Adaptando o Protocolo HTTP ao Ambiente de Computação Móvel [1]

Daniel B. de Faria *Antonio Alfredo F. Loureiro*

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Caixa Postal 702, 30123–970 Belo Horizonte, MG
E-mail: {dbfaria, loureiro}@dcc.ufmg.br

Resumo

Protocolos de comunicação sempre foram projetados levando-se em consideração características do ambiente onde são executados. Com o advento da computação móvel, protocolos que foram especificados sob o paradigma das redes fixas passaram a apresentar uma série de deficiências. O protocolo HTTP, que adquiriu importância com o crescimento da *World Wide Web*, encontra-se neste grupo, utilizando de forma indevida os recursos disponíveis em ambiente móveis. Para adaptar o HTTP a este novo cenário, foram implementados os protocolos HTTP-m e MPP, que utilizam codificação de mensagens HTTP e algoritmos de compressão para melhorar a eficiência na utilização do enlace sem fio. Foram feitas simulações destes protocolos e os resultados obtidos se mostraram bastante promissores.

Abstract

The design of communication protocols has always been influenced by the environment where they are supposed to run. As wireless computing became an important research area, some of the existing protocols have shown some drawbacks. The main reason is that resources offered by the wireless medium are not as abundant as in fixed networks. HTTP has gained popularity with the increasing number of computers connected to the World Wide Web, and despite its bad behavior in mobile computing environments, it will probably continue to be used. This work describes the design of the HTTP-m and MPP protocols, which can be used to increase the efficiency of the HTTP protocol. These protocols use compression algorithms and a new method for coding HTTP messages to reduce the amount of data sent by the HTTP entities. The new protocols were simulated and their utilization was considered suitable for a large range of wireless communication systems.

1 Introdução

Desde seu aparecimento, a Internet vem mantendo um ritmo de crescimento bastante acelerado. Este crescimento, aliado à expansão da *World Wide Web* (WWW), fez com que se tenha hoje um grande tráfego de dados pela Internet, gerado em computadores distribuídos por todo o globo. Esta comunicação global dispõe hoje de novas ferramentas, sendo a computação móvel um exemplo das novas tecnologias empregadas.

Com o desenvolvimento das tecnologias de transmissão sem fio e a evolução dos computadores portáteis, a computação móvel aparece hoje como uma área de pesquisa bastante promissora, com inúmeros serviços já disponíveis no mercado. O estudo mostrado neste trabalho foi motivado pelo impacto da computação móvel sobre o padrão de funcionamento da *World Wide Web*.

Por ser baseada em tecnologias mais recentes que aquelas aplicadas às redes tradicionais de computadores, a computação móvel apresenta uma série de características peculiares. A maioria destas propriedades inerentes do ambiente móvel constituem fatores limitantes para a utilização de inúmeros protocolos de comunicação, especificados e projetados ainda sob o paradigma dos computadores fisicamente conectados às redes. Dentre estas características pode-se destacar as baixas velocidades de transmissão, as altas taxas de erros, a mobilidade

[1] Este trabalho foi parcialmente financiado pelo Projeto SIAM/DCC/UFMG, grant MCT/FINEP/PRONEX número 76.97.1016.00

do usuário e restrições de energia da unidade móvel, que fazem este ambiente pouco amigável à primeira vista. Isto torna necessária a evolução dos protocolos de comunicação, para que se adaptem melhor a este novo cenário de computação distribuída.

O protocolo utilizado para transferência de dados na *Web* é o HTTP (*Hypertext Transfer Protocol*) [6, 19]. O HTTP foi projetado com o intuito de ser um protocolo simples para transferência de dados pela Internet. Todavia, tanto a utilização quanto a complexidade deste protocolo aumentaram consideravelmente desde a sua primeira versão, sem que para isto a sua forma de funcionamento tenha sido alterada de maneira significativa. A *Web* constitui uma das grandes fontes de tráfego de dados, fazendo com que possíveis melhorias no HTTP tenham um impacto considerável sobre a Internet.

A forma como o protocolo HTTP define a troca de informações entre clientes e servidores é bastante simples. As mensagens HTTP, requisições dos clientes e as respostas enviadas pelos servidores, são compostas de mensagens textuais, transportadas sobre uma conexão TCP (*Transmission Control Protocol*) [16, 18]. Desta maneira também são transferidos os outros dados, como figuras e arquivos contendo páginas HTML (*Hypertext Markup Language*) [4]. Porém, as informações de controle definidas por este protocolo apresentam baixo valor semântico, se for analisada a quantidade de dados transferidos para este fim. A forma verbosa como requisições e respostas são criadas pelas entidades do protocolo mostra-se ineficaz do ponto de vista da utilização dos recursos da rede. Embora nos moldes tradicionais das redes de computadores este fator não seja tão crítico, o aproveitamento dos recursos disponíveis é crucial na análise de um protocolo para o ambiente de computação móvel. O objetivo deste trabalho consiste em melhorar o modo como os dados do protocolo são transferidos, sem que para isto o comportamento de suas entidades, frente ao recebimento de uma mensagem, seja modificado.

Neste trabalho são propostas duas melhorias para o funcionamento do protocolo HTTP. A primeira consiste em uma forma de codificação para as mensagens (requisições e respostas) deste protocolo. O objetivo principal desta medida é diminuir a quantidade de dados transferida nestas mensagens. A segunda melhoria consiste em aplicar algoritmos de compressão nos objetos transferidos como dados do protocolo (imagens, arquivos HTML, etc), medida que visa também a economia de banda de transmissão. Pelo fato da quantidade de banda passante ser mais escassa em redes móveis, estas novas propostas são mais indicadas para este ambiente.

Para fazer a codificação das mensagens HTTP, foi especificado o protocolo HTTP-m (HTTP móvel). Este protocolo consiste em uma variante do protocolo HTTP, tendo sido alterada apenas a codificação das mensagens, ou seja, como os bytes que trafegam na rede representam estas informações. Deve-se notar que o HTTP-m apenas provê uma nova forma de codificação de mensagens HTTP, antes que estas sejam enviadas pelo enlace sem fio. As entitades nos extremos da comunicação permanecem inalteradas, já que esta tradução é feita de forma transparente.

Embora seja possível diminuir a representação das mensagens de controle do HTTP, estes dados são apenas parte do fluxo gerado pelas entidades do protocolo. Para que se consiga um ganho mais expressivo, é necessária também uma compressão dos arquivos HTML e outros objetos transferidos. Este serviço será realizado pelo protocolo MPP (*Mobile Presentation Protocol*). Como dito anteriormente, o protocolo HTTP se utiliza de conexões TCP para transmissão dos dados. O MPP encontra-se presente entre o HTTP-m e o TCP, fazendo com que os dados provenientes do primeiro protocolo sejam tratados e então passados para o TCP. A utilização desta nova arquitetura de protocolos mostrou resultados encorajadores.

Uma característica muito importante deste projeto é que o comportamento tanto do HTTP-m quanto do MPP pode ser adaptado dinamicamente em função do tipo e tamanho do objeto a ser transferido, das condições do canal de comunicação em determinado momento e da energia disponível na unidade móvel. Ou seja, estes dois protocolos consideram esses fatores no seu comportamento, o que é um ponto central no desenvolvimento de todo o software para computação móvel [14].

