Avaliação da capacidade multimídia de smartphones *

João Victor P. Rezende
Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
joao.peterson@unifesp.br

Bruno N. Moreira Silva
Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
bruno.naponiello@unifesp.br

Arlindo F. da Conceição
Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
arlindo.conceicao@unifesp.br

ABSTRACT
This work explores the technologies used to develop multimedia application for smartphones and evaluates, emphirically, the multimedia comunication capacity of multimedia data in smartphones over IEEE 802.11 wireless networks.

RESUMO
Este trabalho explora as principais tecnologias usadas no desenvolvimento de aplicações multimídia para *smartphones* e avalia, empiricamente, a capacidade de comunicação multimídia destes dispositivos em redes sem fio IEEE 802.11.

Categories and Subject Descriptors
I.6 [**Simulation and Modelings**]: Applications; C.2.3 [**Network Operations**]: Network monitoring

General Terms
Measurement, Verification

Keywords
Smartphones, multimídia, IEEE 802.11

1. INTRODUÇÃO
Os *smartphones* são dispositivos móveis sofisticados que oferecem uma ferramenta multifuncional, normalmente utilizada para comunicação e organização pessoal e para a realização de pequenas tarefas cotidianas. Gradualmente, o custo destes aparelhos tem caído e, ao mesmo tempo, novas funcionalidades estão sendo adicionadas a eles. Itens como GPS, telas sensíveis ao toque, acelerômetros, câmeras, redes sem fio IEEE 802.11 (Wi-Fi) e Bluetooth são algumas das tecnologias presentes nessas ferramentas.

Em particular, acreditamos que a capacidade de captura, processamento, armazenamento e de transmissão de dados multimídia seja uma funcionalidade indispensável nestes dispositivos. Mas, afinal, qual é a capacidade efetiva de comunicação multimídia de um *smartphone*?

Para responder essa pergunta, mapeamos as principais características multimídia de *smartphones*, tais como interfaces de programação e formatos suportados. Além disso, avaliamos experimentalmente os limites efetivos de comunicação UDP (não orientado a conexão) e TCP (orientado a conexão) utilizando *smartphones* em redes IEEE 802.11.

O restante deste trabalho está organizado da seguinte maneira: a Seção 2 avalia os trabalhos relacionados, a Seção 3 contém um breve mapeamento dos principais formatos e APIs para desenvolvimento de aplicações multímidia para *smartphones* e descreve uma aplicação em que utilizamos essas interfaces. A Seção 4 descreve a avaliação da capacidade de comunicação sobre redes IEEE 802.11. Por fim, a Seção 5 enumera os principais desafios enfrentados e as considerações finais.

2. TRABALHOS RELACIONADOS
Em geral, os trabalhos relacionados se concentraram apenas em um aspecto, ora aplicações, ora multimídia, ora redes. Em nenhum trabalho encontramos a conjunção de avaliação empírica de multimídia sobre redes IEEE 802.11 em *smartphones*. Apesar de não dispor de trabalhos fortemente relacionados, contamos com amplo referencial teórico sobre o tema. O que inclui: leituras sobre Java ME [13], Android [14] e APIs de programação [11], leituras sobre redes [16, 7] (sendo algumas sobre padrões de uso [15]) e trabalhos sobre caracterização de tráfegos em redes IEEE 802.11 [5, 2] e de ambientes para computação móvel [8].

3. MULTIMÍDIA EM SMARTPHONES
Os *smartphones* são dispositivos móveis sofisticados que geralmente possuem a capacidade de manipular elementos multimídia, tais como áudio, vídeo e imagens. Os desenvolvedores de aplicativos para estes dispositivos podem criar, para os mais diversos fins, softwares que façam uso dessas capacidades. Os limites para as aplicações, no entanto, estão fortemente associados ao dispositivo e a tecnologia utilizada. Três plataformas merecem destaque: iPhone, Android e Java ME.

Outro aspecto relevante, são os formatos suportados por cada dispositivo. As APIs de programação multimídia da J2ME seguem os padrões estabelecidos pelo MIME *Media*

*Evaluation of the multimedia capabilities of smartphones

Types [10], dessa forma, devem ser definidos no código-fonte segundo o suporte do dispositivo e de acordo com formato para o padrão MIME suportado. A exemplo, os tipos mais comuns de dados multimídia são mostrados na Tabela 1.

