Estudo Comparativo da Comunicação de Dados em Dispositivos Móveis: Métodos *Web services* e *Sockets*

Tiago dos Santos Marini[1], José Antônio Oliveira de Figueiredo[1], Anubis Graciela de Moraes Rossetto[1]

[1]Grupo de Pesquisa em Computação Aplicada – Instituto Federal Sul-rio-grandense (IFSul) - Campus Passo Fundo
Perimetral Leste 150, Lot. São Cristóvão II – Passo Fundo – RS – Brasil

tsm.marini@gmail.com, jose.figueiredo@passofundo.ifsul.edu.br, anubis.rosseto@passofundo.ifsul.edu.br

Abstract. *This work proposes a comparative test among the techniques of network communication referred to as Sockets and Web services. For this comparison, we designed a simple client-server application, first using network communication Socket method as basis; Afterwards, this same application had been rewritten using the Web service method; finally, we implemented captures of communications in both methods. This allowed us to make some observations and considerations about the behavior of each method of network communication on mobile devices.*

Resumo. *Este trabalho propõe um teste comparativo entre as técnicas de comunicação em rede conhecidas como Sockets e Web services. Para esta comparação, uma simples aplicação cliente-servidor foi projetada, usando como base de comunicação de rede o método Socket; esta mesma aplicação foi reescrita usando como base o método Web services; a seguir foram executadas capturas das comunicações feitas nos dois métodos. Isto nos permitiu fazer algumas observações e considerações, registradas a seguir, sobre o comportamento de cada método de comunicação de rede em dispositivos móveis.*

1. Introdução

Conforme o IDC (2013), neste ano, o número de *smartphones* embarcados superou o de celulares convencionais no primeiro trimestre, com uma fatia de 52,2% do mercado mundial de telefones móveis. A empresa afirma ainda, que esta tendência continuará nos próximos anos devido a forte demanda por dados móveis e computação de mão, evidenciado que o consumo por aplicações em rede seguirá a mesma tendência.

Para o desenvolvimento de aplicações móveis em rede, destacam-se os métodos de comunicação *Web services* e *Sockets* que, apesar de diferentes quanto a forma de trabalho e técnicas de programação, apresentam resultados finais muito próximos. Neste trabalho buscamos conhecer e testar, em um estudo de caso, as diferenças entre estes dois métodos de comunicação. Pretendemos desta forma, oferecer ao desenvolvedor de aplicações móveis, que tenha dúvidas sobre qual método usar, alguns parâmetros que possam auxiliá-lo na escolha.

A respeito de trabalhos relacionados, foram encontrados na literatura, estudos que avaliam as tecnologias de maneira independente, como por exemplo, em Moro, Dorneles e Rebonatto (2011), onde são analisadas as diferenças entre os métodos *Web services* REST e WS-*; desta forma, não foram identificados trabalhos de comparativo entre as duas tecnologias, *Web services* e *Sockets*, conforme proposto neste trabalho.

O trabalho inicia com uma pequena revisão bibliográfica sobre as tecnologias; apresentando a seguir o método aplicado para buscar visualizar estas diferenças e finaliza fazendo uma análise dos resultados obtidos, apresentando algumas conclusões.

2. Tecnologias de comunicação

2.1. *Web services*

Web services são aplicações que aceitam solicitações de outros sistemas através da Internet. Moro, Dorneles e Rebonatto (2011) explicam que esta forma de construção de aplicações de rede permite que aplicações interajam entre si e que sistemas desenvolvidos em plataformas diferentes se tornem compatíveis.

Na sua interface, o *Web service* adota padrões de comunicação da Internet, podendo trabalhar com diversos protocolos como HTTP, SMTP, FTP ou ainda outros protocolos proprietários. Os dois principais padrões adotados em *Web services* são *RESTful* e SOAP (*Simple Object Access Protocol).* Conforme Moro, Dorneles e Rebonatto (2011) SOAP é um padrão de comunicação em que a troca de mensagens ocorre no formato XML sobre o protocolo HTTP; é um padrão consolidado e ainda é largamente utilizado. Devido a esta última característica, neste trabalho optamos por abordar nos testes apenas o padrão SOAP.