Este trabalho está organizado como descrito a seguir. Na seção 2 é feita uma rápida descrição do ambiente de computação móvel, suas principais características e seu impacto na utilização do HTTP. A seção 3 discute alguns trabalhos relacionados com esta proposta que são importantes na identificação das contribuições deste artigo. A seção 4 descreve brevemente o funcionamento do HTTP. Nessa seção são mostrados exemplos de requisições e respostas do protocolo, alguns problemas e as principais desvantagens para sua utilização em ambientes móveis. A seção 5 traz uma descrição das principais características do protocolo HTTP-m. Nessa seção é mostrada a codificação aplicada às mensagens HTTP, com exemplos sobre mensagens reais deste protocolo e uma descrição da nova arquitetura resultante da introdução dos protocolos HTTP-m e MPP. Na seção 6 encontra-se a descrição do MPP e de seus serviços. Embora apenas parte de seus serviços seja de interesse do HTTP-m, é feita uma rápida apresentação dos recursos disponibilizados pelo MPP. A seção 7 mostra os resultados obtidos através da simulação dos protocolos propostos. São mostrados os resultados com os novos protocolos e a conseqüente diminuição no tráfego. Esta seção termina por analisar as circunstâncias nas quais a aplicação dos protocolos especificados é viável. Na seção 8 são descritos alguns trabalhos futuros diretamente relacionados com o presente artigo. Serão apresentadas também as conclusões deste trabalho, destacando as suas principais contribuições.

2 O Ambiente de Computação Móvel

Nesta seção será descrito o ambiente de computação móvel e serão focalizadas as suas principais características. Em primeiro lugar, será feita uma descrição da infra-estrutura da rede, ou seja, como as tecnologias de comunicação sem fio são inseridas nas atuais redes de computadores.

2.1 Infra-Estrutura

O modelo de computação utilizado aqui, mostrado na figura 1, consiste em dois tipos distintos de entidades: computadores móveis (CMs) e estações fixas. Algumas das estações fixas, chamadas de estações rádio-base (ERBs) ou estações de suporte à mobilidade, são providas de interfaces de rede sem fio que as permitem comunicar com os computadores móveis. Cada CM se comunica com apenas uma estação base e se utiliza desta para acessar o restante da rede.

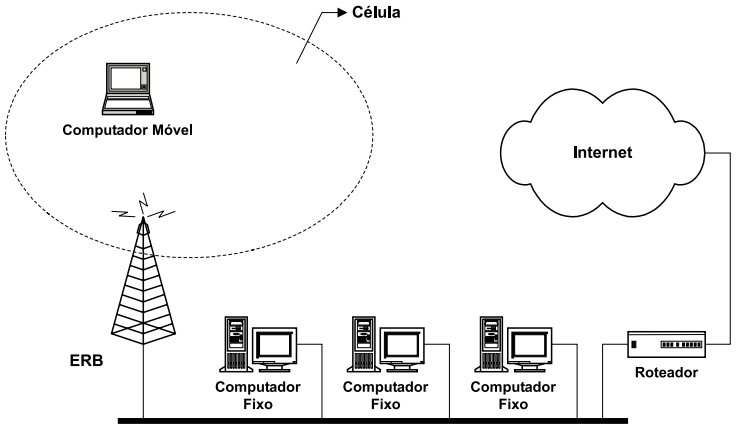

Figura 1: **Infra-estrutura para suporte à mobilidade.**

Este modelo pode ser estendido naturalmente para outras infra-estruturas de comunicação sem fio, como por exemplo redes locais sem fio e comunicação via satélite.

2.2 Principais Características

São várias as características que tornam o ambiente de computação móvel um cenário singular. A grande vantagem desta nova forma de comunicação é a capacidade de mover-se mantendo a conexão com a rede [10]. No entanto, esta contribuição é contraposta a uma série de limitações do ambiente móvel: baixas taxas de transferência, altas taxas de erros, falta de segurança, escassez de recursos, heterogeneidade e assimetria.

Em contraste com os computadores fixos, que ficam conectados a uma mesma rede, os computadores móveis encontram uma grande variedade de serviços de conexão. Em diferentes localidades eles podem encontrar acessos com propriedades bastante desiguais. Se por um lado, tecnologias de comunicação via infravermelho conseguem taxas de 1 a 10 Mbps, redes celulares possuem taxas inferiores a 20 Kbps. É importante para protocolos especificados para estes ambientes o conhecimento destas limitações.

O desempenho dos enlaces de comunicação sem fio é deteriorado ainda mais pela alta taxa de erros causada pelo canal de transmissão. Estes erros ocorrem normalmente em rajadas, causadas por interferências no meio. Enquanto na comunicação sem fio a taxa de bits errados é de aproximadamente 1 bit errado para cada 10^5 ou 10^6 bits transmitidos, numa comunicação através de um enlace de fibra ótica esta taxa é de um bit errado para cada 10^{12} a 10^{15} bits transmitidos [14].

Dispositivos móveis devem possuir algumas características básicas: devem ser leves, compactos, resistentes e sua bateria deve fornecer energia para várias horas. As três primeiras características são mais fáceis de serem atingidas, enquanto que a durabilidade das baterias continua sendo o principal fator limitante da utilização dos computadores portáteis. Independente da funcionalidade, *softwares* especificados para computadores móveis devem consumir o mínimo possível de recursos.

Devido à facilidade de se conectar a um enlace de comunicação sem fio, a segurança de tais sistemas é muito mais fraca que nas redes fixas, especialmente se a transmissão abrange uma grande área geográfica. Questões de segurança se tornam ainda mais complicadas se um CM pode cruzar fronteiras entre domínios diferentes. No cenário mais simples, qualquer computador móvel dentro de uma célula pode "escutar" todas as transmissões executadas.

Diante de todas estas propriedades inerentes da computação móvel, chega-se à conclusão que pelo menos duas medidas devem ser tomadas: diminuição e adaptação do tráfego e aumento da segurança em enlaces de comunicação sem fio. Estes são os principais objetivos dos protocolos HTTP-m e MPP, descritos adiante.

3 Trabalhos Relacionados

A aplicação de técnicas para redução do tráfego gerado por protocolos de comunicação é uma idéia já difundida. Trabalhos publicados na literatura têm utilizado um conjunto de técnicas que envolvem utilização de *cache*, codificação de deltas e algoritmos de compressão. A utilização de *cache* visa diminuir a quantidade de arquivos enviados, fazendo com que somente documentos que tenham sido alterados sejam trocados entre o cliente e o servidor (ou proxy) HTTP. Embora a política de *cache* tenha sido melhorada com a especificação do protocolo HTTP1.1 [6], documentos alterados são enviados inteiros, desperdiçando banda de transmissão. A utilização de codificação de deltas (*delta encoding*) tem como objetivo permitir que apenas as diferenças (deltas) entre duas instâncias do mesmo documento sejam enviadas. Algoritmos de compressão são utilizados tanto sobre os deltas enviados quanto outros objetos transferidos.

