UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma Arquitetura baseada em Componentes para Adaptação de Conteúdo na Internet

Renato André Takayama Claudino

São Carlos – SP
Maio/2005

Renato André Takayama Claudino

Uma Arquitetura baseada em Componentes para Adaptação de Conteúdo na Internet

Orientador:
Prof. Dr. Wanderley Lopes de Souza

Co-orientador:
Prof. Dr. Antonio Francisco do Prado

UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO

São Carlos

M a i o / 2 0 0 5

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C615ab

Claudino, Renato André Takayama.
Uma arquitetura baseada em componentes para adaptação de conteúdo na Internet / Renato André Takayama Claudino. -- São Carlos : UFSCar, 2007.
80 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Adaptação de conteúdo. 2. Política de adaptação. 3. Desenvolvimento baseado em componentes. 4. Reuso de componentes. I. Título.

CDD: 005 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Uma Arquitetura baseada em Componentes para Adaptação de Conteúdo na Internet"

RENATO ANDRÉ TAKAYAMA CLAUDINO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Wanderley Lopes de Souza
(Orientador - DC/UFSCar)

Prof. Dr. Antonio Francisco do Prado
(DC/UFSCar)

Prof. Dr. Cléver Ricardo Guareis de Farias
(UNISANTOS)

São Carlos
Maio/2005

Dedico esta dissertação a meus pais,
Esmeraldo e Clarice .

Agradecimentos

Agradeço imensamente a meus pais, pelo enorme apoio e pelo carinho. Por tudo que fizeram por mim até hoje. Agradeço por me compreenderem e principalmente por sempre acreditarem no meu empenho, no meu esforço e por apoiarem minhas decisões.

Ao meu orientador Wanderley Lopes de Souza pela oportunidade concedida, pelos ensinamentos, pela experiência que pude adquirir durante o mestrado, pelas críticas e idéias que foram essenciais para a realização deste trabalho.

Ao meu co-orientador Antonio Francisco do Prado, principal responsável pelo meu ingresso no mestrado. Agradeço pelos ensinamentos e pelas discussões que sempre visavam o crescimento do trabalho.

Ao programa de pós-graduação pela estrutura e viabilidade de realizar esse trabalho e aos professores do GSDR: Luis Carlos Trevelin, Sérgio D. Zorzo, Hélio Guardia e Célio Moron, pelos conselhos e ensinamentos.

Ao Marcos e Pedro pelas colaborações do projeto.

À Capes, pelo apoio financeiro que permitiu o desenvolvimento deste trabalho.

À minha irmã, Ana Paula, que tem se tornado uma grande amiga e de quem tenho muito orgulho, pelos momentos de alegria e descontração. Aos meus avós Akio e Nair, tios e tias.

Aos grandes amigos que fiz em São Carlos, Wesley (Bel), Vinícius (Baiano) e Mairum, pelo companheirismo e amizade, pelas diversões, festas e alegrias.

A todos os amigos e colegas de São Carlos, Fernando (Bocudo), Ricardo (RAR), Evandro (Catimbinha), Erico (Sabirila), Fernando (Genta), Bruno, Eduardo (Escovar), Jéssica, Daniel Lucredio, Val, Cássio, Darley (Biriguei), André, Fabiana e a todos amigos do futebol.

Aos grandes amigos de Dracena, Michel, Fernando Bajo, Tiago e Kamei, e em especial aos meus irmãos Neto Pacaembu e Rodrigo (Bicudo).

À Mariana, minha namorada, pessoa que conheci há pouco, mas fico cada vez mais admirado e apaixonado. Pelo amor, carinho, compreensão e paciência.

Resumo

O crescimento do número de usuários da Internet, aliado aos avanços das tecnologias de acesso ocorridos em virtude do surgimento da computação ubíqua, gerou a necessidade de adaptar os conteúdos da Internet. A diversidade e heterogeneidade encontrada atualmente na Internet, incluindo usuários, dispositivos, redes de acesso e conteúdos, formam uma quantidade enorme de possíveis combinações a cada entrega de determinado conteúdo.

Em razão dessa necessidade de adaptação, este trabalho apresenta uma arquitetura focada na criação de diversas apresentações de um mesmo conteúdo e na realização de serviços de personalização de conteúdos da Internet. A arquitetura proposta considera os recursos existentes na atual infra-estrutura da Internet, baseia-se no modelo OPES e utiliza o ICAP para realizar a comunicação entre o dispositivo de borda e o servidor de adaptação. Além disso, essa arquitetura foi modelada e desenvolvida baseando-se em componentes de software com o intuito de criar padrões que facilitam seu reuso em outras arquiteturas semelhantes.

Para desempenhar os serviços de adaptação, a arquitetura segue uma política de adaptação, a qual considera as informações sobre usuário, dispositivo, rede de acesso, contrato de serviços e conteúdo. Essa política é controlada por um conjunto de regras de adaptação, que definem quais os serviços devem ser executados, para quem e quando solicitá-los.

Com o intuito de avaliar e validar a arquitetura proposta, este trabalho também descreve sua implementação através de seus estudos de caso e demonstra a viabilidade de seu uso em ambientes de computação ubíqua.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

OPES - Open Pluggable Edge Services

ICAP - Internet Content Adaptation Protocol

OCP - OPES Callout Protocol

DARPA - Defense Advanced Research Projects Agency

IP - Internet Protocol

OSI - Open System Interconnect

ISP - Internet Services Provider

CDN - Content Delivery Networks

PDA - Personal Digital Assistent

WAP - Wireless Application Procotol

WML - Wireless Markup Language

WBMP - Wireless BitMaP

ADSL - Asymmetric Digital Subscriber Line

ISDN - Integrated Services Digital Network

WLAN - Wireless Local Access Network

HTML - HyperText Markup Language

XML - eXtensible Markup Language

JPEG - Joint Photographic Experts Group

GIF - Graphic Image Format

MPEG - Moving Picture Experts Group

AVI - Audio Video Interleave

IETF - Internet Engineering Task Force

API - Application Programming Interface

HTTP - Hypertext Transfer Protocol

IRML - Intermediary Rule Markup Language

URI - Uniform Resource Identifier

BNF - Backus Naur Form

RTP - Real-time Transport Protocol

SOAP - Simple Object Access Protocol

SIP - Session Initiation Protocol

SMTP - Simple Mail Transfer Protocol

FTP - File Transfer Protocol

CSN - Content Services Networks

IDSP - Internet Service Delivery Protocol

CC/PP - Composite Capabilities/Preferences Profile

W3C - World Wide Web Consortium

URL - Unified Resource Locator

Sumário

1 Introdução

A história da Computação pode ser classificada, em função dos ambientes computacionais predominantes, em três eras: a passada dos *mainframes*; a atual dos computadores pessoais; e a futura da computação ubíqua, por alguns autores denominada *pervasive computing* [1]. Esta tem por meta permitir ao usuário, utilizando qualquer dispositivo, a qualquer momento e de qualquer lugar, o fácil acesso e processamento da informação.

A comunicação móvel é um dos fatores que está impulsionando o salto da Computação para essa nova era. A possibilidade de conexão a qualquer momento e de qualquer lugar outorgou aos usuários uma escolha e uma liberdade sem precedentes, permitindo que estes busquem novas e recompensadoras formas para o tratamento de assuntos pessoais e profissionais.

Em apenas uma década as redes móveis atingiram um ritmo de crescimento que demandou quase um século às redes fixas. Entretanto essa mobilidade globalizada demandará novas arquiteturas e protocolos, que viabilizem a fácil conexão das redes móveis aos diversos tipos de provedores de serviços e de conteúdo espalhados pela Internet.