Formato	Tipo	*MIME Type*
wav	áudio	audio/x-wav
au	áudio	audio/basic
mp3	áudio	audio/mpeg
png	imagem	image/png
jpeg	imagem	image/jpeg
bmp	imagem	image/bmp
mpeg	vídeo	video/mpeg
avi	vídeo	video/avi
mov	vídeo	video/mov

Tabela 1: Tipos mais comuns de dados multimídia

A linguagem de programação J2ME [13] foi projetada para funcionar em dispositivos com recursos de memória, processamento e bateria limitados. Esta tecnologia funciona em uma ampla variedade de dispositivos e plataformas e oferece bom suporte à manipulação de multimídia (áudio, vídeo e imagem). J2ME oferece duas APIs para manipulação multimídia, são elas: a *Multimedia API* (MMAPI) [11] e a *Advanced Multimedia Supplements* (AMMS) [12].

3.1 AMMS

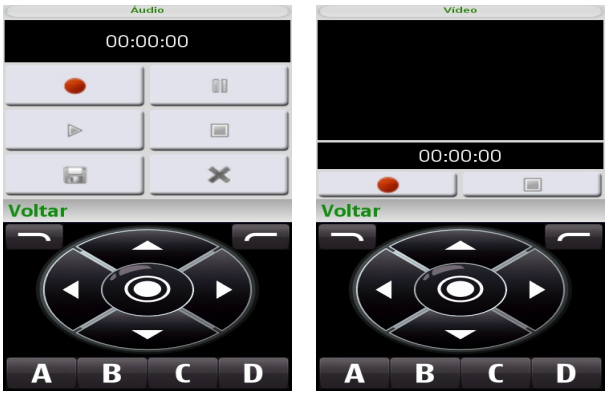
Com a evolução contínua das tecnologias dos dispositivos móveis — que a cada dia oferecem maior capacidade de processamento e de armazenamento de dados — novas possibilidades para manipulação de conteúdo multimídia foram disponibilizadas aos desenvolvedores de aplicações através do pacote opcional AMMS [12], que implementa técnicas avançadas de manipulação de conteúdo multimídia. Uma de suas funcionalidades mais interessantes é a capacidade de executar concorrentemente mais de um *Player*; por exemplo, é possível agrupar os *Players* e fazê-los interagirem entre si, criando, assim, efeitos de som em 3D. Com o uso de AMMS também é possível aprimorar imagens e vídeos, abrindo, desse modo, um novo leque de possibilidades para o desenvolvimento multimídia em dispositivos móveis.

Na prática, poucos dispositivos móveis oferecem suporte a AMMS. De maneira geral, apenas os *smartphones* da Nokia oferecem essa funcionalidade.

3.2 MMAPI

MMAPI é um pacote opcional de J2ME criado e mantido por uma consórcio de empresas (Nokia, France Telecom, Motorola, Philips, Sun Microsystems, entre outras) com o intuito de facilitar o desenvolvimento de aplicações multimídia e a utilização de recursos avançados de multimídia em aparelhos que possuem a tecnologia Java [9]. Fornece os controles básicos para o gerenciamento de mídias e, além disso, oferece suporte para diferentes protocolos e formatos de arquivo.

A Figura 1 mostra interfaces para captura de áudio e vídeo, escritas usando MMAPI e LWUIT[1] e executadas a partir de um celular Nokia N97. A aplicação foi desenvolvida no contexto do Projeto Borboleta (http://ccsl.ime.usp.br/borboleta), uma aplicação móvel para atendimento domiciliar de saúde [3], tornando possível a gravação de áudio, imagem e vídeo dos pacientes. Os dados coletados são gravados localmente no celular, para isso, utiliza-se o mecanismo de *Record Store* de Java ME. Posteriormente, os dados são transmitidos sobre IEEE 802.11 para um servidor central por meio de uma requisição *post* a um *webservice*.



(a) Interface para captura de áudio

(b) Interface para captura de vídeo

Figura 1: Interfaces gráficas Java ME/LWUIT para captura de dados multimídia.