Técnicas de codificação de deltas são utilizadas já há algum tempo. A compressão de cabeçalhos des protocolos da pilha TCP/IP mostrados nos trabalhos de Jacobson [11] e Degermark [5] é baseada em uma técnica de codificação de deltas. Em [21], Williams et al. propõem o emprego de codificação de deltas no funcionamento do protocolo HTTP, em um trabalho sobre políticas de *cache*.

O projeto Mowgli [12] tem como objetivo examinar o comportamento da comunicação sobre enlaces de telefonia celular. Como parte desse projeto, Liljeber et al. mostram em [13] uma arquitetura para melhorar o desempenho do protocolo HTTP nesse ambiente. Essa solução utiliza dois proxies (um no cliente e outro no servidor) que mantém *cache* de documentos. O transporte de dados sobre o enlace sem fio (entre os dois proxies) é feito através do *Mowgli Data Channel Service* (MDCS), que tem como objetivo utilizar melhor os recursos do enlace. Compressão de dados e diminuição dos cabeçalhos do HTTP são medidas sugeridas no trabalho, mas não implementadas.

Em [3], Banga et al. mostram que a utilização de codificação de deltas poderia diminuir a latência na requisição de documentos HTTP. Os *optimistic deltas* propostos nesse trabalho foram especificados para reduzir a latência na transferências de documentos em enlaces com baixas taxas de transferência. Devido ao envio de documentos entre os proxies cliente e servidor em períodos de sub-utilização do enlace, essa solução é mais indicada para modems não compartilhados do que para enlaces sem fio.

O projeto WebExpress [8, 9] foi o primeiro a apresentar resultados concretos sobre a utilização de um tipo de codificação de deltas no protocolo HTTP. O WebExpress foi especificado sem que alterações sobre o protocolo HTTP fossem necessárias, tornando transparente para clientes e servidores as transformações sobre os dados HTTP enviados na rede sem fio. As entidades CSI (*Client Site Intercept*) e SSI (*Server Site Intercept*) funcionam como proxies na comunicação entre cliente e servidor, mantendo uma conexão TCP por onde os dados são enviados. Embora os resultados mostrados sejam impressionantes, eles são bastante limitados e dependentes dos dados presentes no *cache* do cliente sem fio (o *cache* foi pré-carregado e vários dos documentos requisitados pelo cliente possuiam uma versão local). As páginas requisitadas nos testes mostrados em [9] eram páginas de consultas (formulários), sendo que testes com *requests* de outros tipos de páginas não foram executados.

Com a evolução para o protocolo HTTP1.1 e o aparecimento das conexões persistentes, torna-se desnecessária a manutenção da conexão TCP pelo par de proxies presentes no WebExpress. Para manter clientes e servidores inalterados, o WebExpress utiliza três conexões TCP: uma entre o *browser* e o CSI (através da interface de *loopback* do cliente), outra entre o CSI e o SSI (sobre a rede sem fio) e uma última entre o SSI e o servidor da página requisitada (na rede fixa). Como a conexão TCP entre CSI e SSI é permanente (não é fechada a cada objeto requisitado), essa medida retira o atraso do estabelecimento e fechamento de conexões do enlace sem fio, embora este esteja agora presente na interface local do cliente e na rede fixa, representado pelas outras duas conexões TCP.

Uma vez que os resultados mostrados são dependentes do *cache* do cliente móvel e dada a limitação dos recursos do cliente, são necessários testes mais diversificados, com perfis de usuários diferentes, para que o desempenho do *cache* seja melhor quantificado. Um ponto importante é que a arquitetura proposta pelo WebExpress foi especificada com o intuito de melhorar o desempenho apenas do protocolo HTTP, ao contrário do MPP que possui uma aplicação mais diversificada.

No trabalho mostrado em [15], Mogul et al. utilizam algoritmos de compressão, *cache* e codificação de deltas para redução do tráfego gerado pelas entidades do protocolo HTTP. Nesse trabalho são mostrados vários resultados que incentivam a utilização da codificação de deltas, juntamente com algoritmos de compressão, para reduzir o volume das mensagens HTTP entre cliente e servidor. Embora os resultados relatados sejam bastante satisfatórios, Mogul et al. sugerem a alteração do protocolo HTTP para suporte à codificação de deltas, o que demanda alteração de clientes e servidores HTTP. A utilização de *cache* nas entidades clientes é um fator limitante para utilização dessa arquitetura em ambientes de computação móvel, devido à limitação dos recursos presentes nos clientes móveis.

Outros trabalhos utilizam degradação de imagens e outras formas de compressão com perdas (*lossy compression*) para diminuir o fluxo de dados no enlace de comunicação sem fio [7, 17].

Em [7] são mostradas várias formas de filtragem que atuam sobre diferentes tipos de mídias.

Em [17], Seshan et al. defendem a utilização de negociação de conteúdo nas transmissões sobre enlaces de comunicação sem fio. De acordo com o estado da rede, imagens são enviadas com maior ou menor resolução, visando atender sempre um limite de tempo de resposta definido pelo usuário. Estes trabalhos mostraram bons resultados, que serviram de motivação na especificação de alguns serviços prestados pelo protocolo MPP.

Nos trabalhos relacionados acima, pode-se identificar alguns pontos que motivam este trabalho. Em primeiro lugar, não existe uma implementação disponível de um protocolo que implemente os serviços discutidos. Dentre as implementações mostradas nos trabalhos comentados, várias delas são restritas ao ambiente utilizado ou não agregam outros serviços, tornando difícil a sua inserção na pilha TCP/IP.

Alguns trabalhos não foram desenvolvidos especificamente para ambientes de computação móvel, não havendo assim um estudo sobre a sua aplicabilidade nesse cenário. Vários trabalhos se baseiam na utilização de um *cache* de documentos WWW tanto no servidor como no cliente móvel, o que é um ponto que deve ser mais estudado. Diferentes perfis de usuário podem variar o desempenho destas soluções.

A adaptabilidade inserida no serviço de compressão provido pelo MPP é uma característica nova, não identificada em outros trabalhos. Como mostrado neste artigo, o comportamento do MPP varia de acordo com vários parâmetros como estado da rede e tipo de mídia a ser compactada. Os protocolos HTTP-m e MPP fazem parte de uma camada de protocolos que pode ser facilmente inserida na pilha TCP/IP, sem que entidades HTTP precisem ser alteradas.

4 O Protocolo HTTP

O protocolo HTTP é bastante simples, funcionando através do modelo *request-response*. Neste modelo, o cliente estabelece uma conexão com o servidor, realiza uma requisição (*request*) e recebe uma resposta (*response*) do servidor. Os dados retornados pelo servidor podem conter apontadores para outros objetos, que podem gerar novas requisições por parte do cliente.

Existem dois tipos de mensagens definidas pelo HTTP: uma representa as requisições dos clientes e a outra as respostas dos servidores. Estas duas mensagens seguem padrões definidos pelo protocolo. Basicamente, requisições e respostas do protocolo HTTP seguem os formatos descritos na figura 2.

```
request-line
headers (0 or more)
<blank line>
body
```

(a)

```
status-line
headers (0 or more)
<blank line>
body
```

(b)

Figura 2: Formato básico de requisições e respostas do protocolo HTTP.

