UNIVERSIDADE FEEVALE

VANIUS ROBERTO BITTENCOURT

PROTOCOLO THINCLIENT PARA APLICAÇÃO COMERCIAL

(Título Provisório)

Anteprojeto de Trabalho de Conclusão

Novo Hamburgo
2011

VANIUS ROBERTO BITTENCOURT

PROTOCOLO THINCLIENT PARA APLICAÇÃO COMERCIAL

(Título Provisório)

Anteprojeto de Trabalho de Conclusão de Curso, apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

Orientador: Gabriel da Silva Simões

Novo Hamburgo
2011

RESUMO

Atualmente existem várias possibilidades para executar aplicativos através de uma camada *thinclient* independente de plataforma. A alternativa mais utilizada é a de desenvolvimento *web*, que resulta numa visualização em documento HTML através do protocolo HTTP. Esse formato inicialmente foi proposto para visualização de textos e não para executar aplicativos. Por isso não permite visualização em janelas sobrepostas tal como aplicativos *desktops*. Devido ao uso do protocolo HTTP o controle de estados deve ser feito através de *frameworks*. Outras soluções utilizadas são protocolos de tela remota tal como RDP e VNC, que consistem no tráfego de *pixels* do *desktop* do servidor. Para solucionar problemas em comum, este trabalho propõe uma especificação de protocolo para representação visual de aplicações gráficas em janelas. Esta especificação é constituída pelo tráfego de atributos de objetos visuais, através de um formato de fácil entendimento e implementação.

Palavras-chave: protocolo;cliente-servidor;*thinclient*;aplicação.

SUMÁRIO

MOTIVAÇÃO

Existem vários meios e modelos de execução de programas, dentre os quais, um dos mais utilizados é o cliente-servidor. Neste modelo o programa é executado em dois pontos através de uma rede de computadores. O lado do cliente é responsável pela interface com o usuário, enquanto que o lado do servidor é responsável pelo processamento computacional e armazenamento de dados (ACHARYA, 2006). A comunicação e transferência de dados entre os lados são definidas através de um protocolo, geralmente na camada de aplicação. O cliente pode ser uma parte fixa da aplicação como uma extensão ao servidor, formando a aplicação. Neste conceito, o cliente pode executar localmente processamento e regras de negócio, utilizando o servidor apenas como repositório de dados. Outra arquitetura para o modelo de cliente-servidor é o cliente apenas "representar" a execução da aplicação do servidor, de modo que este seja uma mera interface visual. Dentro deste contexto, o mesmo cliente pode representar e executar diferentes aplicações, já que o mesmo é contido na integra no servidor. Este conceito é tratado como *thinclient* (SOSINSKY, 2009).

Na atualidade existem dois modelos comuns de funcionamento de aplicativos cliente-servidor. No primeiro modelo, denominado "*desktop*", a aplicação é executada na estação, ou seja, a aceitação de dados e as regras de negócio são processadas no computador do usuário. Por esse motivo, este modelo também é denominado *fatclient*. O servidor é utilizado principalmente para repositório dos dados (BIDGOLI, 2004, p. 27). Geralmente sua interface gráfica consiste em janelas que são exibidas de forma sobreposta, permitindo manter várias janelas visualizadas simultaneamente. Nesse modelo há componentes visuais elegantes, tal como listas, tabelas, *treeview*, etc. Por ser uma aplicação executada localmente a camada cliente deve estar instalada na estação ou centralizada em sua rede privada. No caso de uma rede de longa distância, há um problema em manter os clientes atualizados. Para isso é necessário ocorrer o *download* do cliente no início da execução, que pode ser uma restrição, dependendo do tamanho da aplicação.

No outro modelo, as aplicações são inteiramente executadas do lado do servidor, o cliente apenas exibe a interface. Esta interface é visualizada através páginas HTML. Com isso, qualquer estação com navegador de internet funciona como cliente. O armazenamento e execução das regras de negócio são feitas no servidor. Esse modelo é denominado "aplicação *web*" (DOOLEY, 2011, p. 53). O formato de documento HTML foi inicialmente desenvolvido para representação de textos – nunca o seu objetivo foi representar aplicações.