A visão futurística da *Internet Móvel* assume usuários com diferentes perfis utilizando diversos tipos de redes de acesso e uma grande variedade de dispositivos móveis, exigindo serviços personalizados que atendam da melhor forma possível as suas necessidades, disponibilidades e localizações. Essa diversidade de dispositivos móveis, geralmente com capacidades limitadas e com acesso à Internet através de redes sem fio, gera a necessidade de disponibilizar várias apresentações de um mesmo conteúdo.

Dada essa nessecidade da realização dos serviços de personalização dos conteúdos e da criação de diversas apresentações de um mesmo conteúdo, uma grande dificuldade encontrada no meio acadêmico é a elaboração de uma estrutura que ofereça esses serviços sem afetar o desempenho da entrega do conteúdo requisitado. Dessa forma a criação dessa estrutura de adaptação é explorada neste trabalho, que define uma arquitetura baseada em componentes para adaptação de conteúdo na Internet. Para executar esses serviços, a arquitetura proposta além de considerar os recursos existentes na atual infra-estrutura da Internet, baseia-se no modelo OPES

(*Open Pluggable Edge Services*) [2] e utiliza o ICAP (*Internet Content Adaptation Protocol*) [3] para realizar a comunicação entre o dispositivo de borda e o servidor de adaptação.

Além da definição da arquitetura, um dos maiores desafios nessa área de adaptação é a implementação de uma política de adaptação. Através dessa política, a arquitetura decide qual adaptação deverá ser realizada, quando esta será solicitada, quem executará a adaptação e qual a ordem de execução das adaptações. Nesse contexto o objetivo deste trabalho é definir e implementar uma política de adaptação para que a arquitetura proposta desempenhe esse processo de decisão.

Dessa forma a política de adaptação da arquitetura proposta é controlada por regras de adaptação e considera os perfis de usuário, dispositivo e contrato de serviços, além das informações sobre a rede de acesso e o conteúdo.

Visto que a arquitetura é desenvolvida baseando-se em componentes de software, este trabalho também apresenta uma experiência de reuso da arquitetura, na qual o processo de decisão da política de adaptação é auxiliada por uma base de conhecimento. Esse tomada de decisão utiliza um mecanismo de inferência implementado em linguagem Prolog.

No próximo capítulo é discutida a evolução na infra-estrutura da Internet, apresentando as características e divisões das redes de conteúdo, assim como os avanços que geraram a necessidade de adaptação de conteúdo. O capítulo 2 descreve ainda as possíveis classificações da adaptação de conteúdo e suas principais aplicações. Esse capítulo também introduz o conceito das redes de serviços.

No terceiro capítulo são relatados os trabalhos de adaptação de conteúdo na Internet. Nesse capítulo são apresentadas as características da rede de serviços OPES, assim como sua arquitetura e suas linguagens de definição das regras de adaptação. No capitulo 3 também são expostos os protocolos para adaptação de conteúdo, ICAP e OCP (*OPES Callout Protocol*). Visando demonstrar a importância deste trabalho, esse capítulo discute os trabalhos correlatos publicados na área de adaptação de conteúdo.

A arquitetura desenvolvida neste trabalho de mestrado é apresentada no capítulo 4. Nesse capítulo são descritos seus módulos principais, assim como os componentes criados para cada módulo. O capítulo 4 apresenta a definição e implementação da política de adaptação utilizada no trabalho desenvolvidos, assim como relata o comportamento da arquitetura desenvolvida. Ainda nesse capítulo apresenta-se uma experiência de reutilização dos componentes definidos na arquitetura e a inserção de um mecanismo de inferência que controla a política de adaptação da arquitetura.

O capítulo 5 descreve um estudo de caso desenvolvido neste trabalho. Nesse estudo de caso foram executadas as seguintes adaptações: escaneamento de vírus, adaptações de imagens e filtragem de conteúdo; sendo que os testes realizados nesse estudo e seus resultados também são discutidos.

No sexto capítulo é apresentado um estudo que realiza adaptações em ambientes de computação ubíqua, assim como adaptação de imagens utilizando emuladores de telefone celular e *palmtop*. Esse capítulo descreve um modelo para análise de desempenho dessas adaptações e um estudo que demonstra a viabilidade de realizar adaptação de vídeo num ambiente de computação ubíqua. Finalmente o capítulo 7 relata as conclusões deste trabalho e discute possíveis trabalhos futuros.

2 *Evolução da Internet*

A comunicação entre computadores teve suas primeiras experiências realizadas no final da década de 60 através da ARPANET, uma rede de computadores desenvolvida pelo órgão norte-americano DARPA (*Defense Advanced Research Projects Agency*). Do seu início até os dias de hoje, esse meio de comunicação não parou de crescer e, a partir do desenvolvimento de novos dispositivos, protocolos e linguagens, chegou-se à Internet atual.

A Internet é um meio de comunicação que interconecta uma comunidade mundial composta por usuários e provedores de conteúdo, os quais são responsáveis pelo fornecimento de vários serviços. A criação dessa rede revolucionou o mundo da computação e das comunicações como nunca se tinha visto anteriormente.

Essa rede nasceu da integração das capacidades de certas invenções como telégrafo, telefone, rádio e computador, proporcionando um novo conceito de comunicação de dados. Através desse novo conceito, a Internet estabelece um mecanismo para disseminação da informação, assim como um meio para colaboração e interação entre indivíduos (e computadores) independente da sua localização geográfica [4].

A Internet vem sofrendo avanços em sua infra-estrutura para suportar seu crescimento e conseqüentemente o progresso da sua importância no cenário mundial. Esses avanços são determinados através da criação de novas camadas de rede, ou *overlay networks*, que foram adicionadas à estrutura essencial da Internet. Essas camadas são constituídas por redes "virtuais" de dispositivos, que oferecem serviços semelhantes entre si e que são conectados num mesmo nível de abstração.

A camada essencial da infra-estrutura da Internet é denominada **redes de pacote** (*packet networks*), que são redes físicas compreendidas de roteadores (*routers*) e outros elementos de rede (e.g., *switches*). A comunicação entre as redes de pacote é realizada através do protocolo IP (*Internet Protocol*), criado para desempenhar serviços da camada de rede. As redes de pacote, que formam a base da infra-estrutura da Internet, são representadas na Figura 1.

A figura 1 mostra um modelo clássico da Internet, no qual cada dispositivo de aplicação é

Figura 1: Redes de Pacote

conectado às redes de pacote a partir dos roteadores. O *usuário* estabelece uma conexão com o *servidor de origem* e se comunica com este através de pacotes que são roteados pela camada de redes de pacote [5]. Esse tipo de comunicação é conhecido como modelo fim a fim, sendo servidor de origem e usuário denominados pontos finais.

Entretanto nas redes de pacote, os roteadores não processam o conteúdo da aplicação. Essas redes simplesmente transmitem os pacotes aos pontos finais [6]. Assim devido ao crescimento do uso da Internet, as redes de pacotes tornaram-se incapazes de solucionar os problemas relacionados à entrega dos conteúdos, assim como o problema dos pontos críticos.

2.1 Crescimento da Internet

Estruturalmente a Internet é uma rede de comunicação formada por milhares de redes autônomas que se comunicam através do IP. Essas redes variam de grandes *backbones*, que constituem a espinha dorsal da Internet, aos pequenos provedores de acesso. Para que a Internet opere como uma única rede, todas as redes autônomas precisam se conectar e realizar a troca de dados, sendo esse processo denominado de parceria (*peering*). Essa parceria, aliada à adição de caminhos de roteamento, torna possível o acesso a todos os servidores e usuários da Internet a partir de qualquer uma dessas redes.

A grande facilidade de acesso ao conteúdo, através de uma única e simples interface (o *browser*), aliada à facilidade de integração dos provedores de conteúdo são os pilares do crescimento da Internet. Esse crescimento atingiu enormes proporções, sendo seu conteúdo acessado por milhares de usuários, como demonstra a Tabela 1 [7].