Uma requisição HTTP (figura 2a) pode conter um ou mais cabeçalhos, todos dentro de um conjunto definido pelo protocolo. Uma requisição tem um campo de dados (*body*) somente se estiver relacionada com o envio de dados, ao invés de um consulta. Uma requisição (*request*) que solicita uma página HTML, por exemplo, pode conter somente a primeira linha, onde é identificado o documento e a versão do protocolo utilizada.

Na resposta do servidor (figura 2b), a primeira linha indica o resultado da operação, ou seja, se a acão solicitada pelo cliente foi realizada com sucesso. Depois é enviado um conjunto de cabeçalhos, para os mais diversos fins. Exemplos de cabeçalhos enviados pelo servidor são aqueles que representam a data de criação dos objetos transmitidos, data e horário em que a requisição foi atendida pelo servidor, definição de políticas de *cache* e caracterização dos dados enviados.

A seguir temos um exemplo de requisição e outro de uma resposta HTTP. Os dados mostrados foram retirados dos testes descritos na seção 7 deste trabalho.

```
GET / HTTP/1.0
Host: www.dcc.ufmg.br
Accept: text/html, text/plain,
  text/sgml, video/mpeg, image/jpeg,
  image/tiff, image/x-rgb, image/png,
  image/x-xbitmap, image/x-xbm, image/gif,
  application/postscript, */*;q=0.01
Accept-Encoding: gzip, compress
Accept-Language: en
Negotiate: trans
User-Agent: Lynx/2.8rel.2 libwww-FM/2.14
```

(a) Requisição HTTP

```
HTTP/1.1 200 OK
Server: Netscape-Enterprise/3.5.1C
Date: Tue, 02 Feb 1999 13:31:28 GMT
Content-type: text/html
Expires: Fri, 21 Aug 1998 18:29:27 GMT
Content-length: 3302
Accept-Ranges: bytes
Connection: close
```

(b) Resposta HTTP

De acordo com estes exemplos, vê-se que estas são constituídas de palavras, ou *tokens*, que descrevem as operações e propriedades envolvidas. Ao contrário de outros protocolos da arquitetura TCP/IP [18, 20], o HTTP não define PDUs (*Protocol Data Units*), ou seja, um formato específico para codificação dos dados do protocolo. Assim, as informações são enviadas entre as entidades do HTTP de forma textual, o que significa a troca de muitos bytes desnecessários.

As requisições HTTP normalmente solicitam o envio de documentos HTML por parte do servidor, juntamente com figuras e outros arquivos multimídia. Embora os formatos empregados para imagens atualmente sejam bastante compactados, o mesmo não acontece com outros objetos transmitidos. Documentos HTML são o exemplo mais direto, pois o texto nesta linguagem não sofre nenhum tipo de tratamento ao ser enviado do servidor para o cliente.

Esta forma de troca de dados faz do HTTP um protocolo com baixa utilização da banda de transmissão disponível. Em redes fixas, com velocidades que chegam hoje a 1 Gbps, esta característica do HTTP não chega a ser tão degenerativa. Em ambientes de computação móvel, no entanto, qualquer desperdício na utilização da rede pode tornar a utilização de um protocolo de comunicação praticamente proibitiva, uma vez que as taxas de transmissão muitas vezes não passam de alguns kbps.

Visando melhorar a utilização dos canais de transmissão em ambientes de computação móvel, foram especificados dois protocolos: HTTP-m e MPP. O HTTP-m fica responsável pela geração das PDUs para representação das mensagens HTTP, enquanto o MPP provê um serviço de compressão adaptativo para ser aplicado sobre os objetos transferidos.

5 O Protocolo HTTP-m

Nesta seção são descritas as principais características do protocolo HTTP-m. Primeiramente, é mostrada a pilha de protocolos resultante da inserção dos protocolos HTTP-m e MPP e, em seguida, é descrito o funcionamento do HTTP-m.

5.1 Modelo de Referência Utilizado

O protocolo HTTP pertence à pilha de protocolos do modelo TCP/IP, situando-se na camada de aplicação [18, 20]. Neste modelo de referência, os protocolos da camada de aplicação utilizam os serviços da camada de transporte, através da interface aplicação-transporte. Os protocolos HTTP-m e MPP provêem serviços ao protocolo HTTP, estando porém acima do protocolo TCP (camada de transporte). Para melhor compreensão da pilha de protocolos resultante da inserção do HTTP-m e MPP, recorrer-se-á ao modelo OSI.

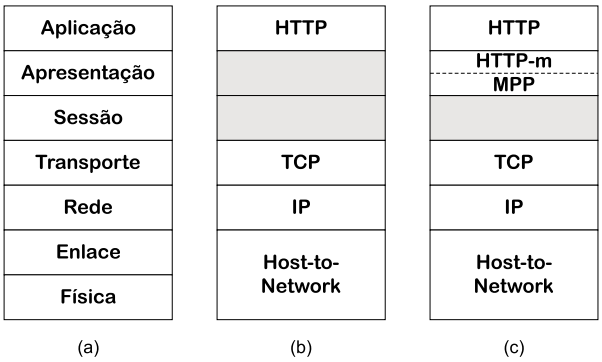

Figura 3: **Pilhas de protocolos: (a) modelo OSI (b) TCP/IP (protocolos) (c) híbrido (protocolos).**

O modelo OSI [20], mostrado na figura 3a, possui as camadas de sessão e apresentação entre a de aplicação e a de transporte. A camada de apresentação oferece serviços comuns a diferentes aplicações. Em particular serviços relacionados com a sintaxe e semântica da informação transmitida. Desta forma, pode-se dizer que os protocolos HTTP-m e MPP possuem características que os colocam na camada de apresentação. A pilha contendo todos os protocolos, depois da inclusão do HTTP-m e MPP, pode ser vista na figura 3c.

Estes protocolos, no entanto, apresentam diferenças significativas. Em primeiro lugar, o HTTP-m está diretamente relacionado a um protocolo de aplicação (HTTP), o mesmo não acontecendo com o MPP. Outra diferença, é que o MPP provê um conjunto de serviços que vão além da compressão de dados que é utilizada pelo HTTP-m. Isto faz com que os serviços do MPP possam ser utilizados por qualquer protocolo de aplicação, o que não é verdade para o HTTP-m. Estas diferenças fizeram com que fossem criados dois protocolos distintos, não agrupando suas funcionalidades.

Esta nova camada, colocada entre a de aplicação e a de transporte, pode ser dividida em duas subcamadas. A subcamada superior contém os protocolos que estão diretamente relacionados a um protocolo de aplicação, ou seja, sua utilização está limitada a um pequeno conjunto de protocolos desta camada. O HTTP-m, por prover um serviço de codificação das mensagens do HTTP, é um representante desta subcamada. A subcamada inferior agrupa os protocolos que disponibilizam serviços gerais, que podem ser utilizados por quaisquer protocolos da camada superior. O MPP encontra-se nesta subcamada.

