Departamento de Ciências e Tecnologias de Informação

MONITORIZAÇÃO REMOTA DE SMARTPHONES: ESTUDO E IMPLEMENTAÇÃO DE UMA PLATAFORMA DE CONTROLO PARENTAL

Bruno Gil

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Software de Código Aberto

Orientador:
Professor Doutor Paulo Ricardo Trezentos,
Professor Auxiliar do Departamento de Ciências e Tecnologias de Informação do ISCTE

Julho, 2011

Departamento de Ciências e Tecnologias de Informação

MONITORIZAÇÃO REMOTA DE SMARTPHONES: ESTUDO E IMPLEMENTAÇÃO DE UMA PLATAFORMA DE CONTROLO PARENTAL

Bruno Gil

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Software de Código Aberto

Orientador:
Professor Doutor Paulo Ricardo Trezentos,
Professor Auxiliar do Departamento de Ciências e Tecnologias de Informação do ISCTE

Julho, 2011

Agradecimentos

Agradeço ao meu orientador, Professor Doutor Paulo Trezentos, cuja motivação, sugestões, críticas e disponibilidade durante o desenvolvimento de toda a investigação, foram fundamentais para a melhoria da qualidade do trabalho apresentado.

Agradeço ao Centro de Investigação em Sistemas e Tecnologias de Informação Avançados - ADETTI pela disponibilidade demonstrada no apoio ao desenvolvimento da investigação, sempre que foi solicitado.

Por fim, agradeço a todos os meus amigos cujo apoio e motivação foram igualmente importantes para a concretização deste trabalho.

Resumo

A investigação realizada propõe um modelo de arquitectura para a transferência de informações entre um Smartphone e um Servidor Web. São efectuados estudos sobre as diferentes técnicas de sincronização e são analisados os impactos de três formatos de intercâmbio de dados, no que se refere à gestão de energia e ao custo de processamento em cada um dos formatos propostos; XML, JSON e Protocol Buffers. Os testes efectuados incidiram sobre duas interfaces de rede, Wi-Fi e 3G. Nesta arquitectura é proposto um novo sistema de *push* de dados onde não há a necessidade de uma ligação permanente ao dispositivo móvel.

Como prova de conceito, neste trabalho é implementada uma plataforma de monitorização da informação privada de um Smartphone, com sistema operativo Android, através da Web. A aplicação desenvolvida é uma aplicação de controlo parental, que sincroniza dados privados (SMS, chamadas, contactos, histórico Web, aplicações instaladas, etc…) e localizações geográficas com um Servidor Web. Esta aplicação proporciona a monitorização da informação de um Smartphone sem ter acesso físico ao mesmo, além de permitir a implementação remota de determinadas acções. A implementação desta prova de conceito, plataforma cliente-servidor de controlo parental, teve como objectivo comprovar o modelo de arquitectura proposto.

Os resultados deste trabalho de investigação contribuem para o desenvolvimento de aplicações móveis que necessitem de transferir dados para um Servidor Web. Exemplos de tais implementações são: aplicações de backup de dados ou aplicações de monitorização de informação, tendo como principais objectivos os impactos de energia e processamento no dispositivo móvel.

Palavras-Chave: Aplicações móveis baseadas na Web, Controlo Parental em Smartphones, Monitorização remota de Smartphones, Formatos de intercâmbio de dados.

Abstract

This thesis proposes an architectural model for transferring information between a Smartphone and a Web Server. The current investigation studies different synchronization techniques, analyzing the impacts of three formats for data exchange and also the energy consumption and the cost of processing each of the proposed formats (XML, JSON and Protocol Buffers) on the device. The tests focused on two network interfaces, Wi-Fi and 3G. In this architecture we propose a new data push system where there is no need for the device to keep a permanent connection.

As a proof of concept, this work presents an implementation of a platform for monitoring the private information of a smartphone with Android operating system, through Web. The application developed is a Parental Control application, which syncs private information (SMS, calls, contacts, Web history, installed applications, etc ...) and geographical locations with a Web Server. This application provides a way of monitoring information of a Smartphone without physical access to the same and allows the implementation of actions remotely. The implementation of these concepts, client-server platform for parental control, aims to demonstrate the proposed architecture model.

The result of this work contributes to the development of mobile applications that transfer data with a Web Server. Example of such implementation are: applications of data backup, monitoring of information or any mobile application that synchronizes data with the Web, having as main objectives the minimization of the impacts of energy and processing in the mobile equipment.

Keywords: Cloud-based mobile applications, Mobile Parental Control applications, Monitoring Smartphones, Data interchange formats.

Índice

Índice de Ilustrações

Índice de Tabelas

Glossário

A

Access Point - Dispositivo numa rede sem fios que realiza a interligação entre todos os dispositivos móveis.

Android – Conjunto de software disponibilizado pelo Google para dispositivos móveis que inclui sistema operativo e um conjunto de aplicações pré-instaladas.

B

Broadcast Receiver – Mecanismo de recepção de eventos transmitidos nos sistemas operativos Andoid.

C

Cartão SIM – *Subscriber Identity Module* é circuito integrado que armazena de forma segura a chave IMSI (*International Mobile Subscriber Identity*) usado para identificar os subscritores dos serviços nos dispositivos móveis. Os cartões SIM são removíveis, e podem ser transferidos entre diferentes equipamentos.

Cloud - utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores remotos.

CyberBulling - prática que envolve o uso de tecnologias de informação e comunicação para dar apoio a comportamentos deliberados, repetidos e hostis praticados por um indivíduo ou grupo com a intenção de prejudicar outrem.

F

Framework SDK da plataforma Android - Ferramentas e APIs necessárias para o desenvolvimento de aplicativos na plataforma Android usando a linguagem de programação Java.

G

GSM - *Global System for Mobile Communications* (GSM: originalmente, Groupe Special

Mobile), é um conjunto padrão desenvolvido pelo European Telecommunications Standards Institute (ETSI) para descrever as tecnologias de segunda geração. No GSM, o sinal e os canais de voz são digitais.

I

Intent – Descrição abstracta de uma operação a ser realizada.

Intent Broadcast – Mecanismo de anúncio de eventos nos sistemas operativos Android.

L

Linux - termo geralmente usado para designar qualquer sistema operativo que utilize o núcleo Linux. Foi desenvolvido pelo finlandês Linus Torvalds, inspirado no sistema Minix. O seu código fonte está disponível sob licença GPL para que qualquer pessoa possa utilizar, estudar, modificar e distribuir de acordo com os termos da licença. Pode ser instalado numa grande variedade de hardware, desde computadores, dispositivos móveis, consolas de jogos, etc....

O

Overhead - processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura de banda ou qualquer outro recurso que seja requerido para ser utilizado ou gasto na execução de uma determinada tarefa.

P

Ping - comando que usa o protocolo ICMP para testar a conectividade entre equipamentos. O seu funcionamento consiste no envio de pacotes para o equipamento de destino e na "escuta" das respostas. Se o equipamento de destino estiver activo, uma "resposta" (o "pong", uma analogia ao famoso jogo de ping-pong) é devolvida ao computador solicitante.

Protocolo HTTP - é um protocolo de comunicação utilizado para sistemas de informação de hipermédia distribuídos e colaborativos. O seu uso para a obtenção de recursos interligados levou ao estabelecimento da *World Wide Web*.

Pull – ligação de um cliente a um servidor. A requisição da ligação é efectuada pelo cliente.

Push - notificação automática dos clientes sempre que haja novos dados no servidor. Pressupõe uma ligação permanente entre o cliente e o servidor.

S

Smartphone - é um telefone móvel com funcionalidades avançadas que podem ser estendidas por meio de programas executados no seu sistema operativo.

SMS - *Short Message Service*, é um serviço de comunicações de texto para telefones, Web, ou sistemas de comunicação móveis, que usa protocolos de comunicação padronizados e que permite a troca de curtas mensagens de texto entre dispositivos de linha fixa ou dispositivos móveis de comunicação.

T

Terceira Geração (3G) - Terceira geração de padrões e tecnologias para telefones móveis e telecomunicações móveis, substituindo o 2G. É baseado na família de normas da União Internacional de Telecomunicações (UIT), no âmbito do Programa Internacional de Telecomunicações Móveis (IMT-2000).

Triple Data Encrytion Standard - é um padrão de criptografia baseado no algoritmo de criptografia DES, desenvolvido pela IBM em 1974 e adoptado como padrão em 1977. O 3DES usa 3 chaves de 64 bits. O tamanho máximo da chave é de 192 bits, embora o comprimento actual seja de 56 bits. Os dados são encriptados com a primeira chave, desencriptados com a segunda chave e finalmente encriptados novamente com uma terceira chave.

W

Wi-Fi - é uma marca registada da Wi-Fi Alliance, que é utilizada por produtos certificados que pertencem à classe de dispositivos de rede local sem fios (WLAN) baseados no padrão IEEE 802.11.

1.Introdução

A evolução das redes móveis de comunicação contribuiu para o surgimento de dispositivos capazes de oferecer novos serviços aos utilizadores. Estes dispositivos evoluíram de básicos equipamentos de comunicação para sofisticados equipamentos de informação e comunicação.

Os dispositivos móveis actuais têm capacidades semelhantes aos computadores pessoais, com o potencial de nos acompanharem nas nossas actividades diárias. Podem estar constantemente ligados à Internet disponibilizando-nos uma grande variedade de serviços. Nesta evolução, vários sistemas operativos surgiram, e um novo mercado aplicacional foi criado, originando milhares de aplicações para diversos fins.

Para o desenvolvimento das aplicações surgiram diversas *frameworks*, quase todas associadas a sistemas operativos. A maioria destas aplicações são desenvolvidas para uma determinada plataforma, à semelhança do que acontece com o desenvolvimento computacional. Além das *frameworks* de desenvolvimento as plataformas operacionais apostam cada vez mais na disponibilização de repositórios aplicacionais, conhecidos como "App Stores", permitindo a divulgação e disseminação de aplicações. Esta abordagem fomenta o crescimento do número de aplicações disponíveis e da própria plataforma, atraindo cada vez mais programadores para o desenvolvimento de produtos e serviços.

Nesta nova era da comunicação e informação, a Internet surge como parte importante das actividades sociais e profissionais das pessoas criando-se serviços e ferramentas que dela dependem. No entanto, apresar dos desenvolvimentos e evoluções registadas, os dispositivos móveis apresentam ainda algumas limitações ao nível do processamento, energia e uso de banda de Internet, que devem ser tomadas em conta quando se desenvolve aplicações.

1.1. Contexto

Muitas das aplicações desenvolvidas para os Smartphones têm necessidade de transferir e recolher dados da Web, no entanto, os Smartphones são equipamentos com capacidades limitadas ao nível dos consumos de energia, largura de banda e processamento. Muitas destas aplicações são autónomas na recolha e envio de dados para a Web degradando rapidamente a energia disponível, quer pela quantidade de dados enviados/recebidos como pelo processamento dos mesmos. Nestas situações é importante que todos os programadores que

desenvolvam aplicações com estas características apostem em arquitecturas de implementação que minimizem este problema. Qual será a melhor abordagem e que meios dispõem os programadores de aplicações móveis para o desenvolvimento de aplicações que monitorizem a informação de um Smartphone através de um Servidor Web? Como minimizar os custos associados às limitações de energia?

Por outro lado, a disseminação dos Smartphones nas faixas etárias mais baixas abre um conjunto de novas preocupações no controlo e monitorização da informação. Estudos recentes (International Telecommunication Union, 2011), no contexto do mercado Norte-Americano, revelaram alguns dados interessantes:

- 27% das crianças usam os seus dispositivos móveis para aceder a conteúdos na Web;
- 43% das crianças admitiu que já foi vítima de CyberBulling;
- 39% dos jovens admitiram que já enviaram mensagens escritas (SMS) contendo teor sexual;
- Mais de 35% dos jovens admitiram que já enviaram mensagens escritas (SMS) a amigos com as respostas dos testes escolares;

Perante estes novos desafios, é importante garantir o acesso aos benefícios da tecnologia móvel protegendo as crianças das suas consequências negativas. Das várias medidas descritas no estudo referenciado existe um cruzamento entre duas entidades fundamentais: pais e indústria. Por parte dos pais as regras só podem ser aplicadas se houver conhecimento da forma como as crianças usam os seus Smartphones, no que respeita ao acesso e divulgação de informação. Relativamente à indústria é necessário que esta disponibilize aplicações de controlo parental, que sejam simples de utilizar e compreender. O acesso a conteúdos, serviços e aplicações disponíveis na Web é cada vez mais facilitado. Por outro lado, existe uma dificuldade no controlo físico dos equipamentos, por parte dos adultos. A monitorização física ou local não é eficiente neste novo paradigma de comunicação, pelo que é necessário criar novas formas de controlar a informação de forma eficiente e transparente.

1.2. Objectivo

É objectivo do presente trabalho a apresentação de um modelo de arquitectura e de um modelo de implementação que permita realizar um controlo parental remoto, monitorizando

os dados de um Smartphone, através da Web, minimizando os custos de energia.

A revisão da literatura será fundamental para a análise dos modelos e tecnologias de transferência de dados mais comuns. Esta recolha de informação será importante para a implementação de um modelo de arquitectura baseado na *framework* de desenvolvimento SDK para a plataforma Android. Para a implementação do modelo de arquitectura será criada uma aplicação de controlo parental para a plataforma Android. Esta aplicação irá monitorizar remotamente, de forma autónoma e manual, a informação correspondente a todas as chamadas efectuadas/recebidas/perdidas, todas as mensagens escritas (SMS), todas as aplicações instaladas, todos os favoritos e históricos Web como também as localizações geográficas do Smartphone. Além disso é proposto um mecanismo de *push* de dados, sem haver a necessidade de uma ligação persistente entre o Smartphone e o Servidor Web. No âmbito deste trabalho será desenvolvida a correspondente parte Web da plataforma.

Esta dissertação agrega um conjunto de boas práticas no desenvolvimento de aplicações móveis que necessitem de monitorizar informações, através da Web, minimizando os custos de processamento e gastos de energia.

A implementação prática da arquitectura proposta será na plataforma Android. Esta plataforma é um dos sistemas operativos para Smartphones mais usados a nível global. Apesar da sua recente chegada ao mercado, a sua base instalada tem evoluído exponencialmente, dispondo actualmente de centenas de milhares de aplicações disponíveis no *Android Market.* A título de exemplo até ao presente momento, no *Android Market*, estão disponíveis as seguintes aplicações de controlo parental:

APLICAÇÃO	FORNECEDOR	OBJECTIVO	CLOUD
Android Parental Control	Smart App Cloud	Filtro de aplicações	Não
Mobile Security	Trend Micro	Anti-Malware Web Security Parental Controls Call & Message Filtering	Não
Picture Alert	Aniello & Company	Monitorização da Camera de vídeo Monitorização do Browser Web Monitorização de anexos das mensagens	Não (e-mail)

Tabela 1.1- Aplicações de controlo parental presentes no Android Market

1.3. Metodologia de Investigação

A metodologia de investigação começou com o estudo do estado da arte, onde foram analisados os modelos, tecnologias e *frameworks* de sincronização de dados da plataforma Android. Foram estudados dois formatos de texto e um formato binário para a transferência de informação em ambientes Web. Foi ainda analisada a *framework* de *push* de dados "*Cloud to Device Messaging"* para a plataforma Android.

A implementação e respectivos testes de usabilidade e impacto energético foram realizados na plataforma Android com recurso à *framework* SDK do Google.

1.4. Problema da Investigação

A crescente disseminação dos Smartphones nas faixas etárias mais baixas levanta algumas preocupações no que respeita ao correcto uso da tecnologia. Desta forma, é necessário que se desenvolvam mecanismos tecnológicos de controlo e monitorização da informação consumida e produzida por este grupo de indivíduos. Os Smartphones são equipamentos móveis que acompanham as crianças nas suas tarefas diárias, dificultando o seu controlo e monitorização parental. O desenvolvimento de aplicações que monitorizem remotamente as informações destes dispositivos surge como uma possível solução para este problema. Muitas aplicações que estão disponíveis actualmente para as plataformas móveis fazem uso de ligações à Web para a transferência de dados. No entanto, estas aplicações têm um impacto negativo na gestão da bateria dos equipamentos móveis, sendo fundamental a racionalização e a implementação de um conjunto de boas práticas no seu desenvolvimento.

2.Revisão da Literatura

No presente capítulo é efectuado um estudo de análise sobre os aspectos mais importantes para o desenvolvimento de aplicações que necessitam de sincronizar dados com a Web. Para a realização da revisão da literatura foi escolhida a plataforma Android, dada a sua disseminação e por ser uma plataforma aberta, no entanto, os conceitos abordados são transversais a qualquer plataforma móvel. A estrutura da revisão da literatura é a seguinte:

- No ponto 2.1 é apresentada a plataforma Android, escolhida para a implementação prática do modelo proposto.
- No ponto 2.2 são apresentados alguns estudos relativos à importância da gestão de energia nos dispositivos móveis, analisando-se alguns estudos efectuados para limitar custos de energia inerentes à sincronização de dados entre um Smartphone e a Web.
- No ponto 2.3 são estudados os métodos de transferência de dados na Web. São descritas as técnicas *pull* e *push* com as suas respectivas variantes.
- No ponto 2.4 é apresentado e estudado a arquitectura do serviço "*Cloud to Device Messaging*" disponibilizado pelo Google para o desenvolvimento de aplicações que necessitem de implementar um sistema de *push* de dados. São, ainda, analisadas as APIs de sincronização disponíveis na *framework* de desenvolvimento da plataforma Android.
- No ponto 2.5 são descritos os formatos de intercâmbio de dados estudados no âmbito deste trabalho. São analisados dois formatos de texto e um formato binário. Além disso é analisada a importância da compressão dos dados no processo de transferência de dados na Web.
- No final deste capítulo, no ponto 2.6 são abordados os Web Services REST e SOAP.

2.1. A plataforma Android

A plataforma Android agrega um sistema operativo livre e aberto, que pode ser personalizado pelos operadores e fabricantes de dispositivos, e uma *framework* de desenvolvimento para a criação de aplicações baseadas na linguagem de programação Java. O sistema operativo é baseado no kernel do Linux, devidamente optimizado para os dispositivos móveis, que dá

acesso a uma interface de baixo nível com o hardware para gestão da memória e processos.

Muitas das aplicações que a plataforma Android disponibiliza são baseadas na Web (ex: Gmail, Contactos, Localização etc…). Este novo paradigma das comunicações móveis só foi possível graças à evolução da velocidade de acesso das interfaces de redes (ex: Wi-Fi, 3G). A tendência futura de evolução das velocidades de acesso à Internet, por parte dos dispositivos móveis, irá criar novos serviços baseados na *cloud*, como se operou no mercado dos computadores. No entanto, o desenvolvimento de aplicações baseadas na Web requer cuidados específicos na gestão eficiente dos recursos de energia de um Smartphone.

2.2. A relevância da bateria

Os Smartphones são equipamentos que nos acompanham nas nossas actividades diárias e agregam um conjunto de tecnologias e serviços que consomem os limitados recursos de energia. O gráfico seguinte demonstra os consumos médios das diferentes tecnologias de um dispositivo móvel:

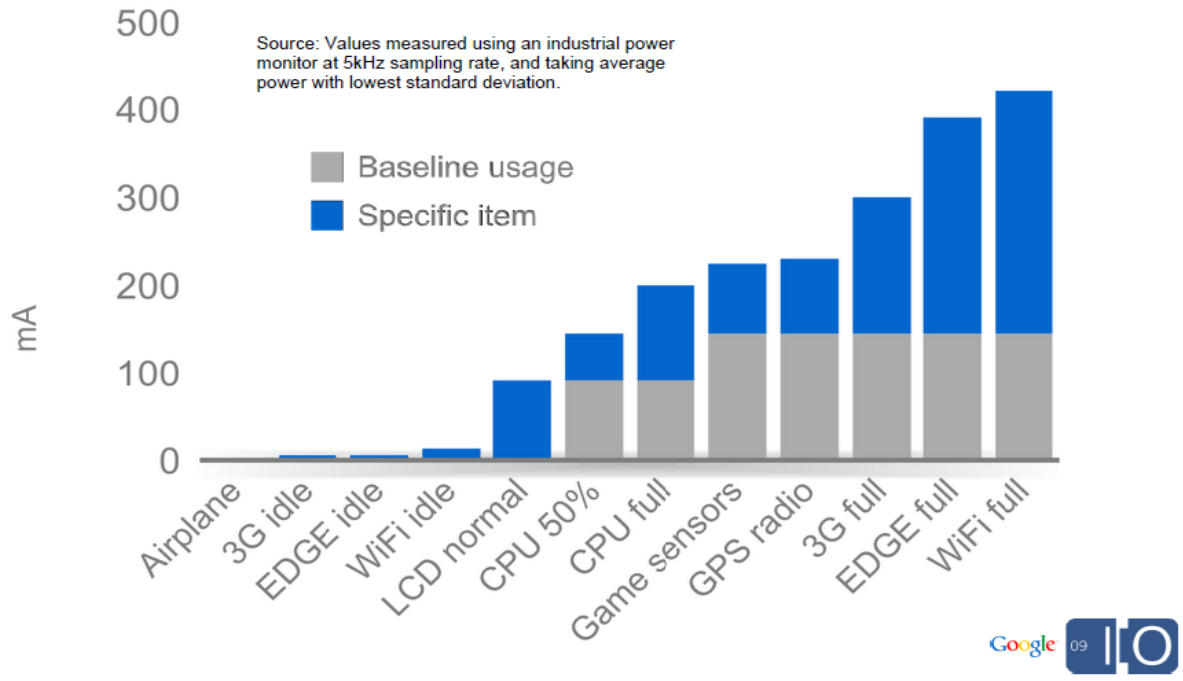


Ilustração 2.1- Consumos das várias interfaces de rede e hardware. Fonte: Sharkey (2009)

Apesar das semelhanças com os computadores, um Smartphone tem limitações que influenciam o desenvolvimento de aplicações que sincronizem dados de forma autónoma com a Web. Uma das limitações mais importantes é o impacto na bateria das várias interfaces de

rede de um Smartphone. Alguns estudos propõem novos modelos de optimização da gestão da energia na utilização dessas várias interfaces. Balasubramanian, et al. (2009), mediram os impactos na bateria de três interfaces de rádio: 3G, Wi-Fi e GSM. Eles concluíram que o consumo de energia está intimamente relacionado com as características da transferência e não apenas com o tamanho total dos dados transferidos. Na interface 3G, os estudos revelaram que a transferência intermitente de poucas centenas de bytes consome mais energia que a transferência contínua de um megabyte, porque foi verificado que a maior parte da energia consumida (cerca de 60%) acontece logo após a transferência dos dados. Esta característica pode ser minimizada através de frequentes e sucessivas transferências, desde que sejam realizadas no curto espaço de tempo em que é mantida a interface ligada. Nas redes GSM a maior parte da energia é gasta na transferência dos dados. Comparativamente com o 3G, o GSM é muito mais eficiente na gestão de energia logo após a transferência. Nas redes Wi-Fi a energia gasta na transferência é significativamente menor do que nas redes 3G independentemente do tamanho dos dados. Quando os custos de procura e associação aos Access Points são contabilizados as redes Wi-Fi são ineficientes comparavelmente às redes GSM mas ainda são mais eficientes que as redes 3G. O modelo que propuseram permite escalonar ou agregar as sincronizações das aplicações que podem tolerar atrasos na sincronização, minimizando o impacto negativo no consumo de energia, tornando as transferências em redes 3G mais eficientes.

