ubíquas personalizadas podem ser entregues através do IoT, enquanto a automatização pode ser aplicada tanto na coleta de dados como a distribuição de baixo custo. A Nuvem oferece uma solução eficaz e barata para se conectar, controlar e gerenciar qualquer coisa de qualquer lugar a qualquer momento usando portais customizados e aplicativos embutidos. A disponibilidade de redes de alta

velocidade permite monitoramento efetivo e controle de coisas remotas, suas coordenações, suas comunicações, e acesso em tempo real aos dados produzidos;

- **Armazenamento:** Internet das Coisas envolve, por definição, uma grande quantidade de fontes de informações (coisas), que produzem uma quantidade enorme de dados não-estruturados ou semi-estruturados, que também têm as 3 características típicas de Big Data: volume (isto é, tamanho de dados), variedade (ou seja, tipos de dados) e velocidade (ou seja, frequência de geração de dados). Armazenamento em larga escala e de longa duração, é possível graças ao virtualmente ilimitado baixo custo, e capacidade de armazenamento *on-demand* fornecidos pela Nuvem.

  A Nuvem é a solução mais conveniente e rentável para tratar de dados produzidos pela IoT e, a este respeito, isso gera novas oportunidades de agregação de dados, integração, e compartilhamento com terceiros. Uma vez dentro da nuvem, os dados podem ser tratados como homogêneos através de APIs (*Application Programming Interface*) bem definidas, podem ser protegidos através da aplicação de segurança de nível superior, e podem ser acessados diretamente e visualizados a partir de qualquer local;

- **Processamento:** Dispositivos da Internet das Coisas tem processamento e recursos de energia limitados que não permitem um complexo processamento de dados no local. Os dados coletados são normalmente transmitidos para os nódulos mais poderosos onde a agregação e o processamento são possíveis, mas a escalabilidade é um desafio de conseguir sem uma infraestrutura adequada. A Nuvem oferece capacidade de processamento ilimitado virtualmente e um modelo empregado sob demanda.

A adoção do paradigma *CloudIoT* permite que novos serviços inteligentes e aplicações baseadas na extensão da nuvem através das coisas lide com uma série de novos cenários da vida real. A literatura mostra como um número de novos paradigmas emergentes da integração entre Nuvem e Internet das Coisas relacionando a várias motivações de forma específica estão impulsionando a integração, alguns deles relacionados com cenários de aplicação específicos.

2.4 Protocolos de Comunicação na Internet das Coisas

Esta seção fornece um resumo dos protocolos mais utilizados para facilitar e simplificar os trabalhos de programadores de aplicativos para IoT. Vários grupos foram criados para fornecer protocolos de apoio, incluindo organizações afim de padronizar tais tecnologias. Nas subseções seguintes, apresenta-se os protocolos XMPP, MQTT, AMQP e CoAP a fim de se detalhar suas arquiteturas, utilização e suas principais funcionalidades.

2.4.1 XMPP

O protocolo XMPP é uma tecnologia aberta para comunicação em tempo real, usando o *Extensible Markup Language* (XML) como o formato de base para a troca de informações. Em essência, XMPP fornece uma maneira de enviar pequenos pedaços de XML a partir de uma entidade para outra quase em tempo real.

Esse protocolo fornece um conjunto de tecnologias abertas para mensagens instantâneas, criptografia de canal, autenticação, presença, mensagens multi usuários, notificações etc [60]. Para obter uma visão rápida dos recursos utilizados por todos os aplicativos baseados em XMPP, a arquitetura geral de sistemas, esquema de endereçamento para as comunicações e mecanismo de *streaming* de XML será discutido nas subseções a seguir:

2.4.1.1 Arquitetura do XMPP

A infra-estrutura da Internet para mensagens instantâneas, presença, e outras formas de comunicação em tempo real é cada vez mais composto por centenas de milhares de servidores Jabber rodando software como ejabberd e Openfire, e milhões de clientes Jabber que executam o software como: Psi, Adium, Gajim, Pidgin, etc. XMPP utiliza uma arquitetura cliente-servidor descentralizada [57] semelhante aos utilizados para a arquiteturas da rede de e-mail na Web como mostra na Figura 2.7:

Como mostrado na Figura 2.7, nessa arquitetura não há nenhum servidor com autoridade central, diferentemente de alguns serviços de mensagens

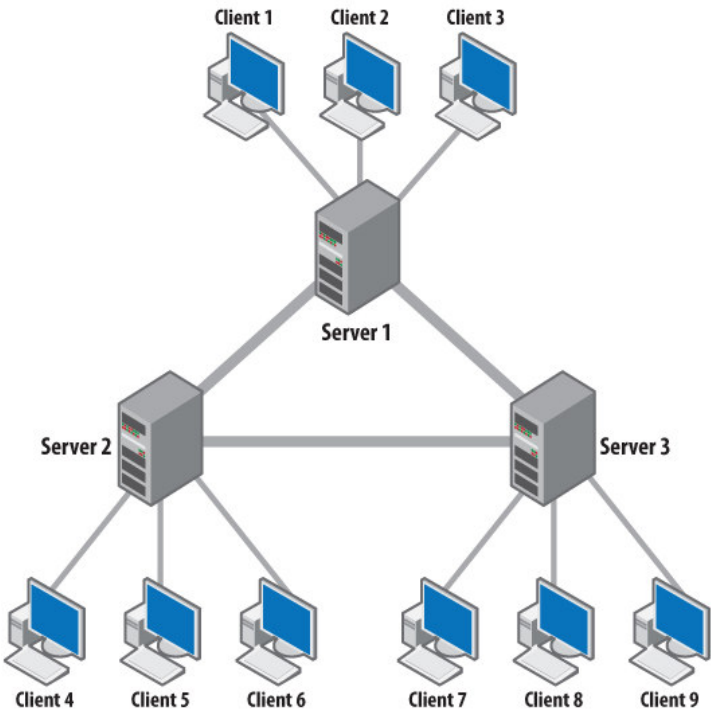


Figura 2.7: Arquitetura cliente-servidor descentralizada XMPP. [57].

instantâneas (IM) onde sua arquitetura é centralizada. Cada aplicação XMPP cliente (aplicação/entidade) está conectada a um servidor, e este conectado a um ou mais servidores. Neste tipo de rede, os clientes conectam-se ao seu servidor XMPP através de um protocolo cliente-servidor que utiliza normalmente uma conexão TCP para comunicar. Os servidores também podem comunicar entre si através de uma conexão TCP.

Na comunicação entre os clientes na rede XMPP utiliza-se como endereçamento um padrão semelhante ao utilizado em um email, chamado Jabber-ID (JID). Esse endereço é composto por um usuário, um domínio (servidor) e um identificador de recurso (geralmente o nome do aplicativo em uso) Exemplo: usuario@dominio.com/idRecurso. Na Figura 2.8 mostra a representação do JID entre vários clientes XMPP:

É possível que um usuário conecte-se ao servidor simultaneamente a partir de mais de um cliente utilizando o mesmo usuário, o que permite inclusive o uso de agentes (*bots*) compartilhando o mesmo JID. Este identificador serve para rotear o tráfego para o cliente correto e é transparente para o usuário final.

A comunicação em XMPP é realizada quando o cliente, através da camada da aplicação envia um conjunto de instruções em XML (*streams*) para o servidor pela

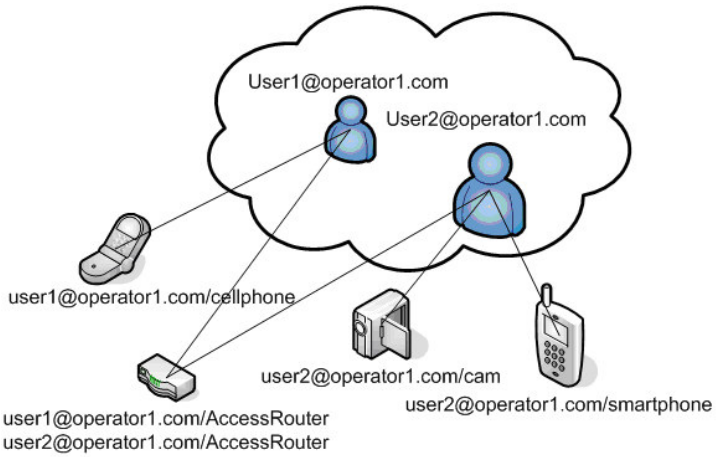


Figura 2.8: Múltiplos Recursos ligada a uma conta. [7].

camada de transporte através do protocolo TCP pedindo a sua subscrição. O servidor por sua vez ao estabelecer a conexão, abre outra conexão permanente para o *stream* originado no servidor com destino ao cliente.

Ao serem abertos esses canais de ida e volta entre aplicação e servidor, três tipos de mensagens - chamadas de XML *stanzas* - são trocados entre as duas pontas: <*presence*/>, <*message*/>, e <*iq*/>. Essas três mensagens são as unidades básicas do XMPP e, após o estabelecimento de conexão entre o cliente e o servidor, podem ser trocadas inúmeras dessas mensagens.

A mensagem de presença (<*presence*/>) controla e reporta a disponibilidade de uma entidade na rede. O servidor de um utilizador envia automaticamente informação de presença para todos os contatos que possuam uma assinatura de presença do utilizador (a assinatura de presença é direcional, ou seja, possui um remetente e um destinatário explicitamente identificados). O *stanza* de mensagem possui o corpo da mensagem a ser entregue. Por último o "iq" que significa *Info/Query* prevê um mecanismo de requisição e resposta para comunicações XMPP.

2.4.1.2 Aplicações do Protocolo XMPP

Alguns cenários para a utilização deste protocolo, incluem salas de conversação e indústrias de jogos, aplicações para IoT (Monitoramento remoto, cidades inteligentes, coleta de dados na área de medicina, aplicações de sensores na agricultura, etc), incluindo informações de presença, funcionalidades de IM e

outras formas de comunicação *Machine-to-Machine* (M2M) (em português máquina a máquina).

O protocolo XMPP é um protocolo aberto, flexível e extensível, tornando-o um dos protocolos de eleição para comunicações instantâneas na Internet. Este protocolo interoperável pode ser importante no contexto M2M, uma vez que os dispositivos inteligentes poderão possuir JIDs e comunicar na rede com outros dispositivos, como já descrito nessa seção. A informação de presença pode representar uma mais valia em determinados cenários nos quais se deseja monitorar o estado de um dispositivo.

Para o site oficial do projeto (xmpp.org) [60], as características importantes do protocolo são destacadas como:

- **Padronizado:** A IETF formalizou o núcleo do XMPP como uma tecnologia de mensagens instantâneas e presença. As especificações XMPP foram publicados como RFC 3920, RFC 3921, RFC 3922 e RFC 3923 propostas em 2004. Em 2011 a *XMPP Standards Foundation* (XSF) revisou as especificações principais resultando nas especificações (RFC 6120, RFC 6121 e RFC 6122);

- **Seguro:** Qualquer servidor XMPP pode ser isolado a partir da rede pública (por exemplo, em uma intranet da empresa); o uso de criptografia de canal usando TLS e autenticação usando o SASL;

- **Extensível:** Usando as funcionalidades do XML, qualquer um pode construir funcionalidade personalizada em cima do núcleo XMPP; para manter a interoperabilidade, as extensões comuns são publicados como um XEP (Protocolo de extensão), mas essa publicação não é necessária e as organizações podem manter as suas próprias extensões privadas, se assim o desejar; Algumas principais extensões do protocolo incluem: *Multi-User Chat* (xep-0045): extensão para criar salas de bate-papo multi usuários; *Publish-Subscribe* (xep-0060): essa extensão implementa o modelo Publicação/Assinatura, no qual um receptor pode assinar um canal do publicador, cada vez que um item for publicado nesse canal, o receptor irá receber; *Internet das Coisas - dados de sensor* (XEP-0323) e várias extensões para Internet das Coisas. Todas as especificações das extensões são encontradas no site do projeto XSF (https://xmpp.org/extensions);

- **Flexível:** Pode ser construídas diversas aplicações utilizando os recursos XMPP, como: ferramentas de gerenciamento de rede, distribuição de conteúdo, ferramentas de colaboração, compartilhamento de arquivos, jogos, monitoramento remoto de sistemas, serviços web, middleware leve, Computação em Nuvem.

2.4.2 MQTT

O protocolo MQTT é um protocolo para troca de mensagens no meio IoT desenvolvido pela IBM e Eurotech. A interação entre as entidades é baseada no padrão *publish/subscribe*, voltado para dispositivos restritos e redes inseguras [38] é adaptado para sistemas de supervisão e coleta de dados do tipo SCADA *(Supervisory Control and Data Acquisition* ou Sistemas de Supervisão e Aquisição de Dados, em português). Segundo (mqtt.org), o protocolo foi projetado para o uso em dispositivos onde se requer o uso de minimo de recurso de hardware e de largura de banda, tentando garantir a confiabilidade e garantia de entrega. O protocolo tem uma baixa sobrecarga (overhead em torno de 2 bytes por mensagem) sendo projetado para suportar redes com perdas e intermitentemente conectadas [46].

2.4.2.1 Arquitetura do protocolo MQTT

O protocolo MQTT segue o modelo cliente/servidor. Segundo T. Jaffey [33], os dispositivos sensores são clientes que se conectam a um servidor (chamado de broker) usando TCP, como mostra a Figura 2.9. As mensagens a serem transmitidas são publicadas para um endereço (chamado de tópico), que inclusive, lembra o conceito de URI, com níveis separados por barras (“/”).

Elementos da rede podem enviar diversos tópicos para o *broker* e subscritores podem escolher os tópicos que desejam. Por exemplo, imagine que, em uma rede de sensores, existam vários sensores diferentes de temperatura e umidade, publicando o valor do sensor como o dado útil *(payload)* e identificando as mensagens com tópicos nos seguintes formatos: Ex: área/ID_da_área/sensor/ID_do_sensor/temperatura, neste

caso as possíveis publicações seriam: (área/10/sensor/5001/temperatura), (área/20/sensor/4000/temperatura), etc.

Os clientes por sua vez podem se subscrever para vários tópicos, tornando-se assim capazes de receber as mensagens que outros clientes publicam neste tópico. Segundo M. Barros [11], apesar de o *broker* representar um elo de fragilidade na rede ao centralizar as comunicações, ele permite um desacoplamento entre as partes comunicantes, algo não possível em modelos de comunicação do tipo cliente/servidor. Vale lembrar que existem soluções de *brokers* redundantes ou operando em *clusters*, na tentativa de mitigar esta fragilidade.

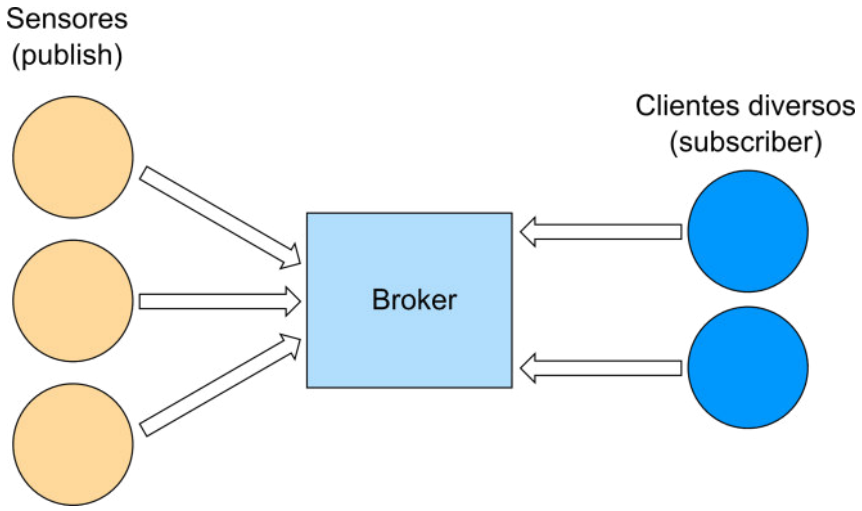


Figura 2.9: Arquitetura do protocolo MQTT. [7].

2.4.2.2 Conexão, segurança e uso em aplicações com MQTT

Segundo T. Jaffey [33], a conexão do cliente ao *broker*, seja ele subscritor ou publicador, é originalmente feita via TCP, com exigência ou não de nome de usuário e autenticação de senha para assegurar a privacidade e uso de criptografia (SSL/TLS). MQTT oferece também um nível de qualidade na entrega de mensagens, onde cada nível possui seus respectivos métodos para garantir que a qualidade do serviço de entrega seja atendida *Quality of Service* (QoS), tendo em vista que o protocolo é baseado na pilha TCP/IP o que garante a entrega da mensagem mas não garante perdas caso a conexão com a rede venha a falhar. Quanto mais alto o nível, mais recursos são necessário. M. Barros [11], cita os três níveis de qualidade de serviço:

- **QoS 0** *(at most once)*: É o que conhecemos como *"best effort"*, ou melhor esforço. Assemelha-se ao protocolo de transporte UDP, onde não se tem confirmações de entrega de mensagem. Quem envia também não tem a obrigação de manter a mensagem armazenada para futuras retransmissões;

- **QoS 1** *(at least once)*: Neste nível existe a confirmação de entrega de uma mensagem. Atende situações onde quem envia acaba gerando várias mensagens iguais possivelmente por um atraso na chegada de confirmação de recebimento. Neste caso, é garantido que uma delas terá o reconhecimento realizado. Existe um armazenamento da mensagem por parte de quem envia até a posterior confirmação;

- **QoS 2** *(exactly once)*: NGarante que a mensagem seja entregue exatamente uma vez, com envio de confirmações de recebimento e confirmações de recebimento de confirmações de recebimento. Existem confirmações nos dois sentidos, para tudo que é trafegado. Enquanto uma mensagem não é confirmada, ela é mantida. É um caso mais próximo do protocolo de transporte TCP.