O protocolo HTTP utiliza indiretamente os serviços dos dois protocolos. As mensagens do HTTP passam pelo HTTP-m, são codificadas e depois o HTTP-m passa estes dados para o MPP, que faz o seu tratamento.

5.2 Descrição do Protocolo HTTP-m

Na arquitetura convencional, o protocolo HTTP utiliza diretamente os serviços do protocolo TCP. Com a inserção dos protocolos HTTP-m e MPP, assim que a entidade HTTP passa os dados para o TCP, estes são interceptados, compactados e então devolvidos para o TCP, que mantém o curso normal da pilha TCP/IP.

Como mostrado na seção 4, o protocolo HTTP possui mensagens textuais extensas, que serão utilizadas pelo HTTP-m para geração de uma representação mais compacta. Pode-se dividir a atuação do HTTP-m em quatro tipos de codificação: requisições, respostas, cabeçalhos e objetos. As três primeiras codificações seguem o mesmo princípio. Em primeiro lugar, são identificados os campos que são fixados pelo protocolo, como tipos de cabeçalhos, métodos disponíveis, códigos de resposta e alguns conjuntos finitos (tipos de mídia, linguagens e codificações conhecidas). Estes campos podem receber uma codificação menor que a empregada

pelo HTTP. Dados como tipo de servidor e cliente utilizados, URL (*Uniform Resource Locator*) requisitada e sua localização não são codificados, uma vez que representam conjuntos muito numerosos.

5.3 PDUs do Protocolo HTTP-m

A codificação de mensagens HTTP resume-se na criação de PDUs do protocolo HTTP-m que possuem as mesmas informações da mensagem original. São definidas quatro PDUs pelo protocolo HTTP-m:

- **HTTPm_REQUEST_PDU:** esta mensagem representa a codificação da primeira linha de um *request* HTTP. Seus campos são: código desta PDU, versão, método, número de cabeçalhos enviados e a URL solicitada. Esta PDU tem tamanho variável, uma vez que a URL é um campo sem tamanho fixo;
- **HTTPm_RESPONSE_PDU:** esta mensagem representa a codificação da primeira linha de um *response* HTTP. Seus campos são: código desta PDU, versão, número de cabeçalhos enviados e um código de operação. Esta PDU tem sempre o tamanho de cinco bytes;
- **HTTPm_HEADER_PDU:** esta mensagem é utilizada para codificar os cabeçalhos enviados tanto em requisições quanto em respostas HTTP. Seus campos são: código desta PDU, identificador de cabeçalho e valor do mesmo. Esta PDU tem tamanho variável, uma vez que o valor de um cabeçalho não tem tamanho definido;
- **HTTPm_FILE_PDU:** esta mensagem é utilizada para codificar os objetos enviados em conexões HTTP, como figuras, arquivos HTML, etc. Seus campos são: código desta PDU, nome do arquivo e seu conteúdo. Esta PDU também possui tamanho variável.

Assim que o HTTP-m recebe uma mensagem HTTP, seja ela uma requisição ou uma resposta, ele gera sua codificação de acordo com estas PDUs. O envio desta mensagem se fará então pelo envio de um conjunto destas unidades de dados.

5.4 Exemplo de Funcionamento

Considere as mensagens HTTP mostradas na seção 4, que consistem em uma requisição e uma resposta. Ao recebê-las do protocolo HTTP, a entidade do HTTP-m gera, ao todo, 15 PDUs (uma para cada linha das mensagens). Para verificar a eficiência da codificação efetuada pelo HTTP-m, montaram-se as tabelas 1 e 2.

Request HTTP	Msg HTTP (bytes)	PDU criada	PDU (bytes)
GET / HTTP/1.0	13	HTTPm_REQUEST_PDU	5
Host: www.dcc.ufmg.br	22	HTTPm_HEADER_PDU	17
Accept: text/html, text/plain, (...)	180	HTTPm_HEADER_PDU	27
Accept-Encoding: gzip, compress	32	HTTPm_HEADER_PDU	4
Accept-Language: en	20	HTTPm_HEADER_PDU	3
Negotiate: trans	17	HTTPm_HEADER_PDU	7
User-Agent: Lynx/2.8rel.2 libwww-FM/2.14	40	HTTPm_HEADER_PDU	30
TOTAL	**324**		**93**

Tabela 1: **Codificação do *Request* HTTP.**

Como poder ser visto pelas tabelas, a codificação reduz bastante a quantidade de dados enviados. Tanto na codificação da requisição (tabela 1), quanto na codificação da resposta (tabela 2), conseguiu-se uma redução de mais de 70% na quantidade de dados enviados. A mensagem de requisição caiu de 324 para 93 bytes, enquanto a de resposta caiu de 215 para apenas 62 bytes.

Response HTTP	Msg HTTP (bytes)	PDU criada	PDU (bytes)
HTTP/1.1 200 OK	16	HTTPm_RESPONSE_PDU	5
Server: Netscape-Enterprise/3.5.1C	35	HTTPm_HEADER_PDU	28
Date: Tue, 02 Feb 1999 13:31:28 GMT	36	HTTPm_HEADER_PDU	7
Content-type: text/html	24	HTTPm_HEADER_PDU	3
Last-modified: Fri, 21 Aug 1998 18:29:27 GMT	45	HTTPm_HEADER_PDU	7
Content-length: 3302	21	HTTPm_HEADER_PDU	6
Accept-Ranges: bytes	21	HTTPm_HEADER_PDU	3
Connection: close	17	HTTPm_HEADER_PDU	3
TOTAL	**215**		**62**

Tabela 2: **Codificação do *Response* HTTP.**

6 *Mobile Presentation Protocol* (MPP)

Nesta seção são expostas as principais características do protocolo MPP. São listadas as unidades de dados do protocolo (PDUs) e um exemplo de seu funcionamento.

6.1 Descrição do Protocolo MPP

O protocolo MPP foi especificado para prover diferentes tipos de serviços importantes para o ambiente de computação móvel. Na seção 2 foram mostradas as principais características deste ambiente, ficando claro que este precisa de vários melhoramentos. O MPP, em sua atual especificação, presta dois tipos de serviço:

- **Compressão de dados:** este serviço consiste na aplicação de diversos algoritmos de compressão. Vários estudos estão sendo realizados para definir uma política de utilização dinâmica destes algoritmos, já que estes podem ter desempenho variável para diferentes massas de dados;
- **Aplicação de algoritmos de segurança:** devido à falta de segurança dos enlaces de comunicação sem fio, o MPP provê uma série de serviços de autenticação e criptografia, baseados em algoritmos conhecidos.

Neste trabalho apenas o serviço de compressão é relevante, constituindo um processo bastante simples. O protocolo da camada superior passa, através de sua interface com o MPP, um conjunto de dados a serem compactados. O MPP executa um algoritmo de compressão sobre eles para que possam então ser enviados. A escolha de um ou outro algoritmo de compressão varia de acordo com o tipo e tamanho da informação que está sendo compactada, já que os algoritmos se comportam de maneira diferente para cada tipo de mídia.