Rodriguez et al. (2010), focalizaram o estudo da eficiência da gestão da bateria no contexto de utilização individual dos dispositivos móveis por parte dos utilizadores. Neste tipo de abordagem é proposta a ideia de um modelo de gestão de recursos e processos que aprenda com o próprio utilizador, permitindo uma gestão mais eficiente da bateria. Este modelo consiste em identificar os picos de uso de aplicações no Smarthone no contexto de espaço e tempo, sendo que esta informação poderá ser utilizada para alocar energia para essas aplicações no futuro.

Numa situação em que é necessário transferir constantemente informações de diferentes Servidores Web o processo comum é o cliente realizar pedidos em intervalos de tempo pré-definidos. Este tipo de metodologia tem um custo elevado na bateria de um Smartphone. Armstrong et al. (2010), propõem um sistema que aumenta a eficiência da bateria transferindo o trabalho de *polling* para o servidor. Neste sistema o utilizador informa os dados que deseja sincronizar e essa informação é guardada no servidor. O servidor é responsável por manter os dados actualizados realizando o *polling* de dados com outros servidores, guardando as

referências de actualizações num *update patch* para cada cliente. A figura seguinte demonstra o sistema proposto:

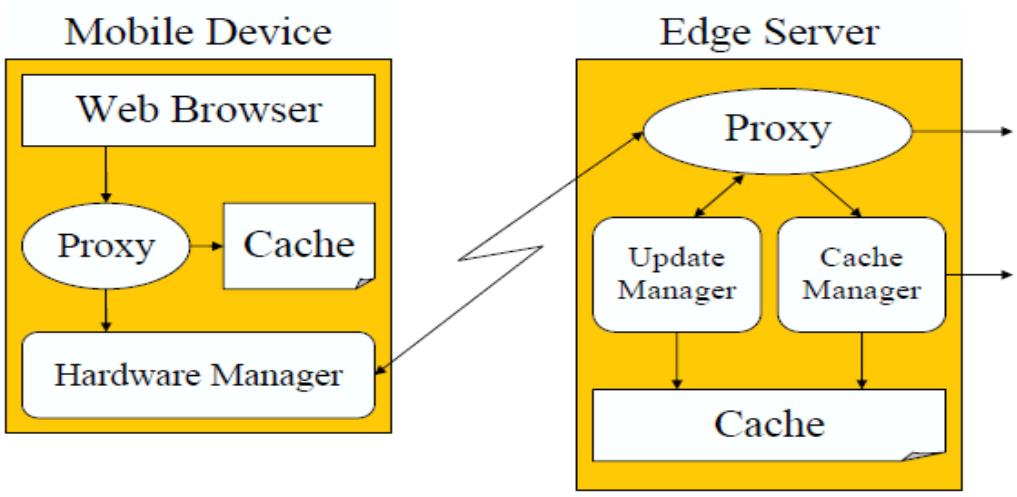


Ilustração 2.2- Sistema eficiente de polling de dados. Fonte: Armstrong et al. (2010)

O sistema inclui um *proxy* (Edge Server) que só realiza um *push* dos dados quando existe uma actualização da parte dos dados que são do interesse do utilizador. O Cache Manager é uma interface responsável por manter o Cache actualizado. O Cache guarda as referências de interesse de cada cliente e sempre que exista uma actualização notifica o Update Manager que regista as alterações no *update patch* de cada dispositivo móvel e notifica ou aguarda um *polling* do cliente. O cliente obtém o *update patch* de três formas possíveis:

- Através de *polling* ao servidor;
- Recebendo um *push* do servidor;
- Recebendo um SMS do servidor com a informação do update;

Esta arquitectura é direccionada para a transferência de dados de um Servidor Web para um Smartphone, no entanto, alguns conceitos de transferência e notificação são transversais às aplicações que transferem dados de um Smartphone para um Servidor Web.

As limitações das baterias dos Smartphones comprometem os programadores de aplicações, que fazem uso das interfaces de rede para a transferência de dados, a um desenvolvimento responsável. Em muitas situações o processo de *polling* de dados não é eficaz e as plataformas têm criado serviços que permitem experiências em tempo real para as aplicações baseadas na *cloud*. Esta abordagem tem impactos positivos na gestão eficiente de energia dos Smartphones mas não abrange um conjunto de aplicações em que as actualizações são menos constantes.

2.3. Técnicas de sincronização: Pull e Push

Muitas das aplicações que fazem parte dos equipamentos móveis usam ligações à Internet para actualizarem os seus dados. As metodologias comuns para estas sincronizações de dados são o *pull* e o *push*. *Pull* é o método que tem sido mais utilizado nas interacções que ocorrem na Internet – um utilizador clica num link e o Browser faz o *pull* de conteúdos de um servidor. Se o servidor quiser enviar novos dados para o cliente, aguarda até nova requisição do cliente (Roden, 2010). A metodologia *pull* tem sido utilizada por várias aplicações que fazem parte do nosso dia-a-dia, como sejam, o protocolo POP de email e fontes RSS. Esta metodologia tem dominado as interacções na Web porque é de fácil implementação e funciona em muitas situações. O que é difícil de determinar são os intervalos de frequência de *pull*. Em muitos casos não existem dados a sincronizar e mesmo assim é feita uma ligação, consumindo bateria e rede.

A ideia da tecnologia *push* é a notificação dos clientes sempre que haja novos dados no servidor. Neste caso, uma ligação é mantida entre um servidor e um cliente e são enviados novos dados sempre que necessário. Este tipo de metodologia permite poupar recursos também nos servidores, além de proporcionar aos utilizadores uma experiência mais atraente (Roden, 2010). Menos pedidos traduzem-se em menos largura de banda e processamento consumido, porque o servidor não está constantemente a verificar e responder a requisições dos clientes.

No ambiente das aplicações para Smartphones as aplicações como o Google Contacts, Calendar e Gmail usam a metodologia *push* para sincronização dos dados.

Duan et al. (2005) descrevem, de uma forma mais completa, estes dois tipos de modelos: *sender-push model* e *receiver-pull model*. No *sender-push model* o servidor sabe a identidade do cliente e faz o *push* da mensagem de uma forma assíncrona. O cliente recebe a mensagem, podendo examiná-la e depois aceitá-la ou recusá-la. Um aspecto importante nesta arquitectura é que a mensagem é inteiramente recebida e só depois há lugar a processamento no cliente.

Ilustração 2.3- Sender Push Model. Fonte: Duan et al. (2005)

Uma variante apresentada no *sender-push model* é o *receiver-intent-based-sender-push*

(RISP) model. Serviços como as mailing lists e aplicações de actualização de software usam esta variante. Uma das vantagens desta variante em relação ao *sender-push model* é que o cliente tem algum controlo no feedback ao servidor. A figura seguinte ilustra esta arquitectura:

Ilustração 2.4- Receiver Intent Based Sender Push. Fonte: Duan et al. (2005)

No *receiver-pull model* é o cliente que contacta o servidor. O servidor espera por requisições e devolve conteúdos. Neste modelo o cliente tem um controlo total sobre a mensagem e confia totalmente no conteúdo devolvido. Os protocolos HTTP e FTP são um exemplo de *receiver-pull model*. A figura seguinte ilustra este modelo:

Ilustração 2.5- Receiver Pull. Fonte: Duan et al. (2005)

A variante apresentada para o *receiver-pull model* é designada de *sender-intent-based-receiver-pull* (SIRP). Neste modelo o servidor informa o cliente que pretende enviar conteúdos e o cliente se tiver interesse contacta o servidor através da tradicional técnica de *pull*. Um exemplo desta arquitectura são as tradicionais mensagens de "*call me*", onde uma mensagem é deixada ao receptor para posterior contacto a esse número. Aqui o cliente tem o controlo de quando e como recebe os conteúdos. A figura seguinte ilustra esta arquitectura:

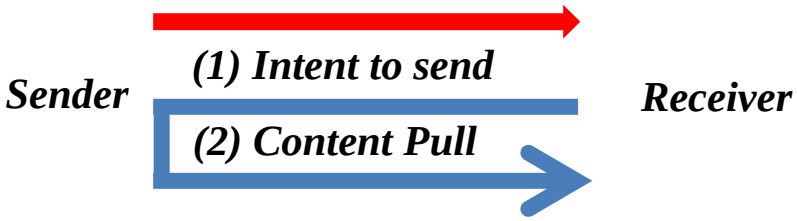


Ilustração 2.6- Sender Intent Based Receiver Pull. Fonte: Duan et al. (2005)

A escolha de utilização de cada um dos modelos deve ter em conta a aplicação que se desenvolve e em alguns casos é necessário uma mistura das duas metodologias em determinados contextos. Deolasee et al. (2001), analisaram estas duas metodologias na transferência de dados dinâmicos na Web. O método de *pull* é caracterizado por ser um

método que não oferece alta-fidelidade quando os dados se alteram frequentemente no servidor. Além disso, este modelo requer uma grande quantidade de comunicações. O método *push* oferece alta-fidelidade quando os dados no servidor alteram frequentemente, no entanto, requerem ligações permanentes com o servidor.

Algorithm	Resiliency	Temporal Coherency	Overheads (Scalability)		
			Communication	***Computation***	***State Space***
Push	Low	High	Low	High	High
Pull	High	Low / High	High	Low	Low

Tabela 2.1- Quadro comparativo das duas metodologias. Fonte: Deolasee et al. (2001)

Neste estudo são propostos dois algoritmos (PaP[1] e PoP[2]), onde em PaP um proxy faz o *pull* dos dados e o servidor faz o *push* dos dados que não foram apanhados pelo *proxy*. O algoritmo PoP, permite ao servidor escolher entre o *push* ou *pull*, dependendo das características como: os recursos disponíveis, os tempos definidos de sincronização ou frequência de mudança dos dados no servidor.

2.4. Sincronização na plataforma Android

Os dispositivos móveis operam em ambientes heterogéneos, caracterizados por ligações lentas, intermitentes e de alta latência. Dadas estas circunstâncias o desenvolvimento eficiente de uma metodologia *push* com uma ligação permanente entre o dispositivo e um servidor é um desafio complicado. Além disso, uma grande concorrência de ligações num mesmo dispositivo tem um impacto negativo na gestão de energia.

Um dos principais trunfos da plataforma Android é a disponibilização e integração de um conjunto de aplicações e serviços Google, baseados na Web. Esta implementação é possível porque a Google mantém uma ligação persistente a cada um dos dispositivos, podendo implementar a técnica de *push* para a sincronização de dados. A plataforma Android, tal como outras plataformas, disponibiliza uma *framework* para a implementação da técnica de *push* para aplicações terceiras, fazendo uso da sua ligação, eliminando os impactos associados à existência de diversas ligações persistentes, especialmente na gestão de energia. Neste estudo foram analisados o serviço "*Cloud to Device Messaging"* e as APIs de Rede.

[1] Push-and-Pull
[2] Push-or-Pull

2.4.1. Cloud to Device Messaging

Cloud to Device Messaging (C2DM) é um serviço disponibilizado na versão 2.2 da plataforma Android com o objectivo de ajudar os programadores a implementar um sistema de *push* de dados nas suas aplicações. Este serviço permite o envio de mensagens de um servidor para uma aplicação no dispositivo móvel. O serviço fornece um mecanismo simples e leve que os servidores podem usar para alertar as aplicações, instaladas nos Smatphones, para iniciarem contacto com o servidor directamente, com o objectivo de sincronizar dados. O serviço C2DM controla todos os aspectos de entrega e gestão de mensagens entre o servidor e a aplicação instalada no dispositivo (C2DM, 2011). As suas principais características são:

- Permite a um servidor enviar mensagens a aplicações instaladas nos dispositivos móveis. É desenhado para o envio de pequenas mensagens apenas para dizer à aplicação que novos dados estão disponíveis. A imagem seguinte ilustra o modelo da *framework* (Ghosh, 2010):

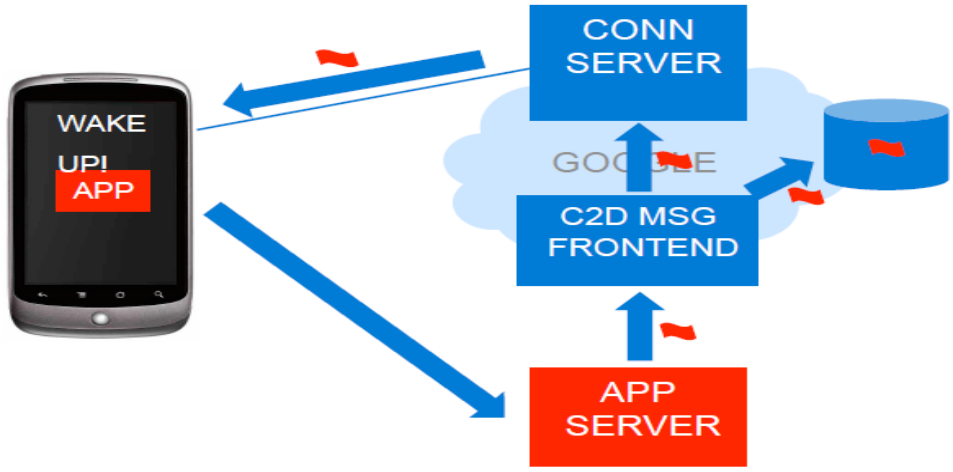


Ilustração 2.7- Funcionamento do sistema C2DM. Fonte: Ghosh (2010)

- A aplicação não precisa de estar a correr em *background* para receber as mensagens. O sistema envia um *Intent Broadcast* quando recebe a mensagem e a aplicação, com o adequado *Broadcast receiver* e respectiva permissão, captura esse *Intent*.
- O tipo de processamento é da responsabilidade da aplicação, este serviço apenas envia mensagens do servidor ao dispositivo.
- Usa a ligação dos serviços Google e requer no mínimo a plataforma Android 2.2, a aplicação Market e uma conta Google configurada no dispositivo.

O C2DM corre um serviço que é iniciado quando a rede está disponível e mantem uma ligação ao servidor através de constantes *pings*, para a verificação do estado da ligação. Usa o alarme do sistema para efectuar os *pings* e aguarda resposta do servidor. O servidor também pode iniciar os *pings*.

A arquitectura faz uma eficiente gestão de entrega de mensagens, sendo capaz de guardar as mensagens caso o dispositivo não se encontre *on-line*, para isso a aplicação informa o sistema sempre que houver *broadcast* de activação/desactivação do ecrã.

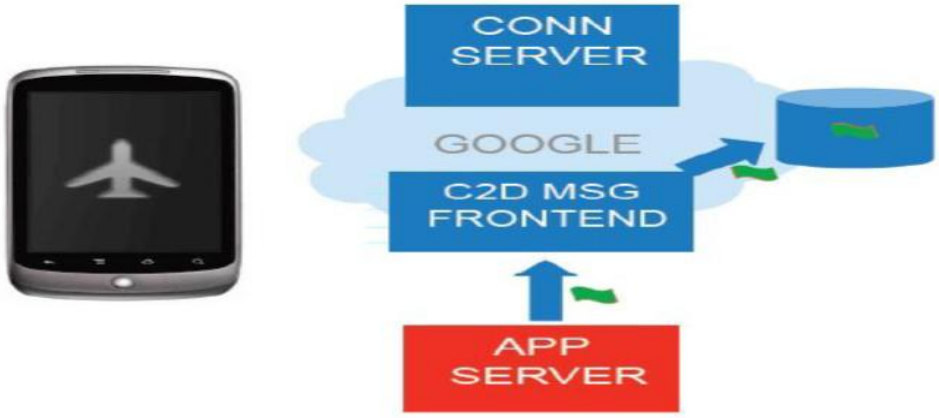


Ilustração 2.8- Gestão de entrega de mensagens. Fonte: Ghosh (2010)

Esta *framework* é eficiente porque permite numa só ligação servir várias aplicações, no entanto, depende dos serviços do Google.

2.4.2. APIs de sincronização

A *framework* aplicacional do Android é composta por (Meier, 2010):

- APIs de base (bibliotecas), que fornecem a maior parte das funcionalidades e que resultam das APIs de base do Java e APIs específicas do Android;
- Máquina virtual Dalvik, especificamente desenhada e optimizada para dispositivos móveis, baseada no kernel do Linux para a gestão dos processos e memória;
- APIs que permitem desenvolver aplicações que sincronizem dados em rede (Android Developers, 2011):
    - Android.net.*, conjunto de classes que ajudam no acesso a redes, além das classes implementadas pelo java.net.* APIs;
    - Java.net.*, conjunto de classes do Java que implementam componentes de

acesso a redes;

- org.apache.http.*, conjunto de classes e interfaces para componentes HTTP. Esta API fornece funcionalidades para o uso do protocolo HTTP, como a representação de mensagens e ligações. Fornece também, classes para o uso de requests e responses

Além destas APIs de base de acesso a redes, quer através do uso de Sockets ou através do protocolo HTTP, a *framework* aplicacional do Android implementa uma API designada de android.content.*.

Esta API contém classes que permitem aceder e publicar dados no dispositivo. Oferece as seguintes classes, que são úteis, na sincronização de dados (Android Developers, 2011):

- PeriodicSync, contem informações sobre uma operação de sincronização periódica, implementada sobre um ContentProvider;
- SyncContext, contexto onde a sincronização ocorre;
- SyncInfo, informações sobre as sincronizações que estão a ocorrer;
- SyncResult, comunica os resultados de uma operação de sincronização com o SyncManager;
- SyncStats, estatísticas sobre o resultado de uma operação de sincronização;

Além destas classes existem métodos nas classes que implementam funcionalidades de publicação e gestão de dados de forma permanente no dispositivo, que permitem o uso e facilitam a sincronizam de dados entre o dispositivo e um Servidor Web.

2.5. Formato de intercâmbio de dados

Um dos factores que influencia o processamento e gestão de energia, na transferência de dados entre um dispositivo móvel e um Servidor Web, é a escolha de um formato de intercâmbio de dados e de um analisador sintáctico eficiente. O impacto na gestão de energia de um conjunto de aplicações que sincronizam dados da Web varia consoante o tamanho e o tempo de processamento dos dados a serem transferidos, sendo importante estudar os formatos mais eficientes para limitar estes custos. De seguida, é feita uma análise a dois dos formatos de texto mais conhecidos (XML e JSON) e a um formato binário desenvolvido pela

Google, designado de Protocol Buffers.

2.5.1. Extensible Markup Language (XML)

XML é um formato simples de texto muito flexível derivado do SGML (ISO 8879). Originalmente concebido para responder aos desafios das grandes publicações electrónicas, XML, desempenha também um papel cada vez mais importante na troca de uma ampla variedade de dados na Web (W3C, 2011).

Apesar de não implementar todas as APIs do Java para o trabalho com o formato XML a plataforma Android oferece as APIs javax.xml.parsers, org.w3c.dom e org.xml.sax que permitem trabalhar com DOM e SAX (Android Developers, 2011). XML é largamente utilizado nas interacções entre aplicações na Internet e serve de base para a comunicação com Web Services do tipo SOAP.

Exemplo de aplicação:

```
<funcionario>
    <nome>Manuel</nome>
    <morada>Rua de baixo…</morada>
    <telefone>6223892673</telefone>
</funcionario>
```

2.5.2. JavaScript Object Notation (JSON)

JSON é um formato de intercâmbio de dados leve. É fácil de ler e escrever para os seres humanos. É fácil para as máquinas de analisar e gerar. É baseado num subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3 ª Edição - Dezembro de 1999. JSON é um formato de texto que é completamente independente, mas usa convenções derivadas do C, incluindo C, C + +, C #, Java, JavaScript, Perl, Python, e outros. Estas propriedades fazem do JSON uma linguagem de intercâmbio de dados adequada para aplicações cliente-servidor (JSON, 2011).

A plataforma Android também suporta JSON na sua *framework* base através da API org.json (Android Developers, 2011).

Exemplo de aplicação:

{Funcionario: { Nome: ‘Manuel', Morada: ‘Rua de baixo…', Telefone:622389267}}

2.5.3. Protocol Buffers

Protocol Buffers é um formato binário de intercâmbio de dados. É uma linguagem e uma plataforma neutra do Google para a serialização de dados estruturados. Para usar este formato temos de definir a estrutura dos dados uma única vez e depois gerar o código fonte para a sua manipulação a partir de uma variedade de fluxo de dados e usando uma variedade de linguagens – Java, C++ ou Python (Protocol Buffers, 2011).

A plataforma Android suporta o formato Protocol Buffers através da inclusão da API com.google.protobufs. Para a implementação deste formato binário é necessário a construção de um ficheiro, com extensão proto, que defina a estrutura das mensagens. Este ficheiro serve de base para a construção do código através do compilador do Protocolo Buffers.

Exemplo de aplicação (ficheiro proto):

```
package pcandroid;

option java_package = "parental.control.messages";
option java_outer_classname = "MessagesProtos";

message Calls{
        required string phoneNumber = 1;
        required int64 date = 2;
        required int32 duration = 3;
        required int32 type = 4;
}
message PortfolioCalls{
        repeated Calls calls = 1;
}
```

2.5.4. Estudos comparativos - XML vs JSON vs Protocol Buffers

2.5.4.1. Analisador sintáctico

Na escolha do formato a utilizar existem algumas variáveis importantes. No desenvolvimento de aplicações para equipamentos móveis os recursos limitados de processamento e de bateria são das condicionantes mais importantes. Vários estudos e testes Sharkey (2009), Galpin (2010) foram realizados em dispositivos móveis para avaliar as velocidades de análise

sintáctica e o tamanho dos dados gerados por cada um destes formatos.