Uma das aplicações atuais do protocolo MQTT encontra-se no Facebook Messenger, para realizar as comunicações com baixo consumo de largura de banda e bateria. Com o uso do MQTT a capacidade de entrega de mensagem de telefone para telefone diminuiu em centenas de milissegundos, em vez de vários segundos [61]. O Amazon Web Services também é uma aplicação que utiliza o MQTT para publicar e subscrever, embora a implementação do *broker* se desvie da especificação.

2.4.3 AMQP

O AMQP ou em português Protocolo Avançado de Filas de Mensagem é um protocolo aberto para sistemas corporativos de troca de mensagens. Especificado pelo AMQP Working Group, o protocolo permite completa interoperabilidade para middleware orientado a mensagens. A especificação define não somente o protocolo de rede, mas também a semântica dos serviços da aplicação servidora [17] .

Também é parte do foco que capacidades providas por sistemas de middleware orientados a mensagem possam estar disponíveis nas redes das empresas,

incentivando o desenvolvimento de aplicações interoperáveis baseadas em troca de mensagens. O AMQP define um protocolo padrão para publicação, enfileiramento, armazenamento e consumo de mensagens. O produtor de mensagens utiliza uma API para publicar as mensagens no servidor. O servidor *broker* irá colocar as mensagens em filas e armazená-las enquanto estas não são consumidas. Por fim, uma aplicação irá consumir as mensagens removendo-as das filas [55].

2.4.3.1 Arquitetura do AMQP

A aplicação pode se inscrever em um determinado canal, e pode enviar mensagens para um canal, que será entregue a todos os aplicativos registrados no canal. A Figura 2.10 mostra o modelo geral do AMQP, o produtor de mensagens utiliza uma API para publicar as mensagens no servidor *Broker*, já no Exchange encaminha as mensagens para as filas de mensagens *(Message Queues)* que correspondem às informações de roteamento, como por exemplo uma chave de roteamento *(routing key)*, com os vínculos previamente definidos para armazená-las enquanto estes não são consumidos. Por fim, uma aplicação irá consumir as mensagens removendo-as das filas *(Subscriber)*.

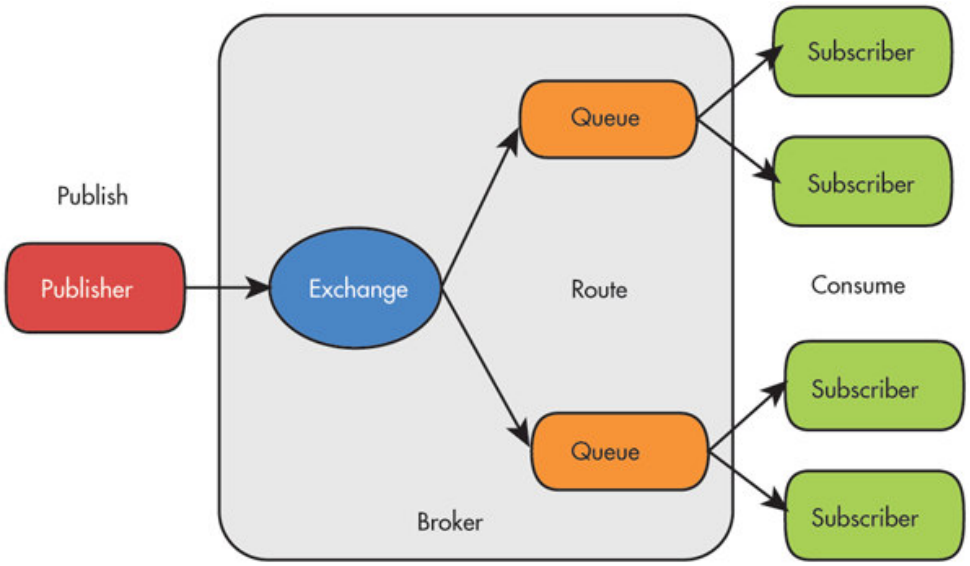


Figura 2.10: Arquitetura do protocolo AMQP. [54].

Uma característica do AMQP é a opção para definir uma mensagem como "persistente". Isto implica que a mensagem é armazenada no servidor AMQP e, sempre que um novo aplicativo entra no sistema, ele receberá todas as mensagens persistentes, a informação sobre o estado atual do sistema. Este mecanismo fornece de forma

automática na inicialização das aplicações, já que eles serão informados sobre o estado atual do sistema com as mensagens armazenadas, as aplicações serão capazes de agir como uma aplicações que já estava no sistema por algum tempo. Por outro lado, se muitas mensagens são armazenadas, a entrega para novas aplicações vai criar muito tráfego e introduzir atrasos imprevisíveis para a inicialização de novas aplicações [5].

Manheim [37], descreve a seguir um resumo dos recursos mais importantes desse protocolo:

- **Eficiente:** É orientado a conexões de protocolos que usa uma codificação binária para as instruções de protocolo e as mensagens de negócios transferidas. Ele incorpora esquemas sofisticados de controle de fluxo para maximizar a utilização da rede e os componentes conectados. Sendo projetado para obter um equilíbrio entre a eficiência, a flexibilidade e a interoperabilidade;

- **Confiável:** Permite que mensagens sejam trocadas com uma variedade de garantias de confiabilidade confirmando a entrega das mensagens;

- **Flexível:** É um protocolo flexível que pode ser usado para oferecer suporte a topologias diferentes. O mesmo protocolo pode ser usado para comunicações de cliente a cliente, agente de cliente e agente de agente.

2.4.4 CoAP

Para Paulo *et al.* [46], CoAP é um protocolo de camada de aplicação projetado para prover uma solução RESTful (i.e., baseada no estilo arquitetural REST – REpresentational State Transfer) e, portanto, modelado seguindo a semântica HTTP, porém bem mais enxuto e com uma abordagem binária em vez de baseada em texto. As vantagens de utilizar um protocolo compatível com o HTTP são a facilidade de integração e o reuso de aplicações.

O CoAP adota uma abordagem tradicional estilo cliente-servidor em vez de uma abordagem de *brokers* e foi projetado para ser utilizado sobre *User Datagram Protocol* (UDP). O CoAP foi criado por um grupo de trabalho do IETF chamado *Constrained RESTful Environments* (CoRE), iniciando suas atividades em março de 2010 com objetivo de prover um framework para aplicações que manipulam recursos

simples localizados em dispositivos interligados em redes limitadas, incluindo desde aplicações que monitoram sensores de temperatura e medidores de energia, até controle de atuadores como interruptores ou trancas eletrônicas, e também aplicações que gerenciam os dispositivos que compõem a rede, conforme o IETF (2010).

O CoAP trata de uma parte deste framework. Sendo assim, a definição do protocolo foi um dos primeiros marcos definidos pelo CoRE. O primeiro *draft (projetodraft.com)* para o CoAP foi publicado no IETF tornando-se um *Request for Comments* (RFC) (documentos técnicos desenvolvidos e mantidos pelo IETF). De acordo com o [32], é um protocolo de transferência voltado para nós e para redes restritas (considerando nós com pequena quantidade de memória RAM e redes em que a taxa de perda de pacotes é alta). O protocolo foi projetado para aplicações M2M como, por exemplo, *smartenergy* e automação residencial. A definição de mensagens deste protocolo consiste em quatro tipos de mensagens [31]:

- **Confirmable (CON):** São mensagens que precisam ser confirmadas no destino. Quando não há perda de pacotes, cada mensagem deste tipo resulta em uma mensagem do tipo *Acknowledgment* ou *Reset*;

- **Non-confirmable (NON):** São mensagens que não necessitam de confirmação de recebimento. Esse tipo é necessário quando uma aplicação que recebe leituras constantes de um sensor de temperatura em um espaço muito curto de tempo, onde a perda de uma ou outra mensagem não é motivo para preocupações;

- **Acknowledgment:** São mensagens que confirmam o recebimento de uma mensagem *Confirmable*;

- **Reset:** Realiza a indicação que outra mensagem (CON ou NON) foi recebida, mas por falta de algum contexto ela não pôde ser devidamente processada. Isso pode ser causando por algum dispositivo ter reiniciado e a mensagem enviada não foi devidamente interrompida.

2.4.4.1 Arquitetura do CoAP

Assim como HTTP, CoAP é um protocolo de transferência de documentos. Ao contrário de HTTP, CoAP foi projetado para as necessidades dos dispositivos

embarcados, sendo seus pacotes muito menores que o HTTP, usando UDP. É um protocolo de camada de aplicação projetado para prover uma solução RESTful baseada no estilo arquitetural REST em cima das funcionalidades HTPP. Essa abordagem é visto como uma forma mais simples para troca de dados entre clientes e servidores, permitindo-os expor e consumir serviços Web assim como *Simple Object Access Protocol* (SOAP), mas de uma forma mais fácil usando *Uniform Resource Identifier* (URI) como métodos de HTTP GET, POST, PUT e DELETE.

Além disso, CoAP modifica algumas funcionalidades HTTP para atender aos requisitos da Internet das Coisas, tais como baixo consumo de energia e funcionamento na presença de perdas nas conexões. No entanto, conversão entre esses dois protocolos REST-CoAP em proxies é simples. A funcionalidade geral do protocolo CoAP é demonstrado na Figura 2.11.

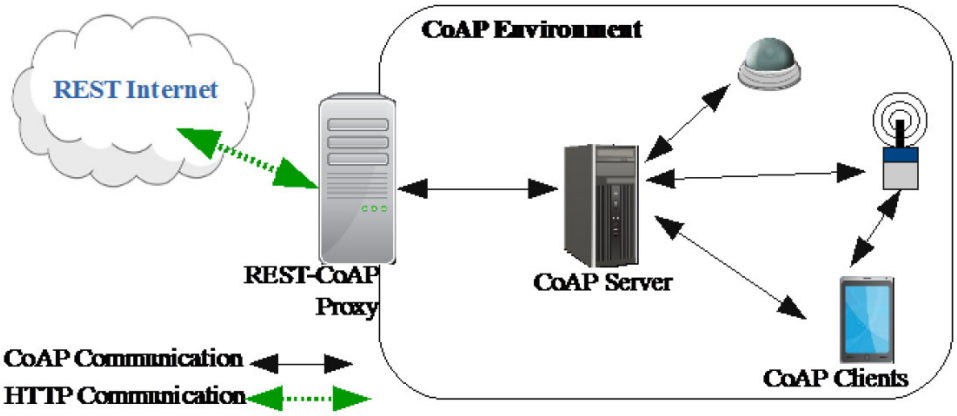


Figura 2.11: Funcionalidade do CoAP. [3].

Para A. Ludovici *et al.* [36], CoAP pode ser considerado como um protocolo de duas camadas. Uma camada de mensagem é utilizada para lidar com UDP, enquanto a outra camada é usada para interações de pedido/resposta utilizando métodos e códigos de resposta, como é feito em HTTP. No entanto ao contrário do modelo de interação de HTTP, CoAP troca mensagens de forma assíncrona. Assim, dividindo-a em duas camadas, interações de pedido/resposta seria transparente à natureza assíncrona da camada de mensagem.

2.4.4.2 Segurança no CoAP

Como CoAP é construído em cima de UDP não TCP, SSL/TLS não estão disponíveis para proporcionar segurança. *Datagram Transport Layer Security* (DTLS), oferece as mesmas garantias que TLS, mas para as transferências de dados através de UDP. Normalmente, DTLS dispositivos CoAP capazes irão apoiar RSA e AES ou ECC e AES.

2.5 Segurança na Internet das Coisas

O fator segurança em ambientes computacionais é um ponto fundamental para garantia de um bom funcionamento de um sistema. Para R. Roman *et al.* [49], a segurança é identificada como um dos obstáculos a serem transpostos para o efetivo uso da Internet das Coisas, os dados transmitidos pela Internet tornam-se vulneráveis a ponto de serem interceptados por usuários indevidos e não autorizados, sendo necessário um modo de garantir que essas informações não sejam modificadas.

Seguindo esta premissa, dispositivos IoT devem possuir mecanismos que supra tal necessidade, ao prover segurança às aplicações, por meio de uma infraestrutura de autenticação e de autorização. Além disso, é preciso garantir o comportamento autônomo dos objetos e a interoperabilidade entre estes.

2.5.1 Requisitos de Segurança na IoT

S. Alam *et al.* [4] e R. Romam *et al.* [49] descrevem de acordo com as características da IoT diversos requisitos de segurança, bem como quais propriedades de segurança devem ser garantidas. J. F. Kurose *et al.* [34] também descreve as características da comunicação segura, sendo estas:

- **Integridade:** Dados armazenados e transmitidos não devem ser alterados, removidos ou incluídos por usuários ou dispositivos não autorizados. Uma das soluções é utilizar as extensões de soma de verificação que encontra-se nos protocolos de transporte e enlace para proporcionar a integridade da mensagem;

- **Confidencialidade:** essa característica descreve que somente o remetente e destinatário possam entender o conteúdo da mensagem. Caso um interceptador obtenha o conteúdo da mensagem, mecanismos de criptografia fazem com que interceptador não possa entender o seu conteúdo. Na IoT dados sensíveis de usuários ou organizações podem estar contidos nas transações e, portanto, a confidencialidade de tais dados deve ser assegurada;

- **Autenticidade:** tanto o remetente como o destinatário precisam confirmar a identidade de cada um na comunicação, para assegurar que realmente é quem alega ser. Os dados de IoT são usados para diferentes processos de tomada de decisão e atuação, sendo que é necessário que tanto o consumidor de recursos/serviços como o provedor sejam autenticados;

- **Privacidade:** se refere a necessidade de prover aos usuários meios para que estes controlem a exposição e a disponibilidade dos seus próprios dados e informações e tenham maior transparência sobre como e por quem seus dados são usados [58].

De acordo com S. Babar *et al.* [9], outros requisitos de segurança precisam ser garantidos em IoT, dentre estes:

- **Gestão de Identidades:** esse requisito é responsável pela identificação e autenticação dos usuários e dos dispositivos/coisas em um sistema. Verifica-se também o acesso aos recursos associando aos direitos de restrição de acesso de acordo com a identidade;

- **Comunicação segura de dados:** inclui a autenticação dos pares da comunicação, assegurando a confidencialidade e integridade dos dados transmitidos, impedindo o repúdio de uma transação e protegendo a identidade das entidades [58];

- **Acesso seguro à rede:** assegura a conexão de rede ou o acesso a um serviço apenas para dispositivos autorizados;

- **Resistência à violação:** mesmo quando o dispositivo for acessado fisicamente por um atacante os aspectos de segurança são satisfeitos.

Na IoT, há vários obstáculos significativos que precisam ser superados para aumentar a sua aceitação por parte dos usuários, sendo o principal deles a segurança. A Internet e seus usuários já estão sob ataques constantes que exploram as suas fraquezas, fato que pode ser mais proeminente na IoT, que incorpora vários dispositivos com recursos limitados como energia, processamento e memória [42].

Segundo R. Roman *et al.* [49], com a popularização da IoT é suscetível ao avanço de novas e engenhosas ameaças de segurança. Com isso o grande desafio é impedir o crescimento de tais ameaças ou pelo menos mitigar o seu impacto. Na seção seguinte iremos descrever algumas ameças suscetíveis nesse cenário.

2.5.2 Ameaças e Ataques na IoT

H. Akram e M. Hoffmann [2] destaca que algumas característica dos sistemas na IoT, por exemplo, serem ubíquos e transparentes para os usuários, podem se tornar uma ameaça a privacidade dos usuários, impondo dificuldade para a garantia da confidencialidade e a integridade dos usuários.

O compartilhamento de dispositivos com outras pessoas é uma das principais ameaças de segurança contra a privacidade dos usuários, pois os dados podem ser facilmente obtidos por pessoas não autorizados, uma vez que esta pessoa bastaria ter acesso físico ao dispositivo.