O comportamento do MPP é ditado dinamicamente, de acordo com os serviços aplicados, tipos de dados trabalhados, além das condições do canal de comunicação e da energia disponível. Este é um aspecto importante do MPP, que adapta dinamicamente o seu comportamento em função dessas características. A entidade MPP que fica no lado da rede fixa pode executar ainda outros serviços, como transformar imagens coloridas em preto e branco ou eventualmente não transmitir essas imagens. Todas estas características têm como objetivo permitir que a aplicação desses serviços seja executada sempre que esta adicione realmente melhorias à transmissão, sem que a prejudique sob outras condições. Em última instância, estas propriedades devem ser definidas pelo usuário.

No âmbito deste trabalho, os protocolos envolvidos na transmissão dos dados são o HTTP, HTTP-m e o MPP. O processo pelo qual os dados gerados pelo HTTP atravessam a pilha de protocolos sofreu algumas modificações em relação à pilha TCP/IP original. Primeiramente, o HTTP gera uma mensagem (*request* ou *response*) da forma habitual e a passa para a camada

de transporte. Antes que estes dados sofram qualquer tipo de tratamento pelo protocolo da camada de transporte (neste caso o TCP), estes dados são interceptados, trabalhados pelos protocolos HTTP-m e MPP e então devolvidos à pilha. Só então eles serão processados pelo TCP. Este processo é transparente aos clientes e servidores HTTP.

A entidade do protocolo HTTP-m, ao receber uma mensagem HTTP, faz a sua codificação gerando PDUs como aquelas mostradas na seção anterior. Depois que estas mensagens codificadas são geradas, elas são passadas para o MPP, que vai fazer a compactação de todos estes dados. É importante notar que alguns dados, como cabeçalhos HTTP, são primeiro codificados e depois ainda compactados, podendo gerar resultados ainda melhores que aqueles mostrados nas tabelas 1 e 2. Outros, como arquivos HTML e figuras, não têm seu tamanho modificado pelo HTTP-m, apenas pela compactação executada pelo MPP.

6.2 PDUs do Protocolo MPP

O MPP recebe uma massa de dados através de sua interface com a camada superior, executa seus algoritmos e gera um novo conjunto de dados compactados. Estes dados são enviados através de uma PDU definida no MPP:

- **MPP_DATA_PDU:** esta PDU é utilizada para transportar os dados depois que os serviços de compressão e criptografia tenham sido executados. Seus campos são: tipo de serviço, cabeçalho de compressão, cabeçalho de segurança e o campo de dados.

O campo tipo de serviço define quais algoritmos estão sendo aplicados sobre estes dados, ou seja, compressão sem segurança, compressão com segurança, ou outra variante possível. É através deste campo que a entidade que recebe a PDU sabe o processo que deve ser executado para gerar os dados originais. Para cada serviço que o protocolo presta, existe um cabeçalho nesta PDU. Cada cabeçalho contém informações necessárias para que este determinado serviço possa ser executado. O cabeçalho de compressão, por exemplo, apresenta um campo que define o algoritmo utilizado, para que o receptor possa aplicá-lo aos dados compactados. O cabeçalho de segurança possui informações como tipo de algoritmo, tamanho da chave e outras informações relevantes para este serviço.

O protocolo MPP foi especificado de forma a permitir a inclusão de novos serviços, assim que estes se tornarem necessários. Para que os dados transportados pela MPP_DATA_PDU seja processada por outro algoritmo que não seja de compressão ou segurança, basta que esteja presente um cabeçalho relativo a esse serviço com todas as informações necessárias.

6.3 Exemplo de Funcionamento

Como será aplicado apenas o serviço de compressão sobre os dados gerados pelo HTTP, considere o seguinte cenário: utilizar-se-á o algoritmo LZ77 (*Lempel-Ziv*) para compactação dos arquivos HTML. As informações necessárias para este serviço consistem apenas na identificação do algoritmo usado (aqui o LZ77). O MPP então, recebe uma massa de dados do HTTP-m, que representa um conjunto de mensagens HTTP já codificadas. Aplicando o algoritmo LZ77 sobre estes dados, o MPP cria uma instância da MPP_DATA_PDU cujo campo de tipo de serviço indica apenas compressão (sem segurança) e o cabeçalho de compressão informa o algoritmo utilizado. O resultado da aplicação do algoritmo sobre os dados recebidos é colocado no campo de dados da PDU.

Como os cabeçalhos e os outros campos da PDU são pequenos, o ganho que se tem nesta codificação é praticamente a taxa de compressão do algoritmo LZ77. Um fator importante na análise do desempenho deste protocolo é o atraso adicional (*overhead*) gerado pela compactação. Os testes executados, mostrados na seção 6, indicam que a taxa de compressão de arquivos HTML se mostrou boa, com tempo de compressão relativamente baixo.

7 Experimentos e Resultados Obtidos

Nesta seção são descritos os experimentos realizados e os principais resultados obtidos. Para testar os protocolos propostos, estes foram parcialmente implementados no Laboratório de Redes de Alta Velocidade da Universidade Federal de Minas Gerais, como parte do projeto SIAM (Sistemas de Informação em Ambientes de Computação Móvel) [2]. Para mostrar a viabilidade de sua utilização, foram feitos vários experimentos que têm como objetivo estudar o desempenho da proposta apresentada neste trabalho.

7.1 Ambiente de Simulação

O processo de simulação começa pela aquisição de um conjunto de dados, descritos abaixo, para servir de entrada para os protocolos HTTP-m e MPP. Com estes dados coletados, simulou-se o comportamento destes protocolos, tendo como resultado a eficiência obtida com a sua utilização.

A máquina utilizada para a simulação consiste em um computador pessoal equipado com processador Pentium de 233 MHz e 64 MB de memória primária. Os mesmos testes mostrados adiante foram também executados em uma máquina menos poderosa, com freqüência de 75 MHz e 40 MB de memória, obtendo porém resultados na mesma ordem de grandeza.

7.1.1 O *Browser* Lynx

Servidores HTTP normalmente geram arquivos de log contendo informações sobre todas as requisições recebidas, como o nome do documento solicitado, a máquina que enviou esta requisição, a data e hora do pedido. No entanto, o *request* completo, contendo todos os cabeçalhos, não é armazenado pelo servidor. Para que todos estes dados sejam armazenados, utilizou-se o *browser* Lynx [1].

O Lynx é um programa de navegação sem recursos gráficos disponível na Internet. Por oferecer uma interface bastante simples e ser distribuído juntamente com seu código fonte, o Lynx foi utilizado para gerar os logs necessários. Alterou-se seu código de forma que todo o texto enviado e recebido dos servidores HTTP fosse armazenado, para posterior utilização. Assim, constam agora nestes logs todos os cabeçalhos e demais informações trocadas com os servidores HTTP. Nos experimentos efetuados, descritos na seção 7.2, utilizou-se a versão 2.8 do Lynx.