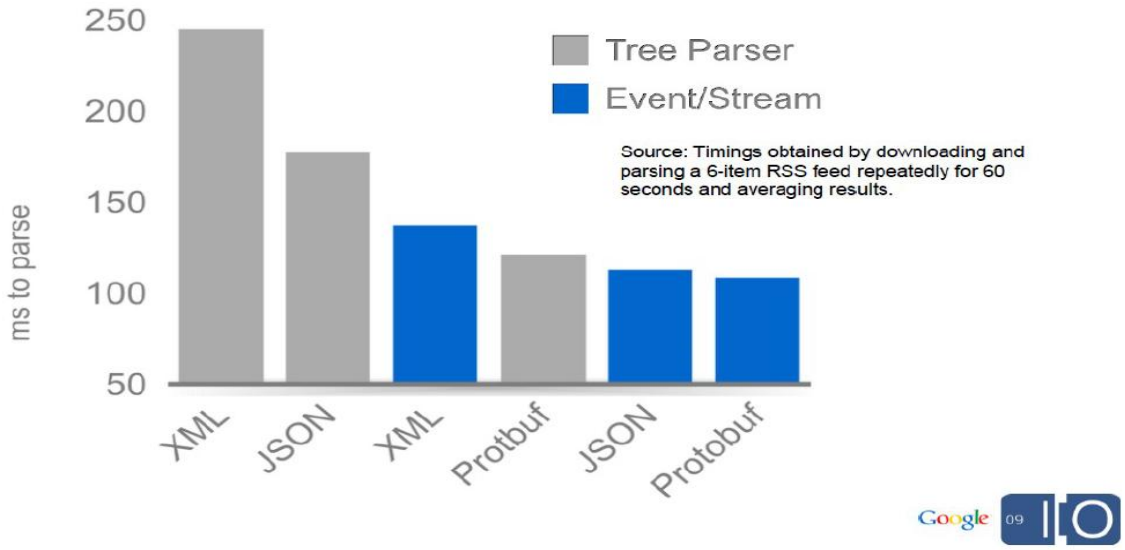


Ilustração 2.9- Velocidade de análise sintáctica dos dados. Fonte: Sharkey (2009)

Galpin (2010) realizou um estudo comparativo, que simulou a requisição de dados a um servidor e mediu o tempo desde a requisição até os dados serem visualizados no dispositivo. Este estudo foi efectuado 50 vezes para cada formato em dois tipos de dispositivos: Motorola Droid e HTC Evo, utilizando a interface de rede 3G.

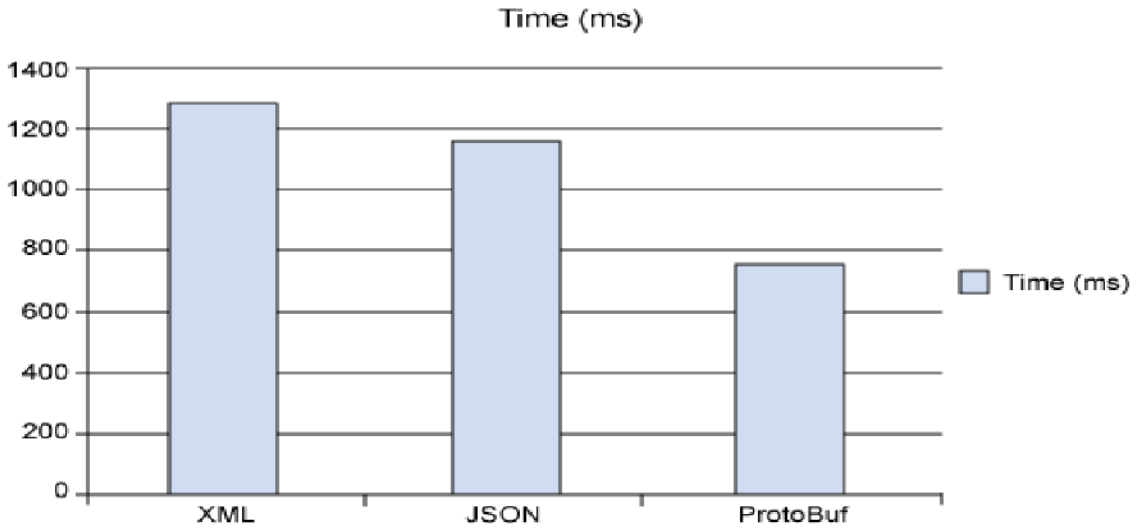


Ilustração 2.10- Comparativo de performance entre formatos de dados. Fonte: Galpin (2010)

O gráfico demonstra que os Protocol Buffers (750 ms) obtêm o melhor resultado comparativamente ao XML (1300 ms) e ao JSON (1150 ms). Um dos factores que influencia

a performance ao longo da transmissão é o tamanho dos dados. Os Protocol Buffers ao serem um formato binário têm um tamanho menor comparativamente aos formatos de texto, como o XML e JSON.

2.5.4.2. Compactação dos dados

A compactação dos dados de texto é fundamental para a redução do tamanho da informação a sincronizar. Após a aplicação da compactação dos dados, os formatos de texto conseguem uma melhor performance, invertendo a situação anterior. O gráfico seguinte ilustra o tamanho dos dados enviados, relativos ao estudo anterior, com e sem compactação, através do Gzip:

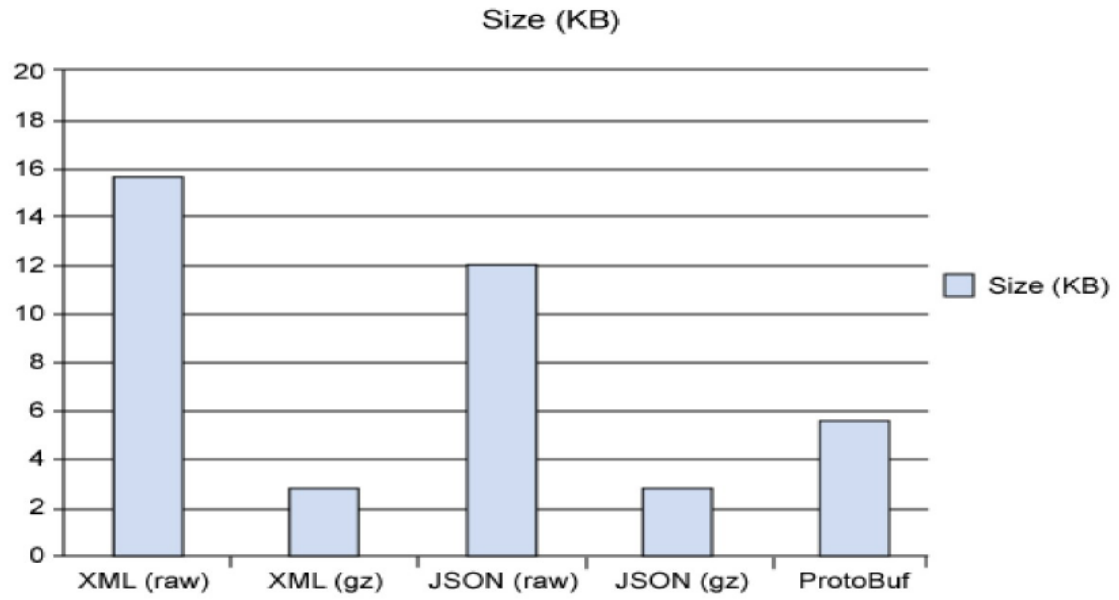


Ilustração 2.11- Tamanho dos dados com e sem compactação. Fonte: Galpin (2010)

Como se pode verificar, o formato Protocol Buffers é mais eficiente relativamente ao tamanho dos dados gerados quando não existe compactação. No entanto, assim que se compactam os dados de texto, o formato Protocol Buffers passa a ser menos eficiente que os formatos de texto compactados. Nos dispositivos móveis, com recursos limitados de processamento é necessário avaliar o custo-benefício da compressão de dados, porque apesar de se poupar energia no tempo de transferência esta poderá ser consumida no processo de descompactação.

Outro estudo, apresentado por Sharkey (2009), compara os tempos médios de download dos dados com compactação e sem compactação em diferentes interfaces de rede. Os resultados são ilustrados no seguinte gráfico:

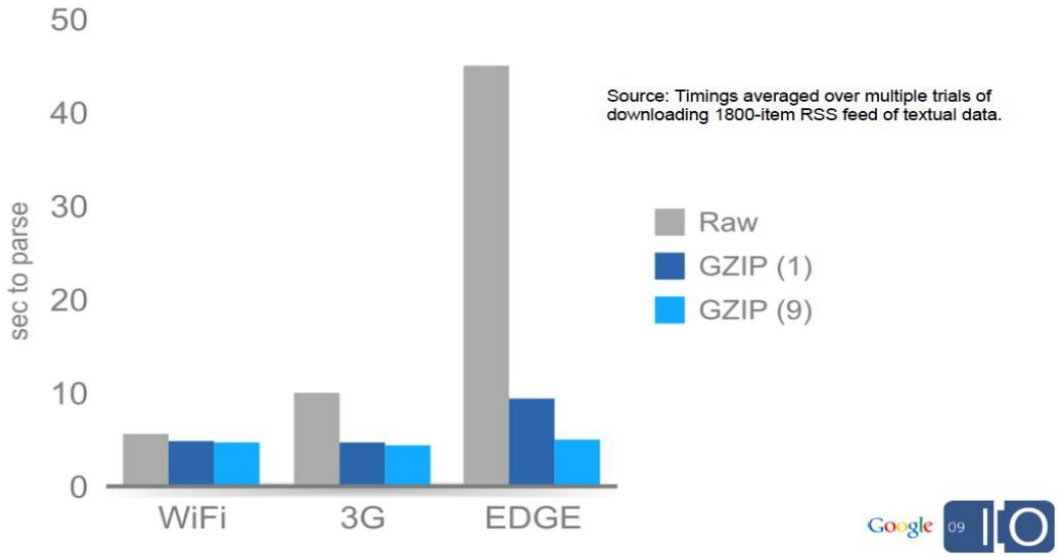


Ilustração 2.12- Tempos médios de download de dados compactados e não compactados nas várias interfaces de rede. Fonte: Sharkey (2009)

Da comparação entre os dados sem compactação (Raw) com os dados com compactação (Gzip) conclui-se que o tempo de sincronização varia de uma forma mais acentuada consoante a velocidade da interface de rede. A compactação dos dados ganha maior importância em interfaces de redes mais lentas.´

Nurseitov, et al (2009), realizaram uma comparação entre os dois formatos de intercâmbio de dados de texto, XML e JSON, no que se refere à utilização de recursos, especialmente memória e CPU, e relativamente à performance das aplicações que usam este tipo de formatos de intercâmbio de dados. Os resultados mostraram que o formato JSON é significativamente mais rápido e usa menos recursos que o XML.

2.6. Web Services

As aplicações dos dispositivos móveis, que fazem transferência de dados com a Web, consomem Web Services para a disponibilização dos dados aos utilizadores. Um Web Service é um tipo de aplicação *Webified*, ou seja, uma aplicação que funciona sobre o protocolo HTTP. Um Web Service é, portanto, um aplicativo distribuído onde os seus componentes podem ser implementados e executados em dispositivos distintos. Por exemplo, um Web Service pode consistir em vários componentes de código, hospedados num servidor, e o Web Service pode ser consumido por PCs, dispositivos móveis ou outro tipo de equipamentos (SunMicrosystems, 2008).

Um Web Service deve conter (Cerami, 2002):

- Uma descrição de si próprio. Quando é publicado um novo Web Service, deve também ser publicado a interface desse Web Service através de um documento num formato de leitura humana para ser fácil a integração do serviço. Esta interface deve ser escrita em XML, identificando todos os métodos públicos, seus argumentos e valores que devolvem.
- Deve poder ser descoberto. Deve conter um mecanismo que permita a todos os interessados descobrir o serviço e a localização da sua interface pública. Este mecanismo pode ser um sistema descentralizado ou um sistema de registo centralizado.

Um Web Service é um serviço que (Cerami, 2002):

- Está disponível na Internet ou numa rede privada (intranet);
- Usa protocolos abertos;
- Não está associado a nenhum sistema operativo ou linguagem de programação;
- Utiliza o XML para descrever os seus métodos;
- Está acessível através de mecanismos de procura simples;

Os Web Services permitiram a passagem do paradigma "*Human-centric Web*", onde as requisições eram iniciadas maioritariamente por pessoas para "*Application-centric Web*" onde as aplicações iniciam autonomamente as requisições e manipulam as respectivas respostas (Cerami, 2002).

Actualmente existem dois grandes grupos de Web Services: Simple Object Access Protocol (SOAP) e Representational State Transfer (REST).

2.6.1. Simple Object Access Protocol (SOAP)

SOAP é um protocolo de comunicação para troca de informações entre computadores na Internet e é baseado na linguagem XML. SOAP pode ser utilizado em diversos sistemas de mensagens e pode utilizar vários protocolos de transporte, sendo o mais comum o HTTP. SOAP permite às aplicações, de forma simples, ligarem-se a servidores remotos para invocarem métodos nesses servidores.

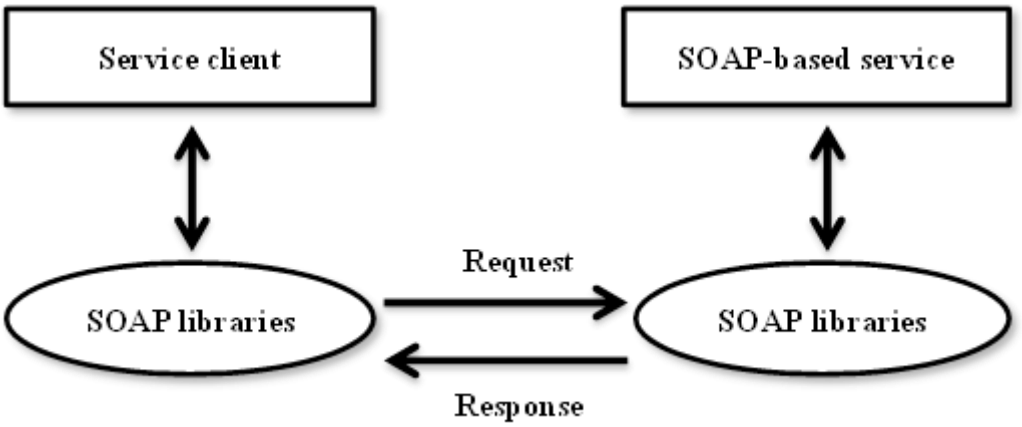


Ilustração 2.13- Sistema SOAP. Fonte: Kalin (2009)

A descrição dos Web Services na arquitectura SOAP é feita através do WSLD (Web Service Description Language). O WSLD é a especificação para descrever os Web Services através de XML. WLSD descreve:

- Interface com todos os métodos públicos disponíveis;
- Tipo de dados de todas as mensagens;
- Protocolo de transporte a ser usado;
- Localização de cada serviço;

2.6.2.Representational State Transfer (REST)

REST é um modelo de arquitectura que define como os dados são representados, acedidos e modificados na Web. Na arquitectura REST, os dados e métodos são considerados recursos e são acedidos através de URIs (Uniform Resource Identifiers), tipicamente links na Web. A arquitectura REST é uma arquitectura cliente-servidor que utiliza o protocolo HTTP. A representação dos recursos entre os clientes e os servidores é baseada numa interface e protocolos standards. Estes princípios fazem das aplicações REST, aplicações simples e leves (SunMicrosystems, 2008).

RESTful Web Services são aplicações Web que utilizam a arquitectura REST e utilizam os quatro métodos HTTP para criar, obter, actualizar e eliminar recursos. A imagem seguinte ilustra o sistema REST:

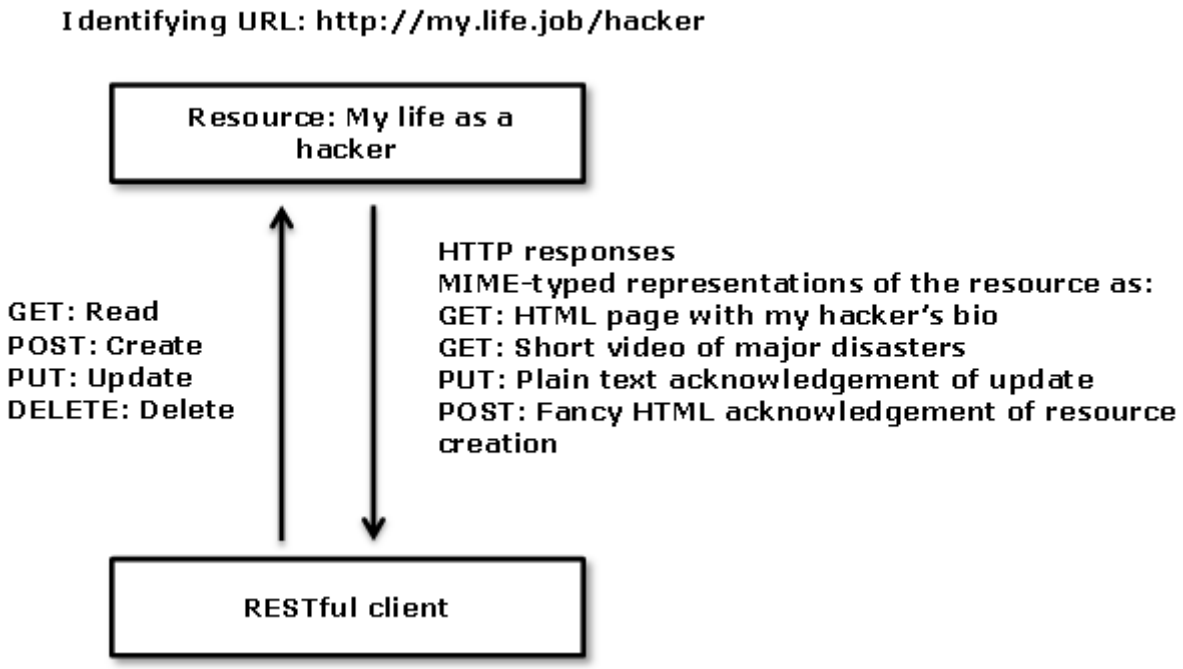


Ilustração 2.14- Sistema REST. Fonte: Kalin (2009)

3.Prova de conceito

A prova de conceito consistiu no desenvolvimento de uma plataforma de monitorização remota de dados de um Smartphone para controlo parental. A sua principal função é disponibilizar uma forma de controlo da informação pessoal e das localizações geográficas dos Smartphones sem necessidade de acesso físico aos equipamentos.

A plataforma engloba uma aplicação que corre no dispositivo móvel e que é responsável pelo envio da informação para um Servidor Web que processa os dados recebidos e os disponibiliza aos utilizadores. A aplicação móvel implementa muitos dos conceitos analisados na revisão da literatura que serão descritos neste capítulo. No entanto, tornou-se necessário o desenvolvimento de uma aplicação Web para testar a plataforma. Apesar da plataforma de implementação ser Android, a aplicação desenvolvida utiliza conceitos transversais a qualquer plataforma variando apenas as técnicas de implementação.

Este capítulo divide-se em três áreas fundamentais que descrevem de forma detalhada toda a infra-estrutura e testes realizados. As áreas são as seguintes:

- Arquitectura
    - Descrição do modelo de arquitectura e das funcionalidades de toda a plataforma;
- Implementação
    - Descrição técnica da implementação prática dos conceitos analisados na plataforma Android;
- Testes
    - Descrição e apresentação dos resultados obtidos nas seguintes métricas:
        - Tempo médio despendido na compressão dos dados a transferir segundo os diversos formatos (XML, JSON e Protocol Buffers);
        - Tempo médio despendido no processo de transferência da informação com e sem compressão de dados;
        - Tempos médios de análise sintáctica, no servidor, de cada um dos formatos;
        - Avaliação dos impactos na gestão da bateria do Smartphone ao nível do

processamento e ao nível do uso das interfaces de rede para cada um dos formatos estudados, com e sem compactação de dados;

3.1. Arquitectura

A plataforma consiste numa aplicação móvel, designada de "Android Parental Control", que funciona de forma autónoma, sem intervenção do utilizador, ou de forma activa em que o utilizador opera activamente as acções. A aplicação móvel tem a particularidade de só correr quando for estritamente necessário, mantendo-se desligada caso não tenha qualquer operação a realizar. Todas as definições de envio e definições de gestão da aplicação móvel requerem o acesso físico ao dispositivo. Um utilizador pode monitorizar vários Smartphones através da Web e operar acções em cada um deles. A plataforma só funcionará correctamente caso encontre um cartão SIM instalado no Smartphone.

A arquitectura geral da plataforma é ilustrada na imagem seguinte, e incorpora duas técnicas de sincronização: sincronização de dados em intervalos pré-definidos e sincronização por *push* de dados, sendo implementada a variante "*Receiver Intent Based Sender Push*".

Receiver Intent Based Sender Push da plataforma Android Parental Control, consiste no envio de uma mensagem para a aplicação móvel a pedir uma sincronização de dados

Ilustração 3.1- Arquitectura da plataforma Android Parental Control

A aplicação móvel, instalada nos Smartphones, realiza o papel mais activo na plataforma, enviando os novos dados ao servidor através de frequentes sincronizações. A arquitectura da plataforma incorpora um sistema de *push*, onde os dados são enviados através de uma sincronização de dados quando a aplicação móvel recebe uma mensagem do utilizador com a intenção de receber novos dados. Este sistema é uma variante do sistema de *push* e é designado de "Receiver Intent Based Sender Push", já analisado na revisão da literatura (Ilustração 2.4). Este tipo de arquitectura não necessita de uma ligação permanente ao Smartphone, evitando o desgaste de bateria e as dificuldades de manutenção de uma ligação activa, no entanto, depende da existência de um cartão SIM activo. Esta estratégia é eficiente para as aplicações que estão fortemente ligadas ao servidor, isto é, aplicações que partilham os dados privados dos cartões SIM, numa base de confiança mútua. A aplicação móvel mantem-se desligada até que seja instruída, quer pelo sistema ou pelo utilizador, a efectuar uma sincronização de dados para o servidor. Desta forma, evita-se o consumo de energia desnecessário, aumentando a longevidade da bateria.

O correcto funcionamento da plataforma depende da existência de um cartão SIM instalado no Smartphone, pois é a partir da análise dos dados do cartão que o servidor contextualiza a informação recebida. Este tipo de abordagem permite que o correcto funcionamento da plataforma não esteja dependente do dispositivo físico possibilitando a migração para qualquer outro dispositivo desde que o cartão SIM seja o mesmo.