Com a proliferação dos computadores pessoais, além de outros motivos, o custo do acesso à Internet diminuiu, sendo esta disseminada para grande parte da população mundial. Assim a Internet é acessada cada vez mais por usuários de países subdesenvolvidos, conforme mostra a Tabela 2 [7, 8]. Essa expansão da Internet tende a descentralizar seu acesso, ampliando a diversidade dos seus usuários.

Tabela 1: Crescimento do número de usuários da Internet

Data	Número de usuários	% da população mundial
Dezembro 1995	16 milhões	0,39
Dezembro 1996	36 milhões	0,88
Novembro 1997	76 milhões	1,81
Setembro 1998	147 milhões	3,60
Agosto 1999	195,19 milhões	4,64
Agosto 2000	368,54 milhões	6,07
Agosto 2001	513,41 milhões	8,46
Maio 2002	580,78 milhões	9,57

Tabela 2: Distribuição dos usuários da Internet (em milhões)

	Dezembro 2000	Maio 2002	Março 2004
EUA/Canada	177,78 (42,5%)	182,67 (31,4%)	221,43 (24,9%)
Europa	133,97 (32,0%)	185,83 (32,0%)	259,65 (29,2%)
Ásia	104,88 (25,1%)	167,86 (28,9%)	302,25 (34,0%)
América Latina	16,45 (3,9%)	32,99 (5,7%)	56,22 (6,3%)
África	3,11 (0,7%)	6,31 (1,1%)	13,46 (1,5%)
Oriente Médio	2,40 (0,6%)	5,12 (0,9%)	19,37 (2,2%)
Oceania	—	—	16,26 (1,8%)
TOTAL	418,59	580,78	888,68

Além do acréscimo de usuários nos últimos anos, o aumento do acesso a recursos multimídia com fluxo contínuo (áudio e vídeo), o desenvolvimento do comércio eletrônico e o crescimento de fluxo de informações criado pela atual globalização estão gerando a necessidade de alterações na parte estrutural da Internet a fim de evitar o seu colapso. A Figura 2 demonstra esse aumento no fluxo de informações dessa rede, que também é refletido no elevado número de *hosts* existentes na Internet, atingindo a marca de 285.139.107 em julho de 2004 [9].

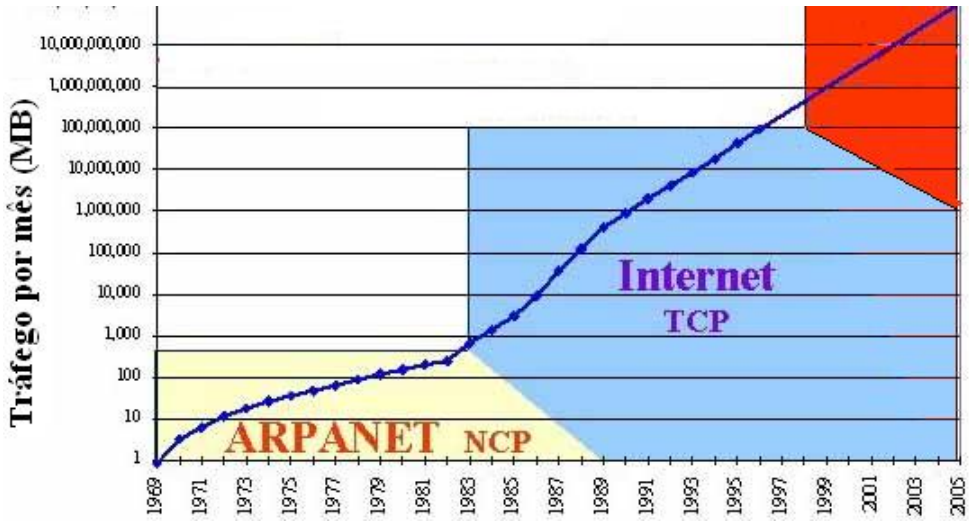


Figura 2: Aumento do fluxo de dados da Internet

A Figura 2 é mostrado que o limite do fluxo de dados suportado pela ARPANET é menor que 1.000 *megabytes* por mês, o qual foi alcançado em 1983, sendo então substituída pela Internet. Da mesma forma, projetou-se que o limite do fluxo de dados suportado pela Internet é de 100.000.000 *megabytes* mensais. Em razão do grande crescimento de usuários, serviços e tráfego de dados na Internet, esse limite foi superado, levando ao congestionamento dessa rede.

Essa sobrecarga do uso da Internet provocou o surgimento dos seus pontos críticos [10], que podem levar o desempenho dessa rede a índices insustentáveis. A Figura 3 apresenta a localização desses pontos críticos (gargalos) na estrutura da Internet.

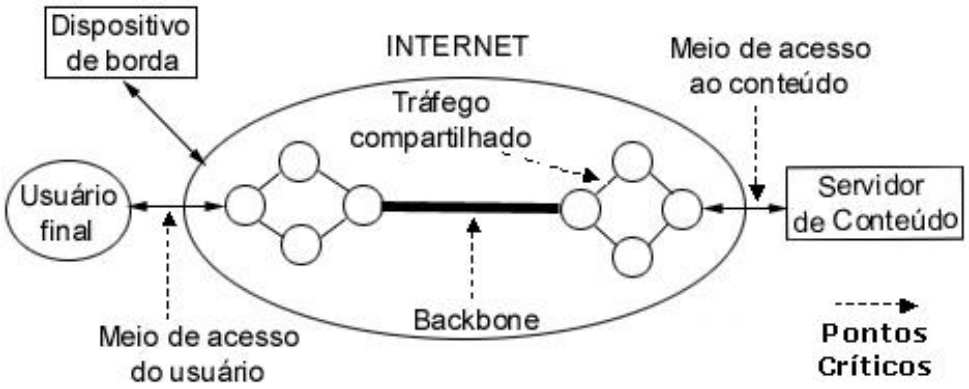


Figura 3: Pontos críticos da infra-estrutura da Internet

O primeiro ponto crítico está na capacidade máxima de dados a serem transmitidos entre o servidor de origem, que responde às solicitações do usuário, e a Internet. Nesse modelo centralizado todas as solicitações de dados, serviços e informações, originadas por usuários distribuídos pelo mundo afora, devem passar pelo meio de acesso ao conteúdo, o qual interliga o servidor de origem à Internet.

O segundo ponto crítico está no tráfego compartilhado entre as redes autônomas. Embora essa troca de dados seja uma necessidade para ambas as partes envolvidas, esta produz custos consideráveis de configuração e não sendo remunerada faz com que a capacidade de tráfego destinada às parcerias seja baixa ou opere no limite, não se adaptando aos picos de tráfego.

O terceiro ponto crítico está no *backbone*. Como no modelo atual todo o tráfego de dados geralmente atravessa uma ou mais redes *backbone*, a capacidade dessas redes deve crescer tão rapidamente quanto o tráfego da Internet, para que não seja gerado um gargalo. O problema é que os investimentos em *backbones* são altíssimos, pois além dos custos inerentes aos meios físicos de comunicação têm-se os custos relativos aos dispositivos de roteamento e comutação.

O quarto ponto crítico compreende na capacidade do meio de acesso do usuário ao provedor. Para grande parte dos usuários que usam *modem* a velocidade de acesso está limitada a 56 Kbps,

o que impede o acesso a determinados recursos multimídia (e.g., vídeo sob demanda de boa qualidade). A inserção de novos modos de acesso, operando a taxas mais elevadas, resolve esse problema, mas ao mesmo tempo agrava os problemas nos outros pontos críticos.