M. S. Wangham *et al.* [58] apresenta uma divisão em cinco categorias dos tipos de ataques, relacionadas a seguir:

- **Ataques Físicos:** são ataques que violam o hardware do dispositivo;

- **Ataques no canal de comunicação:** são ataques baseados em dados recuperados dos dispositivos responsáveis por operações criptográficas. Esses dados são obtidos através de análise de temporização, radiação emitida, potência consumida, dentre outras fontes, que permitem que a chave de criptografia usada seja inferida;

- **Ataques de software:** exploram vulnerabilidades dos softwares presentes no dispositivo. São exemplos desse ataque: exploração de estouro do *buffer (buffer*

overflow) e uso de programas cavalos de tróia, vírus para injetar código malicioso no sistema e worms;

- **Ataques de análise de criptografia:** ataques com foco no texto cifrado, buscando encontrar a chave de criptografia para assim obter o texto em claro. Exemplo desse tipo de ataque: Homem do Meio *(Man in the Middle - MITM)*;

- **Ataques de rede:** são ataques que utilizam das vulnerabilidades da transmissão por difusão *(broadcast)* para captura e análise de tráfego, corrupção de mensagens, ataques de roteamento, etc. É exemplo dessa categoria o ataque de negação de serviço *(Denial of Service – DoS)*.

2.5.3 TLS

Na comunicação entre dispositivos na Internet das Coisas se faz necessário utilizar algum mecanismo de proteção entre as mensagens. Há muitos aspectos a ser capaz de proteger conexões entre dispositivos cliente e servidores. O passo principal é criptografar corretamente o fluxo de dados para que ele não forneça dados confidenciais a alguém que tenha sido capaz de interceptar as mensagens e evitar que um usuário mal-intencionado personifique o dispositivo cliente ou o servidor.

Um protocolo de criptografia muito usado para comunicações de dispositivo para servidor é o TLS. Mesmo quando a camada de link é criptografada (como o WPA2 para Wi-Fi), é importante usar esse método para criptografar o fluxo de dados, prevenindo contra ataques no canal de comunicação, ataques de análise de criptografia como mencionado na seção anterior. O principal objetivo do protocolo TLS é garantir a privacidade e a integridade dos dados em uma comunicação entre duas aplicações.

O protocolo TLS é baseado no protocolo *Secure Sockets Layer* (SSL) descrito no RFC 5246 [21]. Este protocolo foi projetado para utilizar TCP (*Transmission Control Protocol*) para fornecer um serviço de segurança confiável fim a fim [59], ofertando segurança em comunicações na Internet. É situado entre a camada de aplicação e a de transporte, Kurose [34], do modelo *Open System Interconnection*(OSI). A TLS é independente do protocolo utilizado seja HTTP, FTP, etc. Esta não é limitada à Web

por meio de navegadores, podendo ser implementada em servidores e aplicativos para comunicação na Internet.

Segundo S. William e W. Stallings [59], o protocolo tem dois conceitos fundamentais:

- **Sessão TLS:** é a associação entre o cliente e servidor, criado pelo protocolo de estabelecimento de conexão. Serve para definir os parâmetros criptográficos compartilhados entre várias conexões;

- **Conexão TLS:** é um transporte apropriado a um tipo de serviço. As conexões são ponto a ponto e estão associadas a uma sessão.

Para L. Pereira [47], o principal objetivo desse protocolo é o de promover a confidencialidade e a integridade dos dados entre dois aplicativos que se comunicam. A arquitetura do TLS é composto por duas camadas: a *Record* e a *Handshake* como mostra a Figura 2.12.

Figura 2.12: Camada SSL/TLS. [47].

A camada *Record* é utilizada para encriptar os dados da camada de Aplicação, de forma a não ficarem vulneráveis caso sejam interceptadas por terceiros. A camada *Handshake* é divida em três protocolos, o *Handshake*, o *Change Cipher Spec* e o *Alert*.

O *Change Cipher Spec* trata-se de uma mensagem, que notifica o receptor, de que as mensagens seguintes serão cifradas segundo as chaves acabadas de negociar. O *Alert* também é uma mensagem, mas pretende notificar uma ocorrência de erros ou uma ocorrência importante.

O *Handshake* faz negociações entre cliente e servidor, realiza a autenticação entre cliente e servidor, negocia o MAC *(Message Authentication Code)*, algoritmos criptográficos e chaves que serão utilizadas para proteger os dados transmitidos. Utiliza criptografia de chaves públicas para realizar a negociação inicial, em que a chave simétrica de sessão gerada aleatoriamente sendo transferida por um meio seguro.

Para uma melhor explicação dos procedimentos do TLS, S. William e W. Stallings [59] ilustra através da Figura 2.13 o esquema com base na troca de pacotes.

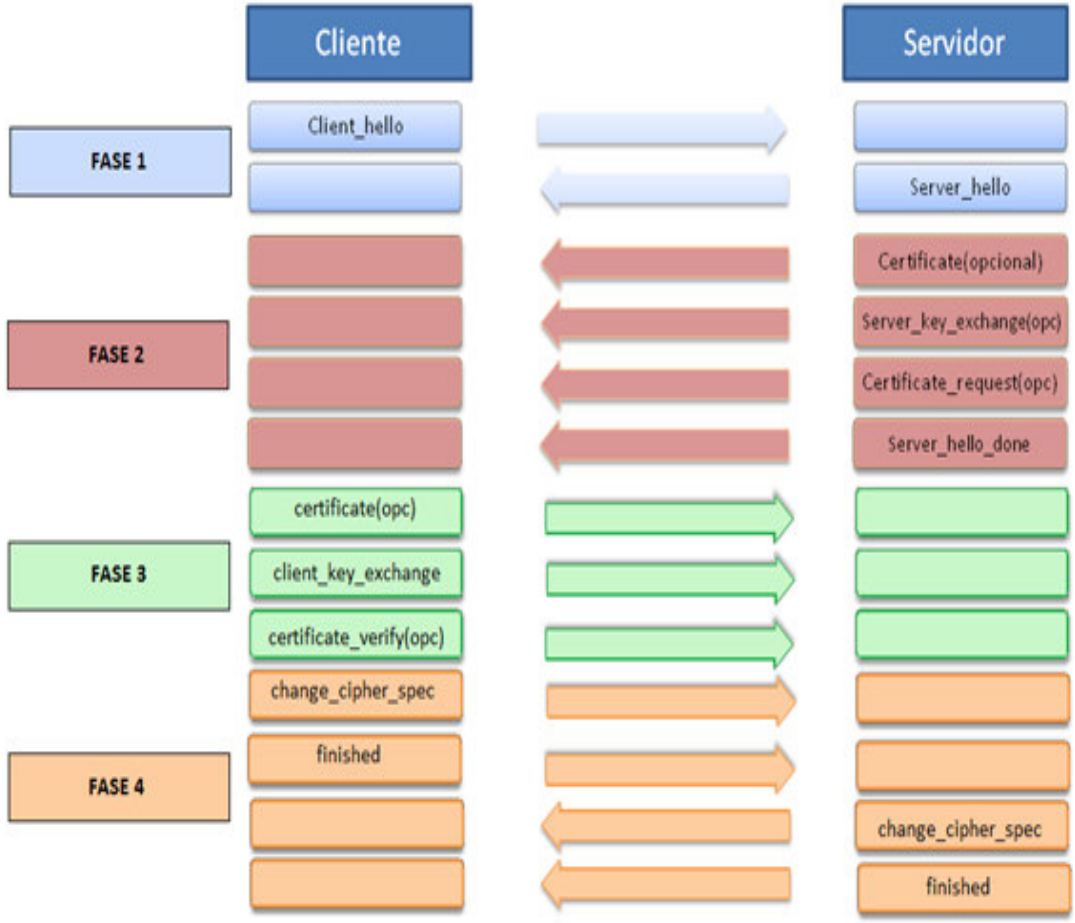


Figura 2.13: Protocolo do *handshake* [59]

Na fase 1, o "Cliente" deve contactar o "Servidor", mostrando interesse em comunicar com o mesmo. Nesta fase são estabelecidas as capacidades de segurança pelos seguintes parâmetros: versão do TLS, valor aleatório para impedir ataques por repetição, ID de sessão para atualizar ou criar uma nova conexão na sessão, o conjunto de cifras suportado e uma lista de métodos de compactação admitida.

Na apresentação mútua, uma mensagem *client hello* define os parâmetros da forma que o cliente admite e a mensagem *server hello* contém os mesmo parâmetros da mensagem *client hello*, desta forma negociando quais parâmetros irão utilizar.

Na fase 2, o "Servidor" entrega o seu certificado ao "Cliente". Seguidamente o "Cliente" procura obter um certificado da "Autoridade de Certificação" e valida o certificado que recebeu do "Servidor". Uma mensagem é enviada sinalizando o término da fase de apresentação mútua.

Na fase 3, Depois de este ser validado com sucesso o "Cliente" envia o seu certificado ao "Servidor". uma mensagem com a troca de mensagens (caso seja o tipo RSA este gera um pré-segredo-mestre aleatório e criptografa-o com a chave pública fornecida pelo servidor) e uma mensagem para oferecer explicitamente uma verificação do certificado do cliente.

Na fase 4, tanto o cliente e o servidor enviam uma mensagem *change cipher spec* que faz parte do protocolo *Change Cipher* e não do protocolo de estabelecimento de sessão para determinar o serviço de criptografia, e uma mensagem de conclusão (*finished*), por meio do qual esta verifica se o processo de autenticação e troca de chaves foram bem sucedidos.

2.6 Síntese

Neste Capítulo, foi apresentado o conceito inicial de Internet das Coisas, que tem por finalidade interconectar qualquer tipo de dispositivo que tenha capacidade de conexão com a Internet. Apresentamos as tecnologias que fazem parte desse cenário, os ambientes de aplicação e protocolos de comunicação que são aplicados de acordo com suas características.

Na Seção Computação em Nuvem, apresentamos o conceito, as vantagens dessa tecnologia na qual proporciona ao usuário final serviços de fácil acesso e de baixo custo, garantindo características como disponibilidade e escalabilidade. Seguido por destacar na Seção Computação em Nuvem na IoT, a integração entre a Internet das Coisas e Computação em Nuvem tendo em vista que Computação em Nuvem supre algumas das necessidades da IoT, como limitação de armazenamento e processamento.

Na Seção protocolos de comunicação para Internet das Coisas, foi apresentado os protocolos XMPP, MQTT, AMQP e CoAP, sendo estes os protocolos mais utilizados para facilitar e simplificar os trabalhos de programadores de aplicações na IoT.

Foi mostrado na Seção Segurança na Internet das Coisas os requisitos de segurança, como integridade, confidencialidade, autenticidade e privacidade, destacando a tecnologia TLS para segurança confiável fim a fim para proteção na comunicação entre os dispositivos na IoT.

3 Estado da Arte

A heterogeneidade das tecnologias da Internet das coisas, o grande número de dispositivos e sistemas, e os diferentes tipos de usuários e funções criam desafios importantes neste contexto. Em particular, os requisitos de segurança, autenticação, confiabilidade, confidencialidade, integridade e privacidade e gerenciamento de contexto são difíceis de resolver, mesmo com o enorme volume de trabalho existente. Na literatura, existem várias maneiras de fornecer esses requisitos os trabalhos [18], [12], [45], [41] e [10], que serão comentados nas próximas seções.

3.1 *The VIRTUS Middleware: an XMPP based architecture for secure IoT communications*

O trabalho proposto por Conzon *et al.* [18] apresenta o middleware Virtus, o qual é orientado a eventos e tem como base o protocolo XMPP possuindo mecanismos de segurança que contribuem com a interoperabilidade, como a federação de servidores e o mapeamento sobre o HTTP. No VIRTUS, os protocolos TLS e SASL são utilizados para garantir a integridade e confidencialidade das mensagens e para a autenticação das partes envolvidas, respectivamente. A Figura 3.1 apresenta os módulos do middleware VIRTUS.

Os Módulos do Virtus são descritos a seguir:

- Os módulos personalizados: também chamados de *bundles* (p.ex. Zigbee bundle), são usados para permitir a comunicação entre dispositivos com restrições computacionais e o Servidor VIRTUS, traduzindo as mensagens específicas da tecnologia do dispositivo para XMPP. Geralmente, são implementados em um intermediador entre o dispositivo e o resto do middleware;

- Servidor VIRTUS: utilizado para gerenciar a comunicação entre os componentes do middleware, sendo que há um servidor para cada rede (domínio) na qual o middleware é implementado;

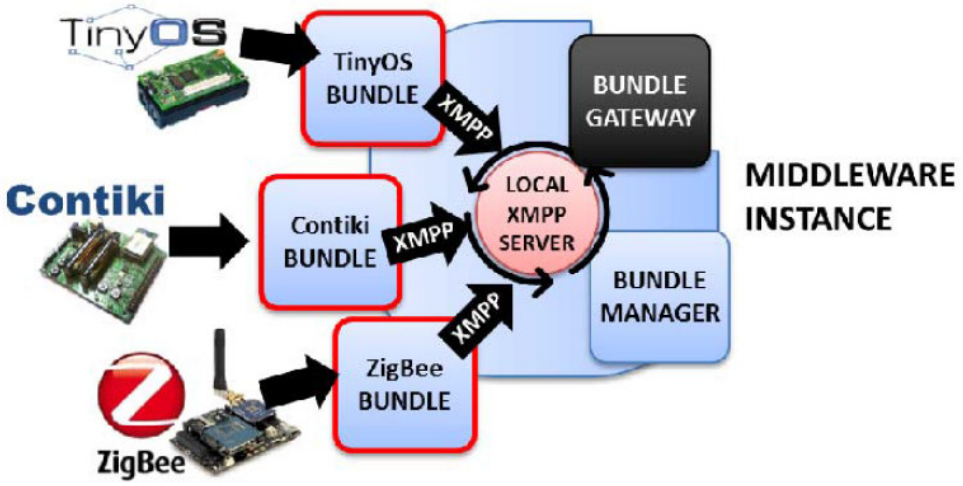


Figura 3.1: Módulos presentes no middleware VIRTUS [18]

- *Manager*: gerencia a conexão entre os diversos módulos do *middleware*. Este também provê uma lista dos módulos disponíveis e faz o gerenciamento de dependências;

- *Gateway*: se comunica com a instância local do Servidor VIRTUS, assim como com instâncias remotas do middleware. Este componente é o intermediário entre qualquer aplicação que deseje se comunicar com qualquer dispositivo dentro do middleware.

O *middleware* considera a existência de três tipos distintos de dispositivos: dispositivos com muitos recursos – como servidores, que implementam todo o *middleware*; dispositivos restritos – como *smartphones*, que implementam módulos cliente XMPP para interagir com outros módulos; e dispositivos simples – como sensores e etiquetas RFID, que tem suas mensagens encapsuladas em formato XMPP por um outro dispositivo com mais recursos computacionais. A comunicação intra-domínio e inter-domínio é feita através de troca de mensagens XMPP, contudo há módulos que permitem a comunicação com outras arquiteturas, como por exemplo com Serviços Web.

3.2 *A Service Infrastructure for the Internet of Things based on XMPP*

O trabalho proposto por S. Bendel *et al.* [12] apresenta uma plataforma de serviços com base na XMPP para o desenvolvimento e provisão de serviços de infra-estruturas pervasivas. O artigo tem com base dois estudos de caso (controle de robô e e-mobilidade) no cenário IoT, demonstrando a capacidade para suportar aplicação de tempo real - TR da arquitetura.

Esse trabalho fornece um ambiente de execução para múltiplos serviços em tempo de execução. Com base no concentrador de dados, uma parte do ambiente de execução de serviços conecta os serviços com vários objetos do mundo real. A plataforma conecta os serviços com vários objetos do mundo real como sensores, dispositivos móveis. Outra funcionalidade está no gerenciamento de dispositivos. Ele lida com o provisionamento de aplicativos para dispositivos móveis, bem como o gerenciamento de usuários e dispositivos. O trabalho aborda questões de segurança, autenticação sobre responsabilidade do protocolo XMPP. A Figura 3.2 mostra a arquitetura do trabalho.

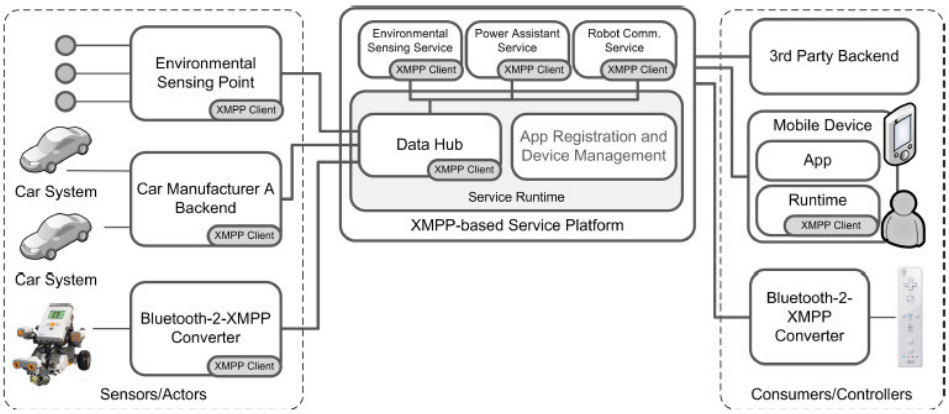


Figura 3.2: *Arquitetura proposta por* [12]

3.3 *SCondi: A Smart Context Distribution Framework Based on a Messaging Service for the Internet of Things*

O trabalho proposto por J. Park e M.-J. Lee [45] apresenta um *framework* de distribuição de contexto baseado na Internet das Coisas, que fornece um mecanismo eficaz e confiável para distribuição de informações de contexto. O mecanismo define a utilização de requisitos de segurança utilizando um canal de contexto o qual provê uma comunicação abstrata confiável adaptável divulgando e coletando informações dos provedores de serviços. Além disso, o canal de contexto proporciona um filtro de autenticação e autorização provendo a segurança das informações.