Por isso, para desenvolver aplicações neste modelo, é necessário uma série de ferramentas e aplicativos adicionais. Sem auxílio deles seu desenvolvimento é complexo (MURUGESAN; DESHPANDE, 2001, p. 5). Por padrão, o HTML não permite visualização em janelas. Para navegar entre as telas (páginas) ocorre a recarga da interface. No servidor as aplicações podem ser desenvolvidas em diversas linguagens, mas a aplicação deve resultar em documentos HTML. Isso praticamente obriga o desenvolvedor *web*, além de dominar a linguagem da aplicação, dominar o formato HTML e prever o seu comportamento em diferentes navegadores. O protocolo HTTP é utilizado para fazer a transferência do documento HTML até o cliente. Este protocolo tem por característica não armazenar estado de conexão, pois é encerrada quando a transferência é concluída. Sendo assim, no lado do servidor, deve haver mecanismos para recuperar e armazenar os dados da sessão (MACDONALD, 2010, p. 258). A vantagem deste modelo é que funciona como um *thinclient*, sendo que não há preocupação em distribuir e atualizar os clientes. Praticamente todas as estações, de diferentes sistemas operacionais, possuem navegador de internet.

Na década de 80, quando surgiram os primeiros conceitos de telas gráficas, o MIT (Massachusetts Institute of Technology) criou um protocolo de comunicação entre terminais semelhante ao VT100, utilizado em terminais caracteres. O protocolo se denominava System Window X, ou simplesmente X (SCHEIFLER; GETTYS, 1986). Este protocolo foi proposto para ser independente de hardware e plataforma. Nos sistemas Unix este protocolo foi largamente adotado e é utilizado até hoje. Apesar de ser um protocolo para ser utilizado em uma arquitetura cliente-servidor, é praticamente um padrão de desenvolvimento de telas gráficas para aplicações *desktop*. Considerando a necessidade atual de soluções em aplicações gráficas na internet, seu uso é questionável. Este protocolo gera muitas interações entre o cliente e servidor, que consiste em trafegar definições detalhadas da tela. Com isso, este protocolo é praticamente inviável para ser utilizado em meios externos de alta latência, tal como a internet (MEERSMAN; TARI, 2005, p. 782).

Visualizando a dificuldade em desenvolver aplicações visuais para internet, a Adobe – desenvolvedora do Flash – criou uma ferramenta chamada Flex, onde o programador desenvolve a aplicação em linguagem Java. Esta aplicação é executada no servidor e sua visualização é feita através do cliente, utilizando o *plugin* Adobe Flash, que está instalado em quase todos navegadores de internet (PISA, 2009). Com isso o resultado é uma aplicação rica, semelhante a aplicações *desktop*, sem necessidade de domínio do formato HTML. Existem outras soluções para aplicações Java, tal como CaptainCasa, que pode tanto exibir a tela do

cliente em navegador quanto em janelas *desktop* (CAPTAINCASA, 2011). Assim como, estas existem outras soluções proprietárias, que aplicaram suas próprias soluções de servidor, cliente e protocolo de comunicação entre eles.

Na internet, muitas tecnologias e protocolos são padronizados, de forma aberta e livre. Para evitar o crescimento de tecnologias proprietárias o W3C está definindo modificações para padrão HTML, agora na versão 5 (PILGRIM, 2010). Está sendo definida característica do protocolo para visualizações ricas, tal como desenhos vetoriais e exibição de vídeo. O principal objetivo do HTML5 é ser uma alternativa ao Flash (HARRIS, 2011, p. 4). O HTML5 possui instruções básicas de desenho, mas não necessariamente para exibir janelas semelhantes às aplicações *desktop*. Para ter esse recurso deve ser desenvolvida uma aplicação que roda sobre o HTML5. A nova versão do HTML não terá recursos prontos para exibição de aplicações tal como o Adobe Flex.