Visto que as redes de pacote tornaram-se incapazes de solucionar os problemas inerentes à ampliação da Internet, foram desenvolvidas as *redes de conteúdo*, que aproximam o conteúdo ao usuário através dos dispositivos de borda, evitando assim os gargalos da rede entre o usuário e o servidor de conteúdo. Essas redes buscam minimizar os problemas ocasionados pelo crescimento da Internet, como o problema dos pontos críticos.

2.2 Redes de Conteúdo

As redes de conteúdo foram adicionadas à infra-estrutura da Internet com a preocupação de manipular a entrega de seus conteúdos. Essas redes incorporam protocolos e aplicações que foram criados para alocação, *download* e uso de conteúdos. As ferramentas desenvolvidas nas redes de conteúdo incluem: *web caching proxies*, ferramentas de gerenciamento de conteúdo, *web switches* inteligentes e ferramentas avançadas para análise de *logs*, entre outros.

Enquanto as redes de pacote se concentram no roteamento e transferência de *frames* e pacotes, as redes de conteúdo são focadas no roteamento e transferência de requisições e respostas de conteúdo. As unidades de dados transportadas nas redes de conteúdo (assim como imagens, vídeos ou músicas) são geralmente muito grandes e podem equivaler a milhares de pacotes [11].

As redes de conteúdo foram criadas como uma nova camada virtual sobre as redes de pacote e, através dos dispositivos de borda (*edge devices*), viabilizam uma infra-estrutura necessária para oferecer as ferramentas voltadas à entrega do conteúdo. A Figura 4 ilustra a disposição dessa camada na infra-estrutura da Internet.

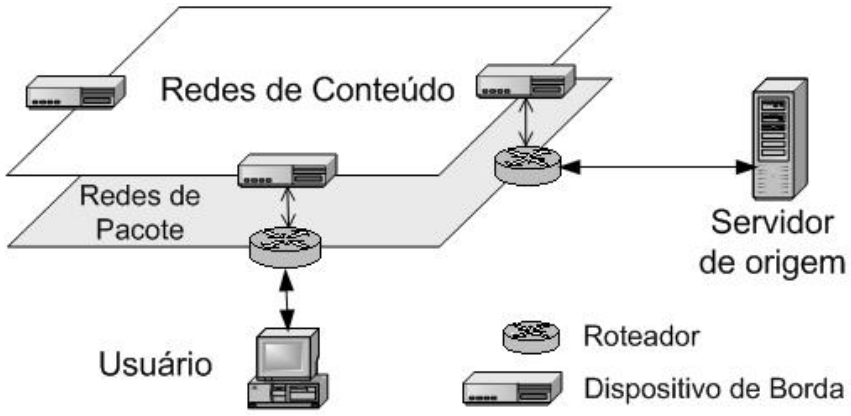


Figura 4: Sobreposição das Redes de Conteúdo

Apesar dos dispositivos de borda serem adicionados na infra-estrutura da Internet, cada dispositivo de aplicação (*usuário* e *servidor de origem*) continua conectado às redes de pacote através dos roteadores. Contudo os dispositivos de borda são inseridos no percurso das requisições e respostas de conteúdos e fornecem ferramentas que processam a entrega do conteúdo.

Segundo Day [11] e Nurmela [12], as redes de conteúdo podem ser dividida em três categorias: complexo de servidores, dispositivo de borda com *cache* e redes de entrega de conteúdo.

Um típico **complexo de servidores** (*server farm*) utiliza um *switch* "inteligente", o qual manipula as informações contidas nas camadas de de alto nível do modelo de rede OSI (*Open System Interconnection*). Um complexo de servidores é constituído por vários servidores de conteúdo que armazenam cópias parciais ou integrais de um conteúdo original e o *switch* inteligente examina as requisições dos usuários para escolher qual o servidor desse complexo será solicitado. Um modelo de complexo de servidores é apresentado na Figura 5(a), no qual a requisição *req_A* é direcionada ao servidor *Z*, enquanto a requisição *req_B*, ao servidor *X*

Entre os objetivos dessas redes estão a impressão de que existe um único servidor de origem, o balanceamento de carga das requisições sobre todos servidores do complexo e o roteamento automático das requisições quando houver falha de um servidor.

Embora forneçam alguma escalabilidade e diminuam a possibilidade de sobrecarga de um servidor de origem, esses complexos de servidores têm limitações. Visto que esse complexo tipicamente é desenvolvido próximo ao servidor de origem, cada requisição de usuário gera um fluxo de dados que percorre toda a Internet, não diminuindo os problemas de congestionamento dessa rede (e.g., o problema dos pontos críticos).

Já os **dispositivos de borda com *cache*** (*caching proxies*) visam beneficiar o acesso dos usuários à Internet e geralmente são implementados por ISPs (*Internet Service Provider*), que também são considerados dispositivos de borda. Esses dispositivos são alocados próximos aos usuários para que estes requisitem um conteúdo através desses dispositivos ao invés de requisitá-lo diretamente ao servidor de origem.

Interessados em melhorar o desempenho da entrega do conteúdo e reduzir a largura de banda utilizada, esses dispositivos de borda armazenam cópias dos conteúdos originais acessados anteriormente por seus usuários. Dessa forma, se o conteúdo requisitado estiver alocado no dispositivo de borda, este o entrega diretamente ao usuário, diminuindo assim o percurso do fluxo de dados pela Internet. A Figura 5 (b) ilustra um modelo dessas redes de conteúdo, no qual as requisições *req_C* e *req_E* obtiveram suas respostas no próprio dispositivo de borda com *cache* e não foram direcionadas aos servidores de origem.

Apesar de não resolver os problemas inerentes a esse g<F2>argalo específico, entre usuário e o dispositivo de borda, este alivia o tráfego nos três primeiros pontos críticos da Internet, abrindo espaço para a adoção em grande escala das redes de banda larga, além de diminuir o tempo de resposta à requisição do usuário.

Os dispositivos de borda com *cache* podem amenizar o problema de congestionamento da Internet, desde que os mesmos estejam alocados próximos ao usuário. Porém o armazenamento dos conteúdos originais é baseado somente na demanda dos usuários e isso pode ser ineficiente. No pior caso, se um conteúdo for requisitado *n* vezes através de *n* dispositivos de borda distintos, provocará *n* requisições ao servidor de origem, exatamente o mesmo comportamento se não houvessem esses dispositivos.

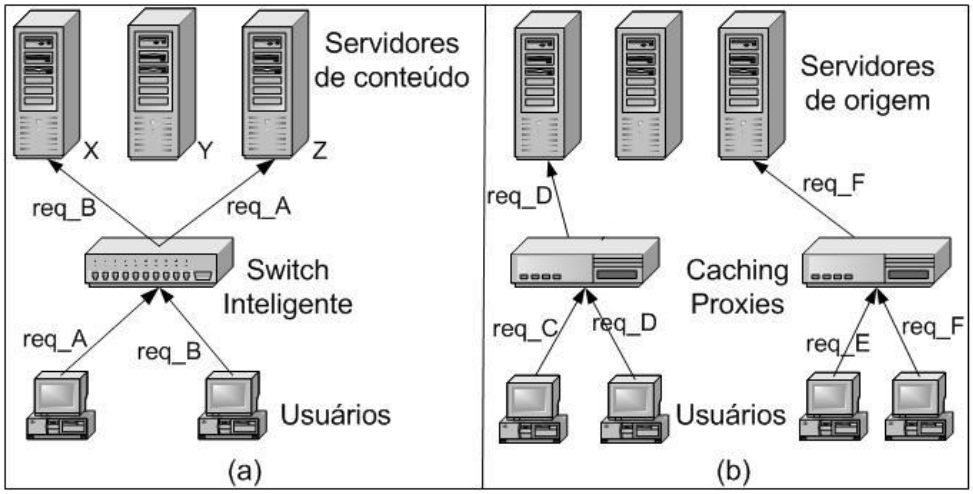


Figura 5: (a) Complexo de servidores; (b) Dispositivos de borda com *cache*

A criação das **redes de entrega de conteúdo** (*Content Delivery Networks* - *CDNs*) ocorreu porque a Internet não foi projetada para suportar a transmissão de grandes conteúdos por longas distâncias [13]. As CDNs, também denominada por *Content Distribution Networks*, diferem dos *caching proxies* essencialmente em dois pontos. Os dispositivos de borda com *cache* armazenam os conteúdos requisitados recentemente e com mais freqüência, baseando-se na configuração local (usuários), enquanto as CDNs são utilizadas pelos servidores de origem para armazenar conteúdos que são especificados por um administrador da rede. Um segundo ponto é que as CDNs podem prover acesso a conteúdos que geralmente são *uncacheable*, incluindo conteúdos de *streaming*, dinâmicos ou conteúdos seguros [14].