3.3 A plataforma Android

A plataforma *Android* foi lançada em Novembro de 2007 pela *Google* na *Open Handset Alliance* (Aliança de Telefonia Móvel Aberta), em conjunto com a primeira versão beta do SDK (*Kit* de Desenvolvimento de Software) para a plataforma. Segundo [14], o *Android* tem o potencial para remover barreiras para o sucesso no desenvolvimento e venda de uma nova geração de softwares aplicativos de telefonia móvel. No campo do desenvolvimento multimídia, o *Android* oferece de forma bastante simples um grande suporte à manipulação e integração de áudio, vídeo e imagens pelas aplicações, permitindo a codificação/decodificação destes nos formatos mais comuns [1]. Além do suporte à multimídia, o *Android* também é capaz de fazer uso de tela sensível ao toque, acelerômetros, GPS e aceleração de gráficos 3D.

4. AVALIAÇÃO EMPÍRICA DA CAPACIDADE EFETIVA DE COMUNICAÇÃO EM REDES SEM FIO

Sabe-se que a capacidade efetiva de um protocolo é inferior a taxa nominal de transmissão [2]. Em *smartphones*, esse problema aparentemente é agravado, pois as políticas de economia de energia e a menor capacidade de processamento afetam negativamente o desempenho.

[1] *Lightwieght User Interface Toolkit* (LWUIT) é uma biblioteca para Java ME, lançada em 2009 pela *Sun Microsystems*, que permite a expressão de interfaces avançadas. Incorpora novos objetos gráficos, por exemplo, botões animados, e melhora a padronização gráficas das interfaces, isto é, as interfaces se mostram exatamente iguais, mesmo em celulares diferentes.

Desse modo, a fim de avaliar a capacidade efetiva destes dispositivos, realizamos experimentos que consistem em geração de tráfego sintético de *streaming* entre um *smartphone* e um PC. Avaliamos, tanto para UDP quanto para TCP:

- ***Throughput* máximo**. Taxa efetiva, em Mbps, transmitida através da rede IEEE 802.11. Este é um dos fatores de maior impacto em aplicações, por exemplo, de videoconferência.
- ***Round Trip Time*** (RTT). A latência bi-direcional ou o tempo, em ms, para um pacote ir do cliente para o servidor e voltar ao cliente.

Para realização dos experimentos, foi adaptado para dispositivos móveis (Java ME) um gerador de fluxos contínuos [4] escrito em Java SE. O gerador implementa o modelo cliente/servidor, onde o servidor envia fluxos contínuos de pacotes UDP ou TCP para as unidades móveis. Em suma, o cliente envia uma requisição para o servidor, a requisição contém o tipo de conexão (TCP ou UDP) e o número de pacotes e o tamanho de cada pacote (em bytes) a serem enviados. Para conexões UDP, na requisição também é informado o intervalo entre pacotes, em milissegundos.

Em nosso ambiente experimental, o servidor utilizado foi um Pentium 4 HT 3 GHz, com 512 Mb de Memoria RAM, sobre a plataforma Windows 2003 Server, rodando a aplicação Java através do GlassFish 3.0. O roteador sem fio utilizado foi um DLink DIR-635. A maioria dos testes foram realizados utilizando *smartphones* Nokia N95, Nokia N97 e HTC Touch.

4.1 Throughput máximo sobre TCP

O *throughput* máximo, ou taxa máxima de transferência, é obtido dividindo-se quantidade de bits transmitidos (8 * Nº de Pacotes * Nº de Bytes de Carga Útil) pelo tempo total da transferência; utilizamos diferentes combinações de tamanhos de pacotes e números de pacotes. Os resultados apresentados foram obtidos utilizando um aparelho Nokia N95.

A Tabela 2 mostra que *throughput* máximo no ambiente avaliado é da ordem de 1,3 Mbps. Os resultados foram sempre muito estáveis, tal que a diferença entre o mínimo (1,226 Mbps) e o máximo (1,363 Mbps) é de apenas 10%. Consolidado os testes não notamos variação significativa no *throughput* máximo em função dos tamanhos dos pacotes utilizados.

Cabe observar que esse valor pode variar significativamente em função do *smartphone* utilizado.

4.2 Throughput máximo sobre UDP

O *throughput* máximo em UDP, assim como para TCP, é calculado em função da quantidade de bits efetivamente recebidos dividida pelo tempo total da transferência. Os experimentos foram implementados a fim de transmitir uma quantidade fixa de dados (1 Mbyte), variando o tamanho dos pacotes (entre 100 e 2100 bytes) e o tempo de intervalo entre dois pacotes (entre 5ms e 21ms).