O fluxo de informação entre o Smartphone e o servidor faz-se nos dois sentidos, no entanto, a predominância é no sentido Smartphone para o Servidor Web, visto ser esta a génese da aplicação, recolha e envio de dados do Smartphone para o Servidor Web. Estes dados respeitam uma estrutura pré-definida que será detalhada na fase de implementação da plataforma. As sincronizações de dados, na plataforma, podem ser feitas por agregação ou de forma autónoma. Na sincronização por agregação, a aplicação móvel regista-se no sistema para ser sincronizada juntamente com outras aplicações, limitando os custos de energia, especialmente nas redes 3G, como analisado na revisão da literatura. A imagem seguinte ilustra a sincronização por agregação da plataforma:

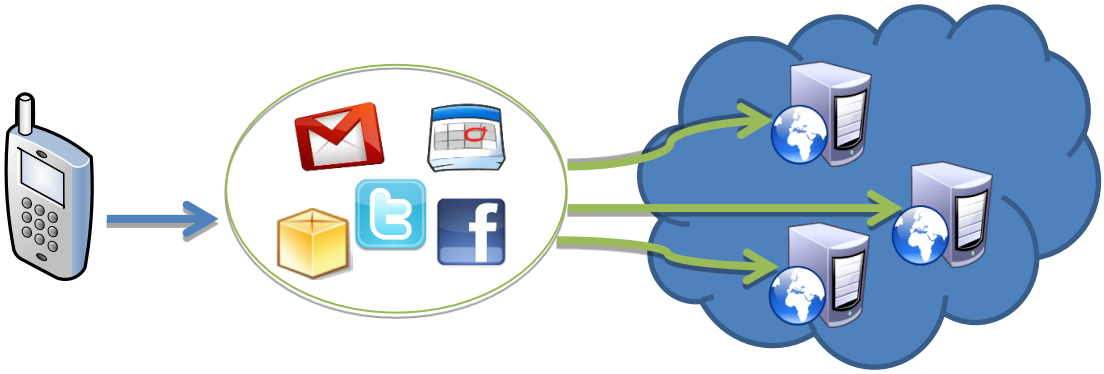


Ilustração 3.2- Plataforma em modo de sincronização por agregação

Neste tipo de modelo, os dados não são enviados em intervalos de tempo precisos e é mais indicada para as aplicações que permitem uma flexibilidade nos tempos de envio. É mantida uma ligação de rede activa até que todas as aplicações sincronizem os seus dados, tornando a gestão de energia mais eficiente, embora os custos de utilização de energia sejam diferentes para as diversas interfaces de rede existentes, como analisado na revisão da literatura.

Na forma de sincronização autónoma, a arquitectura é diferente na medida em que as aplicações acedem aos recursos das interfaces de rede de forma autónoma. A imagem seguinte ilustra a arquitectura de funcionamento da plataforma quando é utilizado este modo de envio.

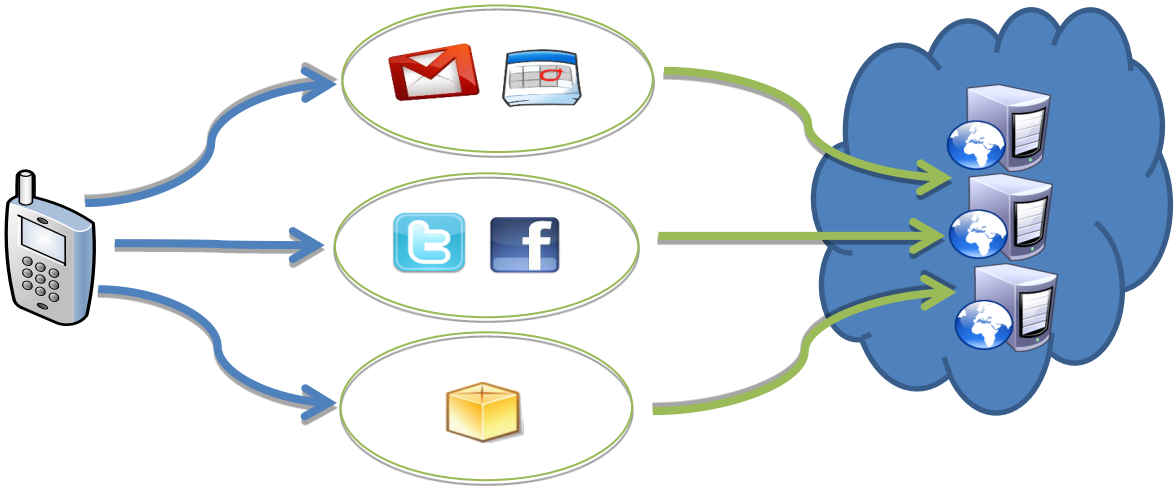


Ilustração 3.3- Plataforma em modo de sincronização autónoma

Neste tipo de modelo a aplicação realiza as sincronizações de forma independente, degradando mais rapidamente a bateria do Smartphone, mas garantido por outro lado, que as transferências de dados são realizadas nos intervalos de tempo pré-definidos por cada aplicação. Este tipo de arquitectura é ideal para as aplicações que não permitem atrasos no processo de sincronização.

Os dados recebidos pelo Servidor Web são processados e estruturados numa Base de Dados sendo retornado ao Smartphone as datas das últimas sincronizações. O papel do Servidor Web é oferecer uma interface de manutenção e monitorização dos dados aos utilizadores e responder, através do processamento de análise sintáctica, à informação recebida dos Smartphones autenticando sempre a origem da informação.

O uso de interfaces de rede é da responsabilidade do sistema operativo. A aplicação usa o interface activo, no entanto, a gestão dos interfaces de rede pode ser configurada nas definições da plataforma Android.

Além das sincronizações autónomas, que têm um impacto negativo na gestão de energia, a arquitectura da plataforma permite sincronizações por acção activa dos utilizadores. Estas sincronizações podem ser por acesso directo ao dispositivo ou por acesso remoto. Desta forma, existe a possibilidade de fazer sincronizações por indicação expressa dos utilizadores, limitando os custos de processamento desnecessário. As imagens seguintes retratam esses processos.

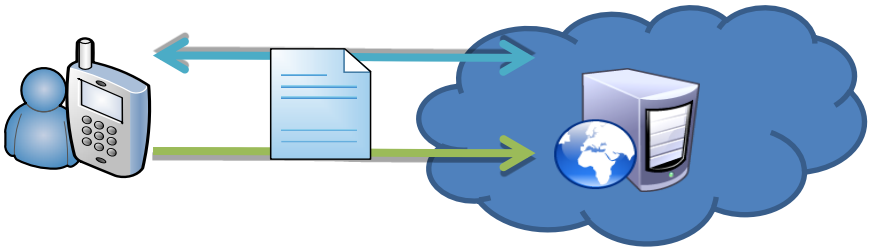


Ilustração 3.4- Sincronização por acesso directo ao dispositivo

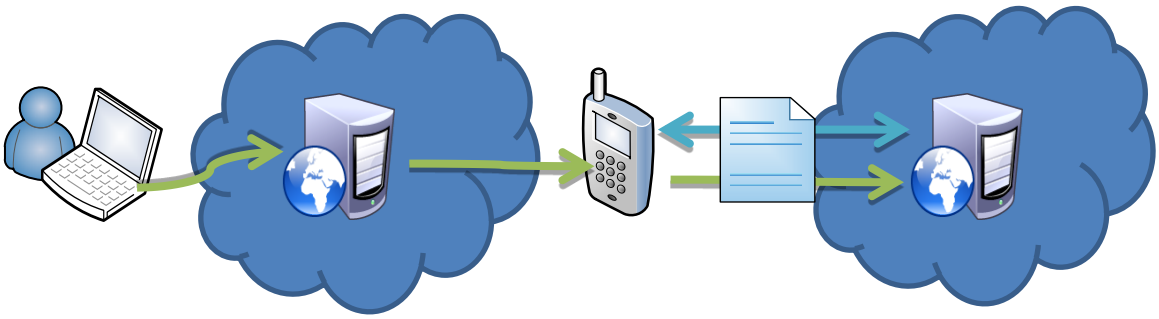


Ilustração 3.5- Sincronização por acesso remoto ao dispositivo

A análise da arquitectura e das suas especificidades ilustra um modelo de sincronização, onde o fluxo de informação faz-se, maioritariamente, no sentido Smartphone para Servidor Web. Apesar do modelo ser desenhado para aplicações que realizam o controlo de informação remotamente, pode ser aplicável a outras situações como sejam as aplicações de backup de

dados ou de partilha de conteúdos, alterando-se pontualmente o modelo de implementação.

A interacção do utilizador com a plataforma faz-se através da aplicação Web e da aplicação móvel. Para a monitorização dos dados é obrigatório o uso da aplicação Web, podendo o utilizador requisitar sincronizações ao dispositivo. Na arquitectura idealizada o utilizador assume diferentes tipos de acções:

- Configuração da aplicação móvel com a conta individual do servidor e definição do modo de envio e tipo de dados a enviar;
- Sincronização, remota ou local, com o servidor a pedido do utilizador;
- Monitorização da informação no Servidor Web;

A imagem seguinte (ilustração 3.6), representa o diagrama de casos de uso da plataforma.

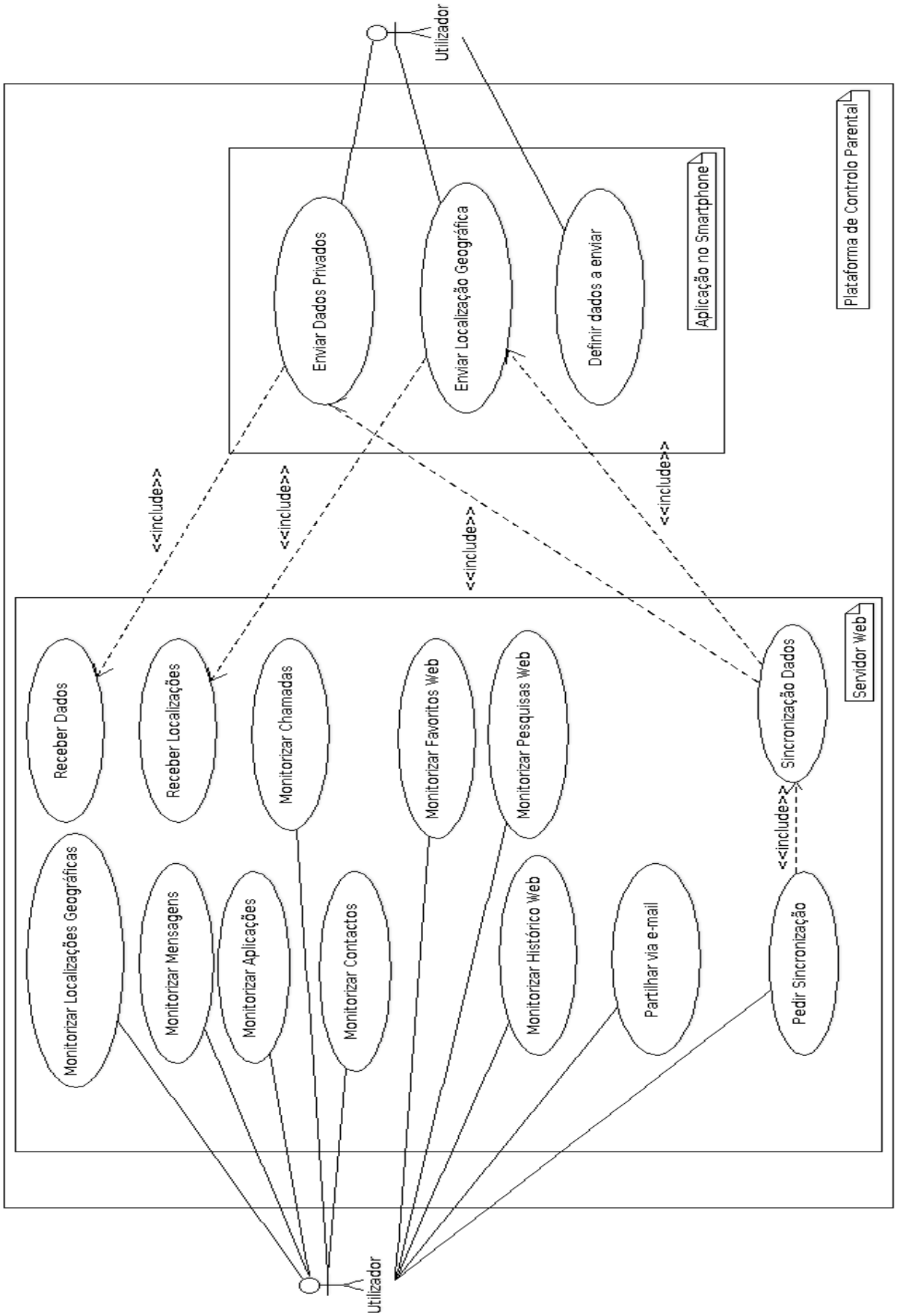


Ilustração 3.6- Diagrama de casos de uso da plataforma Android Parental Control

3.2. Implementação

A implementação da arquitectura proposta teve duas dimensões distintas. A primeira dimensão correspondeu ao desenvolvimento da aplicação móvel e da gestão das sincronizações com o Servidor Web. A segunda dimensão corresponde à implementação da aplicação Web e pela forma como esta implementa a interacção entre utilizador e dispositivo móvel. As imagens seguintes ilustram a plataforma nestas duas dimensões.

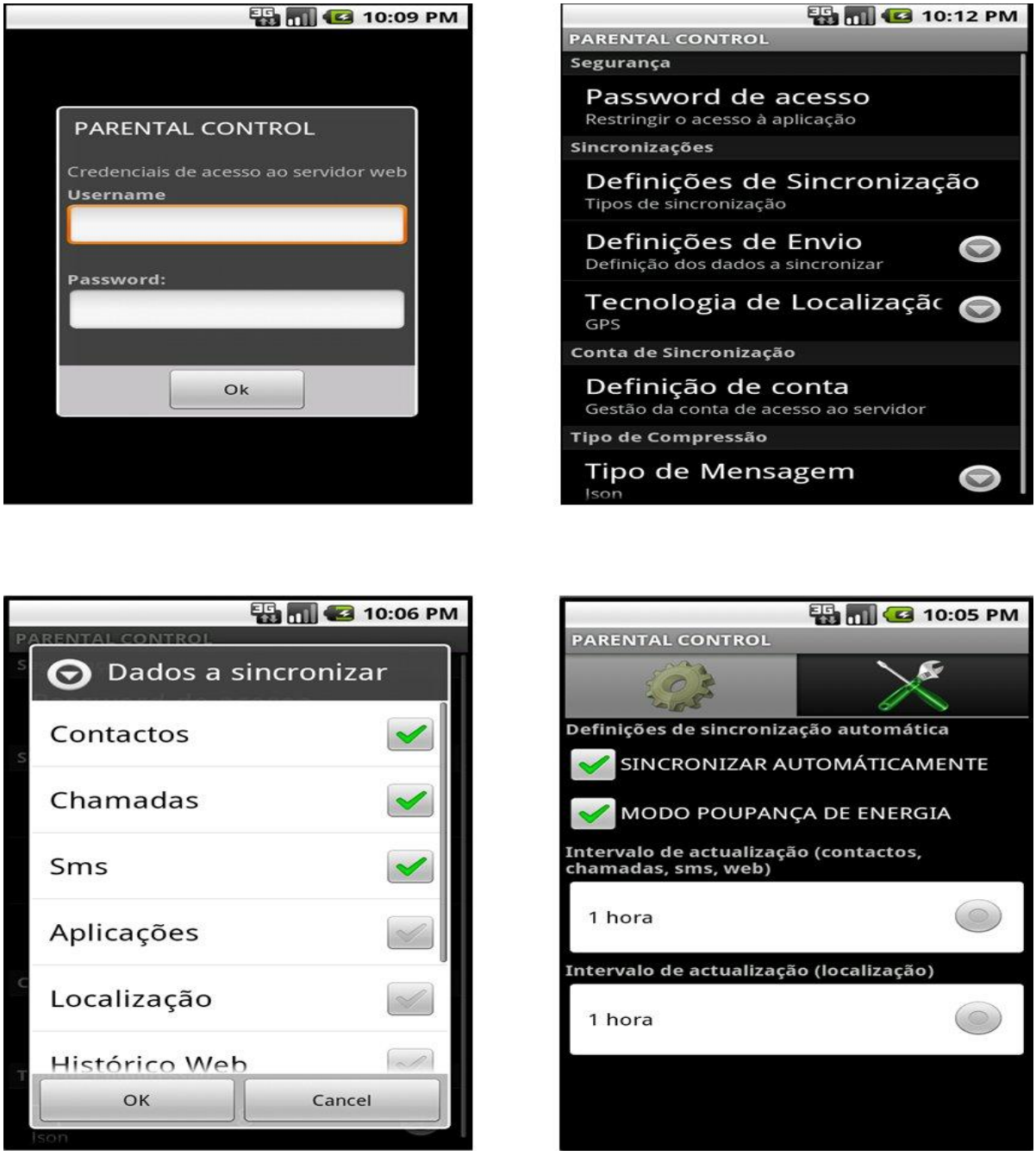


Ilustração 3.7- Imagens da aplicação móvel da plataforma

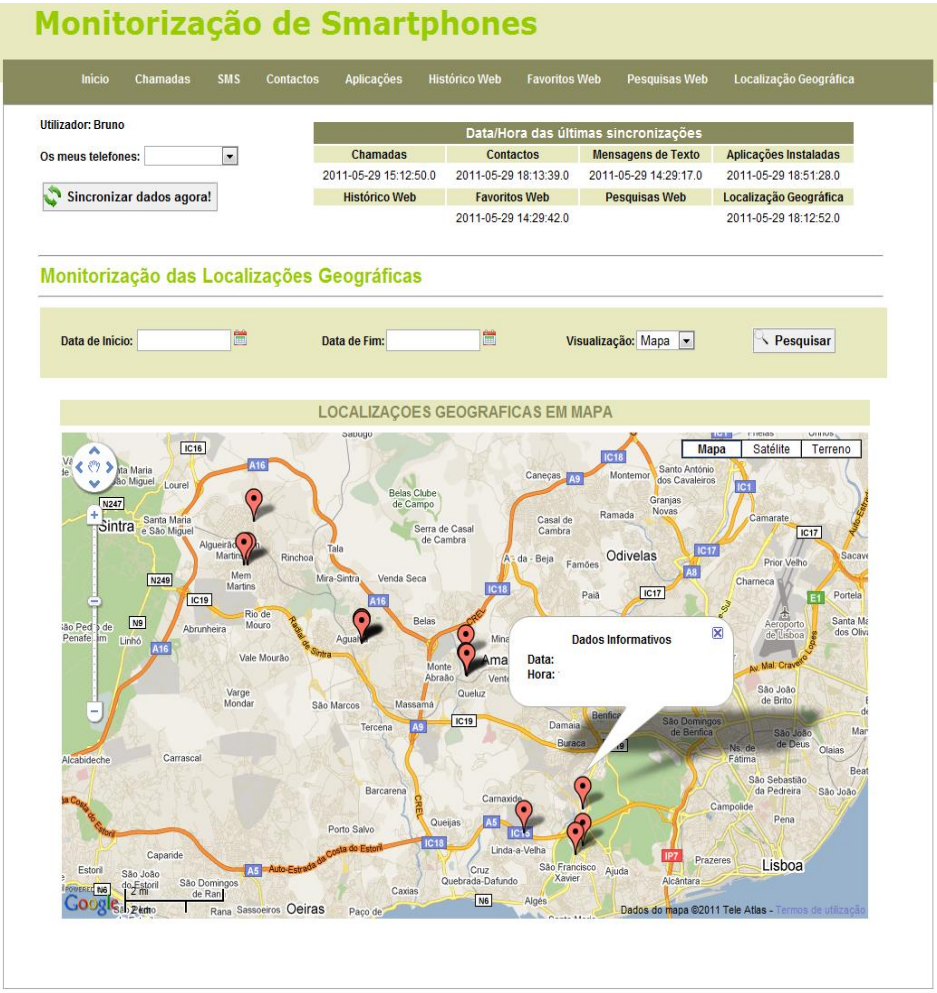


Ilustração 3.8 - Imagem da interface da aplicação Web da plataforma

A aplicação móvel foi desenhada para a plataforma Android e utiliza a *framework* de desenvolvimento SDK do Google. A sua função consiste em recolher os dados privados e as localizações geográficas e transferi-las para o Servidor Web. Para isso, é necessário associar cada dispositivo a uma conta no Servidor Web. Esta associação consiste no registo e autenticação da conta do Servidor Web, na aplicação móvel, e no envio do número de telefone e número de série do cartão SIM, para associar à conta no Servidor Web.

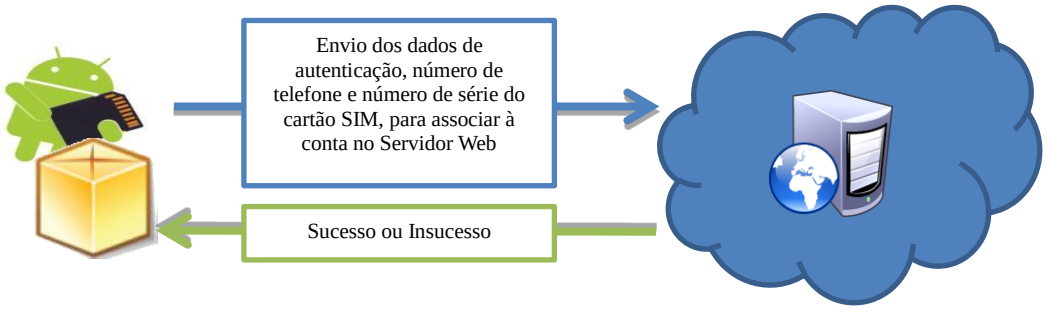


Ilustração 3.9- Processo de autenticação e configuração da aplicação móvel

Caso o número de telefone não esteja disponível é usado o número de série do cartão SIM, para associação à respectiva conta. Após o sucesso da autenticação, os dados da conta do servidor são guardados na base de dados interna da plataforma. Após este passo obrigatório, já é possível efectuar sincronizações, sendo que a aplicação recorre aos dados, relativos à conta do servidor, para autenticar cada sincronização.

Quando existe uma nova autenticação no servidor, a plataforma utiliza a arquitectura REST para a disponibilização/obtenção das datas das últimas sincronizações. Desta forma, mesmo que haja uma desinstalação da aplicação móvel, é garantido que não haja redundância de dados no servidor quando se voltar a instalar a aplicação.

Os processos de sincronização podem ser autónomos e periódicos ou por acção explícita dos utilizadores. Nos dois casos a aplicação responde a eventos emitidos ao sistema. Em casos onde o utilizador defina que a aplicação funcione de forma autónoma, com sincronizações pré-definidas, a plataforma regista dois tipos de eventos no alarme do sistema, um para os dados privados e outro para as localizações geográficas, permitindo a independência nas sincronizações com o Servidor Web. O alarme do sistema funciona como "despertador" da aplicação, poupando-se energia, visto não haver a necessidade de ter permanentemente a aplicação activa em background. A sua função é emitir *Intents Broadcasts* para o sistema com os eventos registados em períodos de tempo pré-definidos. A plataforma de Controlo Parental responde aos dois eventos, por si registados, com processos de sincronização. Se as sincronizações não são autónomas mas despoletadas por acção directa do utilizador, quer localmente ou de forma remota, é a própria aplicação que emite o *Intent Broadcast* para o sistema. As imagens seguintes ilustram a implementação das sincronizações na plataforma nestes dois processos.