Utiliza o serviço de mensagens que suporta o protocolo MQTT em redes TCP/IP, a estrutura do Scondi suporta comunicação anônima entre editores de mensagem e assinantes, fornecendo entrega confiável com base em MQTT QoS (Quality of Service) em ambientes heterogêneos e redes *wireless*.

3.4 *SecKit: A Model-based Security Toolkit for the Internet of Things*

O trabalho proposto por R. Neisse *et al.* [41] apresenta um modelo baseado no conjunto de ferramentas de segurança que é integrado em uma estrutura de gerenciamento para dispositivos na IoT, suportando a especificação e avaliação das políticas de segurança de modo a permitir a proteção dos dados do usuário. Para isso no trabalho é aplicado em um cenário *Smart City* a fim de demonstrar a sua viabilidade e desempenho.

Para permitir que requisitos de segurança, cenários de ameaça, relações de confiança, e políticas de controle de uso sejam satisfeitos, o projeto se baseia em modelagem de gerenciamento de segurança. Concebendo ao usuário a possibilidade de projetar um conjunto de políticas de segurança e privacidade completamente personalizados.

O SecKit pode ser usado para especificar e impor políticas de confidencialidade, retenção de dados, controle de acesso, não-repúdio, e

gerenciamento de confiança. Os metamodelos definidos incluem, dados, tempo, identidade, regras, contexto, estrutura, comportamento, risco, confiança e metamodelos de regras implementados usando o *Eclipse Modeling Framework* (EMF).

3.5 *Design and Development of Integrated, Secured and Intelligent Architecture for Internet of Things and Cloud Computing*

O trabalho proposto por P. Bai e S. A. Rabara [10] fornece uma estrutura IoT para oferecer serviços inteligentes e aplicativos seguros em qualquer lugar, a qualquer hora, qualquer empresa, de qualquer dispositivo e em qualquer rede independente da tecnologia. Para tanto o trabalho oferece essa arquitetura utilizando a *Inteligente Smart Card* (ISC) (Um Cartão Inteligente) que facilita o acesso seguro de aplicações diversificadas e serviços distribuídos em um ambiente de nuvem.

Utilizando um número único de identificação (UID) para cada cliente através do sistema inteligente, este processa os dados, em seguida, envia os dados necessários para a nuvem através da rede IP/MPLS Core. Essa rede é a parte principal da arquitetura composta que liga a plataforma de nuvem e IoT habilitando os sistemas inteligentes.

A informação sobre a ISC é enviado para vário centros de dados para melhorar o processo de autenticação e da disponibilidade de dados a qualquer hora, em qualquer lugar e de qualquer empresa uma vez que todos os setores estão interligados através de uma plataforma integrada para trocar e recuperar informações instantaneamente e de forma inteligente. Este sistema adota criptografia de curva elíptica e certificado digital para garantir a autenticação, a confidencialidade, privacidade e integridade. A Tabela 2.1 apresenta uma análise comparativa entre os mecanismos descritos anteriormente.

Tabela 3.1: Mecanismo de Segurança para Internet das Coisas

Abordagens	[18]	[12]	[45]	[41]	[10]
Transport Layer Security	✓	-	-	-	-
Computação em Nuvem	-	-	-	-	✓
Suporte para varias aplicações	✓	-	✓	✓	✓
Uso em diversas Plataformas	✓	✓	✓	-	✓
Utilização de contexto	-	-	✓	✓	-
Confidencialidade	✓	-	-	✓	✓
Confiabilidade	-	✓	✓	-	✓
Autenticação	✓	✓	✓	-	✓

3.6 Síntese

Neste capítulo, os conceitos relacionados a Internet das Coisas, computação em Nuvem, computação IoT na nuvem, protocolos de comunicação na IoT e segurança em redes foram apresentados. Na Internet das Coisas, tem-se como principal objetivo permitir a interligação de objetos do mundo real com o mundo virtual por meio de sensores, trocando informações sobre status, localização, funcionalidades, problemas, etc.

A computação em nuvem é um modelo de acesso conveniente aos seu recursos compartilhados em rede e dos recursos computacionais os serem provisionados dinamicamente, a partir da necessidade do dispositivo, tornando-se bastante atraente na IoT onde as grandes informações geradas por algoritmos poderão ser armazenados em servidores na nuvem.

Na seção de computação em nuvem integrada com a Internet das Coisas, argumenta-se que a IoT pode se beneficiar das capacidades e recursos da Nuvem virtualmente ilimitadas para compensar as suas limitações tecnológicas, por exemplo, armazenamento, processamento, comunicação, etc. Na seção protocolos de comunicação entende-se como facilitadores afim de simplificar os trabalhos programadores de aplicativos para IoT fornecendo padrões de comunicação em um ambiente.

A seção de segurança em redes mostra as vulnerabilidades que os dados de usuários de aplicações estão sujeitos e algumas soluções utilizadas.

Na seção de trabalhos relacionados, alguns projetos aplicados para Internet das Coisas e suas principais características foram apresentados. E por fim uma comparação entre as características dos trabalhos foi apresentada. Com essa comparação sintetiza-se que há necessidade de implementar mecanismos que forneçam segurança na comunicação mantendo as características da Internet das Coisas, como comunicação com dispositivos heterogêneos, confiabilidade, gerenciamento de usuários e dispositivos.

4 *Framework* de comunicação seguro e confiável para IoT - FRI

Este capítulo tem como objetivo apresentar e analisar a arquitetura do *Framework de comunicação seguro e confiável* - FRI, sua divisão em camadas com suas funcionalidades, destacando seus detalhes, classes e interações. O *framework* é organizado a partir da visão em camadas, detalhando suas funções, a fim de demostrar seu comportamento. Sempre que necessário as principais peculiaridades e dificuldades de implementação encontradas no decorrer do projeto são destacados, assim como discorremos sobre os motivos pelos quais optamos pela solução adotada.

4.1 Objetivos almejados com o FRI

As abordagens atuais para a Internet das Coisas se concentram principalmente em protocolos de comunicação para integrar "coisas" com padrões de protocolo na Internet considerando computação e memória recursos limitados, bem como a disponibilidade de largura de banda e disponibilidade de energia [12]. O ambiente heterogêneo no contexto da Internet das Coisas torna-se a comunicação indispensável.

As informações do usuário dos variados dispositivos, dados de contexto, informações sensíveis tornam-se vulneráveis ao serem transmitidas na rede de computadores, medidas de segurança devem ser realizadas para dificultar aos invasores o seu acesso. A segurança tem um nível de complexidade e uso de recursos computacionais muito alta para ser empregada e a grande variedade de arquiteturas torna uma tarefa dispendiosa para os desenvolvedores de aplicações.

Os vários tipos de dispositivos na IoT muitas vezes tem recursos limitados de memória, processamento, armazenamento. De forma a resolver esses problemas, o FRI utiliza-se de protocolo voltado para o baixo consumo de recursos, provendo mecanismo de confiabilidade no envio da comunicação, e segurança na

conectividade com a computação em nuvem visando fornecer uma maior capacidade de armazenamento dos dados e informações trafegadas pela rede para prover uma economia dos recursos dos dispositivos. Neste trabalho, foi proposto:

- O FRI deve permitir um desenvolvimento simplificado e eficiente em termos de custos e implantação de serviços de comunicação em tempo real entre objetos;
- Realizar toda comunicação entre objetos de forma segura e confiável em termos de conexão;
- Deve apoiar a integração de aplicativos em dispositivos móveis e a Nuvem.

4.2 Arquitetura do FRI

A arquitetura proposta nesta dissertação tem como objetivo fornecer um framework, ou seja um conjunto de classes com finalidade de provê funções de conexão seguras e confiáveis no ambiente IoT. Desta forma o FRI possui funcionalidades bem definidas entre o framework e as aplicações, a visão geral do ambiente de uma solução baseada em FRI é ilustrada através da Figura 4.1 que serão discutidos a seguir.

Dividimos o FRI em camadas, ilustrado através da Figura 4.2, onde cada uma provê uma série de funcionalidades para as demais, provendo em conjunto a solução FRI. A divisão em camadas nos dá a oportunidade de visualizar as entidades centrais da solução.

O coração do *framework* é composto pelo protocolo XMPP Core usado para comunicação simples XML do sistema (RFC 6120). O Core do protocolo é destacado pelos exemplos de código XML *stanzas* abaixo, composto na arquitetura básica do protocolo, visto (Seção 2.4.1.1).

```
1
2
3 <?xml version="1.0"?>
4 <stream:stream xmlns:stream="http://etherx.jabber.org/streams"
5 xmlns="jabber:client" to="amessage.de">
```

```
6
7  <iq type="set" id="auth_2" to="amessage.de" >
8    <query xmlns="jabber:iq:auth">
9      <username>kuusipuu</username>
10     <password>mypassword</password>
11     <resource>Work</resource>
12    </query>
13 </iq>
```

Mecanismo de requisição e resposta para comunicações XMPP:

```
1  <iq from="amessage.de" id='auth_2' type='result'/>
```

Envio de Mensagem:

```
1
2  <message to="tero@exemplo.com" >
3    <subject>teste 1449</subject>
4    <body>teste 1449</body>
5  </message>
6  <presence type="unavailable" >
7    <status>Logged out</status>
8  </presence>
9  </stream:stream>
```

Todos os usuários incluindo serviços e aplicações são clientes XMPP que podem ser identificados por JID (por exemplo usuario@servidor.com). Um segundo grande bloco está os componentes de *Factory Connection, Manager Reliability, Manager Security/TLS e Authenticity/SASL. (SASL).*

No Componente *Factory Connection*, um ambiente em tempo de execução é fornecido, onde cria-se a conexão com os servidores Jabber e/ou nuvem.

O Componente *Manager Reliability* fornece um ambiente em tempo de execução o qual se analisa para múltiplas comunicações de forma dinâmica o envio da mensagem verificando o recebimento com sucesso pelo destinatário. Esse mecanismo verifica casos em que o servidor está indisponível e/ou destinatário indisponíveis realizando o tratamento dessas exceções, solicitando para Fábrica de Conexão a

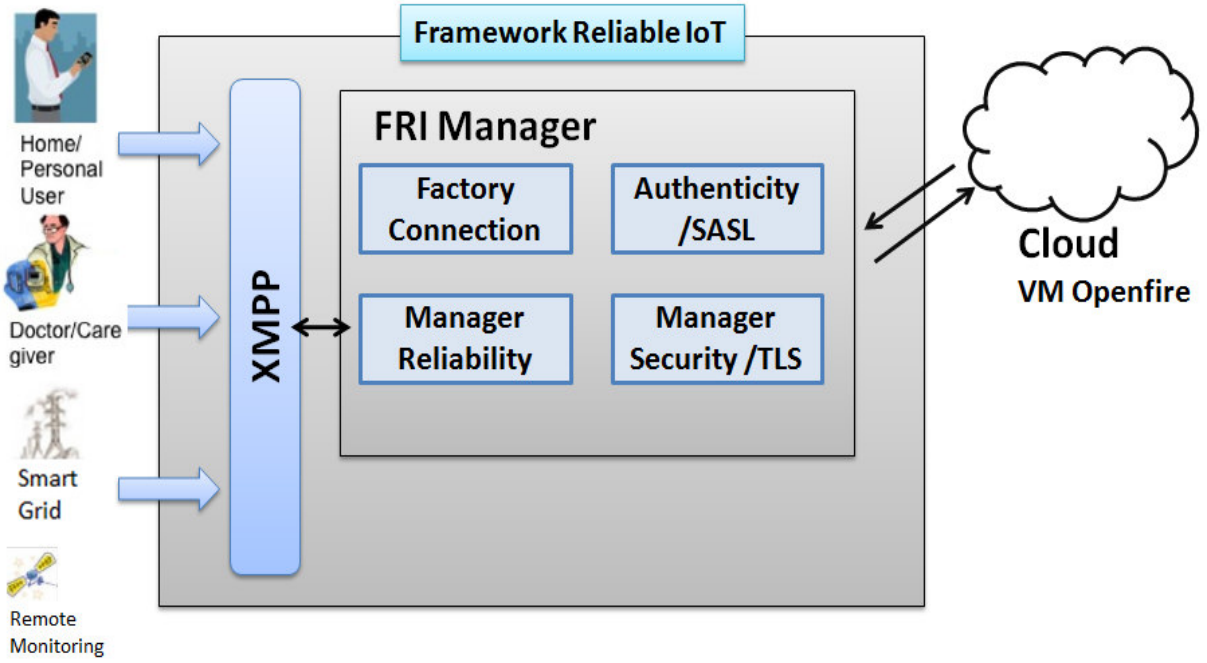


Figura 4.1: Arquitetura do ambiente FRI.

mudança da conexão para outro servidor ou gerando exceções para as aplicações em tempo de execução .

O componente *Manager Security/TLS e Authenticity/SASL* fornece a comunicação o uso de TLS (Transport Layer Security - RFC 2246 [20]), garantindo a integridade e confidencialidade de dados e criptografia de fluxos XML.

Na Figura 4.2, verificam-se os componentes de funcionamento da arquitetura FRI que são: *Application, Framework Reliable* com o protocolo XMPP Core e *Reliable Mechanism, XMPP Server, Java VM e Operating System*.

Na camada *Application* localiza-se as aplicações que dependem do ambiente no qual está empregado a solução, por exemplo aplicações de monitoramento de performance de equipamentos industriais, aplicações de controle de consumo de energia, aplicações de temperatura (sensores no *datacenter*), etc.

Na camada *Framework Reliable* localiza-se o *Framework* proposto composto pelo protocolo de comunicação XMPP Core definido pelo RFC 6120 (protocolo sem adição de extensões, apenas no formato do protocolo simples (Seção 2.4.1)). o componente *Reliable Mechanism* e todas as classes de conexão do FRI, os detalhes dessa camada são descritas nas próximas seções.

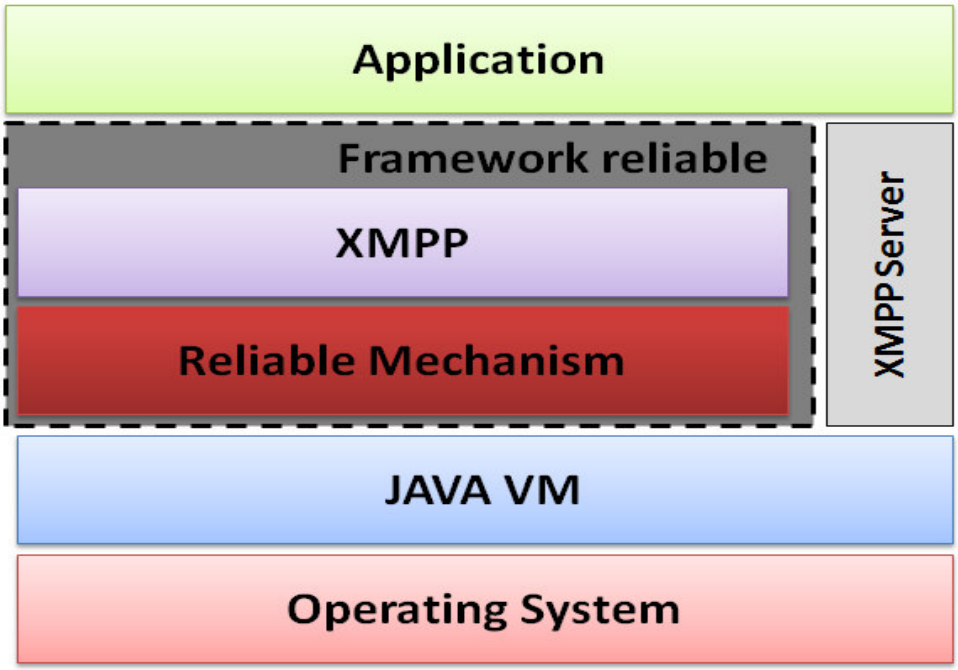


Figura 4.2: Arquitetura em camada do FRI.

Na camada *XMPP Server*, o servidor de comunicação em tempo real multi-plataforma baseada no protocolo XMPP é utilizado.

Na camada *Java VM*, a arquitetura baseada em Java como núcleo sendo a linguagem utilizada para desenvolvimento do FRI é utilizada.

No componente *Operating System*, plataformas típicas da Internet das Coisas, como TinyOS, Contiki, Raspbian podem ser utilizados.

As classes compostas pelo FRI serão descritas com mais detalhes nas próximas seções.