Nº de Pacotes	**Tamanho do Pacote (*Byte*)**	***Throughput* (kbps) N95**
89	2200	1323
98	2000	1344
109	1800	1310
123	1600	1270
141	1400	1363
165	1200	1305
199	1000	1334
200	2000	1258
222	1800	1235
249	800	1254
250	1600	1234
285	1400	1342
333	600	1311
333	1200	1335
400	1000	1287
500	400	1288
666	600	1271
1000	400	1226
2000	200	1270

Tabela 2: Throughput TCP para N95

Em nosso ambiente experimental, o valor máximo de *throughput* UDP efetivo foi de aproximadamente 3,3 Mbps. Observamos uma anomalia, os resultados apresentavam variação significativa quando eram utilizados pacotes com tamanho entre 1500 e 1700 bytes.

A primeira hipótese para esse comportamento foi a fragmentação de pacotes, *threshold* do IEEE 802.11, cujo valor *default* é de 2346 bytes, tanto o roteador quanto o dispositivo móvel estavam configurados com valores corretos, descartando assim essa possibilidade. Outra hipótese é o congestionamento das filas de comunicação gerado pelo MTU (*Maximun Transmission Unit*) da camada Ethernet, cujo valor é de 1500 bytes; mas não obtivemos resultados conclusivos a este respeito. No entanto, recomendamos, a fim de evitar essas instabilidades, que as transmissões UDP sejam realizadas utilizando-se pacotes com cerca de 1400 bytes.

4.3 Round Trip Time

Round Trip Time(RTT) é o tempo gasto para um pacote ser enviado para o servidor e voltar para o cliente. O tempo é medido em ms, tanto para TCP quanto para UDP. Os experimentos mostram o resultado médio, em milissegundos, de 100 execuções de trocas de pacotes para cada tamanho de pacotes. Na computação dos resultados médios, foram descartados os 3 maiores valores. Também é apresentado o desvio padrão.

Na Figura 2 pode-se verificar que o RTT médio para o dispositivo N95 é da ordem de 20ms, enquanto para o HTC é da ordem de 100ms. Não esperávamos uma diferença de desempenho tão acentuada. Além disso, não se observou variação significativa no RTT em função do tamanho dos pacotes utilizados, induzindo a hipótese de que estes tempos de comunicação não são dominados pela transmissão, mas sim por outros fatores, tais como processamento, sincronização e tempos de *backoff* .

A Figura 3 apresenta a avaliação de RTT sobre o protocolo

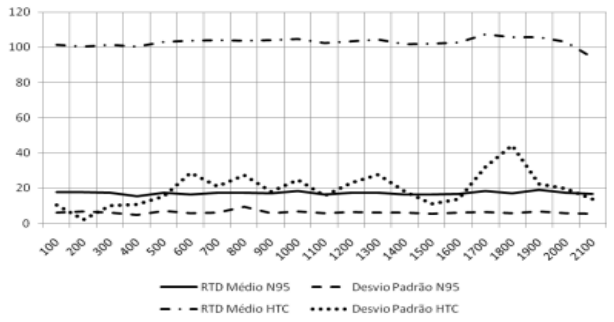


Figura 2: Round Tip Time utilizando protocolo TCP

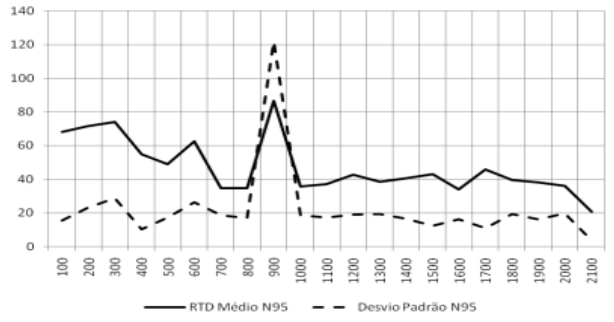


Figura 3: Round Trip Time utilizando protocolo UDP

UDP[2]. Cabe observar que o RTT UDP médio, no N95, é aproximadamente duas vezes maior do que o RTT TCP. A priori, não esperávamos este resultado; provavelmente isso ocorre por uma escolha de projeto, que prioriza comunicações TCP, que são mais frequentes, em detrimento das comunicações UDP.

