BlueGroups: *Middleware* para Construção de Aplicações Distribuídas em Dispositivos Móveis

Thyago Salvá[1], Rafael S. da Silva[1], André P. Vargas[1], Odorico M. Mendizabal[1]

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande - FURG
Campus Carreiros: Av. Itália km 8 Bairro Carreiros – Rio Grande – RS – Brazil

{thyagosalva,rafaelsenna,andre.prisco,odoricomendizabal}@furg.br

Abstract. *Equipped with powerful resources and cheaper prices, mobile devices have achieved wide popularity, increasing the demand for mobile applications. However, developing such applications with the current programming language supports may need advanced programming skills. By dealing with distributed devices and shared data across wireless networks, a solid knowledge about related issues may also be required. In order to make the development simple, this work introduces a middleware that implements group communication primitives over Bluetooth and allows data management through version control mechanisms.*

Resumo. *Equipados com diversos recursos e preços mais baixos, dispositivos móveis atingiram grande popularidade, aumentando a demanda por aplicações móveis. Entretanto, o desenvolvimento de tais aplicações com o suporte atual de linguagens de programação exige habilidades de programação avançadas. Ao lidar com dispositivos distribuídos e dados compartilhados em redes sem fio, um sólido conhecimento sobre problemas relacionados pode também ser requerido. Para facilitar o desenvolvimento, este trabalho introduz um middleware que implementa primitivas de comunicação em grupo sobre Bluetooth e permite gerenciamento de dados usando mecanismos de controle de versão.*

1. Introdução

Devido ao avanço tecnológico e popularização dos dispositivos móveis, telefones celulares e *smartphones* apresentam-se como acessório indispensável no cotidiano das pessoas. Tarefas antes restritas apenas aos computadores pessoais passam a ser realizadas em dispositivos móveis, com a vantagem destes últimos estarem integrados a outros dispositivos, como câmeras digitais e GPS (*Global Positioning System*).

Outra importante característica dos dispositivos móveis em geral é a presença de múltiplas interfaces de rede, que permitem conectividade a quase todo o instante por diferentes meios de comunicação. Dentre os meios de transferência de dados disponíveis está o protocolo de comunicação Bluetooth [SIG 2010], que provê um meio de comunicação através de uma frequência de rádio de curto alcance.

No entanto, desenvolver aplicações distribuídas confiáveis para estes dispositivos ainda é uma tarefa dispendiosa e de difícil validação. Boa parte das bibliotecas e ferramentas de suporte ainda são bastante limitadas, bem como os protocolos existentes. Parte desta limitação é herdada de soluções desenvolvidas para gerações mais antigas de dispositivos móveis, que apresentavam pouco poder de processamento e memória.

Com a crescente oferta de dispositivos mais robustos, cresce também a demanda por aplicações mais complexas, capazes de lidar com um volume razoável de dados. Portanto, é necessário garantir que dados possam ser replicados e atualizados de forma colaborativa, sem violar a consistência e integridade da informação.

Com o objetivo de oferecer maior nível de abstração aos programadores, este artigo descreve a implementação de um *middleware* voltado a soluções distribuídas com compartilhamento de dados. O *middleware*, chamado BlueGroups, além de facilitar o desenvolvimento deste tipo de aplicações, esconde do programador interferências do ambiente em que os aplicativos executam. Por exemplo, falhas decorrentes do meio de comunicação ou falhas em dispositivos são detectadas pelo *middleware*, podendo estas ser tratadas pelo programador. Analogamente, interrupções durante transferência de arquivos são recuperadas sem qualquer intervenção. Dessa forma, o desenvolvedor pode ater-se na lógica de negócio de sua aplicação, tendo a garantia de que a comunicação ocorrerá de forma consistente, mesmo em ambientes não controlados.

A estrutura deste trabalho esta organizada da seguinte forma: a próxima seção apresenta trabalhos relacionados. A Seção 3 descreve o BlueGroups. Uma das principais características do BlueGroups, a notificação de eventos remotos, é apresentada na Seção 4. Uma visão geral do funcionamento do *middleware* é discutida na Seção 5 e as considerações finais sobre este trabalho aparecem na Seção 6.