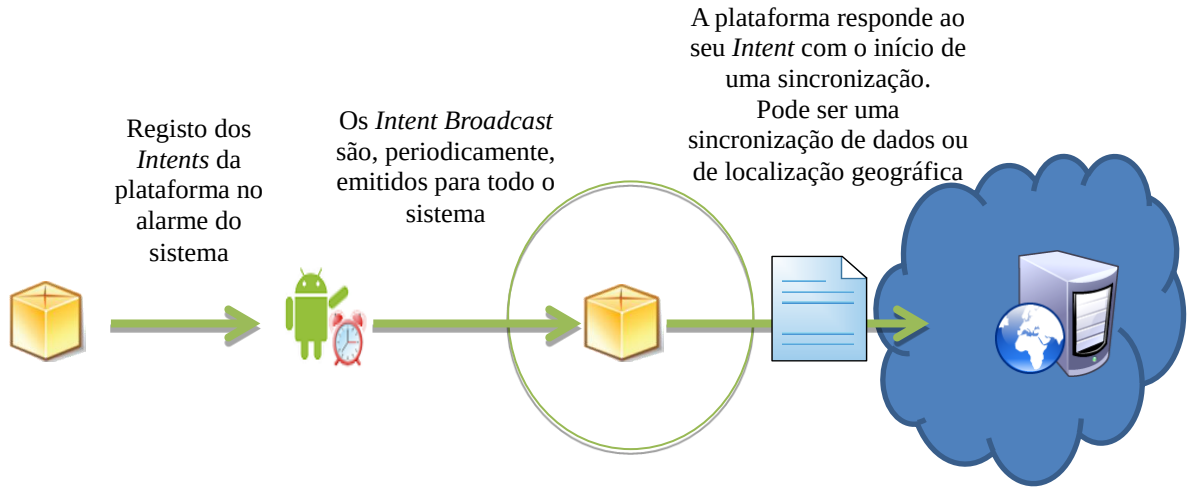


Ilustração 3.10- Sincronização autónoma e periódica da plataforma

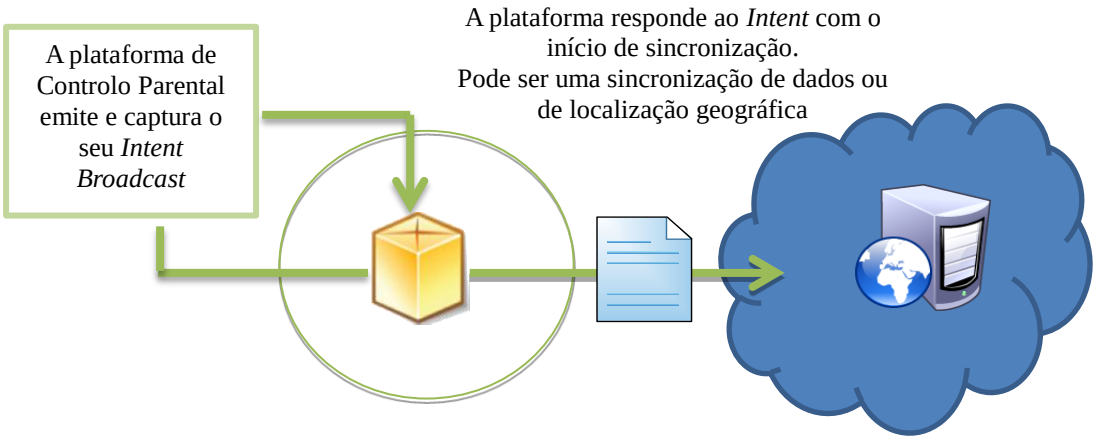


Ilustração 3.11- Sincronização por acção directa do utilizador

A grande vantagem da abordagem de sincronização periódica (ilustração 3.10) é a poupança de energia, visto que a aplicação mantem-se inactiva e só é inicializada para dar resposta aos eventos gerados no sistema. Estas definições são guardadas e registadas novamente a cada reinício do sistema operativo. Outra abordagem prevista na plataforma desenvolvida é o registo dos eventos de forma agregada. Nesta situação é o alarme do sistema que gere os tempos definidos para as sincronizações, agregando-as com outras aplicações que tenham definido períodos de sincronização parecidos. Desta forma, o sistema operativo requisita, apenas uma vez, os recursos da interface de rede activa, para sincronizar várias aplicações, poupando energia.

A transferência de dados é realizada sob um formato de intercâmbio de dados. Por razões de teste, a plataforma permite a escolha entre dois formatos de texto, XML e JSON e um formato binário, Protocol Buffers. Em qualquer das situações os dados são compactados antes do envio. A compactação dos dados é essencial, dadas as limitações de energia e de rádio dos Smartphones. A compactação e o envio dos dados são feitos de forma independente, isto é, as chamadas, os contactos ou as mensagens escritas são enviadas e comprimidas separadamente, no entanto, utilizam apenas uma ligação para o envio dos dados, reduzindo o gasto de energia, sobretudo nas redes 3G.

Nas localizações geográficas, os dados enviados, resumem-se a uma latitude, uma longitude e uma data e hora. Por este motivo os dados não são comprimidos e são enviadas através de JSON. A plataforma não permite o envio por agregação, das localizações geográficas, mas permite a recolha dos valores de latitude e longitude, por rede e por GPS.

A compactação dos dados enviados é feita “on the fly” e não são guardados quaisquer ficheiros. A tecnologia de compactação é o Gzip que é suportado por Servidores Web e

dispositivos Android. Os dados são descompactados, e processados no servidor e associados a respectiva conta, na base de dados da plataforma (Anexo A). Após o sucesso de processamento no lado do servidor, a aplicação móvel guarda informações sobre o último registo transferido, permitindo que na sincronização seguinte sejam transferidos apenas os novos dados.

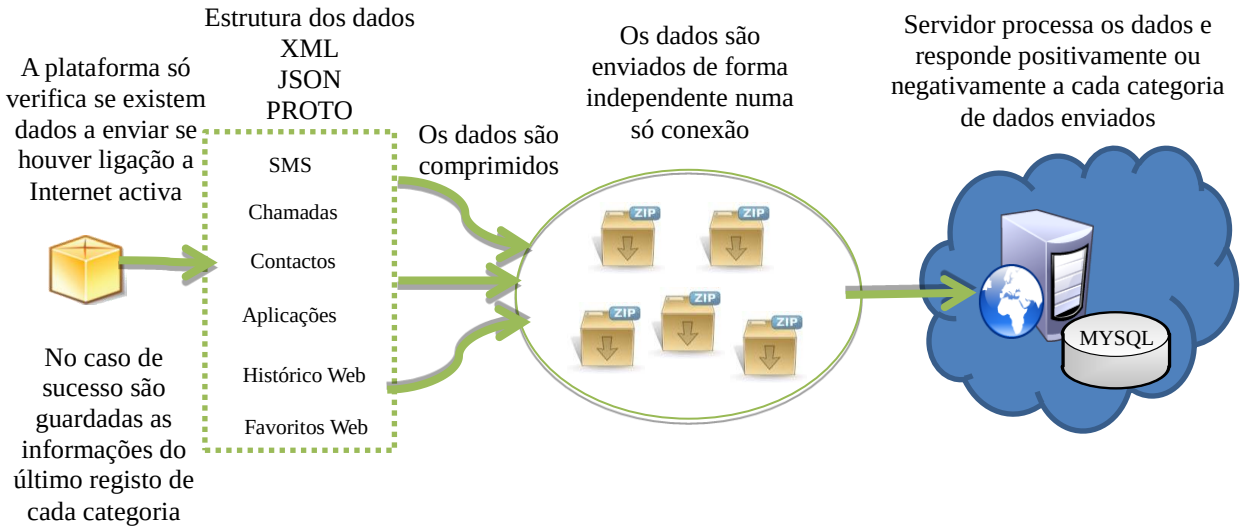


Ilustração 3.12- Processo de implementação da sincronização de dados (Detalhado)

Os processos de sincronização apenas são iniciados se houver confirmação do estado geral da bateria, superior a 10%, e se houver alguma ligação presente à Internet. A análise prévia ao estado do sistema é essencial na gestão da bateria, visto que só existe processamento e respectivas consultas à base de dados do sistema se houver condições para a transferência dos dados para o Servidor Web.

As sincronizações remotas, através da plataforma Web, são implementadas utilizando o sistema tradicional de mensagens escritas. A utilização de um sistema estável e credível de comunicação com o dispositivo, independente do ambiente a que este se encontre é uma garantia para o bom funcionamento da plataforma. Esta implementação resulta da interacção entre da aplicação móvel com a aplicação Web. A aplicação móvel analisa a origem de todas as mensagens escritas recebidas e caso a origem seja a do seu próprio número de telefone, analisa o seu conteúdo. Por questões de segurança as mensagens enviadas são encriptadas de forma simétrica partilhando uma chave secreta. Esta chave secreta corresponde à password de acesso ao servidor da conta de utilizador. Para garantir a autenticidade das mensagens, além da sua encriptação é adicionado o número de série do cartão SIM do equipamento móvel no conteúdo das mensagens, aumentando-se a segurança. O algoritmo de encriptação é o 3DES

(Triple Data Encryption Standard). O utilizador pede uma sincronização através do envio de um pedido, pela plataforma Web, que consiste numa mensagem encriptada com um código específico. A acção a desenvolver é definida no dispositivo móvel. A imagem seguinte ilustra a metodologia *push* da plataforma.

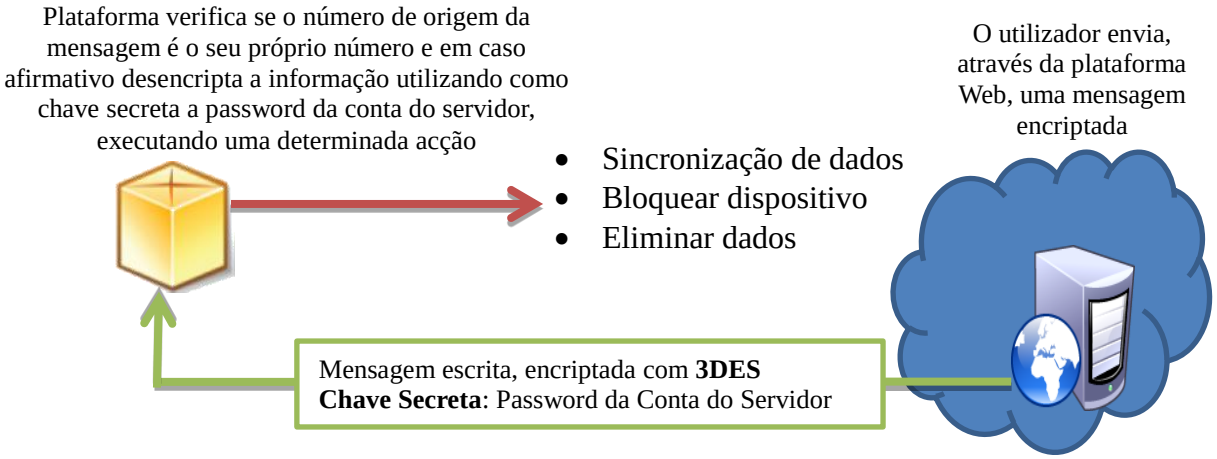


Ilustração 3.13- Representação da metodologia push da plataforma

A aplicação activa de gestão mensagens escritas do sistema não receberá as mensagens destinadas à plataforma de Controlo Parental. Esta abordagem permite a recepção e processamento de mensagens de forma silenciosa e segura.

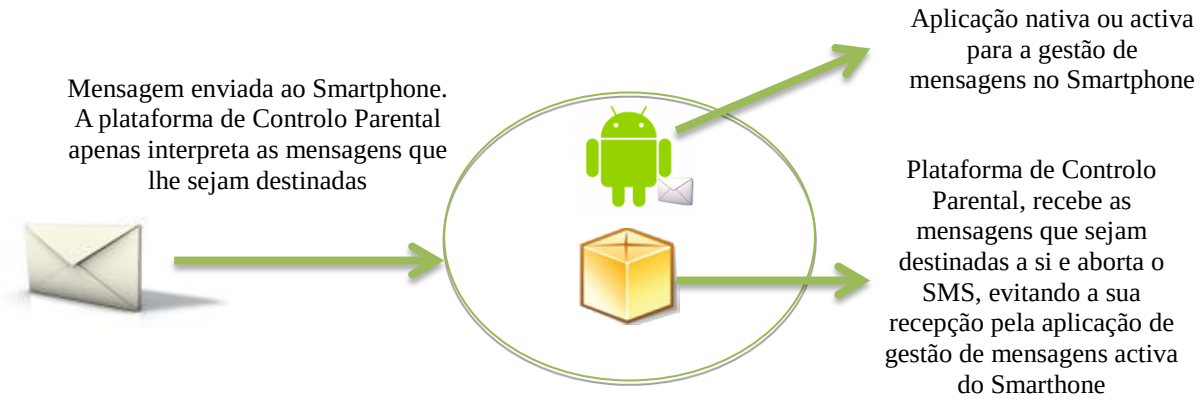


Ilustração 3.14- Sistema de recepção de mensagens escritas da plataforma

A plataforma Web permite a monitorização da informação e a gestão de sincronizações remotas e implementa um conjunto de funcionalidades ao utilizador. Todas as informações sincronizadas podem ser partilhadas através de correio electrónico ou impressas.

3.3. Testes

3.3.1. Introdução

A elaboração de testes a uma plataforma deste tipo, onde existem transferências de dados para a Web, envolve algum grau de incerteza, na medida em que existem factores alheios à própria plataforma que podem influenciar os resultados. A plataforma foi testada em várias abordagens, com diferentes quantidades de dados e em diferentes interfaces de rede. Os pontos centrais da análise foram:

- Tempos médios obtidos na recolha e compressão dos dados, no dispositivo móvel, nos diversos formatos estudados: XML, JSON e Protocol Buffers;
- Tempos médios obtidos na análise sintáctica, no Servidor Web, e respectiva inserção de dados no SGBD, nos diversos formatos estudados: XML, JSON e Protocol Buffers. Teste efectuado para dados compactados (Gzip) e não compactados;
- Tempos médios obtidos na sincronização de dados com o Servidor Web, nos diversos formatos e nas interfaces de rede Wi-Fi e 3G, com compactação de dados (Gzip) e sem compactação de dados;
- Gastos de energia com CPU nas diferentes transferências com os diversos formatos com compactação de dados (Gzip) e sem compactação de dados e nas duas interfaces de rede analisadas;
- Gastos de energia na interface Wi-Fi nas diferentes transferências, com os diversos formatos estudados, com compactação de dados (Gzip) e sem compactação de dados;
- Gastos de energia na interface 3G nas diferentes transferências, com os diversos formatos estudados, com compactação de dados (Gzip) e sem compactação de dados;
- Tamanho dos diferentes dados com e sem compactação nos diferentes formatos estudados;

Os testes incidiram sobre dois volumes diferentes de dados. Foi seleccionado um conjunto de 650 registos de mensagens escritas (SMS) e um conjunto de: 107 registos de informações de contactos pessoais, 650 registos de mensagens escritas (SMS), 133 registos de informações sobre aplicações, 500 registos de chamadas telefónicas, 104 registos de informações relativas

ao histórico Web e 18 registos relativos a informações dos favoritos Web.

Ao nível de hardware o ambiente de teste foi composto por um Smartphone HTC Hero, equipado com um processador Qualcomm® MSM7200A™, 528 MHz, com uma ROM de 512 MB e com uma RAM de 288 MB. O Servidor Web foi composto por um Inter(R) Core(TM)2 Duo CPU T7500 @ 2.20GHz 2.20GHz, com 3GB de RAM.

Ao nível de software o dispositivo móvel estava equipado com a plataforma Android 2.1, corria a Plataforma de Controlo Parental e utilizava uma aplicação externa, designada de PowerTutor[3], de monitorização dos gastos de energia, por aplicação, ao nível da CPU, Wi-Fi e 3G. O Servidor Web estava equipado com o sistema operativo Ubuntu 10.04, com o Servidor Web Tomcat 6 e com o SGBD Mysql 5.1.

A descrição de todos os tempos e valores obtidos encontram-se na secção de anexos (Anexo B) .

3.3.2. Tempos médios de selecção e compressão dos dados

O desempenho de cada um dos formatos é um dado importante para a construção de sistemas mais eficientes, com melhores desempenhos. O processo de teste consistiu em medir o tempo, em milissegundos, decorrente entre o início da selecção dos dados na base de dados do equipamento móvel e o final da compressão dos dados segundo os vários formatos. Neste caso os testes foram efectuados sobre 651 registos de mensagens escritas (SMS) durante 30 vezes para cada um dos formatos, obtendo-se os seguintes dados:

TEMPOS MÉDIOS DE COMPRESSÃO EM MILISSEGUNDOS (651 SMS)			
FORMATO	**XML**	**JSON**	**PROTO**
Média	2001,77	2084,93	1751,40
Desvio Padrão	166,41	165,32	133,00
Desvio Médio	122,74	110,96	90,41
Mediana	1953,5	2029,5	1740,5
Melhor Tempo	1810	1930	1573
Pior Tempo	2543	2622	2239

Tabela 3.1- Tempos médios de selecção e compressão no dispositivo móvel

[3] PowerTutor foi desenvolvido pela Universidade do Michigan. O trabalho foi suportado em parte pelo Google e pela National Science Foundation (under award CNS-0720691).

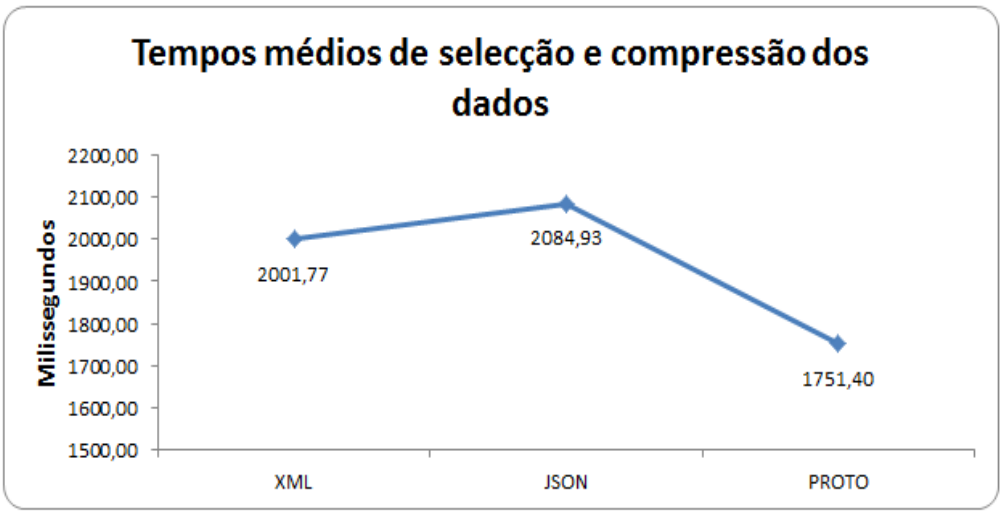
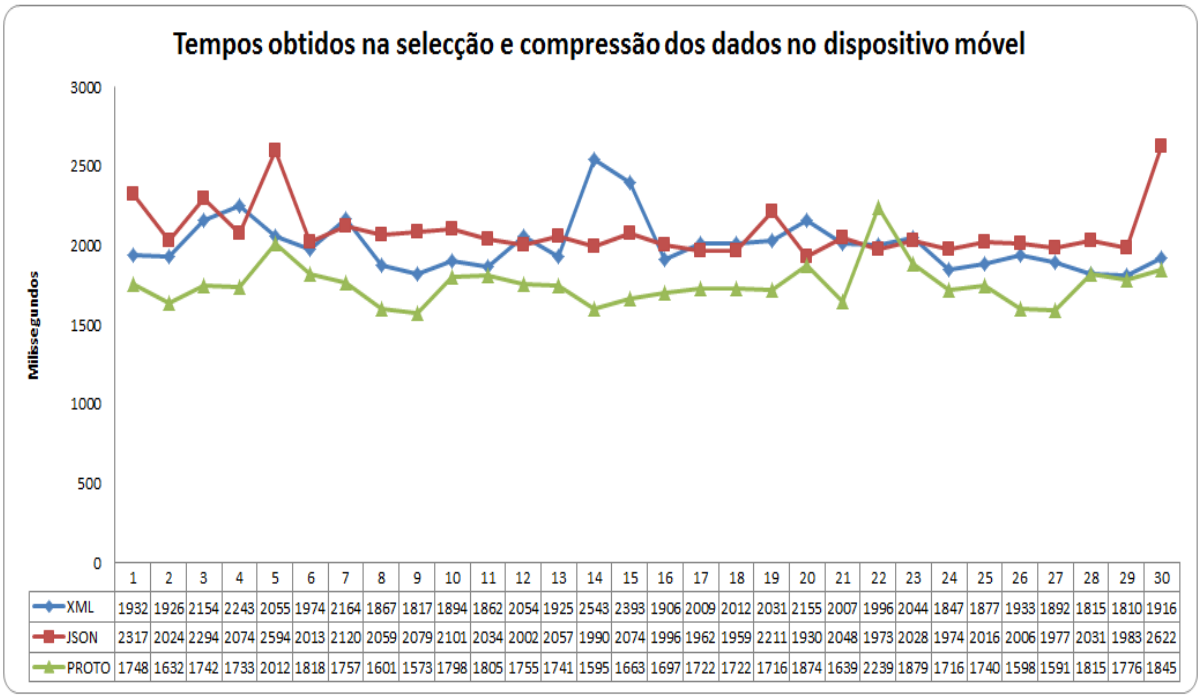


Ilustração 3.15- Tempos médios de selecção e compressão de dados no dispositivo móvel



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
XML	1932	1926	2154	2243	2055	1974	2164	1867	1817	1894	1862	2054	1925	2543	2393	1906	2009	2012	2031	2155	2007	1996	2044	1847	1877	1933	1892	1815	1810	1916
JSON	2317	2024	2294	2074	2594	2013	2120	2059	2079	2101	2034	2002	2057	1990	2074	1996	1962	1959	2211	1930	2048	1973	2028	1974	2016	2006	1977	2031	1983	2622
PROTO	1748	1632	1742	1733	2012	1818	1757	1601	1573	1798	1805	1755	1741	1595	1663	1697	1722	1722	1716	1874	1639	2239	1879	1716	1740	1598	1591	1815	1776	1845

Ilustração 3.16- Valores obtidos na selecção e compressão dos dados nos vários formatos

O formato binário Protocolo Buffers apresentou os melhores tempos médios de desempenho, com 1751,40 milissegundos, e o menor desvio padrão, com 133. Entre os dois formatos de texto os resultados não apresentaram grandes discrepâncias, havendo no entanto um ligeiro melhor desempenho do formato XML, com 2001,77 milissegundos de tempo médio e um desvio padrão de 166,41. O formato JSON obteve uma média de tempos de 2084,93 milissegundos com um desvio padrão de 165,32. No formato XML, 50% dos valores obtidos encontram-se abaixo dos 1953,5 milissegundos, no formato JSON, 50% dos dados obtidos encontram-se abaixo dos 2029,5 milissegundos e no formato Protocol Buffers, 50% dos dados

obtidos encontram-se abaixo dos 1740,5 milissegundos.