2.2. *Socket*

Kurose e Ross (2010) definem *socket* como sendo um mecanismo de comunicação inter-processos. Considerando o modelo de comunicação TCP/IP, este é o mecanismo que faz a interface entre a camada de aplicação e a camada de transporte, levando os fluxos da comunicação de um processo ao outro através da rede. Toda comunicação entre os processo[1] é feita enviando-se mensagens através deste "canal" de comunicação, conforme representado na figura 1.

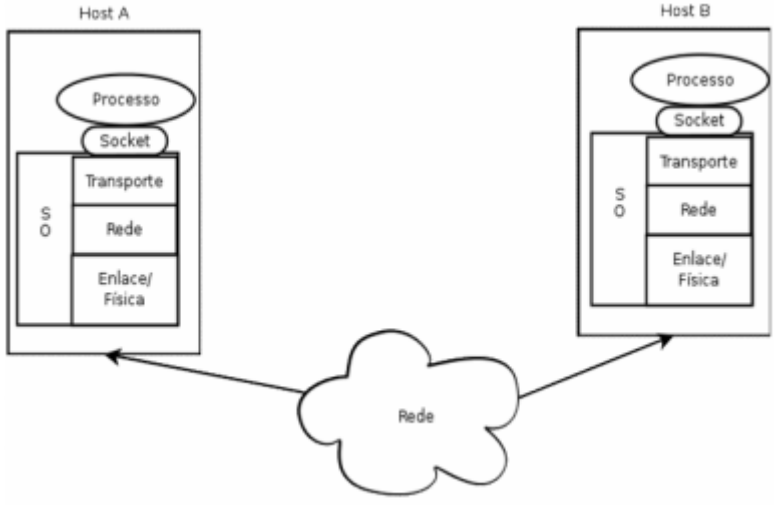

Figura 8. Comunicação por *sockets*

O *socket* também é conhecido por API, *Application Program Interface,* de programação de rede. Esta definição, ainda conforme Kurose e Ross (2010), existe porque o "*socket* é a interface de programação pela qual as aplicações de rede são

1 "Um programa que está rodando dentro de um sistema final" Kurose e Ross (2010).

inseridas na Internet“. Esta tecnologia de programação de redes é bastante antiga e conhecida, sendo frequentemente utilizada em aplicações de rede de porte considerável.

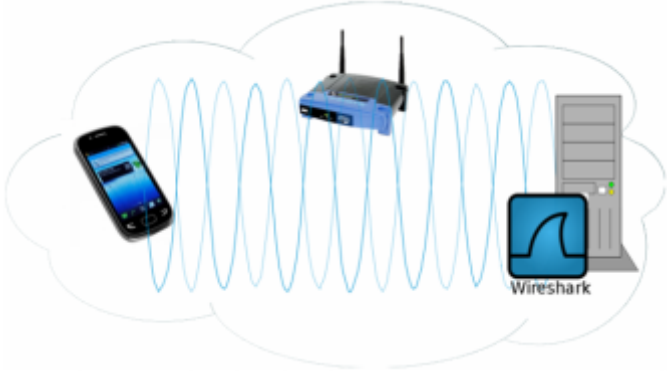

Figura 9. Fluxo de comunicação entre componentes do cenário.

3. Materiais e métodos

Para visualizar as diferenças de comportamento entre os métodos de comunicação *Web services* e *Sockets*, optou-se por comparar o volume de pacotes trafegados e o tempo de resposta de uma aplicação desenvolvida nos dois métodos.

Para isto, uma simples aplicação cliente-servidor foi desenvolvida, onde o cliente roda em um dispositivo móvel (Android) e gera uma quantidade de números aleatórios; esta quantidade é definida pelo usuário no início do teste. Estes números são enviados então para o servidor que faz um simples cálculo de média aritmética entre os números gerados e devolve o resultado ao cliente que solicitou a operação. Estes servidores (*Web service* e *Socket*) foram desenvolvidos em Java e servidor *Socket* foi desenvolvido usando protocolo de transporte TCP.

Destacamos que aspectos não mensuráveis como facilidade de programação, quantidade de aplicativos desenvolvidos no método e outros não foram considerados neste trabalho.