2. Estado da Arte

A construção de aplicações distribuídas para dispositivos móveis requer habilidade do programador, além de um sólido conhecimento sobre as tecnologias utilizadas. Analogamente a este trabalho, outras pesquisas também procuram alternativas para reduzir a complexidade envolvida no desenvolvimento destas aplicações. Esta seção destaca trabalhos relacionadas à construção de *middlewares* para dispositivos móveis que ofereçam suporte a comunicação Bluetooth.

Neste contexto, encontra-se o MOnKey [Albert et al. 2008] (*Middleware on Key*), um *middleware* desenvolvido para tratar a heterogeneidade dos dispositivos, abstraindo diferenças entre plataformas e tecnologias para comunicação. Embora o MOnKey possibilite a utilização de diferentes tecnologias de rede (como Ethernet, WiFi e Bluetooh) e primitivas básicas para a comunicação, o *broadcast* é a única forma de envio de mensagens. Quando utiliza Bluetooth, este *middleware* apresenta limitações, pois não oferece mecanismos de roteamento de mensagens em uma Scatternet. Dessa forma, a comunicação entre dispositivos se dá apenas em redes Piconet [SIG 2010].

Algumas ferramentas para comunicação em grupo, como o Isis [Birman and Renesse 1994], Horus [van Renesse et al. 1996] e JGroups [JGroups 2011a] ganharam destaque nos anos 90 por oferecerem alto nível de abstração ao programador de aplicações distribuídas. No entanto, a única referência sobre ferramentas deste porte para dispositivos móveis é um projeto de implementação do JGroups para Java ME, chamado JGroups-ME [JGroups 2011b]. Entretanto, este projeto foi descontinuado e não foi possível encontrar maiores informações sobre a utilização deste *middleware*.

O BlueGroups baseia-se no modelo de comunicação em grupo para oferecer alto nível de abstração para a implementação de aplicações sobre o protocolo Bluetooth. Se

comparado com [Albert et al. 2008], este modelo oferece maior controle sobre um grupo de dispositivos distribuídos, pois oferece diferentes primitivas para troca de mensagem, gerenciamento de membros e detecção de falhas. Além disso, a implementação de um recurso de notificação de eventos remotos abstrai completamente a necessidade do programador tratar o recebimento de mensagens.

Com relação ao compartilhamento de dados entre dispositivos móveis, alguns trabalhos oferecem mecanismos eficazes. Em [Kotilainen et al. 2005], os autores propõe uma arquitetura de rede P2P na qual dispositivos móveis podem participar. Esta arquitetura, chamada *Mobile Chedar*, permite que dispositivos móveis utilizem Bluetooth para o compartilhamento de arquivos. Esta abordagem faz uso de uma arquitetura híbrida, em que dispositivos móveis conseguem comunicar com nodos roteadores, que se comunicam através de uma rede TCP/IP. Serviços de localização, por exemplo, não são implementados pelos dispositivos móveis.

O BlueGroups também oferece recursos para compartilhamento de arquivos, utilizando o conceito de repositórios e controle de versão. Ele permite retomar transferências de arquivo interrompidas e permite que múltiplas fontes forneçam um mesmo arquivo. A grande vantagem, entretanto, é que além do compartilhamento de arquivos, a comunicação entre dispositivos é facilitada pelas primitivas de comunicação em grupo.

3. BlueGroups

O BlueGroups é um *middleware* que implementa comunicação em grupo e gerenciamento de dados compartilhados. Com o objetivo de prover uma camada de abstração de *software* aos programadores, este *middleware* disponibiliza uma API com as primitivas oferecidas, bastando que o programador importe a biblioteca BlueGroups.

Para utilizá-lo, é necessário que os dispositivos tenham a máquina virtual Java compatível com a configuração CLDC 1.1 e o perfil MIDP 2.1. Além dissso, o dispositivo deve ter a tecnologia Bluetooth embarcada, seguindo as especificações SIG [SIG 2010].

A seguir são descritos os principais serviços oferecidos pelo BlueGroups:

- **Criação e Adesão a Grupos:** O *middleware* abstrai do programador o gerenciamento de entrada de membros e criação de grupos. A ferramenta permite que o dispositivo, ao buscar por um grupo dentro de sua área de alcance, insira-se nele. Caso o grupo não exista, este é automaticamente criado e o dispositivo é designado coordenador. Embora o Bluetooth limite o número máximo de elementos em uma Piconet, o BlueGroups aumenta a escalabilidade permitindo a formação de redes Scatternet [SIG 2010].
- **Roteamento de Mensagens:** O *middleware* permite que membros que não consigam se comunicar diretamente entre si possam participar do mesmo grupo através do roteamento de mensagens. Para isso, basta que um dispositivo pertença a duas ou mais redes Piconet, formando uma rede Scatternet [SIG 2010]. A Figura 1 ilustra uma rede Scatternet composta pela junção das redes A e B. Caso o nodo C envie uma mensagem por *broadcast*, esta será difundida em sua Piconet (rede A) e o nodo D, que neste exemplo atua como roteador, fará a difusão da mensagem na rede B.
- **Estabelecimento e gerenciamento de conexões:** Tradicionalmente, a construção de uma aplicação que necessite se comunicar com processos remotos exige a

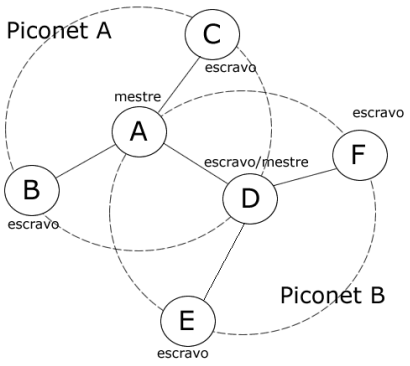

Figura 1. Exemplo de uma Scatternet

criação de canais de entrada e saída e da adoção de um protocolo de comunicação. O envio e recebimento de mensagens devem ser explicitamente gerenciados por *threads* da aplicação. Tais tarefas são dispendiosas e suscetíveis a erros. Por exemplo, o desenvolvedor talvez precise implementar mecanismos para controle de concorrência. Além de suportar troca de mensagem ponto a ponto, o BlueGroups também oferece primitivas *multicast* e *broadcast* [Birman 1997]. Para o recebimento de mensagens, o *middleware* utiliza o modelo de notificação de eventos remotos, conforme será descrito na Seção 4.

- **Recuperação de transferência de arquivos:** Dispositivos móveis frequentemente sofrem interrupções de comunicação, seja por erro de software, carga de bateria insuficiente ou pela falta temporária de alcance de sinal. Para minimizar o impacto de indisponibilidade temporária, o BlueGroups possui um mecanismo de transferência de arquivos próprio. Durante a transferência, o arquivo é dividido em várias partes e cada uma é transferida individualmente. Caso uma conexão seja interrompida durante a transferência, é possível continuá-la a partir da última parte transferida. Assim, somente serão solicitadas as partes restantes do arquivo.
- **Controle de Versão:** O BlueGroups é capaz de verificar se um conjunto de arquivos está desatualizado. Para isso, ele cria um repositório de dados onde os arquivos são armazenados. Outros dispositivos do grupo podem possuir um repositório local e possivelmente uma versão do mesmo arquivo. Quando um dispositivo deseja atualizar seus arquivos, o *middleware* compara as versões dos arquivos nos diferentes dispositivos. Dessa forma, é possível identificar e atualizar conteúdo de forma distribuída.

3.1. Arquitetura

A arquitetura do BlueGroups implementa camadas de serviços responsáveis por prover o gerenciamento do grupo e controle de versão. Para a composição de grupos idealizou-se a criação de grupos fechados, hierárquicos e dinâmicos. Portanto, dispositivos externos não podem interagir com elementos do grupo através do *middleware*, decisões são tomadas por um único membro, o coordenador, e é possível que novos membros entrem no grupo ao longo de sua existência.

A Figura 2 ilustra a arquitetura do BlueGroups e suas camadas são descritas a seguir:

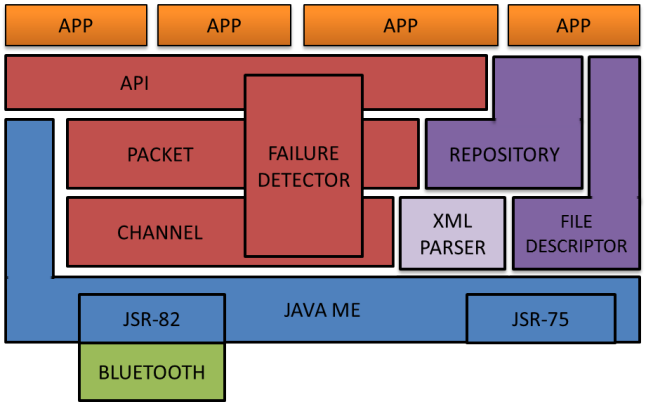

Figura 2. Arquitetura BlueGroups

- **CHANNEL:** Camada responsável pelas conexões entre dispositivos. Ela oferece procedimentos para o descobrimento de dispositivos e serviços, estabelecimento e gerenciamento de conexões e, principalmente, por realizar trocas de mensagens.
- **FAILURE DETECTOR:** Responsável por detectar falhas de comunicação. Caso um dispositivo falhe, este módulo identifica a falha e envia uma mensagem do tipo *Failure* para notificar o coordenador. Observe que tais mensagens ocorrem no âmbito do *middleware*, ou seja, são transparentes para as aplicações.
- **PACKET:** Esta camada empacota e desempacota as mensagens trocadas pelo *middleware*. Os tipos de mensagens suportadas são: *View*, *Failure*, *Notification*, *Checkout*, *File* e *Message*. Exceto pela mensagem *Message*, que é utilizada para envio de mensagens da aplicação, as demais mensagens são trocadas internamente pelo *middleware*, não havendo necessidade do programador tratá-las explicitamente.
- **REPOSITORY:** Esta camada implementa mecanismos para controle de versão sobre um conjunto de arquivos. Metadados que descrevem informações como nome, versão e a lista de arquivos pertencentes ao repositório são salvos em arquivos XML (*Extensible Markup Language*). Um exemplo desse arquivo de metadados aparece na Figura 3. Os identificadores id e $name$ do repositório são atributos do elemento $repository$ (linha 2). O elemento $revision$ (linha 3) possui um atributo chamado $version$ que contém o identificador da última versão. Cada arquivo presente no repositório (linhas 4 e 10) é representado por um elemento $file$ que contém as informações sobre cada arquivo ($name$, id, $address$ e $md5$).
- **FILE DESCRIPTOR:** Camada responsável pela criação e manipulação de arquivos. Cada arquivo contém metadados para o nome, localização, identificador, revisão e um código para garantir integridade de seu conteúdo[1]. O identificador *id* é gerado no momento da adição deste arquivo ao repositório e utiliza o valor obtido pela execução do algoritmo MD5 sobre este arquivo. O valor MD5 também é calculado a cada atualização do arquivo, permitindo a verificação de integridade do arquivo ao final de uma transferência entre dois ou mais dispositivos. Isto é feito antes da operação *commit* e durante a operação *update*. A revisão mantém uma marca de tempo (*timestamp*) do momento em que foi realizada a última alteração no arquivo, ou seja, momento em que foi executada a operação *commit*.

[1]Para este propósito é utilizado o algoritmo MD5 (*Message-Digest algorithm 5*)

```
1  <?xml version='1.0' encoding='UTF−8' ?>
2  <repository id="1320277873319" name="Catalogo">
3      <revision version="1320279335795">
4          <file revision="1320279335795">
5              <name>ReadMe.txt</name>
6              <id>d41d8cd98f00b204e9800998ecf8427e</id>
7              <address>repSample</address>
8              <md5>1055d3e698d289f2af8663725127bd4b</md5>
9          </file>
10         <file revision="1320279104550">
11             <name>Sample.txt</name>
12             <id>9e107d9d372bb6826bd81d3542a419d6</id>
13             <address>repSample</address>
14             <md5>9e107d9d372bb6826bd81d3542a419d6</md5>
15         </file>
16     </revision>
17 </repository>
```

Figura 3. Metadados de um repositório

- **XML PARSER:** Para que os metadados do repositório possam ser manipulados e persistidos no dispositivo, foi criado um módulo para manipulação de documentos XML. Este módulo utiliza a biblioteca kXML2 [Haustein 2011].
- **API:** As aplicações acessam as funcionalidades do BlueGroups através de sua API. Esta fornece um conjunto de funcionalidades para comunicação entre dispositivos além de métodos para o controle de versão. A documentação completa da API pode ser acessada em [BlueGroups 2012].

4. Notificação de Eventos Remotos

A arquitetura do BlueGroups oferece em sua API uma abstração bastante poderosa aos programadores de aplicações distribuídas, a notificação de eventos remotos. Essa abstração permite que cada dispositivo no grupo perceba eventos ocorridos em outros dispositivos como se fossem eventos locais. Portanto, a atualização de uma *view*, o recebimento de uma mensagem ou mesmo a detecção de uma falha, são tratados como eventos.