Uma CDN consiste de um conjunto servidores não-originais (*surrogates*) que tentam aliviar a carga dos servidores de origem, entregando seus conteúdos aos usuários. Os servidores *surrogates*, que contém parte ou todo o conteúdo do servidor de origem, podem ser alocados em qualquer lugar da Internet. Para cada requisição de conteúdo, a CDN procura localizar o *surro-*

gate que está mais próximo do usuário a fim de realizar a entrega do conteúdo mais rapidamente. Sendo que a noção de "próximo" pode considerar a localização geográfica, a topologia da rede ou a sua latência [15].

A arquitetura de uma CDN é formada pela combinação de quatro módulos: a **entrega de conteúdo** consiste no conjunto de servidores *surrogates* que entregam cópias do conteúdo aos usuários; o **roteamento de requisições** é composto por mecanismos que movem as requisições dos usuários para o *surrogate* mais apropriado; a **distribuição** consiste de mecanismos que movem os conteúdos do servidor de origem aos *surrogates*; e o ***accouting*** varre e coleta os dados dos outros três módulos da CDN [11].

A Figura 6 ilustra a arquitetura de uma CDN, onde cópias do conteúdo original estão armazenadas nos *surrogates*. Numa CDN as requisições do *usuário* são transmitidas a um *roteador de requisições*. Este examina essa requisição e define qual o *surrogate* mais próximo do usuário, melhorando assim a eficiência da entrega do conteúdo requisitado.

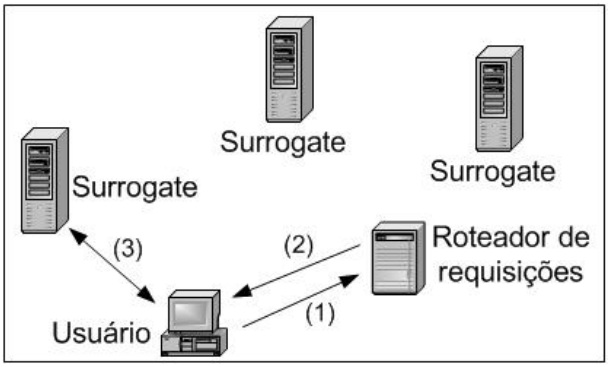


Figura 6: Redes de entrega de conteúdo

2.3 Avanço das Tecnologias de Acesso à Internet

O crescimento do número de usuários e do fluxo de dados da Internet ocasionou um congestionamento dessa rede no fim do século passado e portanto incentivou a criação das redes de conteúdo. Contudo esse número de usuários continua crescente, tornando o uso da Internet cada vez mais diversificado e heterogêneo, ao mesmo tempo que novos tipos de conteúdos se tornaram disponíveis nessa rede.

Além disso nos últimos anos ocorreram outros avanços relacionados ao acesso à Internet, os quais se concentram principalmente nas redes de acesso e nos diversos dispositivos que passaram a acessar os conteúdos da Internet.

- **Dispositivos de Acesso**

A história da computação está passando por uma transição entre a era dos computadores pessoais e a era da computação ubíqua, sendo esta impulsionada pela evolução da computação móvel. Além dos computadores de mesa (*desktops*), foram criados novos dispositivos que possuem acesso à Internet tais como *laptops*, *pamltops*, *handhelds*, *smartphones*, PDAs (*Personal Digital Assistant*) e telefones celulares.

Para atender o crescente uso desses dispositivos, novas tecnologias foram criadas em função de determinadas classes de dispositivos. Por exemplo, visando oferecer acesso à Internet para telefones celulares criou-se o protocolo WAP (*Wireless Application Protocol*) e, a partir desse protocolo, foram desenvolvidos os navegadores WAP, a linguagem de marcação WML (*Wireless Markup Language*) e o formato de imagem WBMP (*Wireless BitMaP*).

Em virtude do desenvolvimento de novas tecnologias e da criação dos meios de conexão que viabilizam o acesso desses dispositivos à Internet, houve uma grande evolução da *Internet Móvel*, na qual esses novos dispositivos têm acesso à rede mundial. A Figura 7 [16] apresenta a porcentagem do acesso à Internet através dos dispositivos móveis em relação aos dispositivos fixos, considerando diversos países.

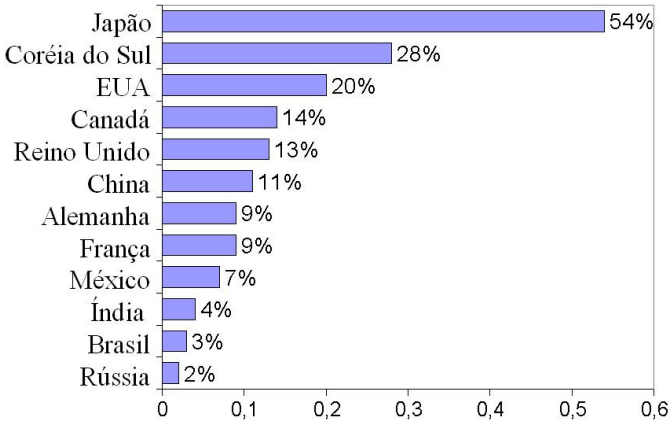


Figura 7: Acesso à Internet móvel

Além disso, os fabricantes têm construído dispositivos móveis com capacidades cada vez mais avançadas, viabilizando assim o uso destes para acessar diversos conteúdos e serviços da Internet, como mostra a Figura 8 [16]. As porcentagens descritas nessa figura representam o acesso aos serviços da Internet através dos dispositivos móveis em relação aos fixos.

Esses novos dispositivos geralmente possuem características bastante variáveis e, em certos casos, capacidades limitadas. Essas características podem ser divididas em **hardware**: memória, tamanho de tela, poder de processamento e escala de cores, entre outros; ou **soft-**

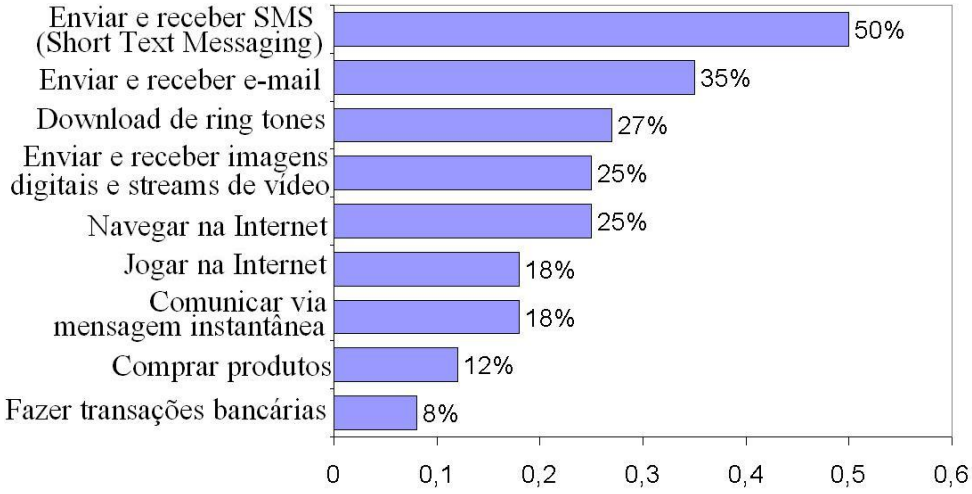


Figura 8: Serviços requisitados através da Internet móvel

ware: sistema operacional, navegador, aplicativos, *codecs*, etc.