3.1. Cenário de testes

O ambiente de testes foi idealizado buscando fazer a coleta de dados de forma exclusiva, evitando interferências de outras comunicações em rede ou mesmo de outros aplicativos. Foram usados um computador, um *smartphone* e um *Access Point*. No computador e no *smartphone*, foram desativados serviços desnecessários e que pudessem interferir nos testes; enquanto que o *Access Point* foi configurado para atender apenas a comunicação executada entre o computador e o *smartphone*. A captura de dados foi feita no próprio computador, com a ferramenta de captura de pacotes *Wireshark*[2]. A figura 2 procura demonstrar este cenário, enquanto a tabela 1 detalha a configuração utilizada em cada um dos dispositivos.

2 Wireshark é um analisador de protocolos de rede. [www.wireshark.org].

Tabela 2 Componentes do ambiente de testes.

Componente	Configuração
Computador	Ultrabook, Dell, Inspirion 14z com: Processador Intel i5 1,7GHz, 4GB de memória, sistema operacional Windows 7 (Home Premium 64bits).
Smartphone	Motorola Razr MAXX; Plataforma Android 2.3.5, com 1GB de RAM; um processador Dual Core 1,2GHz
Access Point	Wireless 802.11g, 2.4GHz

3.2. Coletando os dados

A coleta de dados foi feita da seguinte forma: definiu-se que seriam observados os pontos de 1.000, 10.000, 100.000, 200.000 e 1.000.000 de envios de números aleatórios do cliente móvel para o servidor; para cada ponto foram executadas 7 leituras para análise de volume de tráfego e tempo de resposta. Estes testes foram feitos em cada método de comunicação avaliado. Das coletas feitas, em cada ponto observado e em cada método, foram descartadas as leituras de valores extremos; esta medida foi adotada buscando-se, uma maior precisão nos dados obtidos.

A coleta de dados, para volume de tráfego, foi executada com a ferramenta Wireshark, que oferece mecanismos para análise de volume tráfego e filtros para visualização do que se busca. A tabela 2 apresenta o volume de pacotes gerados por cada um dos métodos em cada um dos testes. Na tabela, já estão descartadas as leituras de maior e menor valor.

Tabela 3: Tráfego gerado em relação ao número de dados enviados e ao tipo de comunicação.

Método	**Envio**	**1.000**	**10.000**	**100.0000**	**200.000**	**1.000.000**
WS	1	20	64	521	1039	*
	2	19	65	513	1030	*
	3	20	64	511	1030	*
	4	20	65	526	1030	*
	5	20	65	529	1030	*
	Média	**20**	**65**	**520**	**1032**	*
Socket	1	17	77	647	1294	6532
	2	14	77	658	1229	6387
	3	15	77	653	1294	6413
	4	17	74	640	1308	6543
	5	17	78	639	1341	6580
	Média	**16**	**77**	**647**	**1293**	**6491**

A obtenção do tempo consumido por cada método de comunicação foi feita com um simples contador de tempo na aplicação móvel; este contador registra o tempo entre o envio da solicitação e a devolução da resposta. Da mesma forma que no teste anterior, foram descartadas as leituras de maior e menor valor. Esta coleta de dados é mostrada na tabela 3 e os valores estão em milissegundos.

Tabela 4: Tempo em relação ao número de dados enviados e o tipo de comunicação.

Método	Envio	1.000	10.000	100.0000	200.000	1.000.000
WS	1	834	2158	13720	40021	*
	2	537	2381	12797	39939	*
	3	882	2375	11452	39876	*
	4	533	2152	13511	40012	*
	5	454	2364	11146	39734	*
	Média	**648**	**2286**	**12525,2**	**39916,4**	*
Socket	1	592	1093	6002	17489	82988
	2	424	1194	6209	16932	82368
	3	870	1192	4989	18515	88368
	4	874	1685	5864	17935	87466
	5	529	1033	5145	17350	83904
	Média	**657,8**	**1239,4**	**5641,8**	**17644,2**	**85018,8**

4. Análise dos resultados

O primeiro ponto a ser destacado está no teste executado com 1.000.000 de números gerados para envio com o método baseado em *Web services*, onde em todas as tentativas, houve estouro da capacidade de memória no dispositivo móvel, ou seja, o aplicativo entrou em erro e não conseguiu fazer o envio. Para este volume de números, a comunicação funcionou corretamente apenas sob o método de comunicação *Socket*.