7.1.2 Java

Para realizar as implementações necessárias para a simulação dos protocolos, utilizou-se a linguagem Java e o pacote JDK (*Java Development Kit*), versão 1.2, da *Sun Microsystems*. A linguagem Java foi escolhida por um conjunto de fatores. Em primeiro lugar, Java é independente de plataforma, ou seja, o mesmo código pode ser executado, sem grandes alterações, em todas as plataformas para as quais existe um interpretador ou compilador Java. O pacote JDK, versão 1.2, provê ainda criação de código nativo, com ganho expressivo de velocidade em relação a versões anteriores do *kit*. Embora estes compiladores Java ainda gerem código com desempenho inferior a códigos C/C++, esta diferença foi bastante reduzida.

7.2 Simulação

Uma vez coletados os dados relativos às mensagens HTTP, foram implementadas as funcionalidades relativas aos protocolos HTTP-m e MPP. Para simular a eficiência dos novos protocolos, são necessários apenas os módulos de criação de suas PDUs.

7.2.1 Geração das PDUs HTTP-m

As requisições e respostas coletadas com o auxílio do Lynx foram utilizadas para criação das PDUs do protocolo HTTP-m. Para cada requisição é gerada uma HTTPm_REQUEST_PDU e uma HTTPm_HEADER_PDU para cada cabeçalho. Para cada resposta é gerada uma HTTPm_RESPONSE_PDU e outras PDUs de cabeçalho.

Para medir a eficiência do protocolo, foram calculados a diminuição conseguida na representação das mensagens HTTP, o atraso introduzido nesta tradução e estimativas de tempo de transmissão com os novos protocolos (seção 7.3). Os resultados da simulação referentes aos dois primeiros parâmetros são mostrados nas tabelas 3 e 4.

Número de requisições codificadas	165
Tamanho total das mensagens HTTP	56 796 bytes
Tamanho total das PDUs geradas	23 593 bytes
Redução	58,5%
Tempo médio de processamento por *request*	20 ms

Tabela 3: **Codificação de *requests* HTTP.**

Número de respostas codificadas	178
Tamanho total das mensagens HTTP	33 676 bytes
Tamanho total das PDUs geradas	12 507 bytes
Redução	62,9%
Tempo médio de processamento por *response*	20 ms

Tabela 4: **Codificação de *responses* HTTP.**

Foram utilizadas 165 requisições HTTP, chegando a um total de 59796 bytes analisados. Depois que as requisições foram codificadas pelo protocolo HTTP-m, o tamanho total foi reduzido para 23593 bytes. O número de respostas HTTP codificadas foi um pouco maior (178), com o tamanho total caindo de 33676 para 12507 bytes. Na codificação das requisições HTTP conseguiu-se pouco mais de 58% de diminuição do tamanho das mensagens, enquanto para as respostas esta taxa foi superior a 62%. Em ambos os casos, o tempo de computação necessário se mostrou baixo, na ordem de dezenas de milissegundos.

É importante salientar que de acordo com a simulação feita, o tráfego HTTP seria codificado em PDUs HTTP-m na origem da comunicação, enviado e então restabelecido através de uma conversão HTTP-m→HTTP no destino. Estas traduções têm como objetivo manter inalterado o comportamento das entidades do protocolo HTTP.

O processo de criação das PDUs (simulado neste trabalho) é computacionalmente mais intenso que o processo inverso, já que se baseia em comparação de cadeias de caracteres extraídos da mensagem HTTP. Assim, o processamento no destino é mais simples que o simulado neste trabalho. Outro ponto interessante é que os clientes utilizados nos computadores móveis, podem gerar diretamente as PDUs do protocolo HTTP-m, dispensando assim este processo de tradução. Neste caso teria-se a mesma diminuição no tráfego mas sem o atraso gerado pela conversão.

7.2.2 Geração das PDUs MPP

Uma vez que as mensagens HTTP são codificadas pelo HTTP-m, este resultado é passado para o protocolo MPP. A entidade do MPP, que neste caso vai aplicar somente o serviço adaptativo de compressão, utiliza o devido algoritmo e gera as suas PDUs.

Dentre os dados recebidos do HTTP-m para compactação, os arquivos (HTML, figuras, etc) representam a maior parcela. Para simular o atraso envolvido neste processo, empregou-se o algoritmo de compressão LZ77 a um conjunto de documentos HTML extraídos da Internet.

Foram coletados 100 documentos, com tamanhos que variam de 200 bytes até 50 Kbytes. Os resultados dos testes são mostrados na tabela 5.

Número de arquivos HTML compactados	100
Taxa média no processo de compressão	530 684 bytes por segundo
Compressão alcançada	71,7%

Tabela 5: **Compressão de documentos HTML utilizando o algoritmo LZ77.**

Vê-se pela tabela 5 que a velocidade de compressão alcançada foi bastante satisfatória, frente à compressão conseguida. Esta velocidade foi encontrada dividindo-se o total de bytes dos arquivos originais pelo tempo gasto no processo de compactação. Considerando-se todos os arquivos compactados, foi conseguida uma redução de mais de 70% nos dados a serem transmitidos. Este valor foi conseguido compactando individualmente cada arquivo HTML e depois comparando a soma dos dados compactados com os originais. Embora estes resultados sejam interessantes, alguns pontos devem ser discutidos.

Em primeiro lugar, foram utilizados somente arquivos HTML, que não representam a totalidade dos arquivos transferidos. Embora isto possa parecer um pouco crítico, os padrões atualmente empregados para codificação de imagens e vídeos já empregam compactação, de forma que estes objetos não precisam ser tratados pelo MPP. Para uma estimativa mais apurada da eficiência desta nova pilha de protocolos, estes devem ser totalmente implementados e submetidos a rotinas exaustivas de testes. Os valores mostrados aqui têm como objetivo mostrar uma primeira idéia da viabilidade de sua utilização.

Outro fator importante que deve ser comentado é a variação no comportamento dos algoritmos de compressão de acordo com o tamanho do arquivo utilizado como entrada. Nos testes realizados, arquivos HTML com menos de 1 Kbyte conseguiram apenas uma redução na ordem de 30% de seu tamanho, a uma taxa de compressão de cerca de 50 Kbytes por segundo. Já arquivos maiores, por exemplo com 40 Kbytes, tiveram seu tamanho reduzido em até 94%, a taxas de mais de 750 Kbytes por segundo. Estas diferenças mostram como é importante o comportamento dinâmico do protocolo, onde a aplicação de um determinado serviço será executada apenas se o ganho desta operação for realmente satisfatório.

7.3 Viabilidade

Para mostrar a viabilidade da utilização dos protocolos HTTP-m e MPP, utilizar-se-ão os exemplos de *request* e *response* mostrados nas seções anteriores. Para os cálculos serão usados os seguintes dados:

- o tamanho médio dos arquivos HTML utilizados na simulação foi de aproximadamente 6000 bytes. Para os cálculos mostrados a seguir será utilizado um deles, com 6030 bytes;
- para *request* e *response* serão utilizados os dados das tabelas 1 e 2. Para codificação destas mensagens HTTP pelo protocolo HTTP-m será considerado um tempo de 20 ms, média mostrada na tabela 3. A codificação reduziu a requisição de 324 para 93 bytes, e a resposta de 215 para 62 bytes;
- a redução de tamanho conseguida para o arquivo HTML considerado foi de 65,2%, com um tempo de compactação de 10 milissegundos. Estes valores foram obtidos compactando-se este arquivo separadamente.