4.3 Componentes do FRI

O FRI foi concebido utilizando a linguagem de programação Java, composto pelas classes *XMPPConnectionManager, NoServerAvaliable, XMPPConfiguration, ContextProviders, BatterInfoReceiver, interface ServerConnectionListener* e *XMPPStatus*. A seguir serão descritas essas classes, bem como seu objetivo. A representação dessas classe é mostrada no diagrama de classe da Figura 4.3 e explicada como segue:

- Classe *XMPPConnectionManager*: Essa classe implementa as funcionalidades essenciais do FRI. Nela consta métodos de conexão, desconexão, reconexão,

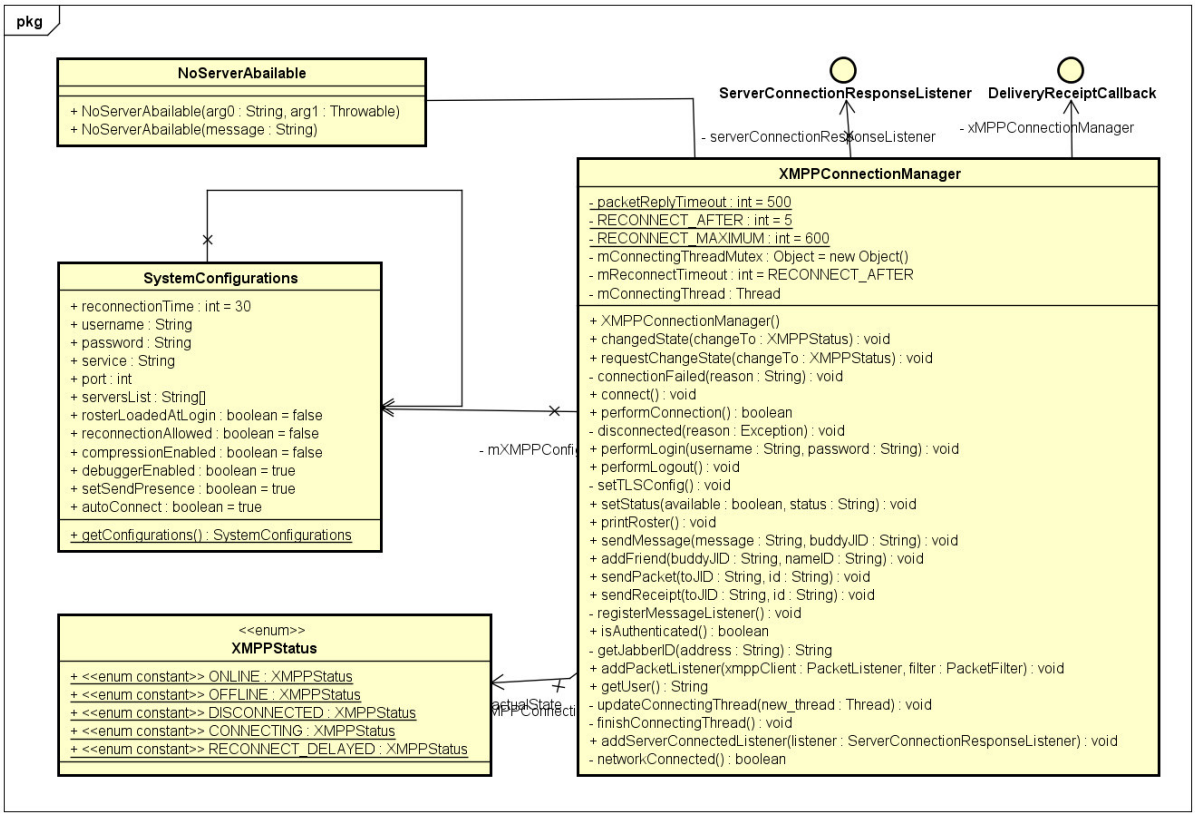


Figura 4.3: Diagrama de classe FRI.

envio de mensagem, verificação do estado da conexão, autenticação, verificação de disponibilidade do canal de envio (rede), segurança (usando TLS), estado do dispositivo (destinatário), envio de estado do dispositivo (remetente), métodos de reconexão com outros servidores, verificação de mensagem recebida;

- Classe *NoServerAvaliable*: Classe pertencente ao pacote de exception, responsável pelas exceções ocasionadas quando nenhum servidor é encontrado para realizar a conexão ou há alguma indisponibilidade no envio da mensagem e problemas de rede;

- Classe *SystemConfigurations*: Classe responsável pelas configurações do *framework*, tais como: lista de IP's de servidor, porta de conexão e nome do serviço para conexão, definição de auto conexão, envio de presença (indisponível, disponível, outra de preferencia definido pelo utilizador), tempo de reconexão, uso *debugger*, usuário e senha para autenticação;

- Componente Enum *XMPPStatus*: Essa estruturas de dados armazenam os tipos de status que a conexão poderá assumir dependendo do seu comportamento;

- Interface *ServerConnectionListener*: É uma interface ouvinte de resposta da conexão (sucesso da conexão ou erro). Essa interface também é um ouvinte quando ocorre algum problema e o servidor fica indisponível;

- Interface *DeliveryReceiptCallback*: É uma interface ouvinte de resposta de envio de mensagem.

4.4 Funcionalidades dos componentes do *Framework*

As funcionalidades apresentadas nesta seção foram realizadas com o dispositivo com recursos limitados (*smartphone*), dispositivo com recursos ricos PC [18] e plataforma RaspberryPI, dispositivos estes que fazem parte do cenário IoT [3]. A aplicabilidade dessas funcionalidades sempre se dá entre camadas próximas, evitando assim que a aplicação do usuário se preocupe com os detalhes de implementação da tecnologia de rede. A seguir serão apresentados algumas funcionalidades dos principais os componentes.

4.4.1 Envio de mensagem com mecanismo seguro

A segurança nas comunicações utilizando XMPP foi formalizado com a publicação das especificações núcleo XMPP (RFC 3920 e RFC 3921 [51] [52]), onde principais extensões relevantes segurança, a autenticação, privacidade e controle de acesso. A proposta de segurança do FRI é destinado a obter uma arquitetura cliente-servidor seguro descentralizada explorando as políticas de segurança (tais como autenticação de usuário, criptografia de canal), alcançando uma segurança intrínseca devido a arquitetura escolhida.

O envio da comunicação se dá no envio da mensagem pelo cliente ao servidor, realiza-se a obtenção dos parâmetros de conexão e a busca do servidor (DNS XMPP). Após a busca é efetuado a verificação da sua disponibilidade utilizando os mecanismos do FRI na tentativa de conexão, caso não esteja disponível naquele instante realiza-se uma tentativa de conexão com outro servidor previamente cadastrado nas preferências do FRI, se houver, essa rotina se dá até ser concluída a conexão.

No momento que há sucesso no pedido de conexão a negociação começa com abertura de uma conexão TCP (duplex). O cliente envia um cabeçalho de fluxo inicial e o servidor abre um fluxo de resposta através do envio de uma marcação de abertura, contendo ID único gerado especificamente para a sessão. Em seguida, o servidor envia uma lista de recursos com suporte para fixar e comprimir a conexão. Após as negociações de autenticação e outros, a comunicação começa, utilizando TLS e SASL.

O TLS fornece um mecanismo confiável para garantir a integridade e confidencialidade de dados, criptografia de fluxos XML conforme definido ao longo de uma extensão "STARTTLS"(RFC 2595 [43]). O SASL garante a validação do servidor por meio de um perfil específico do XMPP, por meio de autenticação. Na Figura 4.4 mostra o diagrama de atividade dessa rotina.

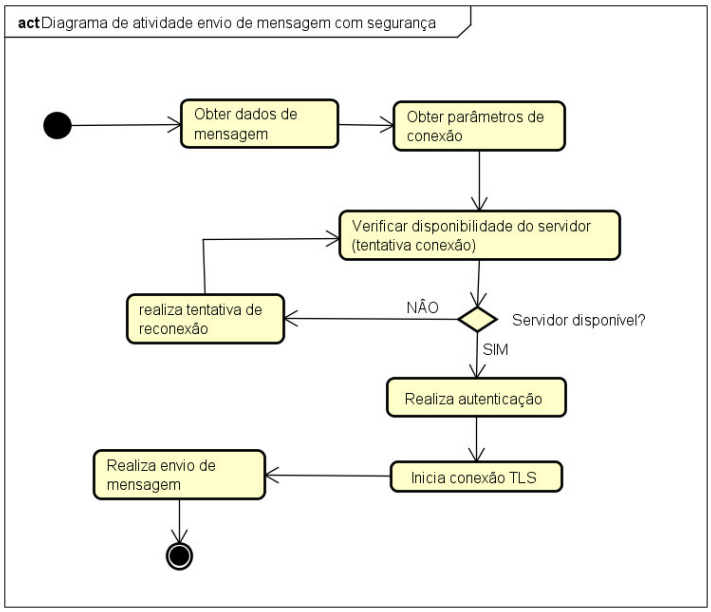


Figura 4.4: Diagrama de atividade envio de mensagem.

4.4.2 Controle da desconexão e reconexão com servidores

A proposta do controle do fluxo de conexão e reconexão utiliza o status da conexão *(online, offline, disconnected, connecting, reconnect delayed)* no momento que a tentativa de conexão é efetuada, as rotinas do FRI realizam a tentativa e atribui esses status.

Caso ocorra falha do envio de dados, indisponibilidade de conexão de rede (Ethernet, Wifi, rede de dados móveis, etc) é atribuído o status offline e esperado o retorno da conexão com a rede de dados. Caso seja efetuado a conexão com a rede de dados, ocorre a Tentativa de Conexão com o servidor *(connecting)*, se iniciada, parâmetros de conexão são estabelecidos e autenticação efetuada.

Caso ocorra problemas de indisponibilidade de serviço *(offline)* nova tentativa *(reconnect delayed)* é efetuada de acordo com o tempo estabelecido pelo usuário com outro servidor caso esteja cadastrado senão é efetuado com o mesmo. Se ocorrer uma desconexão por motivo qualquer, seja por rede ou indisponibilidade do servidor é atribuído em tempo de execução *(reconnect delayed)* e novas tentativas de reconexão são efetuadas. A Figura 4.5 mostra o diagrama de atividade para essas rotinas descritas.

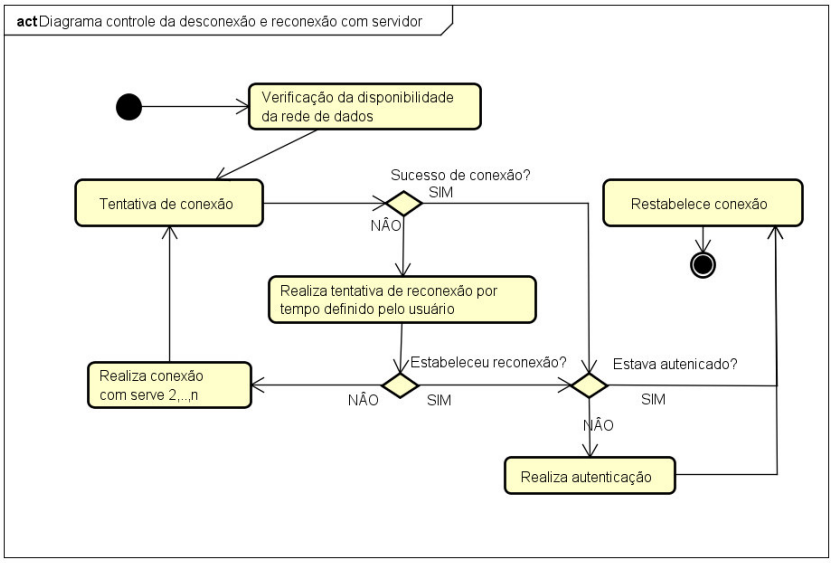


Figura 4.5: Diagrama de atividade desconexão e reconexão com servidores.

4.4.3 Garantia de entrega de mensagem ao destinatário

O FRI utiliza como mecanismo de garantia de entrega de mensagem ao servidor a confiabilidade garantida também pela arquitetura TCP/IP, e por se utilizar a extensão do protocolo XMPP para recibos de entrega de mensagens (XEP-0184) em plataforma mobile (Android), no qual o remetente de uma mensagem pode solicitar uma notificação de que a mensagem foi entregue a um cliente controlado pelo destinatário pretendido [53].

O FRI retorna a mensagem de "entrega efetuada". A Figura 4.6 mostra o diagrama de atividade realizado por essa funcionalidade.

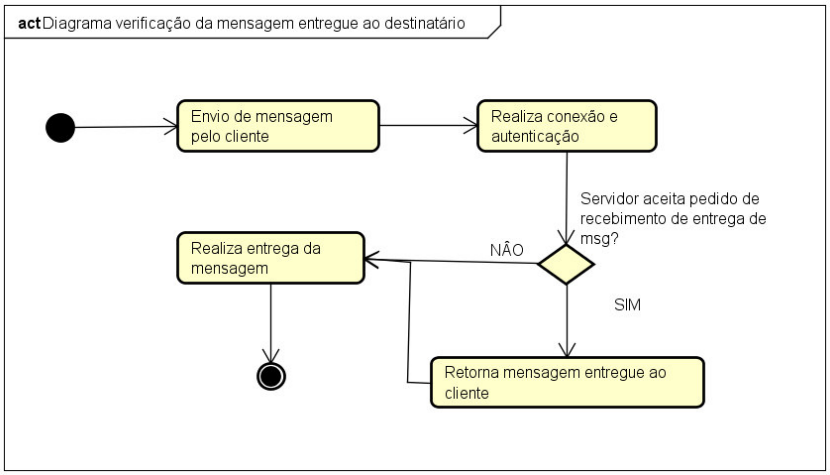


Figura 4.6: Diagrama de atividade de mensagem entregue ao destinatário.

4.5 Metodologia de Aplicação do FRI

Para a utilização do *framework* exposto nesse Capítulo deve-se utilizar as classes e métodos seguindo suas funcionalidades destacadas abaixo:

1. O que instanciar:

   a) A classe **SystemConfigurations** precisa ser instanciada primeiro, pois nela será colocada as configurações de funcionamento (ips, porta, nome de serviço, etc.);

   b) A classe **XMPPConnectionManager** é responsável por gerenciar todo o funcionamento do *framework*;

2. Métodos a chamar: Com a instância de **XMPPConnectionManager** pode-se utilizar os métodos:

   a) **addServerConnectedListener**: para adicionar uma interface ouvinte para o estado da conexão. A interface possui dois métodos, um para obter a resposta se a conexão foi bem sucedida (assim podendo inicializar alguma funcionalidade que precisa da conexão) e outro método para quando a conexão for perdida, ou seja a desconexão não foi voluntária.

b) **requestChangeState**: o controle do estado do framework funciona por solicitações de mudanças de estado, portanto aqui é são declaradas essas intenções, um exemplo: **XMPPStatus.ONLINE** requisita entrar no estado de conexão ativa.

c) **setStatus**: Definir seu status *on-line*, também conhecido como presença.

d) **printRoster**: Recuperar *roster* do usuário e verificar o estado dos contatos existentes.

e) **sendMessage**: Envia mensagem para o destinatário;

f) **addFriend**: Um novo amigo pode ser adicionado à sua lista, utilizando a classe **Roster**;

g) **sendPacket**: Envia pacote pela rede. Podem ser pacotes de informação, apenas;

h) **isAuthenticated**: Verifica se o usuário está autenticado, ou seja, logado com o seu usuário;

i) **addPacketListener**:Para caso haja necessidade de análise específica dos pacotes;

3. Configurações a definir:

a) Exemplo da Configurações do ambiente:

```
1 // cria-se uma instancia de SystemConfigurations:
2 SystemConfigurations system = SystemConfigurations.
     getConfigurations();
3
4 system.serversList = new String[] { "192.168.28.13";"
     192.168.28.14" }; // lista de servidores possiveis
5 system.port = 5222; //porta de conexao
6 service = "xmpp-serve"; //nome do servico
7 username = "labsac"; //nome do usuario para autenticacao
8 password = "java"; //senha de autenticacao
```

a) Uso da mudança de estado:

```
manager.requestChangeState(XMPPStatus.ONLINE); // para solicitar
    conexao
manager.requestChangeState(XMPPStatus.OFFLINE); // para
   solicitar desconexao
```

a) Uso do ouvinte de conexão:

```
manager.addServerConnectedListener(new
   ServerConnectionResponseListener() {
          @Override
          public void responseConnection(boolean response) {
            System.out.println("Resposta: " + response);
              if (response);
                // TODO executa alguma acao
              else;
              // TODO executa outra acao
          }


          @Override
          public void connectionDown() {
            // TODO conexao perdida com o servidor
          }
});
```

4.6 Características do FRI

Algumas das principais características do *Framework FRI* são descritas a seguir:

4.6.1 Utilização da comunicação segura

Aplicações da IoT são vulneráveis a ataques de segurança (vide seção 2.5.2) por várias razões, dentre elas D. Conzon *et al.* [18] cita: os dispositivos são fisicamente

vulneráveis e são muitas vezes deixados sozinhos; é difícil de implementar as contra-medidas de segurança devido à grande escala e ao paradigma descentralizado; a grande maioria dos componentes da IoT são dispositivos com recursos limitados e que não podem suportar complexo medidas de segurança.

A principal preocupação com os problemas de segurança são a autenticação e integridade dos dados, as abordagens de autenticação são difíceis de implementar, uma vez que geralmente requerem um infra-estrutura adequada e várias mensagens trocadas (inviável na maioria dos dispositivos com recursos limitados) [18].