Este paradigma facilita a programação de aplicações distribuídas uma vez que o programador não precisa gerenciar linhas de execução para receber informação dos canais que interligam todos os membros do grupo. Basta que o programador implemente a interface *ReceiveListener* (vide Figura 4) com o código referente ao tratamento do evento.

```
1  public interface ReceiveListener {
2
3      public void view(View view);
4      public void receive(String data);
5      public void failure(String data);
6  }
```

Figura 4. Interface ReceiveListener

Três métodos podem ser implementados pelo programador: *view*, *failure* e *receive*.

O método *view* será executado quando a visão local do dispositivo for atualizada. Por exemplo, se um novo elemento junta-se ao grupo, os outros componentes recebem uma notificação de evento e o método *view* é invocado. Quando uma mensagem for entregue ao dispositivo, o BlueGroups gera um evento do tipo *receive*. Nesse caso, o método *receive* é invocado. Finalmente, o método *failure* será invocado quando ocorrer um evento, gerado pelas primitivas do bluetooth, que indique uma suspeita de falha de outro membro do grupo.

5. Funcionamento do BlueGroups

Esta seção apresenta o funcionamento do BlueGroups. Em [BlueGroups 2012] é possível encontrar exemplos de aplicações que utilizam o BlueGroups.

5.1. Gerenciamento do Grupo

As aplicações que fizerem uso do BlueGroups gerenciam facilmente os membros do grupo. Na Figura 5, observar-se o primeiro passo para a adesão ao grupo: a execução do método *join*. Caso o grupo ainda não exista, ele será criado e o dispositivo assumirá o papel de coordenador (*Device 1*). Se o grupo já existir, o coordenador criará uma nova visão informando todos os dispositivos sobre a existência de um novo membro. Quando outro dispositivo aparecer na área de alcance de um membro do grupo e invocar o método *join*, ele identificará um serviço disponível e fará uma solicitação para se juntar ao grupo (Figura 5, tempo 2). Se o membro do grupo que identificou a solicitação for o coordenador, ele atualizará a *View* com o novo membro e enviará uma mensagem para os membros do grupo com a nova *View*.

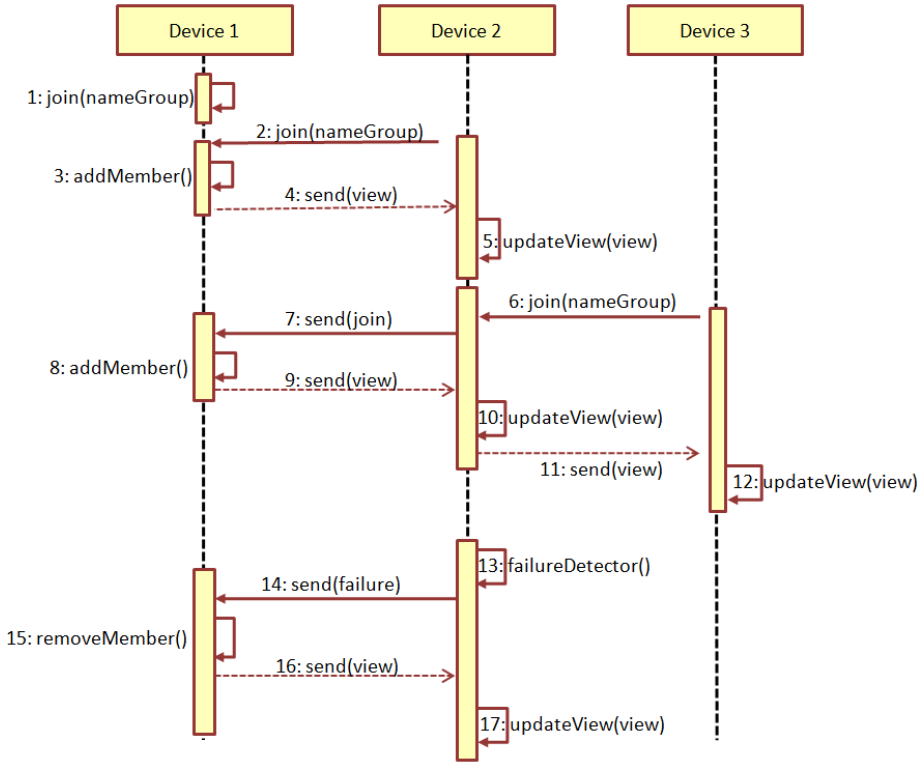

Figura 5. Diagrama de Gerenciamento do Grupo

Se o membro que identificar a solicitação não for o coordenador, ele encaminha o pedido ao coordenador (Figura 5, tempo 7). Enquanto o coordenador não for notificado sobre esta solicitação, os demais membros não terão conhecimento sobre o novo membro. O BlueGroups faz o roteamento de mensagens através dos mestres de cada piconet.