Para contornar o problema de acessar aplicações desktop à distância, surgiram alguns protocolos para permitir acessar a tela do computador remotamente, tal como RDP (Remote Desktop Protocol) e VNC (Virtual Network Computing). Estes protocolos consistem em mapear a área da tela *desktop pixel* a *pixel*, fazendo a transferência da área modificada (CRAFT; BROOMES; KHNASER, 2002, p. 155). Então, se uma janela é movimentada ou minimizada, ocorre a tráfego da área, pois não há armazenamento das janelas em segundo plano. Por representar a tela do usuário, na prática, ocorre a exibição integral da aplicação no servidor, consumindo recursos para cada usuário conectado. Se houver um número elevado de usuários será necessário um servidor bastante robusto (HARWOOD, 2002, p. 159).

Atualmente, são perceptíveis as mudanças das tendências sobre desenvolvimento de aplicações. Percebe-se uma busca pelo resgate dos aspectos centralizados e leves adotados pelos terminais burros (MORIMOTO; KOVACH; ROBERTS, 2003, p. 483). Na internet existem diferentes protocolos de comunicação, mas não há um protocolo difundido que permita a execução de aplicações remotas com aparência em janelas. Igualmente não é encontrado protocolo que realiza o tráfego de atributos de objetos visuais ao invés do tráfego de *pixels*, executada por um cliente-leve independente de plataforma e sistema operacional.

Estra trabalho tem por objetivo criar uma especificação de um protocolo para representação visual de aplicações, visando resolver muitos dos problemas citados acima. A partir das possibilidades atuais é possível extrair um modelo e propor uma normalização. Para auxiliar sua aplicação será adotado um formato de padrão aberto e difundido, de melhor compreensão humana. Será possível desenvolver aplicações centralizadas em servidor, sendo

que para o usuário será exibida somente a interface visual. Essa interface poderá ser semelhante a aplicações *desktop* convencionais, fugindo das limitações de aplicações *web* com HTML. Depois de construídos componentes para interpretar o protocolo, o programador poderá focar seu esforço na regra de negócio da aplicação, e não com processo de execução da interface. Por ser um protocolo aberto, poderão ser construídos clientes para diversos ambientes para funcionar com qualquer aplicação. Da mesma forma do lado do servidor haverá a possibilidade de ser criado um servidor de aplicação que será responsável pelo tráfego do protocolo visual.

OBJETIVOS

Objetivo geral

Especificar protocolo para representação de interface gráfica de aplicativos comerciais

Objetivos específicos

Avaliar outros protocolos

Especificar formato das mensagens do protocolo

Definir sintaxe e significado de palavras reservadas

Definir regra de fluxo de mensagens do protocolo

Enumerar possibilidades de objetos e eventos

METODOLOGIA

Será pesquisado em livros e artigos sobre os atuais protocolos de aplicação cliente-servidor, para identificar benefícios e deficiências. Estudar casos de uso em que atuais protocolos possuem problemas. Pesquisar em bibliografia sobre as tecnologias que serão usadas, tal como TCP/IP e formato JSON, já que as mesmas auxiliarão para compor a especificação do protocolo.

Serão desenvolvidos protótipos de aplicação servidora e cliente, que se comunicarão através do protocolo proposto. Será elaborado cenário com caso de uso para o protocolo ser aplicado, onde será feita experimentação através do protótipo. Serão definidos aspectos que devem ser avaliados. Será registrada a comparação entre os comportamentos esperados com os resultados obtidos. Pretende-se observar um melhor desempenho de tráfego em comparação a outros protocolos. Para o protótipo deverá ser uma definida uma API em Java para abstrair a manipulação do protocolo. No lado do servidor, o protótipo deve executar uma aplicação Java que faz uso da API. Com isso espera-se comprovar a fácil utilização para compor aplicações *thinclients*.