5. CONCLUSÕES

Esta avaliação experimental buscou refinar o conhecimento sobre a real capacidade multimídia de *smartphones*. Os principais aspectos avaliados foram os de *throughput* e de latência. Além disso, puderam ser observados comportamentos inesperados (anomalias), dificuldades para a implementação de aplicações multimídia e a significativa diferença de comportamento entre dispositivos. Esta última, infelizmente, torna a atividade de desenvolvimento de aplicações multimídia para *smartphones* uma atividade fortemente dependente do ambiente utilizado.

Sobre os procedimentos metodológicos adotados, devemos destacar que não havia outro meio de detectar e aprofundar o nosso conhecimento senão pela via empírica. Sem a implementação real e a exaustiva experimentação, jamais constataríamos, por exemplo, as anomalias.

Portanto, concluímos que, em nosso ambiente experimental (que apesar de particular é bastante comum de ser encontrado), as taxas de transferência máximas são de aproximadamente 3,2 e 1,2 Mbits, para UDP e TCP, respectivamente. Verificamos ainda que a latência encontra-se entre 20 e 100 ms. Estas medições indicam que os *smartphones*, sobre redes IEEE 802.11, podem ser utilizados para transmissão de vídeo de baixa qualidade (aproximadamente 1 Mbps), mas que dificilmente podem ser utilizados para aplicações com requisitos fortes de tempo real, tais como aplicações de voz sobre IP [5, 6].

No futuro, desejamos refinar os experimentos e avaliar a capacidade de comunicação na presença de tráfegos concorrentes e analisar como os traços de execução são afetados pela mobilidade. Além disso, desejamos encontrar justificativas para as anomalias observadas.

[2]A figura mostra apenas resultados para o N95, pois o dispositivo HTC (Windows Mobile), apresentou um *bug* em sua máquina virtual java (JBED) que não permitiu as medições UDP.

6. REFERÊNCIAS

[1] Android Developers. Audio and video. Available in: `http://developer.android.com/guide/topics/media/index.html`, 2010.

[2] MG Arranz, R. Aguero, L. Munoz, and P. Mahonen. Behavior of UDP-based applications over IEEE 802.11 wirelessnetworks. In *Personal, Indoor and Mobile Radio Communications, 2001 12th IEEE International Symposium on*, volume 2, 2001.

[3] Rafael Correia, Fabio Kon, and Rubens Kon. Borboleta: A Mobile Telehealth System for Primary Homecare. In *ACM Symposium on Applied Computing*, 2008.

[4] Arlindo F. da Conceição. *Transmissão de voz e vídeo sobre redes IEEE 802.11*. PhD thesis, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo-SP, Brazil, May 2006.

[5] Arlindo F. da Conceição, Jin Li, Dinei A. Florêncio, and Fabio Kon. Is IEEE 802.11 ready for VoIP? In *8th International Workshop on Multimedia Signal Processing (IEEE MMSP)*, Victoria, Canada, October 2006.

[6] Arlindo F. da Conceição, Jin Li, Dinei A. Florêncio, and Fabio Kon. Transmissão de voz sobre redes IEEE 802.11: um levantamento dos principais problemas e restrições. In *12nd Brazilian Symposium on Multimedia and the Web (WebMedia)*, Natal-RN, Brazil, November 2006.

[7] F. de Moraes Jardim. *Treinamento Avançado em Redes Wireless*. Universo dos Livros Editora LTDA.

[8] L. Farkas and K. Aczel. Streaming video from smart phones: A feasibility study. In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008*, pages 1–6, 2008.

[9] V. Goyal. *Pro Java ME MMAPI*. Apress, 2006.

[10] Internet Assigned Numbers Authority. Mime media types. Available in: `http://www.iana.org/assignments/media-types/`, 2010.

[11] Java Community. Jsr 135: Mobile media api. Available in: `http://jcp.org/en/jsr/detail?id=135`, 2006.

[12] Java Community. Jsr 234: Advanced multimedia supplements. Available in: `http://jcp.org/en/jsr/detail?id=234`, 2006.

[13] John W. Muchow. *Core J2ME: Tecnologia and MIDP*. Pearson Makron Books, 2004.

[14] Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. *Desenvolvimento de aplicações Android*. O'Reilly, 2009.

[15] P. Smyth. *Mobile and wireless communications: key technologies and future applications*. Peter Peregrinus Ltd, 2004.

[16] W.R. Stevens, B. Fenner, and AM Rudoff. *Programação de Rede UNIX: API para soquetes de rede*. 2005.