Quando um membro do grupo falhar, membros que estiverem conectados diretamente a ele detectam a falha e notificam o coordenador (Figura 5, tempos 13 e 14). O coordenador, ao receber a notificação, removerá o membro da *View* e enviará a nova formação de membros aos demais participantes do grupo.

Existem casos mais complexos de detecção de falhas. Por exemplo, na Figura 6 os dispositivos A e D detectam a falha de um dispositivo em comum C. No entanto, D não consegue notificar o coordenador do grupo (dispositivo A). Nesse caso, o dispositivo A atualiza a visão do grupo sem os dispositivos C e D. O dispositivo D por não conseguir notificar o coordenador, cria um novo grupo, e torna-se coordenador, caracterizando o particionamento do grupo. O mesmo ocorre quando o coordenador falha, pois os outros dispositivos detectam sua falha e criam novos grupos.

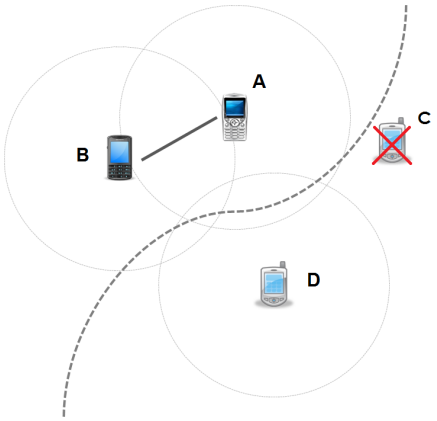

Figura 6. Particionamento do Grupo

O BlueGroups não permite a junção de grupos particionados. Uma vez que um grupo esteja particionado, não há como estes serem reestruturados em um único grupo, mesmo que o nome do grupo seja o mesmo. Versões futuras deverão permitir ao programador configurar o BlueGroups a fim de permitir ou não a junção de grupos particionados.

Uma vez que os grupos estejam estabelecidos, os membros podem trocar mensagens através da primitiva *send*. Quando não informado o destinatário da mensagem, será feita a difusão da mensagem a todos os membros do grupo. O recebimento de mensagens, assim como qualquer notificação do BlueGroups, é feito através de tratamento de eventos.

5.2. Transferência e Atualização de Arquivos

O BlueGroups fornece recursos para aplicações que queiram não somente trocar mensagens, mas também trocar arquivos. Para isso, implementa repositórios. A utilização de um repositório é opcional e definida na adesão ao grupo através da primitiva *join*.

A atualização dos arquivos entre dispositivos remotos é feita em partes. Dessa forma, caso uma transferência seja interrompida, é possível retomá-la e receber apenas as

partes faltantes do arquivo. O BlueGroups mantém um arquivo temporário, no qual serão armazenadas as partes do arquivo transferido. Após receber a primeira parte do arquivo, o membro solicitante recebe também o número de partes em que o arquivo está dividido (ver Figura 7). Esta informação permite que o *middleware* solicite outras partes posteriormente. Quando o arquivo estiver completo, este torna-se disponível no repositório.

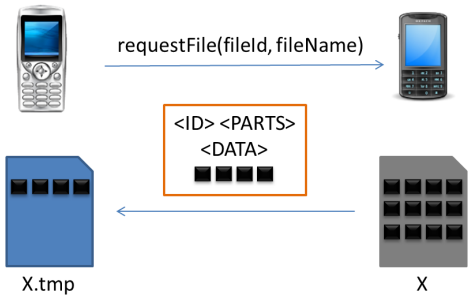

Figura 7. Solicitação por transferência de arquivo

Este mecanismo de transferência de arquivos trás outro benefício, a possibilidade de utilizar múltiplas fontes provedoras para um mesmo arquivo (ver Figura 8).