CRONOGRAMA

Trabalho de Conclusão I

1) Pesquisar livros e artigos sobre os atuais protocolos de aplicação cliente-servidor
2) Estudar casos de uso em que atuais protocolos possuem problemas
3) Pesquisas em bibliografia sobre as tecnologias que serão usadas
4) Definir especificações e funcionamento do protocolo.
5) Redigir TC I

Etapa	Meses						
	Jun	Jul	Ago	Set	Out	Nov	Dez
1	X	X	X	X			
2			X	X			
3				X	X	X	X
4				X	X	X	X
5				X	X	X	

Trabalho de Conclusão II

1) Desenvolver protótipo de aplicação servidor e cliente
2) Elaborar caso de uso para o protocolo ser aplicado
3) Realizar experimentação e registrar resultado
4) Redigir TC II

Etapa	Meses						
	Jan	Fev	Mar	Abr	Mai	Jun	Jul
1	X	X	X	X			
2	X	X					
3		X	X	X	X		
4	X	X	X	X	X		

BIBLIOGRAFIA

ACHARYA, Vivek. **TCP/IP Distributed System.** New Delhi: Laxmi Publications, 2006, 474 p.

BIDGOLI, Hossein. **The Internet encyclopedia:** Volume 1. New Jersey: John Wiley And Sons, 2004, 880 p.

CAPTAINCASA GMBH. **CaptainCasa Enterprise is the Rich Internet Application** solution for Business Applications with demanding users. 2011. Disponível em: <http://www.captaincasa.com/>. Acesso em: 4 ago. 2011.

CRAFT, Melissa; BROOMES, Chris; KHNASER, Elias N.. **Configuring Citrix MetaFrame XP for Windows including future release 1.** New York: Syngress, 2002, 560 p.

DOOLEY, John. **Software Development and Professional Practice.** New York: Apress, 2011, 260 p.

FARLEY, Jim. **Java distributed computing**, O'Reilly, 1998, 392 p.

FREEMAN, Eric; FREEMAN, Elisabeth; SIERRA, Kathy; BATES, Bert. **Head First design patterns**, O'Reilly, 2004, 686 p.

HAROLD, Elliotte Rusty. **Java network programming**, O'Reilly, 3rd edition, 2004, 762 p.

HARRIS, Andy. **HTML5 for dummies quick reference**, John Wiley and Sons, 2011, 224 p.

HARWOOD, Ted. **Inside Citrix MetaFrame XP:**a system administrator's guide to Citrix MetaFrame XP/1.8 and Windows terminal serives. Indianapolis: Addison-wesley Professiona, 2002, 944 p.

MACDONALD, Matthew; FREEMAN, Adam; SZPUSZTA, Mario. **Pro ASP.NET 4.0 in C# 2010.** 4. ed. New York: Apress, 2010, 1616 p.

MEERSMAN, Robert; TARI, Zahir. **On the move to meaningful Internet systems 2005:** OTM confederated international conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 21-November 4, 2005 : proceedings, Parte 1. Birkhäuser: Springer. p. 781-782. 2005.

MILLER, Frederic P.; VANDOME, Agnes F.; MCBREWSTER, John. **JSON**, VDM Publishing House Ltd., 2009, 178 p.

MORIMOTO Rand; ABBATE Andrew; KOVACH Eric; ROBERTS Ed. **Microsoft Windows Server 2003 insider solutions**. Sams Publishing, 2003, 638 p.

MURUGESAN, San; DESHPANDE, Yogesh. **Web engineering:** managing diversity and complexity of Web application development. Berlin: Springer, 2001, 355 p.

PARALLAX, Inc, **Programming and customizing the multicore propeller microcontroller: the official guide**.McGraw-Hil, 2010, 496 p.

PILGRIM, Mark. **HTML5: Up and Running**. O'Reilly & Google Press, 2010, 205 p.

PISA, Filippo Di. **Beginning Java and Flex: Migrating Java, Spring, Hibernate and Maven Developers to Adobe Flex.** New York: Apress, 2009, 500 p.

SCHEIFLER, Robert; GETTYS, Jim, The X Window System, In **ACM Transactions on Graphics**, v.5, n.2, 1986, 109 p.

SOSINSKY, Barrie. **Networking Bible.** Indianapolis: John Wiley And Sons, 2009, 1056 p.