3.3.3. Tempos médios de análise sintáctica dos dados no Servidor

A rapidez de compressão dos dados é um factor importante para o desempenho e tem um impacto positivo na gestão de energia, no entanto, como existe uma interacção entre dispositivo móvel e Servidor Web é importante perceber os tempos de análise sintáctica de cada um dos tipos de formatos de dados presentes nas sincronizações.

De seguida são apresentados os resultados obtidos, nos testes efectuados, com um lote de 650 mensagens escritas com compactação de dados e sem compactação de dados, enviados repetidamente durante 30 vezes e registando-se os tempos médios obtidos de análise sintáctica e inserção de dados no Servidor Web.

TEMPOS MÉDIOS, EM MILISSEGUNDOS, DE ANÁLISE SINTÁCTICA NO SERVIDOR COM COMPACTAÇÃO DE DADOS (650 SMS)			
Formato	**XML**	**JSON**	**PROTO**
Média	284,23	175,87	218,90
Desvio Padrão	70,45	46,80	54,47
Desvio Médio	57,23	39,20	43,59
Melhor Tempo	195,00	137,00	127,00
Pior Tempo	488,00	277,00	394,00

Tabela 3.2- Estatísticas apuradas na análise sintáctica dos dados compactados

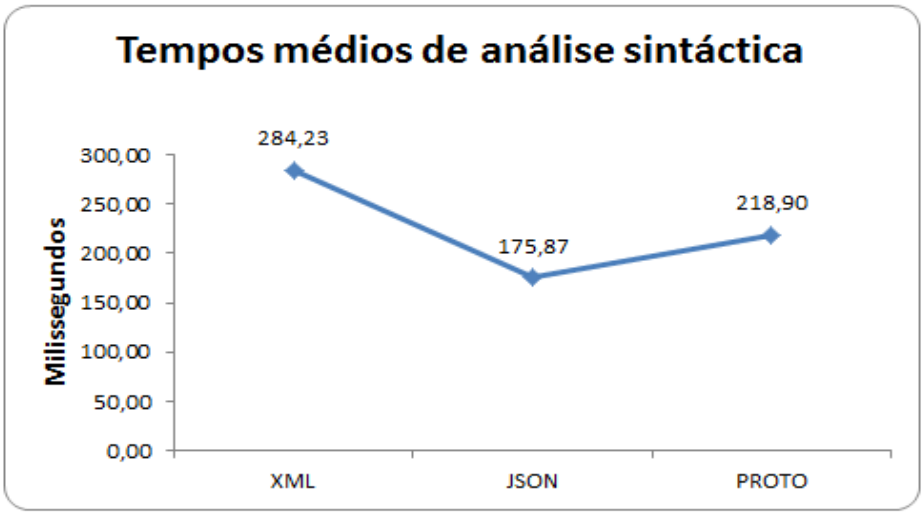


Ilustração 3.17- Tempos médios obtidos na análise sintáctica dos dados compactados

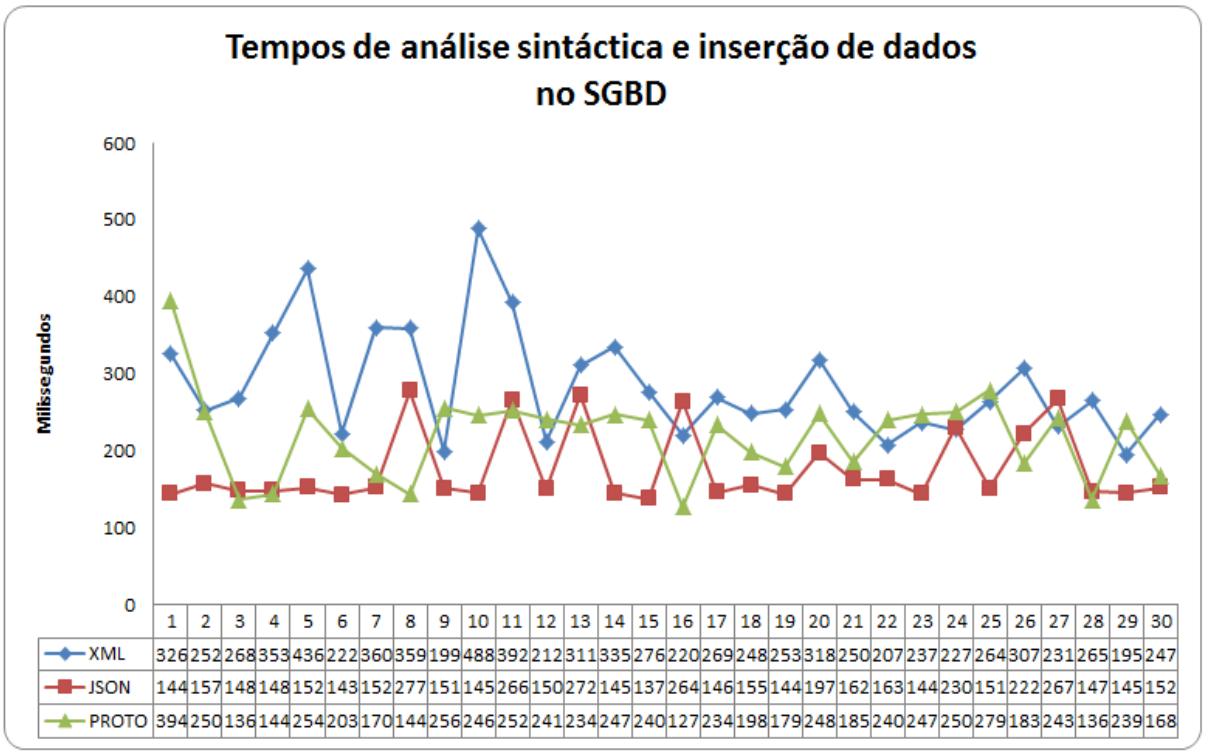


Ilustração 3.18- Tempos obtidos na análise sintáctica de dados compactados

Da análise aos resultados, o formato de texto JSON e o formato binário Protocol Buffers obtiveram a melhor performance com uma ligeira vantagem para o formato JSON, com a melhor média de tempo e o menor desvio padrão, apesar do melhor tempo registado ser do Protocol Buffer. O formato XML revelou ser mais pesado a nível de análise sintáctica com um valor médio de 284,23 milissegundos.

TEMPOS MÉDIOS, EM MILISSEGUNDOS, DE ANÁLISE SINTÁCTICA NO SERVIDOR SEM COMPACTAÇÃO DE DADOS (650 SMS)			
FORMATO	**XML**	**JSON**	**PROTO**
Média	261,57	169,57	205,00
Desvio Padrão	65,75	42,38	57,34
Desvio Médio	47,88	30,64	47,87
Melhor Tempo	161,00	133,00	125,00
Pior Tempo	481,00	271,00	377,00

Tabela 3.3- Estatísticas apuradas na análise sintáctica dos dados não compactados

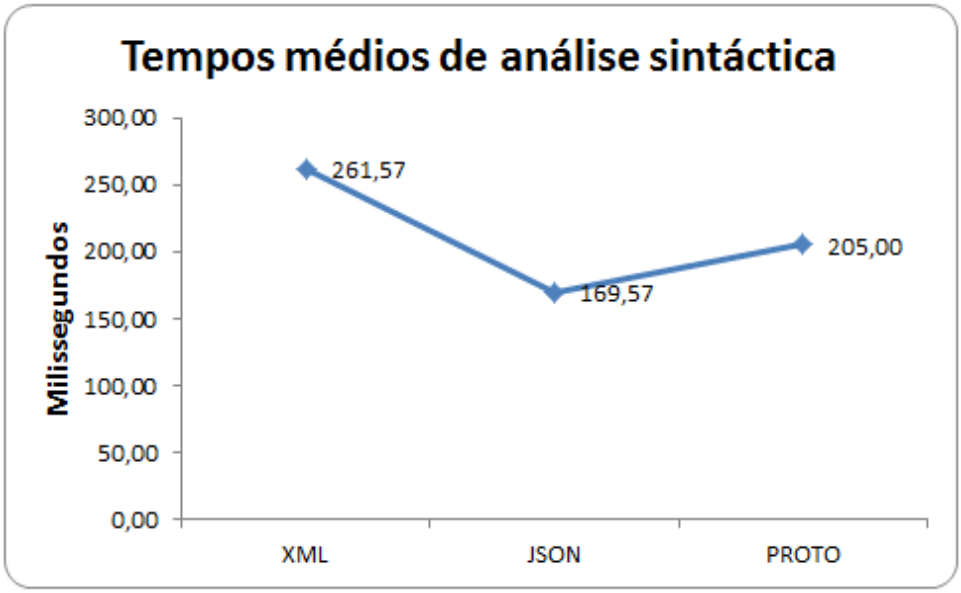
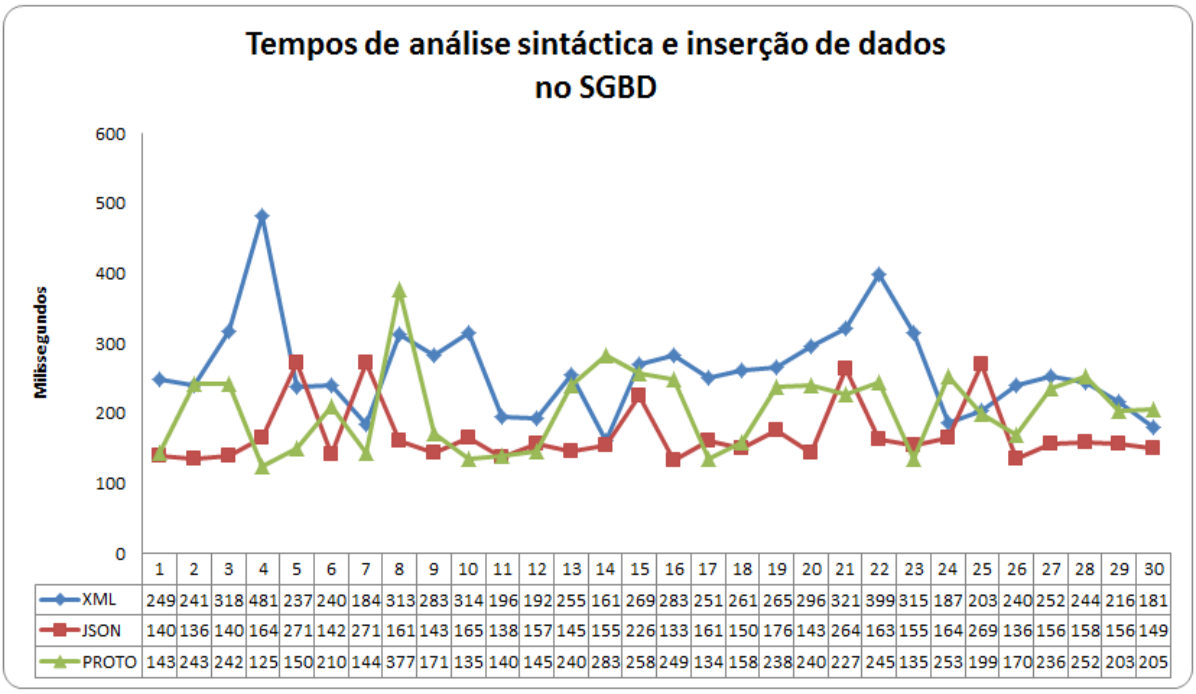


Ilustração 3.19- Tempos médios obtidos na análise sintáctica dos dados não compactados



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
XML	249	241	318	481	237	240	184	313	283	314	196	192	255	161	269	283	251	261	265	296	321	399	315	187	203	240	252	244	216	181
JSON	140	136	140	164	271	142	271	161	143	165	138	157	145	155	226	133	161	150	176	143	264	163	155	164	269	136	156	158	156	149
PROTO	143	243	242	125	150	210	144	377	171	135	140	145	240	283	258	249	134	158	238	240	227	245	135	253	199	170	236	252	203	205

Ilustração 3.20- Tempos obtidos na análise sintáctica de dados não compactados

Na análise sintáctica sem compressão de dados, existe uma ligeira melhoria de performance, no entanto, não é significativa dada a quantidade de dados reduzida. Não existe alteração na ordenação dos formatos pela sua performance, mantendo-se o formato JSON como o mais eficiente com um tempo médio de 169,57 milissegundos, seguido do formato Protocol Buffers com um tempo médio de 205 milissegundos e o formato XML com um tempo médio de 261,57 milissegundos.

3.3.4. Tempos médios de sincronização com o Servidor Web

Na avaliação dos tempos médios de resposta desde o envio da informação do dispositivo móvel até a obtenção de uma resposta do Servidor Web, sendo que esta resposta só é dada depois de processada a informação, foram utilizados duas quantidades diferentes de dados, um lote de 650 objectos, representando um conjunto de mensagens escritas (SMS), com e sem compactação e um lote de 1512 objectos, representado a agregação de registos de informações de contactos pessoais, registos de mensagens escritas (SMS), registos de informações sobre aplicações, registos de chamadas telefónicas, registos de informações relativas ao histórico Web e registos relativos a informações dos favoritos Web. Por uma questão prática, no decorrer do resto do documento os dados são catalogados em dois tipos de lote, descritos na tabela seguinte:

DESCRIÇÃO DA INFORMAÇÃO DE CADA LOTE DE DADOS	
Designação	**Descrição**
LOTE 1 (650 objectos)	• Mensagens escritas (SMS)
LOTE 2 (1512 objectos)	• Registos de informações de contactos pessoais • Registos de mensagens escritas (SMS) • Registos de informações sobre aplicações • Registos de chamadas telefónicas • Registos de informações relativas ao histórico Web • Registos relativos a informações dos favoritos Web

Tabela 3.4- Descrição dos dados de teste

Além dos testes aos dados com compactação e sem compactação, as análises aos tempos obtidos foram efectuados nas duas interfaces de rede mais comuns: 3G e Wi-Fi. Os tempos obtidos nestes testes resultam do cálculo da média dos tempos obtidos em 30 sincronizações, utilizando os mesmos dados nos diferentes formatos. Para uma melhor compreensão dos resultados obtidos é necessário perceber a quantidade, em bytes, dos dados transferidos nos diversos formatos, com e sem compactação.

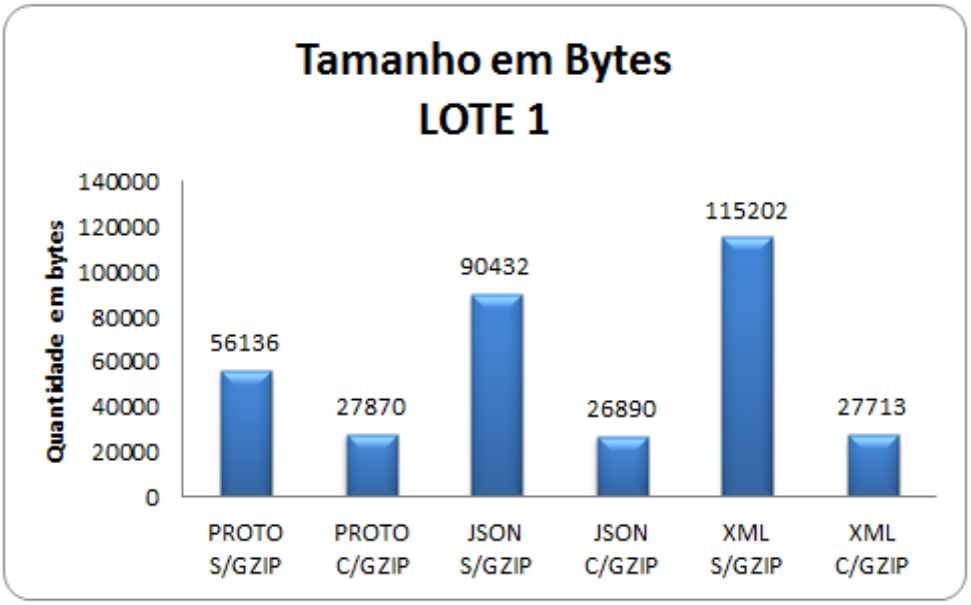


Ilustração 3.21- Tamanho, em bytes, dos dados de teste do LOTE 1

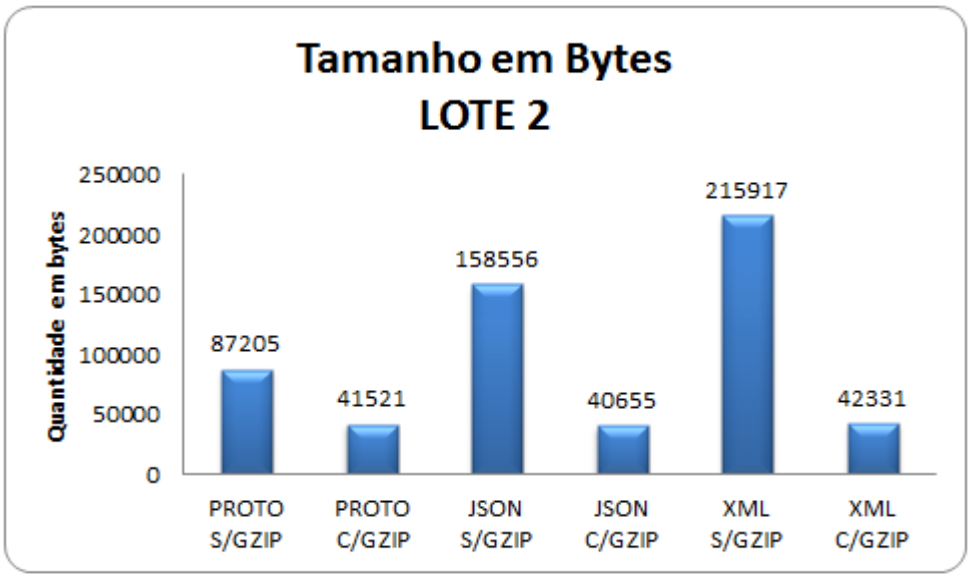


Ilustração 3.22- Tamanho, em bytes, dos dados de teste do LOTE 2

Os dados revelam que o formato Protocol Buffers é muito mais eficiente que o formato JSON e que o formato XML, no que respeita ao tamanho, em bytes, da informação sincronizada. O formato binário apresenta perto de metade do tamanho do formato JSON e uma discrepância ainda mais acentuada em relação ao formato XML. A diferença é um pouco mais acentuada com maiores quantidades de dados, como se pode verificar na ilustração 3.22. A vantagem do formato binário é invertida com a compactação dos dados, havendo uma redução acentuada, cerca de 66% do tamanho inicial, nos formatos de texto e apenas uma redução de 50% no formato binário, Protocol Buffers.

Tempos médios de sincronização

TEMPOS MÉDIOS DE SINCRONIZAÇÃO COM COMPACTAÇÃO LOTE 1 - VIA WI-FI			
FORMATO	**XML**	**JSON**	**PROTO**
Média	660,93	573,10	581,03
Desvio Padrão	81,51	85,59	129,68
TEMPOS MÉDIOS DE SINCRONIZAÇÃO SEM COMPACTAÇÃO LOTE 1 – VIA WI-FI			
Média	783,00	633,90	582,93
Desvio Padrão	74,60	74,38	62,65
TEMPOS MÉDIOS DE SINCRONIZAÇÃO COM COMPACTAÇÃO LOTE 1 – VIA 3G			
Média	1408,30	1321,03	1434,47
Desvio Padrão	172,95	86,91	245,56
TEMPOS MÉDIOS DE SINCRONIZAÇÃO SEM COMPACTAÇÃO LOTE 1 – VIA 3G			
Média	1921,93	1610,43	1551,43
Desvio Padrão	223,85	130,15	291,01

Tabela 3.5- Tempos médios e desvios padrões das sincronizações dos dados do LOTE 1

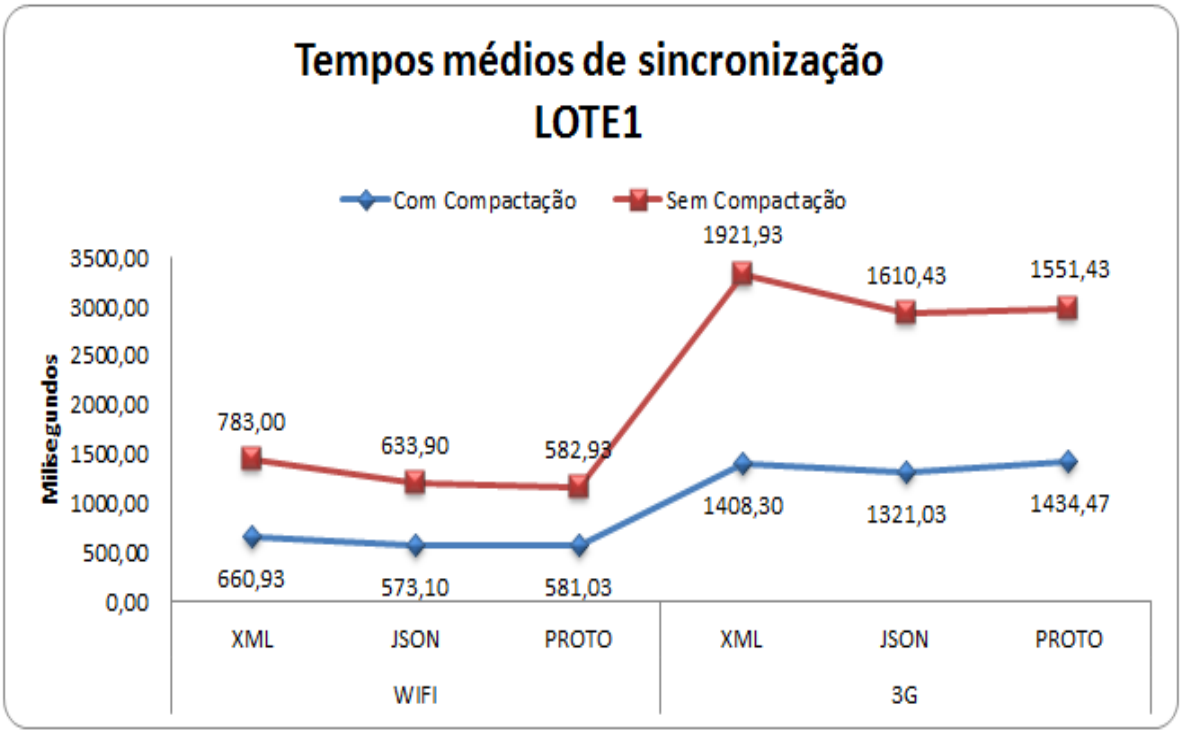


Ilustração 3.23- Tempos médios de sincronização dos dados do LOTE 1

Numa interface de rede rápida o tamanho dos dados não tem um impacto tão evidente na performance das sincronizações, no entanto, podemos verificar que as maiores diferenças são detectadas nos formatos de texto, visto que a compactação reduz em cerca de 66% o tamanho

da informação a sincronizar, não sendo esta diferença tão evidente no formato Protocol Buffers, onde os tempos médios obtidos diferem muito pouco apesar da compactação reduzir em cerca de 50% o tamanho dos dados.