Para satisfazer os requisitos da comunicação segura (vide seção 2.5.1) a abordagem implementadas no FRI é destinada a obter uma arquitetura cliente-servidor segura descentralizada, que explora as políticas de segurança (tais como autenticação de usuário, criptografia de canal e prevenção de falsificação de endereço). A arquitetura do servidor XMPP fora isolado a partir da rede pública usando SASL e TLS alcançando essa característica.

4.6.2 Resolvendo problemas de conectividade

Em relação a recursos de comunicação e de redes, o FRI lida principalmente com o objetivo de apoiar a comunicação contínua entre os dispositivos, canais de comunicação não-confiáveis com intermitente conectividade devido à mobilidade ou de interferência. Para tanto o desenvolvimento de soluções nesse cenário devem suprir tais deficiências.

Segundo A. Fox *et al.* [24] o desenvolvimento móvel e da web tem uma prática chamada "offline primeiro", ou seja, desenvolver aplicativos que funcionem tão bem quanto possível sem a rede de dados para apoiá-los. Desta forma, caso os recursos estejam indisponíveis, a perda e a restauração da conectividade precisam ser um processo contínuo que não afete o consumidor final.

Vários problemas ocasionados pela rede são apoiados pelo FRI, tais como: a rede possa desaparecer a qualquer momento ou mesmo durante uma transferência; as mensagens armazenadas localmente podem ser enfileiradas quando offline ou enviadas novamente no caso de uma falha (visto na seção 3.2.2.3); rotinas de avaliação

(visto na seção 3.2.2.2) da conectividade facilitando a não interferência do usuário para se conectar ou não.

4.6.3 Utilizando poucos recursos

No desenvolvimento de aplicações para IoT uma das questões que devem ser levadas em consideração é a utilização racional dos recursos disponíveis. Dispositivos na IoT ainda possuem restrições de processamento, armazenamento, memória RAM, rede, etc. Seria dispensável desenvolver uma solução que drenasse toda a energia ou mesmo que necessitasse de muito espaço, impossibilitado seu uso concomitante com o armazenamento de arquivos.

O FRI foi concebido utilizando poucas classes em sua arquitetura, mostrando-se com baixa sobrecarga em redes com velocidade rápidas e médias, utilizando pouca memória dos dispositivos. A integração com a computação em nuvem compensa as limitações tecnológicas como armazenamento.

4.6.4 Portabilidade

Para que o FRI possa ser utilizado no maior número possível de dispositivos na IoT, o projeto teve como requisito a portabilidade da solução. Atualmente a heterogeneidade de dispositivos traz consigo diversos problemas por não existir atualmente uma padronização total desse paradigma, diferentes empresas criam solução muitas vezes específicas, não possibilitando a utilização de soluções como as apresentadas pelo FRI.

A arquitetura do FRI foi concebido utilizando algumas classes da biblioteca em Java puro *(Smack)*, e da biblioteca para dispositivos móveis *Smackx* com o mecanismo confiável. A solução FRI pode ser incorporada em qualquer aplicação criando um cliente XMPP completo para integrações XMPP simples. Na construção de aplicações para diferentes plataformas o FRI facilita a construção propondo em uma única solução o conjunto dessas bibliotecas.

4.6.5 Suporte a dados de contexto no dispositivos

A utilização de dados de contexto no FRI é provido pela classe *ContextProviders* tais como uso de memória, uso de cpu, informação de bateria. Estes dados de contexto poderão ser utilizadas em diversas aplicações.

Segundo P. Brown *et al.* [14] uma informação de contexto pode ser capturada a partir de sensores, do estado do dispositivo computacional, informações já existentes (diários, previsões de tempo ou valores de ações), ou pelo histórico da interação do usuário com o equipamento. Por exemplo, sensores de uma sala podem detectar que não há movimento, desta forma entendendo que não há pessoas no local e acionando o desligamento das lâmpadas para que haja economia de energia elétrica.

A administração de contexto tornou-se uma funcionalidade essencial em sistemas de software. Esta tendência está crescendo no paradigma IoT, tendo em vista que há vários cenários de aplicações onde é aplicado o monitoramento, tais como: Monitorando a condição e o uso de componentes conectados, dados de contexto do dispositivo para analisar a performance de aplicações, monitoramento de recursos de dispositivos, etc.

Para tanto o FRI utiliza interface de sensoriamento para monitoramento do ambiente, informações do contexto da rede na qual está inserido para uso do mecanismo confiável, como rede cabeada, rede 3G, 4G. Monitoramento do status dos dispositivos no qual se deseja realizar a comunicação.

4.7 Análise comparativa

Conforme descrito no Capítulo 3, existem inúmeros mecanismos, *frameworks*, *middlewares* para Internet das Coisas, baseados na arquitetura cliente/servidor. A Tabela 4.1 apresenta as características que o *framework* proposto possui em relação aos *frameworks* e *middlewares* citados no Capítulo 3 em trabalhos relacionados.

As características do *framework* proposto, como o uso em diversas Plataformas, computação em nuvem, suporte para varias aplicações, coleta de informações de contexto e utilização de mecanismo de conexão e comunicação segura

como confidencialidade, confiabilidade e autenticação. A utilização de mecanismo de análise de conexão e reconexão facilita aos desenvolvedores de aplicações para IoT a reusabilidade de tais mecanismo sem a necessidade de implementação ou mesmo lidar com funcionalidades de baixo nível para a manipulação de tais objetos. A utilização do protocolo XMPP na IoT reduz o problema da interoperabilidade requerida por esse paradigma, destacando a variedade de modelos de aplicações que podem utilizar essa ferramenta proposta neste trabalho.

O uso da tecnologia TLS para produzir um canal de comunicação seguro com o uso da arquitetura de computação em Nuvem *(CloudIoT)* para fornecer elasticidade de recursos e armazenamento ao dispositivos, e o uso em diversas plataformas, são atributos integrados a partir da contribuição de todos os autores da tabela, que visa a fornecer um mecanismo que abranja em sua totalidade estas características.

As contribuições de cada autor influenciaram o desenvolvimento dessa proposta (modelos de comunicação seguros utilizando o cenário de Internet das Coisas e computação em nuvem) visto que em todos os trabalhos há uma nítida importância em destacar o suporte para varias aplicações ao fornecer segurança.

Tabela 4.1: Relação dos trabalhos relacionados ao mecanismos seguros para Internet das coisas

Abordagens	[18]	[12]	[45]	[41]	[10]	FRI proposto
Transport Layer Security	✓	-	-	-	-	✓
Computação em Nuvem	-	-	-	-	✓	✓
Suporte para varias aplicações	✓	-	✓	✓	✓	✓
Uso em diversas Plataformas	✓	✓	✓	-	✓	✓
Utilização de contexto	-	-	✓	✓	-	✓
Confidencialidade	✓	-	-	✓	✓	✓
Confiabilidade	-	✓	✓	-	✓	✓
Autenticação	✓	✓	✓	-	✓	✓

4.7.1 Síntese

O *Framework* de comunicação seguro e confiável possui a utilização do protocolo TLS, que os trabalhos [12], [45], [41] e [10] não utilizam. O uso da

computação em Nuvem apresentado somente em [10] não sendo utilizados nos demais, o suporte para varias aplicações são apresentados em quase todos somente em [12] esse requisito não fora abordado e o uso em diversas plataformas não sendo destacado em [41], o uso de dados contexto sendo apresentado em [45] e [41], os requisitos de segurança como confidencialidade é apresentado em [18], [41] e [10] a confiabilidade por [12], [45] e [10] e a autenticação não sendo apresentado somente por [41].

5 Avaliação do *Framework*

Este capítulo aborda a avaliação do *framework* proposto na pesquisa, tendo como objetivo verificar seu funcionamento. Para tanto foi desenvolvido, uma aplicação de controle de temperatura na arquitetura Raspberry PI 3, um comunicador instantâneo na arquitetura mobile e um cliente para recebimento de dados de controle de temperatura utilizando a linguagem de programação Java e o *framework FRI*.

Primeiramente, foi necessário utilizar os componentes do *framework FRI* utilizando a metodologia de aplicação (seção 4.5). Com o objetivo de verificar o funcionamento do modelo proposto, foram utilizadas as classes de conexão fornecendo confiabilidade e segurança as comunicações. Na construção do ambiente de testes analisamos as funcionalidades de cada módulo e sua conformidade perante os objetivos impostos do FRI. Os experimentos realizados e seu impacto nas aplicações construídas sobre o *framework* são apresentados a seguir.

5.1 Ambiente de desenvolvimento

Para a realização do trabalho foi criado no LABSAC (Laboratório de Sistemas e Arquiteturas Computacionais) da UFMA (Universidade Federal do Maranhão) um ambiente de teste para que os dispositivos IoT envie dados de sensoriamento ao servidor em funcionamento na infra-estrutura em nuvem. O dispositivo utilizado é um Raspberry PI 3 com SO Raspibian, um dispositivos móvel com sistema operacional de fábrica Android, uma aquitetura PC SO Windows7, e um servidor com o ambiente de Nuvem Eucalyptus executando o servidor na máquina virtual no qual foi instanciada o sistema operacional Ubuntu 14.10. Na Tabela 5.1, apresentam-se as características de *hardware* e *software* utilizados para o ambiente de desenvolvimento.

A máquina virtual ”VMSERVIDOR” possui processamento de 2 núcleos, cedidos pelo servidor de Nuvem, que fora configurado no ato da sua criação. O

Tabela 5.1: Características dos dispositivos do ambiente de testes.

Nome da Máquina	IP	Hardware	Software
NUVEM	192.168.28.85	Intel Core i7 com 3.4 Ghz, RAM 16GB, HD 1 TB	Linux CentOS 12.03 server, EUCALYPTUS
VMSERVIDOR	192.168.28.200	4GB RAM, HD 300GB	Ubuntu 14.10, Java 8, OpenFire Server
VMSERVIDOR	192.168.28.205	8GB RAM, HD 500GB	Ubuntu 14.10, Java 8, OpenFire Server
SMARTPHONE	192.168.0.84	Quad-Core 1.5 GHZ, Memória 16 GB, 16 RAM, Rede HSPA+	Android 5.1.1, Softwares de padrão de Fábrica, protótipo XMPPClient
ARQUITETURA PC	192.168.0.103	Processador Intel Core i3, 4 GB RAM, HD 500GB, Rede: Realtek	SO Windows 7, XMPPClienteDesk, Spark
RASPBERRY PI3	192.168.0.150	1.2GHz 64-bit quad-core ARM Cortex-A53 CPU	SO Raspbian, XMPPClienteRasp

dispositivo móvel possui o protótipo de um cliente instalado, além dos *softwares* instalados de fábrica.

O ambiente de testes com os dispositivos inseridos é conforme a Figura 5.1. A rede utilizada é LABSAC, em que os dispositivos Nuvem e VMSERVIDOR, estão conectados por uma rede virtual interna, o VMSERVIDOR possui um endereço IP externo para que haja a comunicação do *Openfire* com as aplicações desenvolvidas utilizando o FRI, que estão sendo executado no dispositivos Smartphone e PC, estes conectados via um roteador sem fio. Utiliza-se como sensor de temperatura e umidade o modelo DHT11 conectado na arquitetura do Raspberry PI 3 para obter os dados de leitura que foram utilizados nas aplicações desenvolvidas.

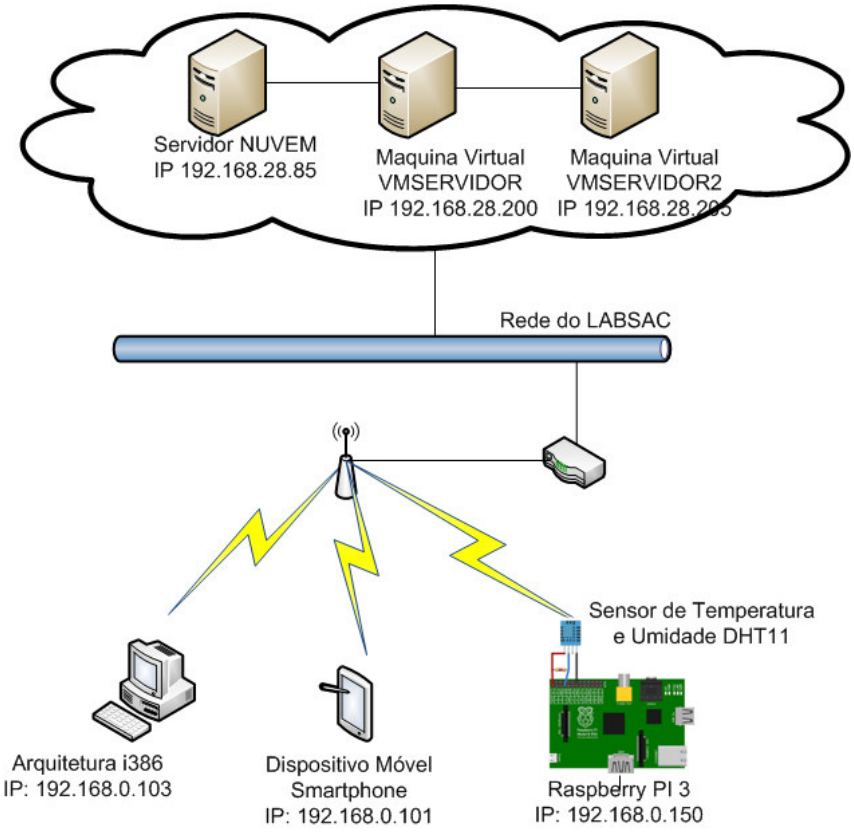


Figura 5.1: Ambiente de testes.

O Servidor *Openfire* inicializado no VMSERVIDOR é um servidor de colaboração em tempo real (RTC) licenciado sob a licença de código aberto Apache, multi-plataforma de comunicação em tempo real. Ele usa o protocolo aberto amplamente adotado pela comunidade que utiliza o protocolo XMPP. Para instalar o servidor e preciso instalar o Banco de Dados MySql, Java, e openssl.

O ambiente que se obtêm os dados de monitoramento de temperatura e umidade foi utilizado o LABSAC na UFMA, sendo este o mesmo laboratório que foi desenvolvido as aplicações.

5.2 Implementação dos protótipos

Como o objetivo deste trabalho é apresentar um *Framework* que auxilie os desenvolvedores nos projetos para diferentes dispositivos no ambiente IoT, identificamos alguns cenário e especificamos algumas aplicações a afim de avaliar os objetivos propostos. Os mais variados cenários podem ser estudados, desde aplicações de monitoramento de temperatura do ambiente até aplicações sofisticadas de distribuição de conteúdo.

Para avaliar o *Framework FRI*, foi realizado o desenvolvimento de uma aplicação cliente que reutiliza as classes do FRI. O cliente desenvolvido utiliza os recursos do FRI para as plataformas heterogêneas com base nas bibliotecas específicas para cada arquitetura, a execução foi efetuada em um ambiente controlado, em que somente os sistemas operacional usados e suas primitivas estariam funcionando, além da aplicações desenvolvidas.

5.2.1 Cliente Raspberry - Controle de Temperatura e Umidade

Foi desenvolvido a partir das classes do FRI, a Aplicação *XMPPClientRasp* com a finalidade de enviar dados do cliente para o servidor em nuvem provendo confidencialidade, integridade, autenticação e não repudio, utilizando como dados de transmissão os valores de temperatura e umidade do ambiente de monitoramento anteriormente descrito na seção 5.1.

A aplicação fora desenvolvida utilizando o Ambiente de desenvolvimento *Eclipse Mars* (https://eclipse.org/mars/) utilizando as classes do FRI, construido sobre a plataforma Raspberry PI 3 com Sistema Operacional *Linux Raspibian*.

A Figura 5.2 mostra a tela de desenvolvimento do protótipo, utilizando o FRI utiliza-se uma interface para aquisição dos valores do sensor de monitoramento. Os dados são lidos e enviados ao servidor e posteriormente aos clientes (Figura 5.3).