Considere o seguinte cenário: o *request* da tabela 1 é enviado do cliente para o servidor, que responde com o *response* da tabela 2 juntamente com o arquivo HTML considerado (6030 bytes). Para medir a viabilidade da utilização dos protocolos propostos, serão medidos os tempos gastos nesta transação nos dois cenários possíveis, definindo como v a velocidade de transmissão da rede sem fio em bytes por segundo.

Tempo sem utilização dos protocolos HTTP-m e MPP

- envio e recebimento da requisição: $(324/v) * 2$;
- envio e recebimento da resposta: $((215 + 6030)/v) * 2$;
- tempo total: $(13138/v)$.

Tempo com utilização dos protocolos HTTP-m e MPP

- envio e recebimento da requisição: $(93/v + 0.020) * 2$;
- envio e recebimento da resposta: $((62 + 6030) * (1 - 0.652)/v + 0.020 + 0.010) * 2$;
- tempo total: $(4426/v + 0.100)$.

Deve ser observado que a compactação conseguida foi aplicada ao arquivo HTML juntamente com a PDU do HTTP-m, uma vez que o conjunto destes dados constitui o corpo da PDU do MPP. No segundo cenário, foram adicionados aos tempos de transmissão os atrasos gerados pela geração das PDUs dos dois protocolos.

Comparando os tempos totais dos dois cenários, pode-se encontrar uma estimativa da aplicabilidade desta nova solução. Para calcular o limite superior de velocidade de transmissão da rede onde os resultados conseguidos pela nova arquitetura é melhor, tem-se que:

$$(4426/v + 0.100) < (13138/v)$$
$$v < 87\ 120\ \textit{bytes por segundo}$$
$$v < 696\ 960\ \textit{bits por segundo}$$

Desta forma, a utilização dos protocolos HTTP-m e MPP se mostrou vantajosa em redes com velocidade de transmissão inferiores a 700 kbps. Este valor é bastante satisfatório, uma vez que tecnologias de transmissão sem fio normalmente não conseguem taxas efetivas de transmissão desta magnitude. Um ponto interessante é que uma implementação real na linguagem C, por exemplo, deve apresentar um desempenho ainda melhor que o mostrado aqui. Os testes mostrados nesta seção são bastante limitados, tendo como único objetivo mostrar a viabilidade da implantação destes protocolos. Pretende-se submeter esta nova arquitetura a testes mais extensos e completos assim que esteja disponível uma implementação completa destes protocolos.

8 Conclusões e Trabalhos Futuros

Os protocolos de comunicação sempre foram projetados levando-se em consideração características do ambiente onde são executados. Com o advento da computação móvel, inúmeros protocolos propostos sob o paradigma tradicional das redes de computadores apresentaram deficiências na utilização dos recursos disponíveis. Isto se deve ao fato dos ambientes móveis, além de proverem mobilidade ao usuário, limitarem consideravelmente o desempenho na comunicação. O HTTP é um exemplo de protocolo especificado antes do aparecimento dos ambientes móveis, não sendo adequado para esse tipo de cenário.

Os protocolos HTTP-m e MPP, implementados com o intuito de diminuir o tráfego gerado pelo HTTP, mostraram resultados bastante satisfatórios. Mostrou-se que a nova arquitetura utiliza melhor os recursos da rede, diminuindo consideravelmente o número de bytes transmitidos sem que o atraso adicionado prejudique o seu desempenho. De acordo com os cálculos mostrados na seção 7.3, a economia aproximada de tempo de transmissão acarretada pelo uso desses protocolos torna-os uma boa opção para grande parte das tecnologias de transmissão sem fio, com velocidades inferiores a 700 Kbps.

A adptabilidade de aplicações ao ambiente de computação móvel é um problema importante e tem sido assunto de pesquisa da comunidade científica da área. Pretende-se continuar

trabalhando nesse problema em algumas direções. Será disponibilizada em breve uma versão completa de domínio público do HTTP-m e MPP, para que outras pessoas possam usar e eventualmente incluir novas funcionalidades. Um serviço que não foi discutido aqui e que será incluído no protocolo MPP, é o provimento de segurança (criptografia) dos dados.

Referências

[1] Lynx. http://lynx.browser.org/.

[2] SIAM - Sistemas de Informação em Ambientes de Computação Móvel. http://www.dcc.ufmg.br/siam.

[3] G. Banga, F. Douglis, and M. Rabinovich. Optimistic deltas for www latency reduction. In *Proceedings of the the 1997 Usenix Technical Conference*, January 1997.

[4] T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0. Technical Report RFC 1866, 1995.

[5] M. Degermark, M. Engan, B. Nordgren, and S. Pink. Low-loss tcp/ip header compression for wireless networks. *Wireless Networks*, 3(5):375–387, 1997.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical Report RFC 2068, Jan. 1997.

[7] A. Fox, S. D.Gribble, E. A.Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 160–170, 1996.

[8] B. C. Housel and D. B.Lindquist. WebExpress: A system for optimizing web browsing in a wireless environment. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking*, 1996.

[9] B. C. Housel, G. Samaras, and D. B.Lindquist. WebExpress: A client/intercept based system for optimizing web browsing in a wireless environment. *Mobile Network Applications*, 3(4):419–431, December 1998.

[10] J. Ioannidis, D. Duchamp, and G. Q. M. Jr. IP-based protocols for mobile internetworking. In *Proceedings of the ACM SIGCOMM'91 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 235–243, Zürich, Suisse, 1991.

[11] V. Jacobson. Compressing TCP/IP Headers for Low-speed Serial Links. Technical Report RFC 1144, Feb. 1990.

[12] M. Kojo, K. Raatikainen, and T. Alanko. Connecting mobile workstations to the internet over a digital cellular telephone network. In *Proceedings of the MOBIDATA Workshop*, NJ, 1994. Rutgers University.

[13] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K. Raatikainen. Optimizing world-wide web for weakly connected mobile workstations: An indirect approach. In *Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments*, Whistler, Canada, June 1995.

[14] G. R. Mateus and A. A. F. Loureiro. *Introdução à Computação Móvel.* 11a. Escola de Computação, Rio de Janeiro, Brasil, 1998.

[15] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for http. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 181–194, 1997.

[16] J. Postel. Transmission Control Protocol. Technical Report RFC 793, Sept. 1981.

[17] S. Seshan, M. Stemm, and R. H. Katz. Benefits of transparent content negotiation in http. In *Proceedings of the Global Internet Mini Conference at Globecom 98*, Sydney, Australia, November 1998.

[18] W. R. Stevens. *TCP/IP Illustrated Volume 1 - The Protocols.* Addison-Wesley, 1994.

[19] W. R. Stevens. *TCP/IP Illustrated Volume 3 - TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols.* Addison-Wesley, 1996.

[20] A. S. Tanembaum. *Computer Networks.* Prentice Hall PTR, 3rd edition, 1996.

[21] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. Removal policies in network caches for world-wide web documents. In *Proceedings of the ACM SIGCOMM'96 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 293–305, Stanford, CA, August 1996.