Numa interface mais lenta, como 3G, o tamanho da informação transferida tem um impacto maior nos tempos médios de sincronização, sendo a compactação dos dados extremamente vantajosa nos formatos de texto, com acentuadas melhorias de performance na sincronização de dados, não se revelando o mesmo para o formato binário, em parte explicado pela menor performance obtida com a compactação dos dados.

A vantagem de se usar a compactação de dados resulta da análise entre o custo de processamento para a compactação dos dados relativamente aos benefícios obtidos na sincronização com o envio dos dados compactados. Para isso forem efectuados os mesmos testes com um lote diferente de dados, com mais informação e consequentemente com maior tamanho. Os valores, em bytes, deste lote de dados estão descritos na ilustração 3.22. Os resultados obtidos, representam os tempos de sincronização com o servidor, incluindo a compactação, a descompactação e análise sintáctica dos dados.

TEMPOS MÉDIOS DE SINCRONIZAÇÃO COM COMPACTAÇÃO LOTE 2 - VIA WI-FI			
FORMATO	**XML**	**JSON**	**PROTO**
Média	2714,43	2576,87	2749,30
Desvio Padrão	201,80	346,76	303,48
TEMPOS MÉDIOS DE SINCRONIZAÇÃO SEM COMPACTAÇÃO LOTE 2 – VIA WI-FI			
Média	3047,00	2846,67	2629,50
Desvio Padrão	182,41	293,25	194,56
TEMPOS MÉDIOS DE SINCRONIZAÇÃO COM COMPACTAÇÃO LOTE 2 – VIA 3G			
Média	5352,83	4744,57	5192,50
Desvio Padrão	438,68	497,64	515,27
TEMPOS MÉDIOS DE SINCRONIZAÇÃO SEM COMPACTAÇÃO LOTE 2 – VIA 3G			
Média	7450,47	6640,97	6020,67
Desvio Padrão	732,60	734,74	695,67

Tabela 3.6- Tempos médios e desvios padrões das sincronizações dos dados do LOTE 2

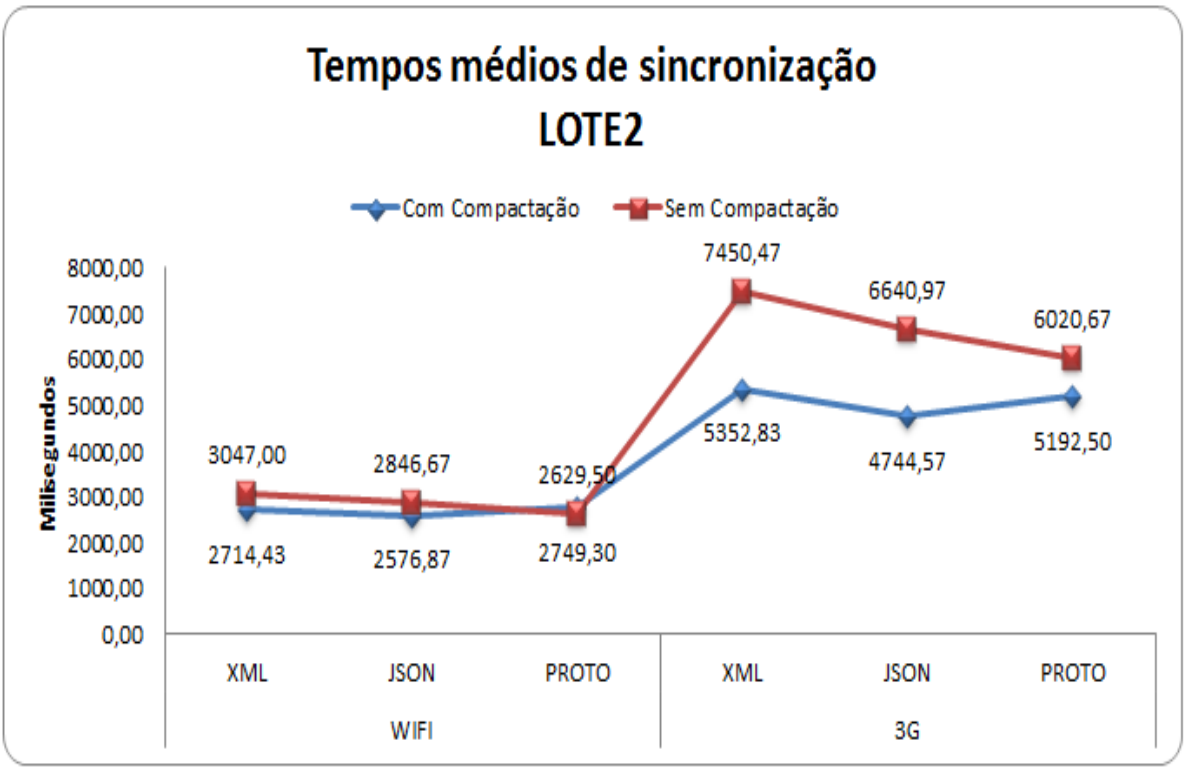


Ilustração 3.24- Tempos médios de sincronização dos dados do LOTE 2

Mesmo com um volume de dados maior, o uso da compactação de dados aumenta a performance de sincronização, à excepção do formato Protocolo Buffers na interface de rede Wi-Fi que obteve um desempenho ligeiramente inferior à sincronização de dados sem compactação. O formato Protocol Buffers é o que apresenta menos discrepâncias entre a compactação ou não compactação dos dados, especialmente em interfaces de redes mais rápidas, revelando um custo superior, ao nível da performance, quando se opta por compactar os dados. Para a interface de rede 3G, a relação entre tamanho e performance é maximizada, sendo vantajoso compactar os dados em todos os formatos, visto que existe uma melhoria significativa no desempenho de sincronização.

3.3.5. Gastos de energia – CPU e REDE

A gestão de energia num equipamento móvel é dos factores mais determinantes no desenvolvimento de aplicações. Para uma tomada de decisão do formato de dados mais adequado para a aplicação é fundamental perceber qual dos vários formatos apresenta melhor performance na gestão de energia. Os testes efectuados consistiram em medir, para as diferentes sincronizações realizadas, os gastos energéticos, em joules, com o trabalho de CPU, nas interfaces de redes Wi-Fi e 3G. Foi utilizada uma aplicação externa à plataforma,

designada de PowerTutor, para monitorizar os gastos de energia. Esta aplicação foi desenvolvida pela Universidade do Michigan com apoio do Google.

Nos casos de teste sobre dados sincronizados, o processo de compactação é contabilizado. Estes testes foram efectuados uma única vez para cada tipo de sincronização em cada uma das interfaces de rede.

Para as sincronizações dos dados do LOTE 1, foram obtidos os seguintes resultados, por interface de rede, com e sem compactação de dados:

GASTOS DE ENERGIA DE CPU (JOULES) LOTE 1		
	COM COMPACTAÇÃO	**SEM COMPACTAÇÃO**
XML	6,1	5,3
JSON	6,1	4,6
PROTO	5,4	5,1

Tabela 3.7- Gastos de energia, em joules, com o processamento dos dados do LOTE 1

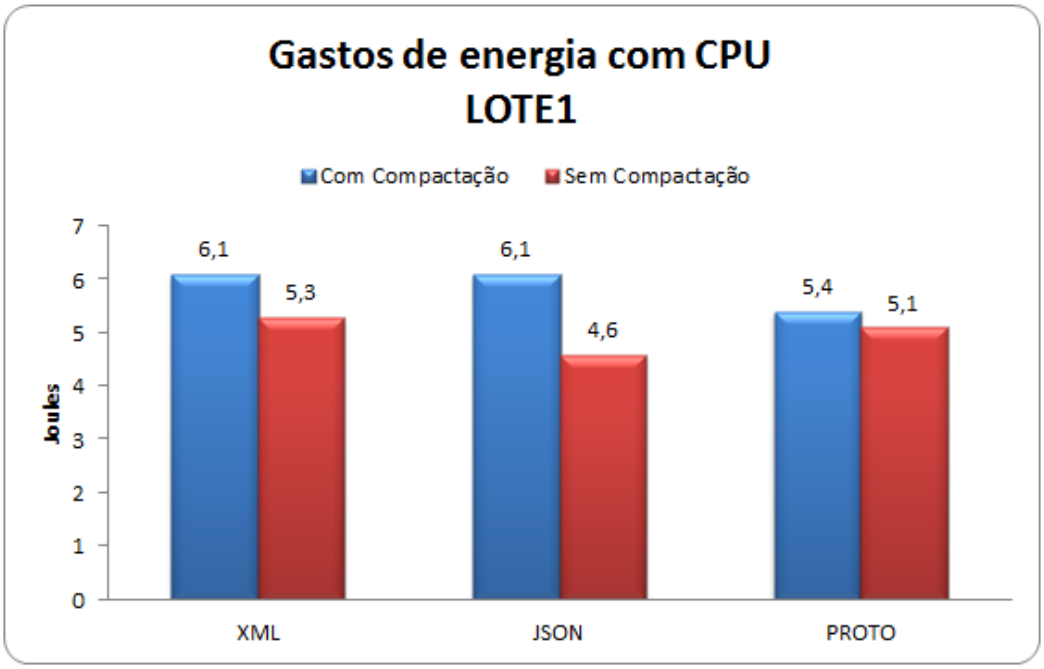


Ilustração 3.25- Gastos de energia com o processamento dos dados do LOTE 1

A nível de gastos de energia com CPU, o que podemos verificar é que existe um ganho de energia em todos os formatos de dados estudados quando não há compactação dos dados, no entanto, é necessário avaliar se este ganho de energia não é perdido no processo de sincronização.

GASTOS DE ENERGIA POR INTERFACE DE REDE (JOULES) LOTE 1			
INTERFACE DE REDE	**FORMATO**	**COM COMPACTAÇÃO**	**SEM COMPACTAÇÃO**
WI-FI	**XML**	10,8	14,7
	JSON	9,4	13,8
	PROTO	10,9	10,2
3G	**XML**	25,1	34,7
	JSON	25,0	29,6
	PROTO	28,6	28,5

Tabela 3.8- Gastos de energia, em joules, por interface de rede (LOTE 1)

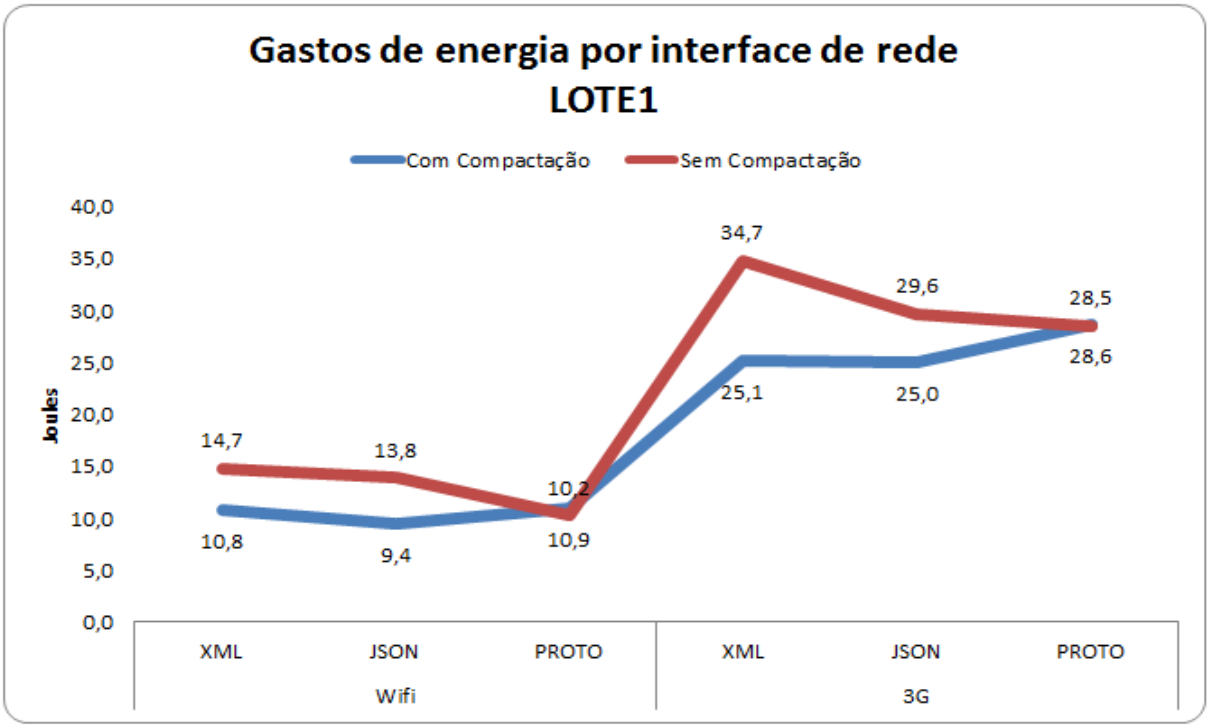


Ilustração 3.26- Gastos de energia por interface de rede (LOTE 1)

No processo de sincronização, para um volume de dados do tipo LOTE 1, a energia gasta na compactação dos formatos de texto, é largamente compensada com uma melhor performance de gestão de energia na sincronização dos dados, qualquer que seja a interface de rede utilizada. O mesmo já não se verifica com o formato binário, Protocol Buffers, que apresenta um maior gasto de energia quando existe uma compactação dos dados. Este maior gasto de energia, apesar de não ser significativo, é revelador da ineficiência da compactação de dados nos formatos binários quando o volume de dados não é significativo.

Para um volume de dados com um tamanho maior, os resultados apurados ao nível de gasto de CPU e Rede foram os seguintes:

GASTOS DE ENERGIA DE CPU (JOULES) LOTE 2		
	COM COMPACTAÇÃO	**SEM COMPACTAÇÃO**
XML	28,2	29,1
JSON	28,6	28,7
PROTO	28,9	29,0

Tabela 3.9- Gastos de energia, em joules, com o processamento dos dados do LOTE 2

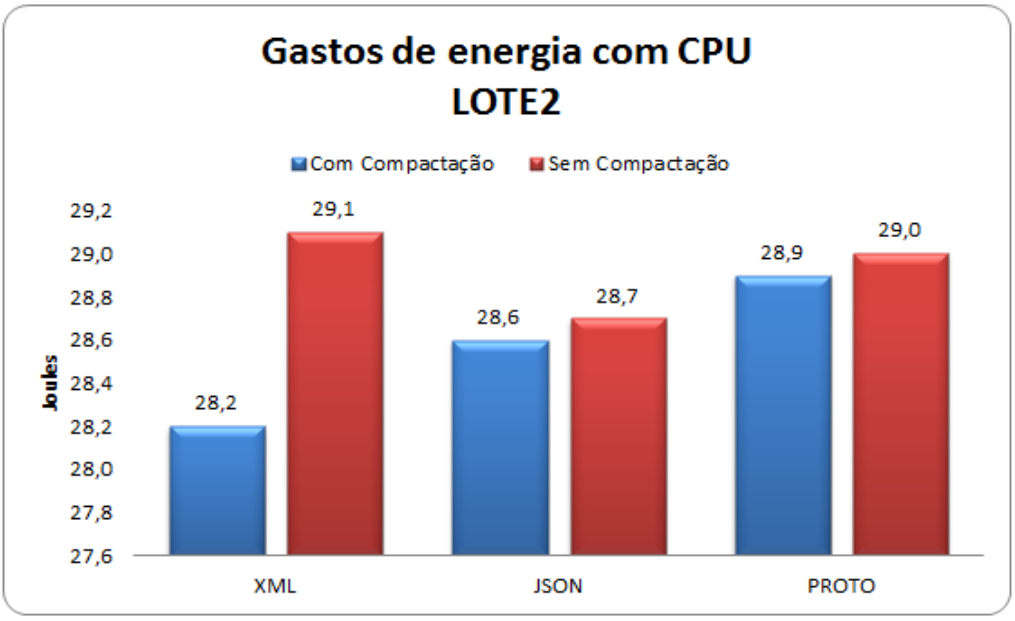


Ilustração 3.27- Gastos de energia com o processamento dos dados do LOTE 2

O resultado obtido com um volume de dados maiores indica que a energia gasta com CPU durante o processo de sincronização com compactação de dados é similar à energia gasta, no mesmo processo, sem a compactação dos dados, à excepção do formato XML, que apresenta uma diferença de quase um joule. Outro aspecto a referir é que para volumes de dados menores, o processo de compactação tem um custo maior nos gastos de energia com processamento do que com volumes de dados um pouco maiores.

GASTOS DE ENERGIA POR INTERFACE DE REDE (JOULES) LOTE 2			
INTERFACE DE REDE	**FORMATO**	**COM COMPACTAÇÃO**	**SEM COMPACTAÇÃO**
WI-FI	**XML**	53,3	60,8
	JSON	51,9	58,6
	PROTO	51,9	50,6
3G	**XML**	95,1	129,9
	JSON	79,8	116,2
	PROTO	91,2	102,0

Tabela 3.10- Gastos de energia, em joules, por interface de rede (LOTE 2)

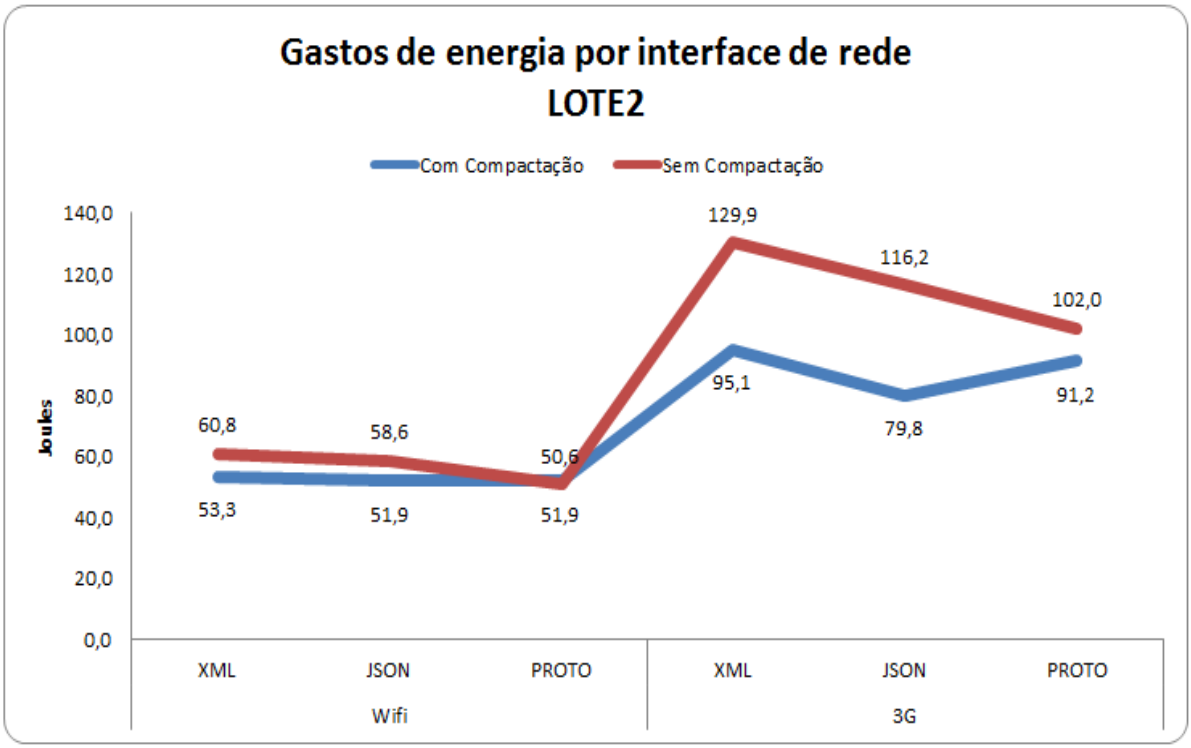


Ilustração 3.28- Gastos de energia por interface de rede (LOTE 2)

Nos processos de sincronização com um volume de dados maior é vantajoso usar a compactação de dados, tendo um impacto positivo nas duas interfaces de rede. Excepção para o formato binário (Protocol Buffers) que apresentou um pior resultado com a compactação de dados na interface Wi-Fi. Este factor pode ser explicado em parte pela baixa performance de compactação dos formatos binários, comparativamente aos formatos de texto, e na sua relação com o custo de análise sintáctica no Servidor Web.

No âmbito destes testes foi produzido um artigo científico referenciado no capítulo relativo às publicações.

4.Conclusões

O crescimento exponencial da tecnologia móvel, especialmente da largura de banda das interfaces de rede, permitiu o crescimento da disseminação dos Smartphones a nível global. Estes equipamentos são dispositivos de informação com características que se assemelham às dos computadores pessoais dispondo de características únicas que os diferenciam dos equipamentos móveis tradicionais. Uma dessas vantagens é a possibilidade dos utilizadores adicionarem aplicações que sincronizem informação com a Web. Esta característica está presente em grande parte das aplicações instaladas nos Smartphones. Apesar da pouca literatura técnica e científica sobre o assunto, este estudo veio permitir analisar as várias abordagens de sincronização para este tipo de aplicações e permitiu perceber a importância da escolha de um formato de intercâmbio de dados na gestão de energia de um Smartphone.

Apesar da implementação ter sido efectuada na *framework* SDK do Google para os sistemas Android, na prova de conceito é proposta uma arquitectura multiplataforma para a monitorização da informação de um Smartphone. Esta arquitectura agrega um conjunto de técnicas que permitem uma gestão eficiente da bateria do equipamento móvel, através da implementação de sincronizações agregadas e de um sistema de *push* de dados que não necessita de uma ligação permanente ao servidor. O sistema de *push* proposto na plataforma utiliza o sistema de mensagens escritas dos telemóveis para operar diversas acções no dispositivo. A segurança das mensagens é implementada através de encriptação simétrica onde a chave de encriptação é personalizada para cada dispositivo, porque é baseada na password de cada conta individual do servidor.