Essa diversidade encontrada nos dispositivos móveis demandou a realização de adaptações sobre os conteúdos solicitados pelos mesmos. Essa adaptação busca viabilizar a entrega de conteúdos para dispositivos com capacidades limitadas (e.g., entregar páginas WML para determinados telefones celulares), assim como disponibilizar várias apresentações de um mesmo conteúdo para melhor aproveitar as capacidades desses dispositivos (e.g., reduzir as dimensões de um vídeo para atender ao tamanho de tela de um dispositivo pequeno).

- **Redes de Acesso**

Outro fator que incentivou a aceitação das adaptações de conteúdo foi o surgimento de novas tecnologias de redes de acesso à Internet. Algumas dessas novas tecnologias possuem grande largura de banda, como conexão via cabo, *Asymmetric Digital Subscriber Line* (ADSL), conexão via fibra ótica e *Integrated Services Digital Network* (ISDN); outras com baixa largura de banda, por exemplo via linha telefônica comum (*modem dial-up*); além das tecnologias com largura de banda variável, como rede local sem fio (WLAN - *Wireless Local Access Network*).

O desenvolvimento dessas tecnologias tem diversificado o acesso à Internet e para cada tipo de rede de acesso, o conteúdo possuirá um desempenho diferenciado na sua entrega. Assim a modificação desse conteúdo pode ser útil para otimizar o uso dessas redes e melhorar seu desempenho. A Figura 9 [16] apresenta essa diversidade das redes de acesso, expondo o meio de conexão que o usuário utiliza freqüentemente (conexão principal) e outras redes que o mesmo também possui acesso, porém esporadicamente (conexões secundárias).

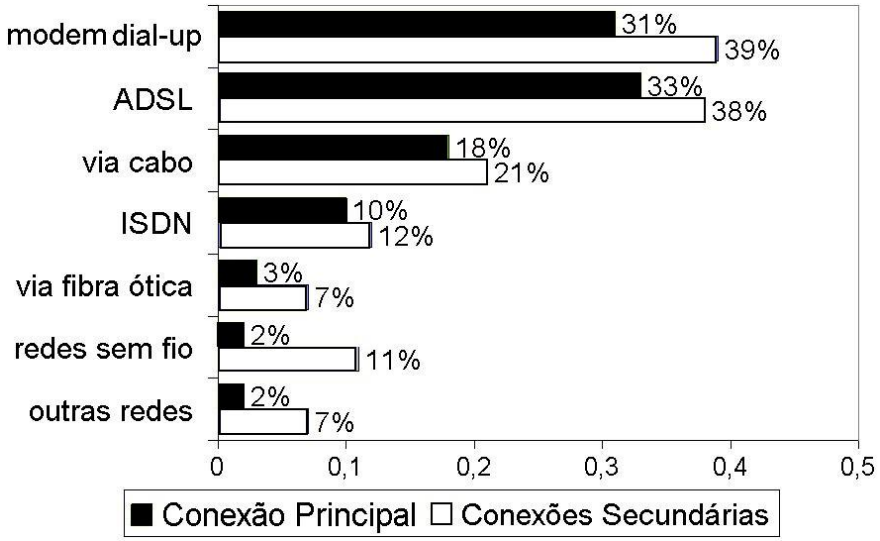


Figura 9: Redes de acesso à Internet

2.4 Serviços de Adaptação de Conteúdo

O termo adaptação de conteúdo se refere à modificação da representação de objetos ou conteúdos da Internet com o intuito de encontrar uma versão ideal desses conteúdos, adequando-os, por exemplo, às capacidades do dispositivo de acesso e às restrições de transmissão impostas pela rede de acesso [17].

Essa adaptação pode ser efetuada sobre diferentes tipos de conteúdos, assim como: **textos e páginas de marcação**, tais como HTML (*HyperText Markup Language*), XML (*eXtensible Markup Language*) e WML; **imagens**, por exemplo JPEG (*Joint Photographic Experts Group*), GIF (*Graphic Image Format*) e WBMP; **vídeo sob-demanda e arquivos de vídeo**, como MPEG (*Moving Picture Experts Group*), AVI (*Audio Video Interleave*) e Quicktime; **aplicações**; e **e-mail**.

Uma forma de classificar a adaptação de conteúdo na Internet é dividindo a mesma em duas categorias: adaptação estática e adaptação dinâmica[18, 19]. Na Figura 10 é demonstrada a diferença entre esses dois tipos de adaptação, que são discutidos a seguir:

- *Adaptação estática*. No momento de sua criação, o conteúdo é pré-adaptado em múltiplas versões que são armazenadas no servidor de origem. No momento da entrega e apresentação do conteúdo, a versão mais apropriada é selecionada.

- *Adaptação dinâmica*. Nesse tipo de adaptação, uma única versão do conteúdo é criada e armazenada no servidor de origem. No momento da entrega do conteúdo, este é adaptado durante seu percurso até o usuário.

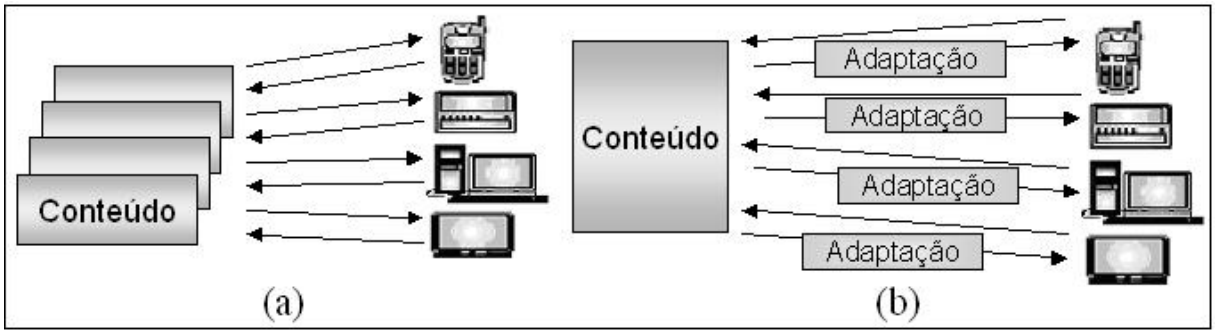


Figura 10: Adaptação de conteúdo: (a) estática e (b) dinâmica

As principais vantagens da adaptação estática são a qualidade da versão criada e o desempenho na entrega do conteúdo. Porém as desvantagens dessa adaptação consistem na repetição do mesmo conteúdo em todas as versões disponibilizadas, no custo com o desenvolvimento dessas versões e das suas atualizações, e na impossibilidade de atender a dispositivos não previstos.

Considerando o cenário atual da Internet, que possui grande heterogeneidade de usuários, dispositivos, redes de acesso e conteúdos, os trabalhos na área de adaptação de conteúdo se focam principalmente na adaptação dinâmica. Nesse caso, existem três pontos onde a adaptação pode ocorrer: no dispositivo do usuário, no servidor de origem ou no dispositivo de borda [18, 20], como mostra a Figura 11.