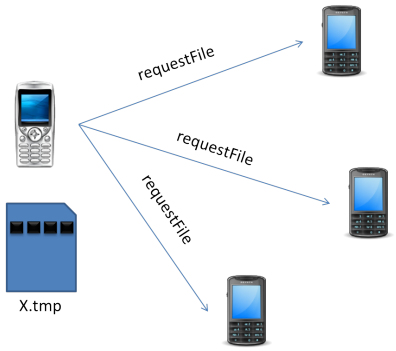

Figura 8. Transferência de arquivo acessando múltiplas fontes

O processo de transferência de arquivos é efetuado automaticamente ao serem executados comandos comuns aos sistemas de controle de versão. Por exemplo, comandos como *add* e *commit* são oferecidos pelo BlueGroups, embora estes não ocasionem transferência de arquivos diretamente. O comando *update*, por sua vez, identifica se há alguma versão mais atual dos arquivos contidos no repositório local em algum repositório do grupo. Em caso afirmativo, inicia a transferência e solicita partes dos arquivos aos membros que possuem cópias atualizadas.

6. Conclusão

Este trabalho apresenta o BlueGroups, um *middleware* para gerenciamento de comunicação em grupos. Voltado principalmente para aplicações distribuídas entre dispositivos móveis, ele oferece uma API simples para envio e recebimento de mensagens. Com o BlueGroups, o programador conta com um ambiente confiável de comunicação em ambientes heterogêneos e suscetíveis a fatores ambientais. Os grupos criados são hierárquicos, fechados e dinâmicos, ou seja, foram concebidos de forma coerente com as características de comunicação móvel.

Dentre as contribuições do trabalho, destaca-se a *Notificação de Eventos Remotos*, conforme apresentado na Seção 4. A utilização de um modelo mais simples e familiar aos programadores é uma vantagem em relação ao modelo de troca de mensagens. Outra contribuição importante é o roteamento de mensagens através de membros de uma scatternet, permitindo a formação de uma topologia que minimize as limitações de alcance do Bluetooth.

Futuramente, pretende-se estender a ferramenta criando módulos configuráveis. Por exemplo, a criação de módulos que implementem outros protocolos de comunicação, como o Wi-Fi. Dessa forma, poderá se estabelecer grupos complexos, utilizando mais de um protocolo de comunicação. Deseja-se também implementar o recurso de junção de grupos, ou seja, oferecer à aplicação a possibilidade de tornar dois grupos em um só de forma transparente. Tal recurso também poderá restaurar grupos particionados por indisponibilidade temporária de um nodo mestre, conforme discutido na Seção 5.1. Embora não tenham sido abordados aspectos como segurança e desempenho, estes devem ser considerados em trabalhos futuros.

Finalmente, o recurso de atualização de arquivos abre a possibilidade para uma série de aplicações antes oferecidas principalmente para dispositivos fixos e estruturas convencionais de comunicação. Controle de versões, murais de avisos e atualizações de software são exemplos de aplicações que, implementadas sobre o BlueGroups, usufruem dos recursos de atualização de arquivos. O modelo de repositórios descentralizados permite também que a informação tramite entre os grupos de forma otimizada e que possa ser acessada e atualizada mesmo que o dispositivo esteja fora do grupo.

Referências

Albert, J., Castang, J., and Chaumette, S. (2008). MOnKEY - A Portable Middleware on Key. In *20th IASTED (PDCS 2008)*.

Birman, K. (1997). Building secure and reliable network applications.

Birman, K. P. and Renesse, R. V. (1994). *Reliable Distributed Computing with the ISIS Toolkit*. IEEE Computer Society Press, Los Alamitos, CA, USA.

BlueGroups (2012). Bluegroups. http://www.bluegroups.furg.br (acesso em julho de 2012).

Haustein, S. (2011). kxml 2. http://kxml.sourceforge.net/kxml2/ (acesso em outubro de 2011).

JGroups (2011a). JGroups – A Toolkit for Reliable Multicast Communication. http://www.jgroups.org/ (acesso em outubro de 2011).

JGroups (2011b). JGroups-ME – JGroups for J2ME (cell phones and PDAs). http://community.jboss.org/wiki/JGroupsME (acesso em outubro de 2011).

Kotilainen, N., Weber, M., Vapa, M., and Vuori, J. (2005). Mobile chedar - a peer-to-peer middleware for mobile devices. In *PerCom 2005 Workshops.*

SIG, B. (2010). *Specification of the Bluetooth System version 4*.

van Renesse, R., Birman, K. P., and Maffeis, S. (1996). Horus: a flexible group communication system. *Commun. ACM*, 39:76–83.