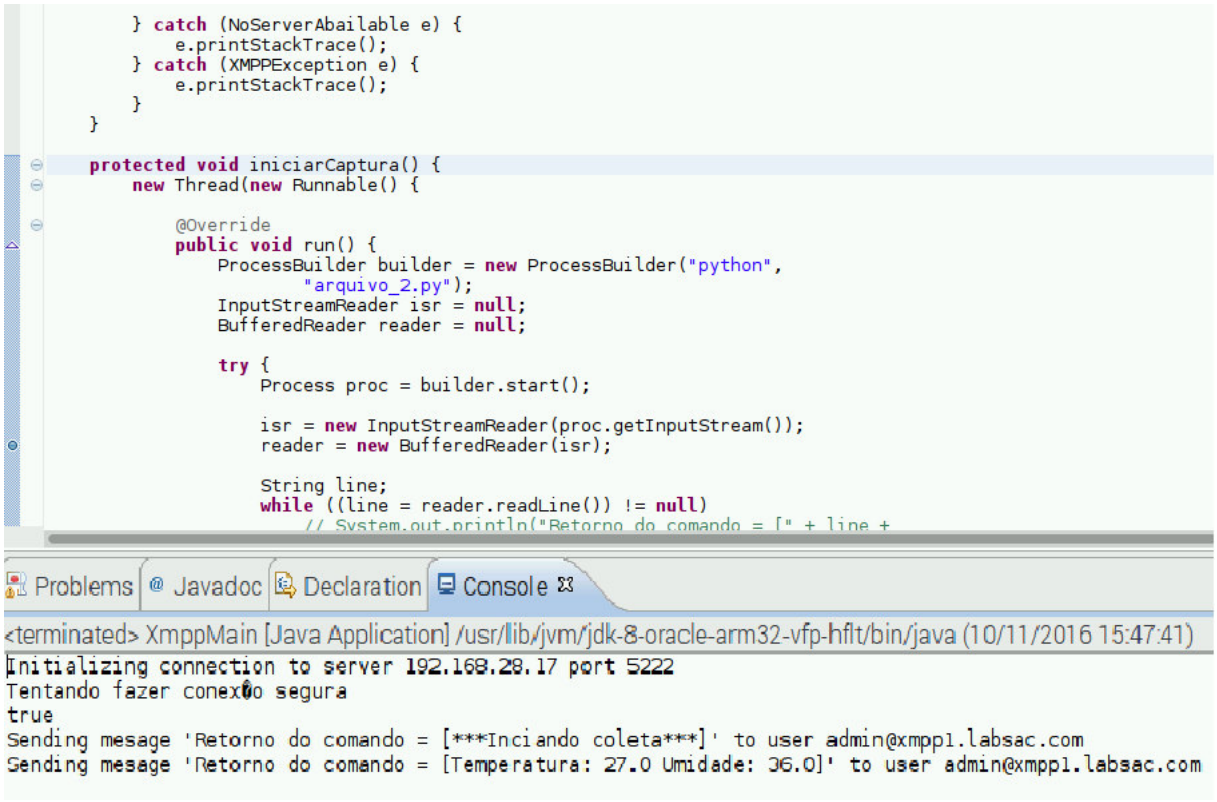


Figura 5.2: Implementação do Cliente Raspberry

5.2.2 Cliente Android - Controle de Temperatura e Umidade

Foi desenvolvido a partir das classes do FRI, a Aplicação *XMPPClient* com a finalidade de receber dados do dispositivo de sensoriamento do Raspberry. A aplicação também envia dados para outros clientes tendo em vista que fora utilizados toda estrutua de desenvolvimento do FRI.

O Aplicativo possui em sua implementação a utilização de métodos de comunicação como destacado nas seção 4.4 (Funcionalidades dos componentes do Framework), tais como envio de mensagem com mecanismo seguro, controle

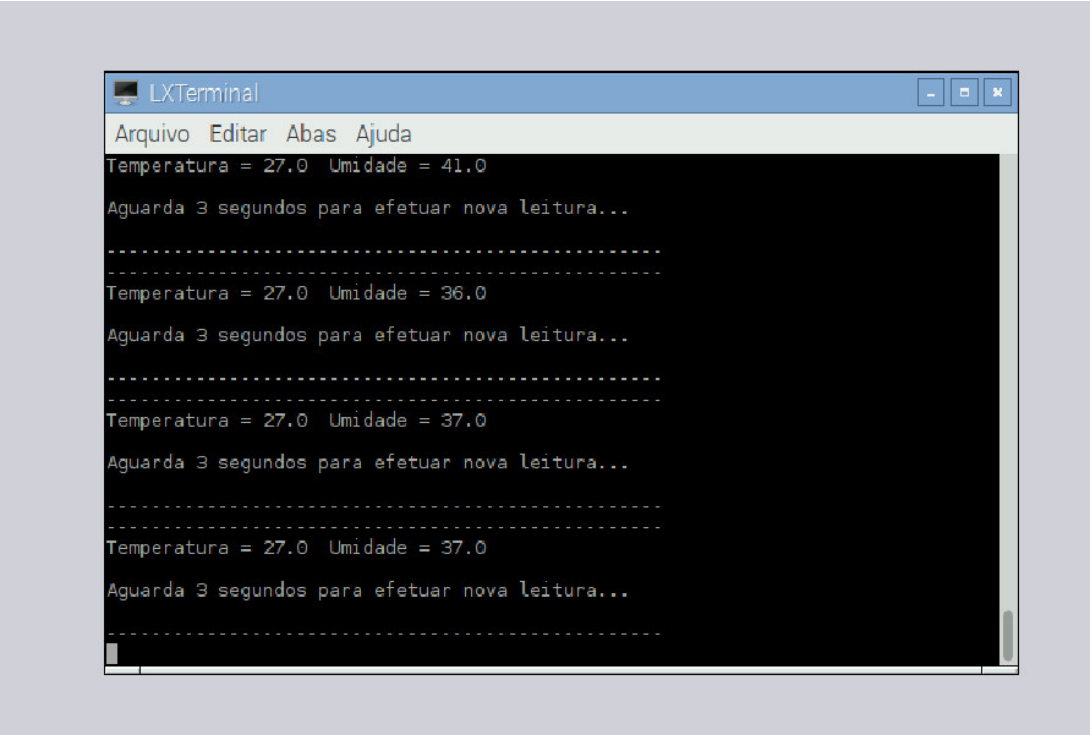


Figura 5.3: Aquisição de dados de sensoriamento

da desconexão e reconexão com servidores, garantia de entrega mensagem ao destinatário, envio de status.

A Figura 5.4 apresenta as telas iniciais da aplicação cliente. A Imagem destacada por a)mostra a tela inicial da aplicação e as opções de menu (Conexão, Desconexão e Configurações). Na Imagem destacada por b) mostra as configurações de conexão. Na imagem c) mostra a tela de recebimento dos dados do sensoriamento.

5.2.3 Outras aplicações

São várias as aplicações que podem utilizar o FRI em diferentes cenários. Qualquer aplicação ou dispositivo que utilizará como funcionalidade principal ou uma das funcionalidade a conexão com a Internet ou a trocar informação com outros objetos. Onde esses dados precisam ser processados, enviados para nuvem ou mesmo a um sistema que realize o tratamento, o FRI se faz útil.

São exemplos de aplicações que podem utilizar o FRI: sistemas de monitoramento de segurança, controle de temperatura de ambiente, gerenciamento de iluminação integrados, envio de dados de câmeras se segurança, envio de dados de alarmes contra incêndio, aparelhos de ar condicionado, lâmpadas e outros itens

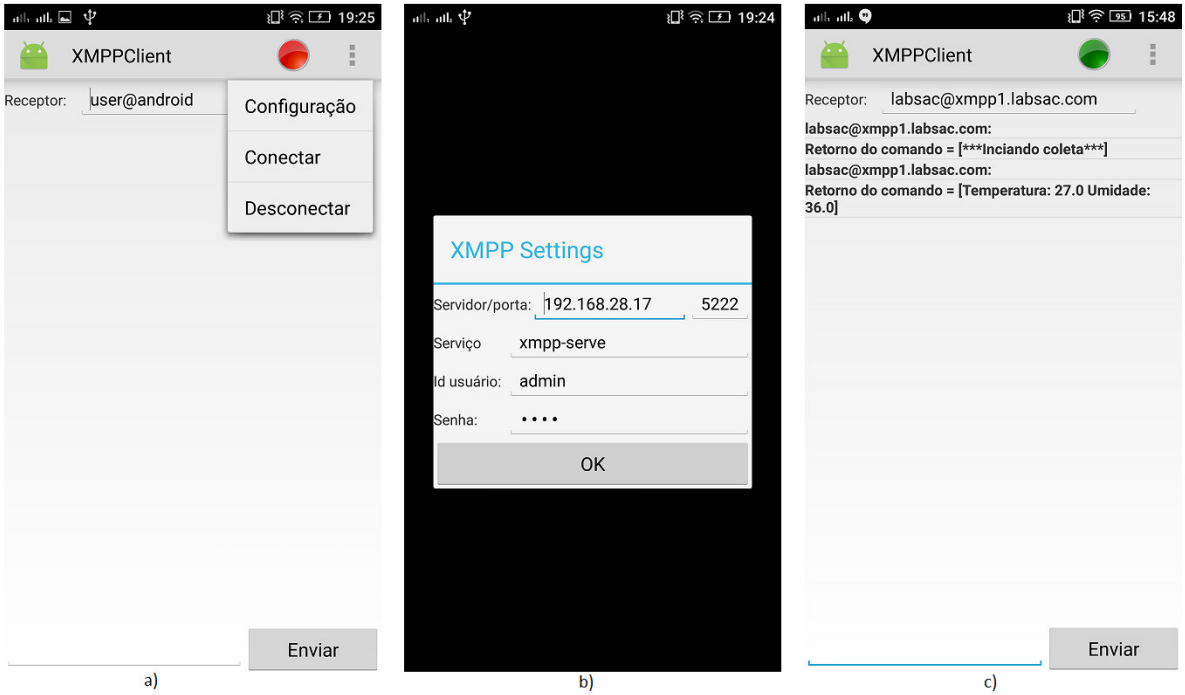


Figura 5.4: Aplicação Cliente no Android.

são que são enviados para um sistema que controla cada aspecto. Aplicações para industria onde há envio e recebimento de dados por máquinas para obter dados estatístico de produção, etc.

Por fim, as possibilidades são inúmeras e, com a integração com novos serviços, novos cenários serão descobertos, originando aplicações mais sofisticadas capazes de tornar o uso dos dispositivos mais práticos.

5.3 Testes e Resultados

Com o objetivo de testar o impacto do FRI ao usar suas funcionalidades verificamos o tempo de envio e recebimento ao realizar uma comunicação no Ambiente de Desenvolvimento (Seção 5.1), e a análise da criptografia de canal. Os teste efetuados de tempo poderão sofrer alterações caso sejam aplicados em outros ambientes, pois o numero de saltos de rede e o tempo de processamento por exemplo, poderão ter influencia direta.

5.3.1 Tempo de Envio e recebimento de mensagem

Para avaliar o FRI utilizou-se o envio de 250 mensagem durante 10 ciclos do cliente do Raspberry PI 3 para o cliente no Android, a fim de avaliar o tempo que o FRI realiza o processamento de análise do ambiente e envia para o destinatário. Executando em um ambiente controlado, em que somente o sistema operacional Raspibian e o Android e suas primitivas estariam funcionando, além da aplicação cliente desenvolvida.

A Figura 5.5 mostra o resultado do envio e recebimento de confirmação. Somente seria realizado o envio da próxima mensagem caso o FRI detectasse a confirmação de entrega da mensagem anterior. Utilizou-se do cálculo (Taxa de entrega = tempo de recebimento - tempo de envio) tendo como tempo dado milisegundos. Seguindo um comportamento regular e quase constante, a taxa de entrega fora calculada utilizando a média simples, considerando os picos de desvio, ficando a 0,16 segundos, como mostra a Figura.

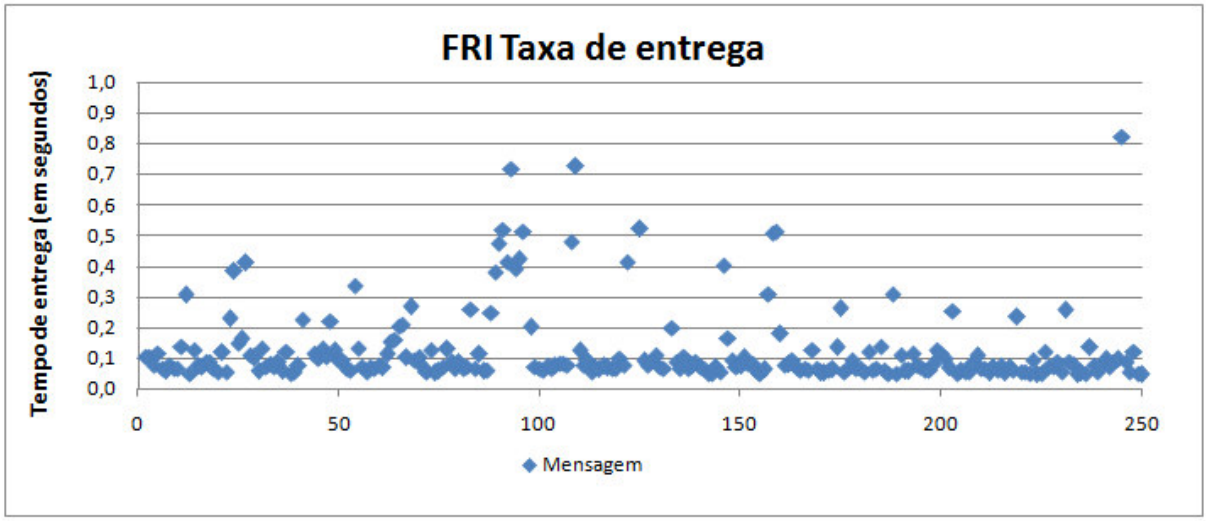


Figura 5.5: Tempo de envio e recebimento do *Framework FRI*.

5.3.2 Análise da criptografia de canal

Como teste da verificação do canal de criptografia foi utilizado o *software Wireshark versão 1.12*. Para maiores detalhes sobre o *software*, acesse o endereço eletrônico (https://www.wireshark.org/). O Wireshark é um programa que analisa o tráfego de rede, e o organiza por protocolos. O Wireshark mostra o tráfego de uma

rede a saber tudo o que entra e sai do computador em diferentes protocolos, ou da rede à qual o computador está ligado.

Neste teste foi analisado o tráfego da rede no Ambiente de teste proposto na seção 5.1. Podemos verificar na Figura 5.6 a análise do protocolo XMPP encriptado, garantindo assim a confidencialidade e a integridade e não-repudio. O detalhamento dessa Figura está exposto nas Figuras a), b) e c). Na Figura 5.7 exibe os pacotes que estão trafegando pela rede, neste caso foi analisado um pacote de mensagem XMPP que foi detalhado em camadas do modelo ISO/OSI pela Figura 5.8. E na Figura 5.9 mostra o conteúdo do pacote, onde neste caso está criptografado.

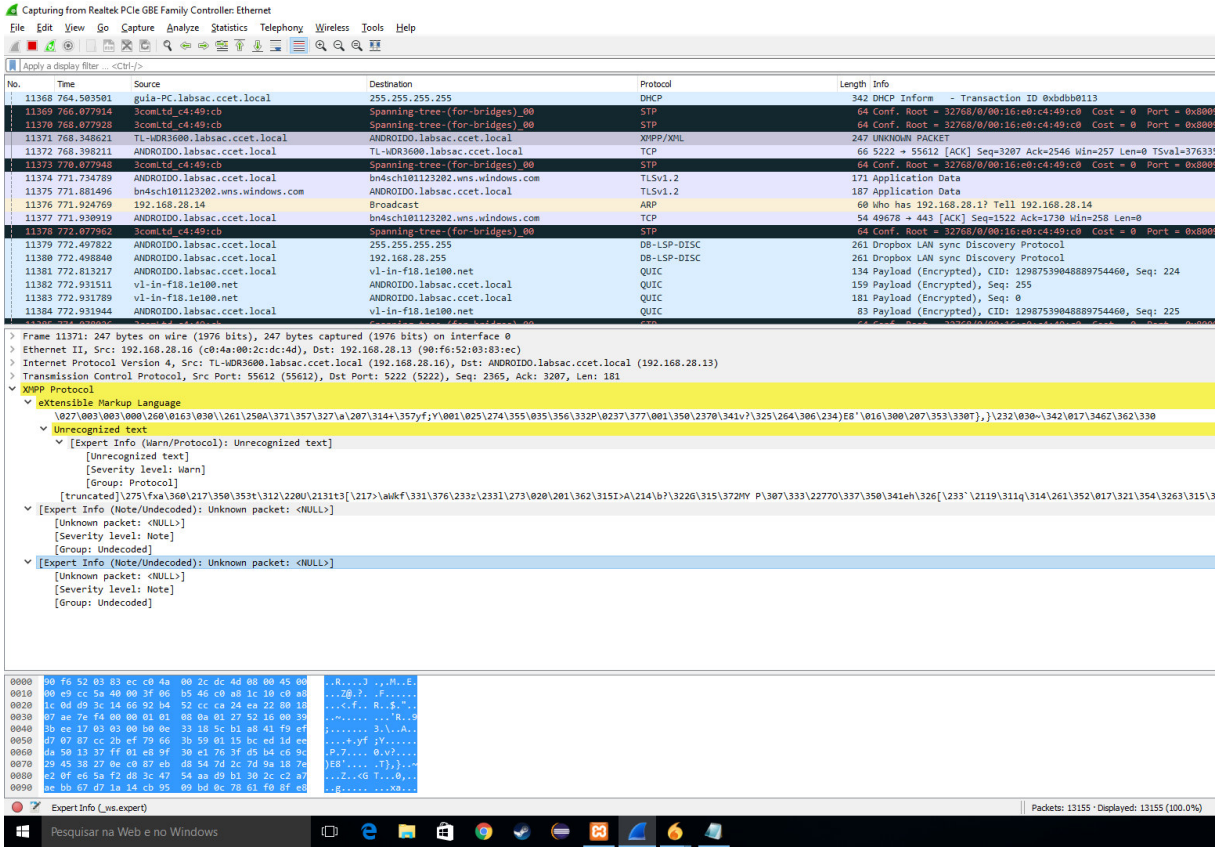
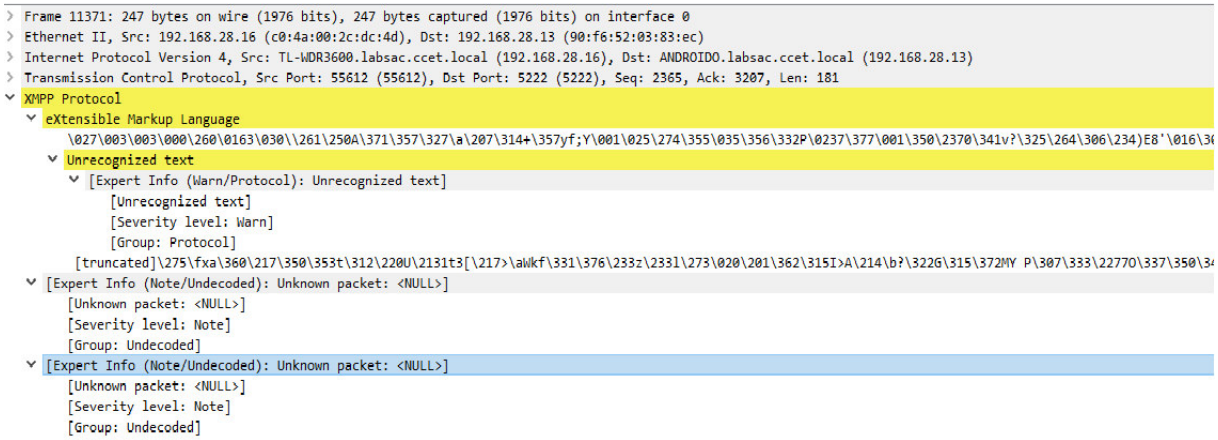


Figura 5.6: Análise de criptografia de canal do FRI.