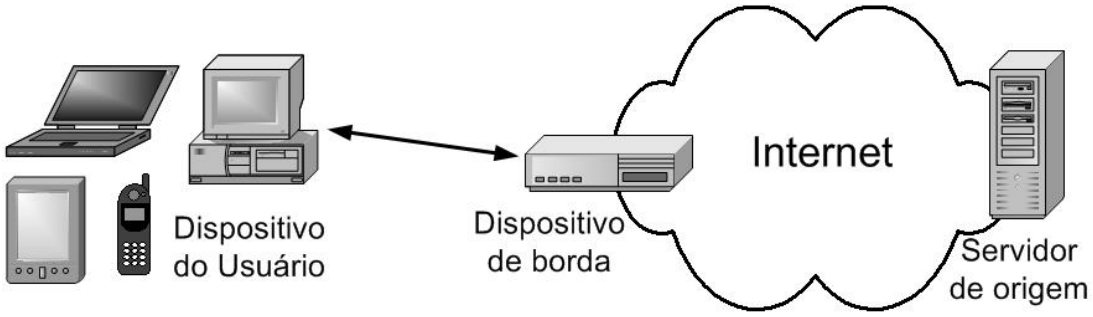


Figura 11: Pontos de adaptação de conteúdo

- *Adaptação no dispositivo do usuário*. Esse tipo de adaptação depende principalmente das capacidades e limitações desse dispositivo. Essa adaptação é executada quando o dispositivo do usuário recebe o início do conteúdo respondido pelo servidor de origem. Essa adaptação pode ser vantajosa se o servidor de origem ou o dispositivo de borda estão impossibilitados de determinar quais são as características do dispositivo do usuário. Entretanto essa alternativa tende a ser ineficiente ou até impossível quando o dispositivo do usuário possui recursos limitados, caso típico dos dispositivos móveis.

- *Adaptação no servidor de origem*. Nessa adaptação o servidor de origem armazena o conteúdo original e, a partir da requisição do usuário, executa a adaptação mais adequada àquele usuário. No entanto torna-se complicado para o servidor de origem obter dinamicamente e armazenar todos os dados necessários para uma adaptação adequada (e.g., informações relativas aos usuários, às características dos dispositivos do usuário e às capacidades de transmissão das redes de acesso)

- *Adaptação no dispositivo de borda*. Nesse tipo de adaptação, o usuário tem acesso à Internet através de um dispositivo de borda (também chamado de *proxy*). Este adapta o conteúdo retornado pelo servidor de origem e envia a versão modificada desse conteúdo ao usuário. A adaptação no dispositivo de borda oferece vantagens como: possuir melhores condições para obter dinamicamente e armazenar uma grande quantidade de informações relativa ao ambiente de adaptação; e não sobrecarregar o servidor de origem e nem o dispositivo do usuário.

Além da divisão entre adaptação dinâmica e adaptação estática, da distinção dos possíveis pontos de adaptação de conteúdo e das diferentes aplicações, existem outras classificações que podem ser consideradas em relação à adaptação de conteúdo [18]. Primeiramente, considerando o foco final da adaptação de conteúdo, esta pode ser dividida em duas categorias:

- *Adaptação segundo capacidades técnicas*. Essa adaptação considera as capacidades do dispositivo do usuário e as capacidades de transmissão da rede de acesso. Esses conceitos técnicos são relevantes, visto que os dispositivos "pervasivos" atuais contém amplas variações de suas características e as redes de acesso utilizadas variam de cabo à sem fio, possuindo diferenças significativas de largura de banda e latência de rede. Como exemplo desse tipo de adaptação pode-se citar a redução do tamanho de uma imagem para atender um dispositivo com tela de tamanho reduzido.

- *Adaptação segundo as preferências do usuário*. Para tipos específicos de conteúdo, os usuários podem ter predileções individuais e estas devem ser consideradas. Enquanto uns usuários preferem áudio a vídeo, outros podem ser diferentes. Assim como uns preferem imagens em preto-e-branco, outros preferem imagens de tamanho reduzido. Enfim existem diferenças de interesses entre os usuários e nesse tipo de adaptação essas diversidades são consideradas.

Pode-se também categorizar a adaptação considerando o tipo de mídia do conteúdo. Essa classificação compara o tipo da mídia resultante da modificação em relação ao tipo da mídia do conteúdo original. Assim existem outras duas classificações de adaptação de conteúdo:

- *Adaptação intra-mídia*. Nessa adaptação o conteúdo adaptado contém o mesmo tipo de mídia que o conteúdo original. Por exemplo, conteúdos como imagens podem sofrer alterações na qualidade, resolução, tamanho ou no formato, mas o resultado dessa adaptação continua sendo uma imagem. Da mesma maneira conteúdos textuais podem ter modificações de idioma ou de linguagem de marcação, obtendo como resultado outro texto.

- *Adaptação inter-mídia*. Nesse caso o processo de adaptação altera o tipo da mídia do conteúdo. Por exemplo, a transformação de vídeo num conjunto de imagens, disponibilizando o acesso da informação desse vídeo para dispositivos limitados. Sob outro aspecto, para usuários com deficiências visuais, pode-se adaptar um conteúdo de texto para áudio.

Com o intuito de demonstrar a grande funcionalidade da área pesquisada neste trabalho e exemplificar o uso das adaptações definidas acima, apresenta-se a seguir as principais aplicações dos serviços de adaptação de conteúdo [21–23].

- *Escaneamento de vírus*: capacidade de desempenhar dinamicamente a busca por vírus sobre o conteúdo requisitado e entregá-lo ao usuário livre de infecções. Esse serviço é particularmente muito útil quando o usuário não possui um *software* anti-vírus instalado em seu dispositivo. Com esse serviço o usuário pode acessar conteúdos da Internet sem se preocupar com a possibilidade de receber um conteúdo infectado.

- *Inserção de propaganda*: capacidade de inserir propagandas num conteúdo (e.g., páginas Web), baseando-se nos interesses do usuário ou na sua localização geográfica. Assim as propagandas originais desse conteúdo podem ser substituídas por outras focadas num usuário específico.

- *Tradução de linguagens* Markup: permite a dispositivos que não suportam páginas HTML (e.g., determinados telefone celular) receberem essas páginas em outro formato (e.g., WML). Esse serviço além de disponibilizar um número maior de conteúdo para os dispositivos limitados, elimina o trabalho de desenvolver várias versões do mesmo conteúdo.

- *Compressão de dados*: permite ao servidor de origem enviar seus conteúdos comprimidos, então o dispositivo de borda faz uma descompressão desses dados, economizando a largura de banda utilizada na comunicação entre os mesmos.

- *Filtragem de conteúdo*: capacidade de redirecionar uma requisição não autorizada, ou então bloquear um resposta também não permitida. Esse serviço permite, por exemplo,

aos pais e aos gerentes de uma empresa controlarem, respectivamente, o acesso de seus filhos a conteúdos impróprios e o acesso de seus empregados a conteúdos que não estejam relacionados com a sua produtividade.

- *Transcodificação e filtragem de imagens*: capacidade de manipular imagens, podendo realizar serviços como: transformar o formato de uma imagem, possibilitando que dispositivos limitados possam recebê-las (e.g., certos telefones celulares somente suportam imagens no formato WBMP); reduzir o tamanho, a resolução ou a escala de cores de uma imagem; remover imagens de uma página Web, economizando a utilização da largura de banda da rede de acesso e da memória do dispositivo.

- *Tradução de idiomas*: traduz uma página Web de um idioma para outro, por exemplo de português para chinês. Pode ser útil quando o servidor de origem não tiver todas as traduções necessárias para entregar um mesmo conteúdo em múltiplos idiomas.

- *Adaptação de* streaming: capacidade de processar arquivos de fluxo contínuo, assim como vídeo e áudio. Esses serviços podem adequar o conteúdo para garantir a qualidade de serviço da rede de acesso.