4.1. Volume de tráfego

Em um cenário de redes de dispositivos móveis, temos frequentemente, menor capacidade de energia disponível, enquanto utilizamos redes sem fio que consomem mais energia. Assim, fica evidente, que o melhor método será o que apresentar o menor volume de pacotes gerados. Pela coleta de dados feita, podemos observar que o método de comunicação *Socket* apresenta melhor eficiência apenas no teste feito com envio de 1.000 números; ficando atrás do método *Web service*, mesmo que por uma pequena diferença, nos testes com 10.000, 100.000 e 200.000.

Este comportamento é verificado no gráfico da figura 3, onde a linha vermelha representa o método *Socket* e a linha azul representa o *Web service*. No gráfico pode-se notar que o método *Web services* não completou o envio com 1.000.000 dados.

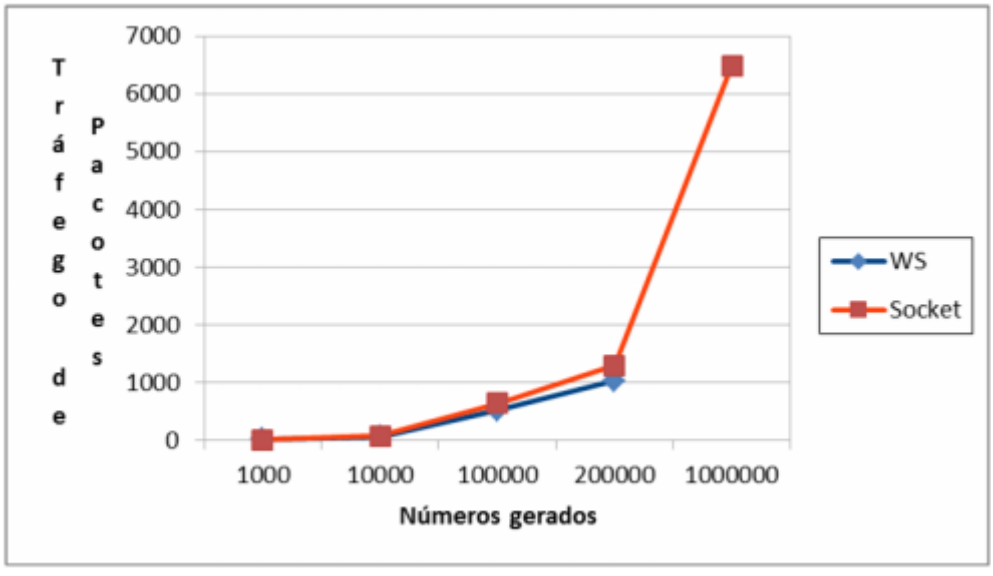

Figura 10. Gráfico representando volume de dados.

4.2. Tempo de resposta

Quando usamos um dispositivo móvel, a velocidade na resposta de uma solicitação é fundamental para que o usuário tenha uma boa experiência no uso de qualquer aplicação móvel em rede. Fica evidente então, que o melhor método será o que apresentar menor tempo de resposta (for mais rápido). Pelas medidas feitas, o método *Web service* é mais rápido apenas no teste com 1.000 números; nos testes com 10.000, 100.000 e 200.000 é o método *Socket* que apresenta menor tempo de resposta.

Este comportamento é verificado no gráfico da figura 4, onde a linha vermelha representa o método *Socket* e a linha azul representa o *Web service*. Novamente, o teste com 1.000.000 de números não ocorre com o método *Web service* pelo motivo explicado na seção 4.