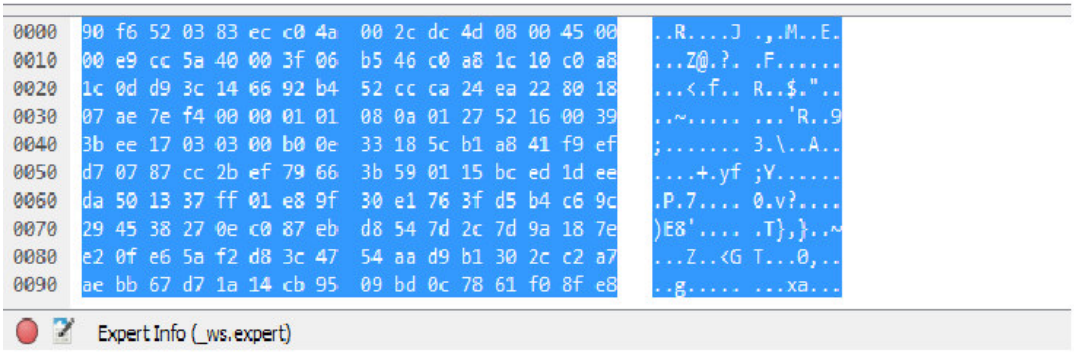
a)

Figura 5.7: Lista de pacotes XMPP trafegando pela rede.



b)

Figura 5.8: Pacote XMPP sendo analisado.



c)

Figura 5.9: Visualização do conteúdo do pacote XMPP.

5.4 Síntese

Neste capítulo, foi apresentado a avaliação do *Framework FRI* no cenário IoT para computação em nuvem proposto nesta dissertação. Realizaram-se testes nos ambientes montados em laboratório demostrando a viabilidade na construção de aplicações e funcionalidades de segurança a partir do uso do mecanismo para o ambiente de rede. As funcionalidades foram comparadas com outros *frameworks* para o mesmo cenário concedendo destaque para as características do trabalho proposto. No capítulo seguinte apresentaremos a conclusão do trabalho, bem como as sugestões para trabalhos futuros.

6 Conclusões e Trabalhos Futuros

Este capítulo apresenta as contribuições deste trabalho, conclusões sobre os resultados alcançados para trabalhos futuros.

6.1 Contribuição do trabalho

Baseado nas pesquisas existentes na área da Internet das Coisas, foi proposto, nesta dissertação, como contribuição, o *Framework Reliable for Internet of Things* - FRI, cujo objetivo é auxiliar o desenvolvimento de aplicações provendo segurança e confiabilidade nas comunicação entre a computação em nuvem e Internet das Coisas. O FRI provê diversos serviços, como gerenciamento de conexão, gerenciamento de segurança, autenticação, informações de contexto e troca de mensagem, possuindo uma arquitetura simples e organizada.

Um solução baseado no FRI é composta por cinco camadas: camada de aplicação na qual localiza-se as soluções para os mais diversos problemas, camada do framework FRI na qual apoia toda comunicação, a camada de servidor, camada da máquina virtual java, tendo em vista que a solução FRI se baseia nessa plataforma, e o sistema operacional, que pode ser os mais variados dependendo da arquitetura especifica. A combinação dessas camadas permite alcançar as metas da solução proposta.

Por fim, desenvolveu-se um protótipo funcional utilizando o *framework* proposto. A principal contribuição é provê uma solução genérica, com capacidade em tempo real de comunicação com vários servidores verificando a confiabilidade na comunicação fornecendo segurança na comunicação entre os dispositivo. Os resultados obtidos na avaliação no Capítulo 5 são considerados satisfatórios, tendo em vista que a solução prover funcionalidades de grande importância na construção de aplicações IoT destacando a sua aplicabilidade em vários cenários de aplicações frente a outras soluções com arquiteturas fechadas e específicas para cada cenário.

6.2 Publicações

O presente trabalho foi aceito para publicação em uma conferência que ocorrerá nos dias 14 e 15 de Dezembro de 2016, na cidade de Pattaya, Tailândia. O evento acontecerá na Mercure Pattaya Ocean Resort. A conferência é intitulada: 10th International Conference on Computer Science and Information Technology (ICCSIT'2016).

6.3 Limitações

Nesta proposta, as seguintes limitações foram identificadas:

- O *framework* se limita a usar outros tipos de sensores de análise de ambiente. Portanto caso seja necessário usar outros tipos de sensores será necessário usar a biblioteca correspondente;
- O FRI por não ser testado em dispositivo sem compatibilidade com a tecnologia Java, fica limitado ao uso em outras plataformas com essa característica;
- O FRI não contempla redes de rádio *bluetooth* para envio de mensagem.

6.4 Trabalhos Futuros

Com o desenvolvimento deste trabalho, algumas possibilidades para trabalhos futuros foram identificados, dentre as quais podemos sugerir:

- Adicionar interface de comunicação para sensores;
- Adicionar compressão de fluxos XML usando EXI -*Efficient XML Interchange* para diminuir a sobrecarga de rede, em redes com limitações;
- Prover interação entre dispositivos próximos que não estejam visíveis através do encaminhamento de mensagens, utilizando redes *bluetooth*;
- A integração com serviços de localização geográficas tornando possível o desenvolvimento de aplicações vinculadas a esse cenário.

Referências Bibliográficas

[1] R. Aitken, V. Chandra, J. Myers, B. Sandhu, L. Shifren, and G. Yeric. Device and technology implications of the internet of things. In *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*, pages 1–4. IEEE, 2014.

[2] H. Akram and M. Hoffmann. Laws of identity in ambient environments: The hydra approach. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*, pages 367–373. IEEE, 2008.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *Communications Surveys & Tutorials, IEEE*, 17(4):2347–2376, 2015.

[4] S. Alam, M. M. Chowdhury, and J. Noll. Interoperability of security-enabled internet of things. *Wireless Personal Communications*, 61(3):567–586, 2011.

[5] M. Albano, L. L. Ferreira, L. M. Pinho, and A. R. Alkhawaja. Message-oriented middleware for smart grids. *Computer Standards & Interfaces*, 38:133–143, 2015.

[6] N. Alhakbani, M. M. Hassan, M. A. Hossain, and M. Alnuem. A framework of adaptive interaction support in cloud-based internet of things (iot) environment. In G. Fortino, G. Di Fatta, W. Li, S. Ochoa, A. Cuzzocrea, and M. Pathan, editors, *Internet and Distributed Computing Systems: 7th International Conference, IDCS 2014, Calabria, Italy, September 22-24, 2014. Proceedings*, pages 136–146, Cham, September 22-24. 2014. Springer International Publishing.

[7] M. Almeida and A. Matos. Bridging the devices with the web cloud: A restful management architecture over xmpp. In J. Rodriguez, R. Tafazolli, and C. Verikoukis, editors, *Mobile Multimedia Communications: 6th International ICST Conference, MOBIMEDIA 2010, Lisbon, Portugal, September 6-8, 2010. Revised Selected Papers*, pages 136–150, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[8] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[9] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad. Proposed security model and threat taxonomy for the internet of things (iot). In *International Conference on Network Security and Applications*, pages 420–429. Springer, 2010.

[10] T. D. P. Bai and S. A. Rabara. Design and development of integrated, secured and intelligent architecture for internet of things and cloud computing. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 817–822. IEEE, 2015.

[11] M. Barros. Mqtt - protocolos para iot. Disponível em: http://www.embarcados.com.br/mqtt-protocolos-para-iot. Acessado em: 8 de Outubro de 2015.

[12] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, and M. Ameling. A service infrastructure for the internet of things based on xmpp. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 385–388. IEEE, 2013.

[13] A. Botta, W. de Donato, V. Persico, and A. Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[14] P. Brown, W. Burleson, M. Lamming, O.-W. Rahlff, G. Romano, J. Scholtz, and D. Snowdon. Context-awareness: some compelling applications. In *Proceedings the CH12000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness*, 2000.

[15] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *A system of patterns: Pattern-oriented software architecture*. Wiley New York, 1996.

[16] R. Buyya, J. Broberg, and A. M. Goscinski. *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons, 2010.

[17] T. d. R. CARMO. Uso do padrão amqp para transporte de mensagens entre atores remotos. Master's thesis, Dissertação (Mestrado em Ciência da Computação)—Universidade de São Paulo, 2012.

[18] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito. The virtus middleware: An xmpp based architecture for secure iot communications. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–6. IEEE, 2012.

[19] T. C. de França, P. F. Pires, L. Pirmez, F. C. Delicato, and C. Farias. Web das coisas: conectando dispositivos físicos ao mundo digital.

[20] T. Dierks, C. Allen, W. Treese, P. Karlton, and A. Freier. P. kocher,"the tls protocol version 1.0. Technical report, RFC 2246, January, 1999.

[21] T. Dierks and E. Rescorla. Rfc 5246: The transport layer security (tls) protocol. *The Internet Engineering Task Force*, 2008.

[22] R. U. M. e Moraes. Ssacc - serviço de segurança para autenticação ciente do contexto: para dispositivos móveis no paradigma da computação em nuvem. Master's thesis, UNIVERSIDADE FEDERAL DO MARANHÃO, 2014.

[23] N. Fernando, S. W. Loke, and W. Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013.

[24] A. Fisher. Melhores práticas para desenvolvimento de iot. *Disponível em: http://www.ibm.com/developerworks/br/library/iot-mobile-practices-iot-success/*, Acessado em: 20 de Abril de 2015.

[25] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[26] Gartner. *Gartner says the Internet of Things will transform the data center*. Disponível em: http://www.gartner.com/newsroom/id/2684616. Acessado em: 20 de Abril de 2014.

[27] Gartner. *Internet of Things definition*. Disponível em: http://www.gartner.com/it-glossary/internet-of-things. Acessado em: 20 de Janeiro de 2015.

[28] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49(11):58–67, 2011.

[29] M. M. Gomes, R. d. R. Righi, and C. A. da Costa. Future directions for providing better iot infrastructure. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 51–54. ACM, 2014.

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

[31] I. E. T. F. (IETF). *The Constrained Application Protocol (CoAP) - Request for Comments: 7252*. Disponível em: https://tools.ietf.org/html/rfc7252. Acessado em: 21 de Outubro de 2015.

[32] J. L. Z. Ismael Rodrigues Martins. Estudo dos protocolos de comunicação mqtt e coap para aplicações machine-to-machine e internet das coisass. 2013.

[33] T. Jaffey. *MQTT and CoAP, IoT Protocols*. Disponível em: http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. Acessado em: 20 de Dezembro de 2015.

[34] J. F. Kurose, K. W. Ross, and A. S. Marques. *Redes de Computadores ea Internet: Uma nova abordagem*, volume 1. Addison Wesley, 2003.

[35] I. Lee and K. Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.

[36] A. Ludovici, P. Moreno, and A. Calveras. Tinycoap: a novel constrained application protocol (coap) implementation for embedding restful web services in wireless sensor networks based on tinyos. *Journal of Sensor and Actuator Networks*, 2(2):288–315, 2013.

[37] S. Manheim. *Suporte ao AMQP 1.0 no Barramento de Serviço*. Disponível em: https://azure.microsoft.com/pt-br/documentation/articles/service-bus-amqp-overview/. Acessado em: 20 de Outubro de 2015.

[38] I. R. Martins and J. L. Zem. Estudo dos protocolos de comunicação mqtt e coap para aplicações machine-to-machine e internet das coisas1, 2. *Americana*, 3(1):64–87, 2015.

[39] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, NIST, Gaithersburg, MD, United States, 2011.

[40] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan. A survey on security issues and solutions at different layers of cloud computing. *The Journal of Supercomputing*, 63(2):561–592, 2013.

[41] R. Neisse, G. Steri, I. N. Fovino, and G. Baldini. Seckit: A model-based security toolkit for the internet of things. *Computers & Security*, 54:60–76, 2015.

[42] O. Neto et al. Síntese de requisitos de segurança para internet das coisas baseada em modelos em tempo de execução. 2015.

[43] C. Newman. Rfc 2595 - using tls with imap, pop3 and acap. *Disponível em: http://ietf. org/rfc/rfc2595. txt.*, Acessado em: 23 de Fevereiro de 2016.

[44] E. Oelinton. *O futuro é a Internet das Coisas – lide com ela*. Disponível em: http://suprimatec.com/2015/10/31/o-futuro-e-a-internet-das-coisas-lide-com-ela/. Acessado em: 16 de Outubro de 2015.

[45] J. Park and M.-J. Lee. Scondi: A smart context distribution framework based on a messaging service for the internet of things. *Journal of Applied Mathematics*, 2014, 2014.

[46] Paulo, F. C. Pires, T. Delicato, T. Batista, E. Barros, M. Cavalcante, and Pitanga. Plataformas para a internet das coisas. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–60, 2015.

[47] R. S. L. Pereira. Análise de desempenho e usabilidade em sistemas voip seguros. 2015.

[48] S. R. d. A. Rita Suzana Pitangueira Maciel. Middleware uma solução para o desenvolvimento de aplicações distribuídas. *CienteFico*, I(3):1–16, 2004.

[49] R. Roman, J. Lopez, and P. Najera. A cross-layer approach for integrating security mechanisms in sensor networks architectures. *Wireless Communications and Mobile Computing*, 11(2):267–276, 2011.

[50] H. Ruschel, M. S. Zanotto, and W. C. MOTA. Computação em nuvem. *Pontifícia Universidade Católica do Paraná, Curitiba, Brazil*, 2010.

[51] P. Saint-Andre. Rfc 3921: Extensible messaging and presence protocol (xmpp): Instant messaging and presence, oct. 2004. *Status: PROPOSED STANDARD*.

[52] P. Saint-Andre. Rfc 3920: Extensible messaging and presence protocol (xmpp). *Core, IETF*, 2004.

[53] P. Saint-Andre and J. Hildebrand. Message delivery receipts. 2011.

[54] S. Schneider. *What's The Difference Between DDS And AMQP*. Disponível em: http://electronicdesign.com/embedded/what-s-difference-between-dds-and-amqp. Acessado em: 5 de Outubro de 2015.

[55] K. Takahashi. *AMQP Básico Ilustrado*. Disponível em: http://blog.locaweb.com.br/artigos/tecnologia/amqp-bsico-ilustrado/. Acessado em: 8 de Outubro de 2015.

[56] L. Tan and N. Wang. Future internet: The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–376. IEEE, 2010.

[57] Y. Tian, T. Hara, and T. Springer. Rtc for mobile-learning: Current state of the technology.

[58] M. S. Wangham, M. C. Domenech, and E. R. de Mello. Infraestrutura de autenticaçao e de autorizaçao para internet das coisas. *Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais—SBSeg*, 2013.

[59] S. William and W. Stallings. *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[60] XMPP. *Extensible Messaging and Presence Protocol*. Disponível em: http://xmpp.org/about/technology-overview.html. Acessado em: 16 de Outubro de 2015.

[61] L. Zhang. *Building Facebook Messenger*. Disponível em: https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920. Acessado em: 9 de Outubro de 2015.

[62] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[63] J. Zhou, T. Leppänen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, and H. Jin. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 651–657. IEEE, 2013.