Para este tipo de aplicações, que sincronizam dados com um Servidor Web de forma repetitiva e autónoma, a gestão de energia é um factor fundamental. Os testes realizados à plataforma de Controlo Parental permitiram concluir que dos formatos estudados (XML, JSON e Protocol Buffers), o que melhor performance registou foi o formato JSON com compactação de dados. Este formato registou os melhores resultados nos seguintes pontos: tempos de sincronização com o Servidor Web; análise sintáctica e gestão de bateria. O único registo negativo, comparativamente com os outros formatos, foi relativo à selecção e compressão de dados no dispositivo móvel, obtendo a pior performance, no entanto, este valor negativo foi completamente absorvido pelos bons resultados ao nível da análise sintáctica, dos tempos de sincronização e da gestão de energia. O formato XML apresentou resultados mais fracos, comparativamente ao formato JSON, a todos os níveis excepto, na gestão da bateria com os

gastos de processamento do CPU, quando existe compactação de dados. O formato binário, Protocol Buffers, apresentou os melhores resultados em todos os níveis, comparativamente aos formatos de texto, quando não houve compactação, no entanto, revelou-se pouco eficiente comparativamente aos formatos de texto, quando existe compactação de dados. Este formato revelou-se mais indicado para transferência de grandes volumes de dados, onde o processo de compactação é mais penoso ou em situações onde não é desejável que haja compactação dos dados. Uma das vantagens dos formatos binários, em relação aos formatos de texto, é que permitem a inclusão de informação no formato binário na sua estrutura, possibilitando a transferência de outro tipo de informação como sejam, imagens, vídeos, áudio etc…

A compactação de dados revelou-se de grande importância, nos formatos de texto, optimizando as velocidades de sincronização e a gestão de energia, dada a sua boa performance na redução do tamanho da informação a sincronizar, reduzindo a informação em cerca de 66% do seu tamanho original. Os resultados mais expressivos são obtidos nas interfaces de rede mais lentas, como a interface 3G, não se verificando uma discrepância tão grande nas interfaces de rede mais rápidas, como na interface Wi-Fi. Nos dois tipos de volume de dados (LOTE 1 e LOTE 2) analisados a compactação teve sempre um impacto positivo.

Assim, no processo de escolha de um formato adequado para a transferência de informação entre um Smartphone e a Web é necessário avaliar três tipos de requisitos: tipo de informação a ser sincronizada; tamanho da informação a ser sincronizada e interface ou interfaces de rede utilizadas.

Desta forma, o formato de intercâmbio de dados a utilizar para as sincronizações em plataformas deste tipo é o formato JSON, com compactação de dados. Os resultados foram obtidos para este tipo de plataforma específica (monitorização da informação), onde os dados sincronizados são do tipo texto. Em contextos diferentes a eficiência do formato JSON pode não se revelar a mesma. Apesar das discrepâncias de valores entre os diversos formatos serem, em alguns casos, pouco significativas, têm impacto mais acentuado na gestão de bateria, visto que o processo de sincronização é repetitivo. Como analisado na revisão da literatura, a plataforma deve recorrer sempre que seja desejável ou possível, ao processo de sincronização por agregação e à utilização do sistema de *push* proposto, aumentando-se desta forma a longevidade da bateria e reduzindo os tempos dos processos de sincronização.

5.Publicações

GIL, B., TREZENTOS, P. (2011). Impacts of data interchange formats on energy consumption and performance in Smartphones (OSDOC '11). ACM, New York, NY, USA

6.Bibliografia

ANDROID DEVELOPERS (2011). Package Index. http://developer.android.com/reference/packages.html (última visita Janeiro de 2011).

ARMSTRONG, T., TRESCASES, O., AMZA, C. AND LARA, E. (2010). Efficient and Transparent Dynamic Content Updates for Mobile Clients. Department of Electrical and Computer Engineering and Department of Computer Science, University of Toronto, Canada.

BALASUBRAMANIAN, N., BALASUBRAMANIAN, A. AND VENKATARAMANI, A. (2009). Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. Department of Computer Science, University of Massachusetts Amherst.

CERAMI, E. (2002). Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDLEthan Cerami. Publisher: O'Reilly. First Edition February 2002, ISBN: 0-596-00224-6, 304 pages.

C2DM, (2011). Google Projects for Android: C2DM. Android Cloud to Device Messaging Framework. http://code.google.com/intl/ptPT/android/c2dm/index.html (última visita em Janeiro de 2011).

DEOLASEE, P., KATKAR, A., PANCHBUDHE, A., RAMAMRITHAM, K. AND SHENOY, P. (2001). Adaptive Push-Pull: Disseminating Dynamic Web Data. Department of Computer Science and Engineering, Indian Institute of Technology Bombay and Department of Computer Science, University of Massachusetts.

DUAN, Z., GOPALAN, K. AND DONG, Y. (2005). Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic. Florida State University and University of Hawaii.

GALPIN, M. (2010). Using Internet data in Android applications Parse XML, JSON, and protocol buffers data. © Copyright IBM Corporation 2010.

GHOSH, D. (2010). Building Push Applications for Android, Google I/O 2010 – Developer Conference. http://www.google.com/events/io/2010/sessions/push-applications-android.html

INTERNATIONAL TELECOMMUNICATION UNION, (11 June 2010) Geneva, Switzerland. http://www.itu.int/council/groups/wg-cop/second-meeting-june

2010/CommonSenseSmartPhonesSmartKidsWhitePaper.pdf

JSON (2011). http://www.json.org/ (última visita em Janeiro de 2011).

KALIN, M. (2009). Java Web Services: Up and Running, February 2009: First Edition. ISBN: 978-0-596-52112-7.

MEIER, R. (2010). Professional Android 2 - Application Development. Published by Wiley Publishing, Inc, ISBN: 978-0-470-56552-0.

NURSEITOV, N., PAULSON, M., REYNOLDS, R. AND XIZURIETA, C. (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. Department of Computer Science Montana State University – Bozeman, Montana, 59715, USA.

PROTOCOL BUFFERS (2011). http://code.google.com/intl/pt-PT/apis/protocolbuffers/ (última visita em Janeiro de 2011).

RODEN, T. (2009). Building the Realtime User Experience. Published by O'Reilly Media, Inc., July 2010: First Edition, ISBN: 978-0-596-80615-6.

RODRIGUEZ, V. N., HUI, P., CROWCROFT, J. AND RICE, A. (2010). Exhausting Battery Statistics. Understanding the energy demands on mobile handsets.University of Cambridge and Deutsche Telekom Laboratories.

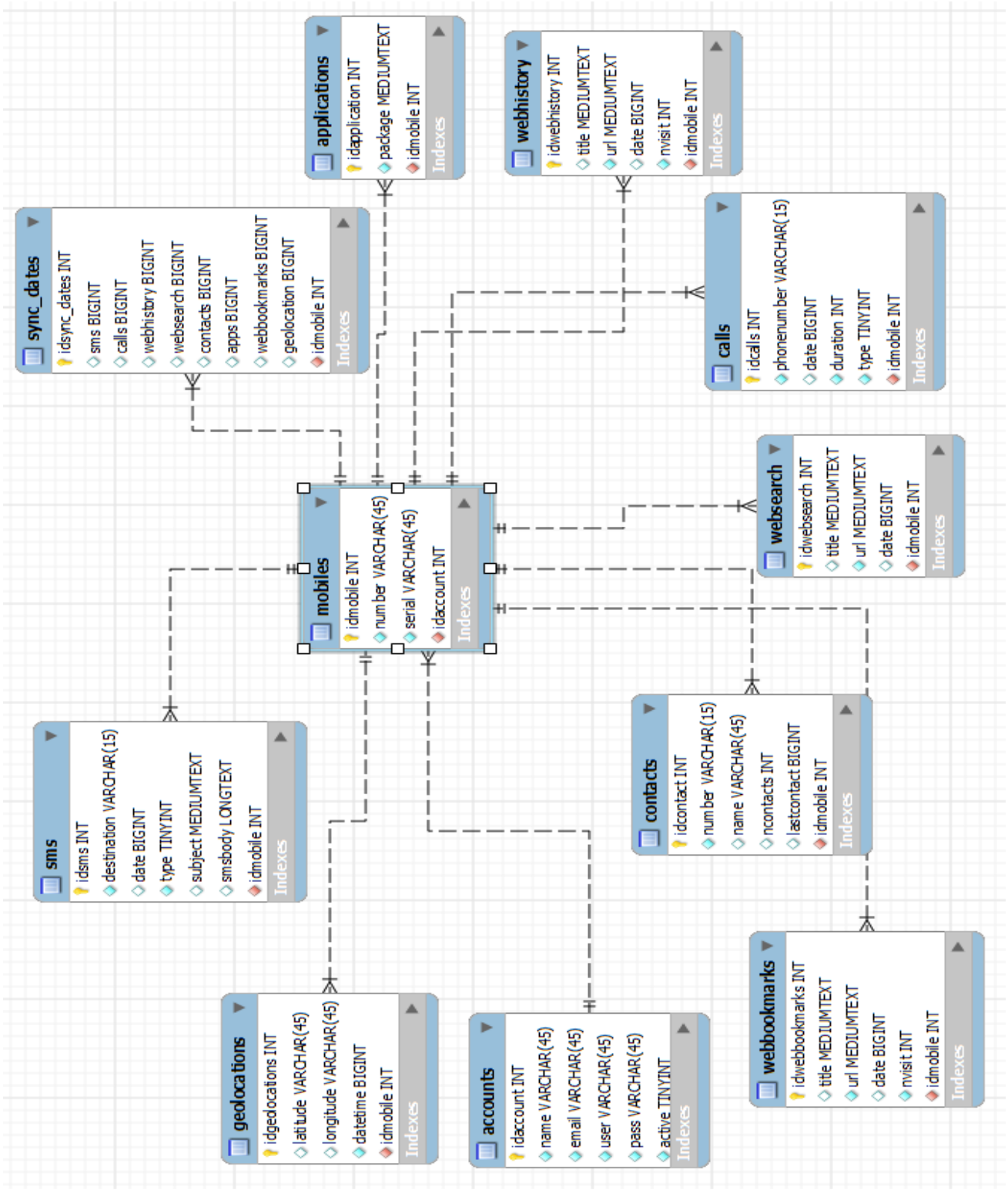
SHARKEY, J. (2009). Coding for Life -- Battery - Life, That Is. Google I/O 2009 – Developer Conference. http://www.google.com/events/io/2009/sessions/CodingLifeBatteryLife.html

SUNMICROSYSTEMS (2008). RESTfulWeb Services Developer'sGuide, SunMicrosystems, Inc. PartNo: 820–4867–05, May 2008.

W3C (2011), Extensible Markup Language. http://www.w3.org/XML/ (última visita em Janeiro de 2011).

7.Anexos

Anexo A - Diagrama de Entidades Relacionamentos



Anexo B – Dados obtidos nos testes de performance

(651 SMS)	**Compressão, no Smartphone, para cada um dos formatos Tempos em milissegundos**		
Nº. / Formato	**XML**	**JSON**	**PROTO**
1	1932	2317	1748
2	1926	2024	1632
3	2154	2294	1742
4	2243	2074	1733
5	2055	2594	2012
6	1974	2013	1818
7	2164	2120	1757
8	1867	2059	1601
9	1817	2079	1573
10	1894	2101	1798
11	1862	2034	1805
12	2054	2002	1755
13	1925	2057	1741
14	2543	1990	1595
15	2393	2074	1663
16	1906	1996	1697
17	2009	1962	1722
18	2012	1959	1722
19	2031	2211	1716
20	2155	1930	1874
21	2007	2048	1639
22	1996	1973	2239
23	2044	2028	1879
24	1847	1974	1716
25	1877	2016	1740
26	1933	2006	1598
27	1892	1977	1591
28	1815	2031	1815
29	1810	1983	1776
30	1916	2622	1845
Média	**2001,77**	**2084,93**	**1751,40**
Desvio Padrao	**166,41**	**165,32**	**133,00**
Desvio Médio	**122,74**	**110,96**	**90,41**
Mediana	**1953,5**	**2029,5**	**1740,5**
Minimo	**1810**	**1930**	**1573**
Máximo	**2543**	**2622**	**2239**

LOTE 1 (650 SMS)	**Análise sintáctica e inserção de dados no DBMS - Web Server Tempos em milissegundos**					
	Com Compactação			**Sem Compactação**		
Nº/Formato	**XML**	**JSON**	**PROTO**	**XML**	**JSON**	**PROTO**
1	326	144	394	249	140	143
2	252	157	250	241	136	243
3	268	148	136	318	140	242
4	353	148	144	481	164	125
5	436	152	254	237	271	150
6	222	143	203	240	142	210
7	360	152	170	184	271	144
8	359	277	144	313	161	377
9	199	151	256	283	143	171
10	488	145	246	314	165	135
11	392	266	252	196	138	140
12	212	150	241	192	157	145
13	311	272	234	255	145	240
14	335	145	247	161	155	283
15	276	137	240	269	226	258
16	220	264	127	283	133	249
17	269	146	234	251	161	134
18	248	155	198	261	150	158
19	253	144	179	265	176	238
20	318	197	248	296	143	240
21	250	162	185	321	264	227
22	207	163	240	399	163	245
23	237	144	247	315	155	135
24	227	230	250	187	164	253
25	264	151	279	203	269	199
26	307	222	183	240	136	170
27	231	267	243	252	156	236
28	265	147	136	244	158	252
29	195	145	239	216	156	203
30	247	152	168	181	149	205
Média	**284,23**	**175,87**	**218,90**	**261,57**	**169,57**	**205,00**
Desvio Padrão	**70,45**	**46,80**	**54,47**	**65,75**	**42,38**	**57,34**
Desvio Médio	**57,23**	**39,20**	**43,59**	**47,88**	**30,64**	**47,87**
Melhor Tempo	**195,00**	**137,00**	**127,00**	**161,00**	**133,00**	**125,00**
Pior Tempo	**488,00**	**277,00**	**394,00**	**481,00**	**271,00**	**377,00**

LOTE 1	Tempos de Sincronização com Web Server Tempos em milissegundos					
	Dados Compactados (GZIP)					
	Tempo Resposta WiFi			Tempo Resposta 3G		
Nº /Formato	XML	JSON	PROTO	XML	JSON	PROTO
1	784	701	732	1444	1230	1948
2	676	533	535	1449	1249	1340
3	567	547	585	1210	1267	1571
4	710	520	600	1438	1338	1197
5	838	703	517	1369	1229	1330
6	545	518	616	1410	1378	1428
7	569	559	501	1329	1267	2238
8	690	601	490	1288	1278	2158
9	705	866	590	2038	1358	1387
10	670	501	564	1828	1207	1320
11	697	522	544	1520	1207	1386
12	622	633	878	1309	1277	1287
13	744	641	420	1308	1468	1250
14	710	505	628	1228	1261	1378
15	813	533	502	1518	1376	1418
16	598	499	768	1409	1430	1450
17	791	691	639	1326	1458	1466
18	573	568	619	1326	1398	1429
19	541	533	477	1349	1298	1370
20	534	512	398	1591	1256	1267
21	677	655	658	1297	1489	1280
22	591	495	510	1299	1377	1365
23	688	500	475	1337	1359	1361
24	629	533	680	1308	1218	1210
25	665	631	494	1329	1286	1308
26	679	572	437	1578	1199	1289
27	577	521	951	1370	1319	1315
28	615	486	502	1190	1316	1330
29	719	630	408	1366	1508	1379
30	611	484	713	1488	1330	1579
Média	**660,93**	**573,10**	**581,03**	**1408,30**	**1321,03**	**1434,47**
Desvio Padrao	**81,51**	**85,59**	**129,68**	**172,95**	**86,91**	**245,56**
Desvio Médio	**68,01**	**68,07**	**101,50**	**120,76**	**72,90**	**157,92**
Mediana	**673**	**533**	**554**	**1357,5**	**1307**	**1367,5**
Melhor Tempo	**534**	**484**	**398**	**1190**	**1199**	**1197**
Pior tempo	**838**	**866**	**951**	**2038**	**1508**	**2238**

LOTE 2	Tempos de Sincronização com Web Server Tempos em milissegundos					
	Dados Compactados (GZIP)					
	Tempo Resposta WIFI			Tempo Resposta 3G		
Nº /Formato	XML	JSON	PROTO	XML	JSON	PROTO
1	3484	4101	3543	6700	7216	6824
2	2649	2503	3282	4797	4438	5029
3	2472	2462	2819	4883	4517	5948
4	2967	2507	2667	5275	4636	6078
5	2674	2309	2415	5337	4614	5666
6	2931	2494	2787	5161	4524	5924
7	2595	2316	2667	5576	4558	4973
8	2728	2361	2787	5108	4360	5357
9	2654	2486	2502	5340	4738	4909
10	2615	2216	2452	4898	4810	5134
11	2651	2360	2749	4668	4406	6197
12	2699	2527	2528	4868	4690	5566
13	2829	2465	2878	5179	4678	4888
14	2819	2462	2773	5427	5139	4809
15	2634	2516	2454	5680	5018	4816
16	2707	2439	2640	5578	4568	4877
17	2688	3385	2545	5308	4587	4858
18	2663	2418	2680	5888	4659	4821
19	2952	2874	2552	5519	4447	4866
20	2579	2485	2450	5467	4739	4718
21	2842	2655	3489	5428	4368	4980
22	2553	2639	2373	6506	4636	5078
23	2772	2474	2652	5380	4963	5068
24	2985	2619	2615	5716	4446	5086
25	2501	2443	2673	4887	4708	4800
26	2461	2596	3375	5198	4878	4856
27	2618	2607	2783	5159	4513	5000
28	2663	2517	2966	5222	4787	4798
29	2530	2474	2862	5073	4747	4697
30	2518	2596	2521	5359	4949	5154
Média	**2714,43**	**2576,87**	**2749,30**	**5352,83**	**4744,57**	**5192,50**
Desvio Padrao	**201,80**	**346,76**	**303,48**	**438,68**	**497,64**	**515,27**
Desvio Médio	**144,31**	**192,01**	**223,49**	**309,14**	**253,73**	**401,33**
Mediana	**2663,00**	**2490,00**	**2670,00**	**5322,50**	**4647,50**	**4990,00**
Melhor Tempo	**2461,00**	**2216,00**	**2373,00**	**4668,00**	**4360,00**	**4697,00**
Pior Tempo	**3484,00**	**4101,00**	**3543,00**	**6700,00**	**7216,00**	**6824,00**

LOTE 1	Tempos de Sincronização com Web Server Tempos em milissegundos					
	Dados Não Compactados					
	Tempo Resposta WIFI			Tempo Resposta 3G		
Nº /Formato	XML	JSON	PROTO	XML	JSON	PROTO
1	799	704	774	1824	1395	1206
2	811	589	526	1840	1450	1298
3	753	622	597	1757	1488	1364
4	655	612	587	1807	1469	1260
5	746	556	610	1809	1488	1279
6	781	591	594	1728	1515	1307
7	743	529	518	1896	1550	1341
8	901	691	549	2298	1568	1308
9	726	717	615	2067	1459	1387
10	663	603	611	2071	1638	1298
11	894	594	632	1918	1818	1380
12	904	590	739	1846	1589	1390
13	758	698	526	1678	1550	1246
14	767	585	543	1892	1565	1319
15	675	578	481	2854	1595	1358
16	779	613	622	1822	1571	1314
17	776	622	568	1766	1569	2428
18	880	791	606	1777	1678	1870
19	778	747	548	1858	1680	1889
20	753	724	523	1926	1585	1809
21	735	510	632	1831	1719	1969
22	711	702	631	1778	1590	1640
23	722	827	515	1749	1797	1805
24	881	634	558	1769	2038	1597
25	753	607	544	1828	1768	1680
26	822	598	537	2035	1579	1707
27	969	646	547	2190	1527	1608
28	833	570	537	2108	1689	1669
29	723	599	575	1938	1729	1898
30	799	568	643	1998	1657	1919
Média	**783,00**	**633,90**	**582,93**	**1921,93**	**1610,43**	**1551,43**
Desvio Padrao	**74,60**	**74,38**	**62,65**	**223,85**	**130,15**	**291,01**
Desvio Médio	**58,67**	**60,54**	**48,80**	**151,04**	**99,75**	**251,20**
Mediana	**771,5**	**609,5**	**571,5**	**1843**	**1582**	**1388,5**
Melhor Tempo	**655**	**510**	**481**	**1678**	**1395**	**1206**
Pior Tempo	**969**	**827**	**774**	**2854**	**2038**	**2428**

LOTE 2	Tempos de Sincronização com Web Server Tempos em milissegundos					
	Dados Não Compactados					
	Tempo Resposta WIFI			Tempo Resposta 3G		
Nº /Formato	XML	JSON	PROTO	XML	JSON	PROTO
1	3675	3599	3339	7271	9174	9185
2	3288	2745	3032	6559	5937	5838
3	2750	2909	2317	6357	5769	5679
4	3298	2687	2687	6403	5747	5667
5	3142	2627	2544	6613	5809	5869
6	2943	2783	2486	7188	6326	6043
7	2930	2827	2707	7465	6330	5693
8	2945	2867	2563	7370	6039	6147
9	3162	2625	2832	6969	5978	5908
10	3067	2606	2492	7089	7038	5825
11	3068	2720	2576	7288	5688	6331
12	2964	2846	2506	9388	5856	6407
13	2832	2937	2580	8006	6220	6467
14	2949	2600	2693	8058	5967	7228
15	2998	2703	2632	7988	6489	5727
16	3263	3045	2519	7759	7608	5931
17	3008	2910	2653	8366	6888	6019
18	3235	2814	2422	7028	6746	6587
19	2866	2709	2558	7096	7259	5776
20	2920	2855	2500	7132	7018	5918
21	2942	2743	2587	6949	6819	5750
22	2840	2718	2440	7396	7080	5649
23	2937	2783	2517	7229	6957	5677
24	2876	4024	2545	7089	7367	5778
25	3196	2671	2881	8259	7008	5569
26	2988	2523	2499	7757	6538	5769
27	3008	3008	2633	7078	6809	5549
28	3055	2834	2662	9536	7698	5317
29	3102	2660	2790	7618	6517	5590
30	3163	3022	2693	7210	6550	5727
Média	**3047,00**	**2846,67**	**2629,50**	**7450,47**	**6640,97**	**6020,67**
Desvio Padrao	**182,41**	**293,25**	**194,56**	**732,60**	**734,74**	**695,67**
Desvio Médio	**140,20**	**180,62**	**136,70**	**549,66**	**566,36**	**415,31**
Mediana	**3003,00**	**2783,00**	**2578,00**	**7250,00**	**6544,00**	**5801,50**
Melhor Tempo	**2750,00**	**2523,00**	**2317,00**	**6357,00**	**5688,00**	**5317,00**
Pior Tempo	**3675,00**	**4024,00**	**3339,00**	**9536,00**	**9174,00**	**9185,00**