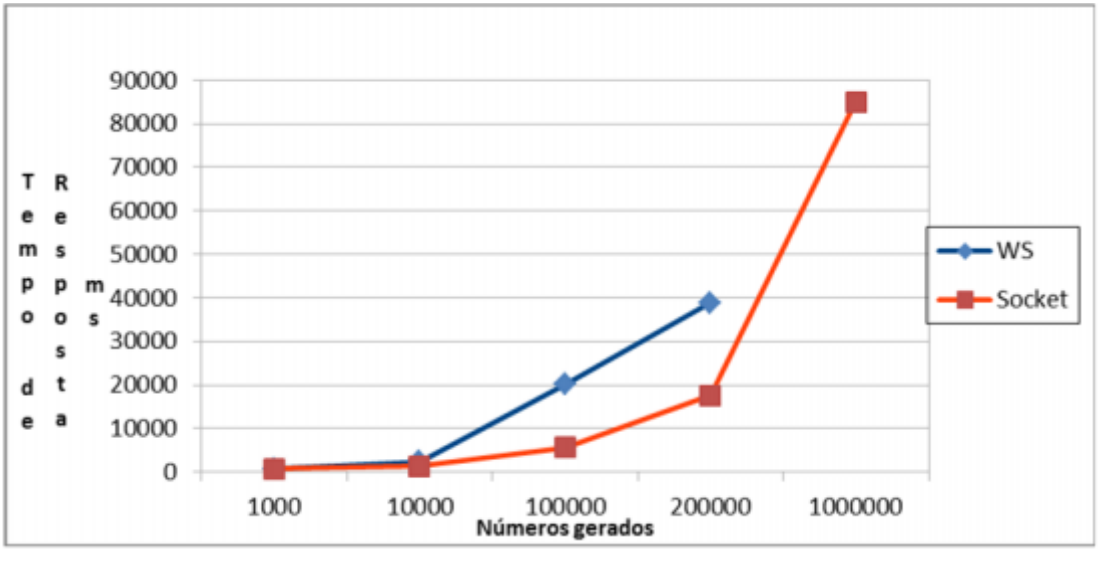

Figura 11. Gráfico representado o tempo e resposta dos métodos testados.

5. Considerações finais

Buscamos neste trabalho observar as diferenças de funcionamento, em nível de rede, entre os métodos de comunicação *Web services* e *Sockets*. Não pretendemos com isto

concluir se um é melhor do que outro, mesmo porque para isto teríamos que definir um conjunto muito maior de testes. Contudo, diante dos testes executados, podemos obter algumas conclusões interessantes.

A principal conclusão a ser destacada é que, para quantidades grandes de variáveis a serem transmitidas, o método de comunicação *Socket* consome menos memória do dispositivo móvel, ao contrário do método *Web service* que causou um erro na execução do aplicativo. Portanto, caso um desenvolvedor tenha o objetivo de enviar grandes quantidades de variáveis poderá preferir o uso de *Sockets*.

Os testes mostraram que o método *Web service* gerou menor quantidade de pacotes trafegados na rede, no entanto, fica claro que a diferença entre os métodos foi mínima, ou seja, ambos tem um comportamento de consumo de rede muito parecido, o que os torna equivalentes no aspecto volume de tráfego.

Nos testes de tempo de resposta, o método *Socket* é mais rápido em relação ao método *Web service*, na medida em que a quantidade de variáveis a serem transmitidas aumenta. Todavia, esta diferença está mais relacionada com o consumo de memória de execução do que com a rede propriamente dita, aspecto que poderá ser melhor avaliado em outros procedimentos de teste.

Referências

ALVES, Otávio, Pedro; SILVA S. Robson; Tecnologias De Sistemas Distribuídos Implementadas em Java: Sockets, RMI, RMU-IIOP e CORBA. Universidade para o Desenvolvimento do Estado e da Região do Pantanal, 2009.

COMER, Douglas. Interligações de Redes com TCP/IP: princípios, protocolos e arquitetura. 5a edição. 2006

COULOURIS, George; DOLLIMORE, Jean; KINDEMBERG, Tim. Sistemas Distribuídos: conceitos e projeto. 4a edição. Bookman, 2007.

KUROSE, James F.; ROSS, Keith W. Redes de Computadores e a Internet: uma abordagem top-down. 5ª edição. 2010.

MERGEN,Sérgio; GEYER, Cláudio; RAMOS, M. Emanuel: *Web services*: Conceitos, SOAP, WSDL, UDDI. Instituto de Informática- UFGRS, Julho, 2008.

MORO, Tharcis D.; DORNELES, Carina F.; REBONATTO, Marcelo T.: *Web services* WS- versus *Web services* REST. REIC - Revista de Iniciação Científica - UFRGS, 2011. Disponível em: <http://seer.ufrgs.br/reic/article/download/22140/12928>. Acesso em: 20/12/2012.

SMARTPHONES Expected to Grow 32.7% in 2013 Fueled By Declining Prices and Strong Emerging Market Demand, According to IDC Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS24143513> Acesso em: julho/2013.