2.5 Redes de Serviços

O principal objetivo das redes de conteúdo é replicar o conteúdo em diversos locais e assegurar que este é entregue ao usuário de maneira confiável e rápida, melhorando o desempenho da Internet. Entretanto essas redes não modificam o conteúdo, adicionando valores ao mesmo [24]. Visto que a Internet continua evoluindo e aumentando sua diversidade, faz-se necessária a criação de uma nova camada em sua infra-estrutura, que estenda as capacidades das redes de conteúdo e realize os serviços de adaptação de conteúdo. Essa nova camada foi denominada de *redes de serviços* (ou *Edge Services Networks*) [5]. A Tabela 3 mostra uma comparação, sob alguns aspectos, entre as redes de serviços e as redes de conteúdo.

As redes de serviços oferecem um mecanismo de direcionamento do fluxo do conteúdo para módulos de serviços, que processam e modificam esse conteúdo. Esse mecanismo é constituído por um conjunto de regras e condições que determina qual o serviço a ser processado sobre o conteúdo. A sobreposição das redes de serviços perante as outras camadas da infra-estrutura da Internet é apresentada na Figura 12.

Nessa figura percebe-se que o *usuário* continua se comunicando com o *dispositivo de borda* através de uma rede virtual, sendo que os conteúdos são transmitidos por pacotes roteados pelas

Tabela 3: Comparação entre redes de serviços e redes de conteúdo

	Redes de Conteúdo	**Redes de Serviços**
Overview	Rede de *caching proxies*	Rede de *proxies* de aplicação
Serviços	Armazenamento e réplica	Processamento
Tipo de recursos	Disco e memória	CPU e processador
"Consumidores"	Servidores de origem e ISPs	Servidores de origem, usuários, ISPs e CDNs
Distribuição dos recursos para	Conteúdos Web	Serviços de valor agregado e aplicações

redes de pacote. Porém nesse modelo de rede um conteúdo transmitido é analisado por regras instaladas no dispositivo de borda e, se certas condições forem satisfeitas, esse conteúdo é enviado para um *módulo de serviço de conteúdo* (*content service module*). Nesse módulo o conteúdo é adaptado e seu resultado dessa adaptação é retornado ao *dispositivo de borda*.

A inserção dessa nova camada na infra-estrutura da Internet impulsionou os trabalhos na área de adaptação de conteúdo na Internet, assim como o modelo OPES (*Open Pluggable Edge Services*) e o protocolo ICAP (*Internet Content Adaptation Protocol*).

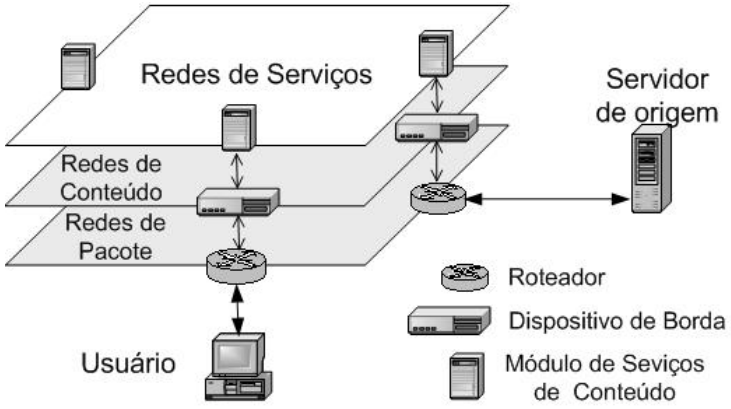


Figura 12: Sobreposição das Redes de Serviços

3 *Adaptação de Conteúdo na Internet*

Com a diversificação dos dispositivos e das redes de acesso à Internet, os usuários passaram a requisitar novos serviços de valor agregado, em particular serviços de adaptação de conteúdo tais como: tradução de idioma, filtragem de conteúdo, adaptação de imagens e vídeos, escaneamento de vírus, inserção de propaganda, montagem dinâmica de conteúdo e adequação de conteúdo para dispositivos móveis [24].

Dada essa necessidade de adaptar conteúdos, surgiram modelos de redes que visam oferecer esses serviços de adaptação. Os primeiros trabalhos dessa área realizavam as adaptações nos dispositivos de borda, porém com o desenvolvimento das *redes de serviços*, formulou-se a idéia de distribuir as adaptações para servidores dedicados, explorada no modelo *Open Pluggable Edge Services* (OPES).

3.1 Open Pluggable Edge Services

Buscando aprimorar as estruturas das redes para adaptação de conteúdo na Internet, a IETF (*Internet Engineering Task Force*) estabeleceu uma nova frente de pesquisa, resultando na criação de um modelo de redes para adaptação de conteúdo, o Open Pluggable Edge Services.

O modelo OPES é baseado na infra-estrutura das redes de serviços e oferece linguagens, APIs (*Application Programming Interface*) e protocolos que facilitam a implementação dos serviços de adaptação. Esse modelo foi construído por um grupo de trabalho, criado pela IETF, chamado OPES WG (*Working Group*) [1] que inicialmente definiu a arquitetura desse modelo.

Na arquitetura OPES tem-se o processador OPES, que desempenha o papel do dispositivo de borda, e os servidores remotos (módulos de serviços de adaptação). Nessa arquitetura o processador OPES e os servidores remotos são chamados de agentes OPES. Um sistema OPES é um conjunto de agentes OPES que interagem na realização de um serviço de adaptação.

[1] www.ietf-opes.org

A arquitetura OPES [2] é definida em termos de três conceitos relacionados: entidades, fluxo e regras OPES. As *Entidades* OPES, ilustrada na Figura 13, são aplicações que operam sobre o fluxo de dados entre o servidor de origem e o usuário. Essas aplicações podem ser:

- Aplicações de serviços OPES, que analisa e possivelmente modifica as mensagens trocadas entre o servidor de origem e o usuário. Essa entidade pode residir num processador OPES ou em servidores remotos (servidores de adaptação); ou
- Despachante de dados, que invoca um serviço OPES baseado numa política de adaptação, seguindo um conjunto de regras previamente conhecidas.

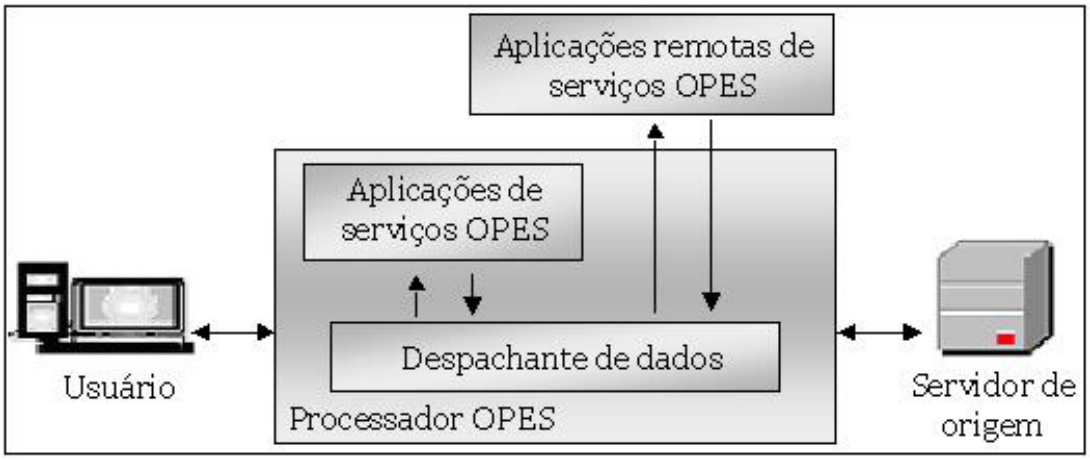


Figura 13: Entidades OPES

O *Fluxo OPES* compreende o fluxo de dados entre o servidor de origem, o usuário, as aplicações de serviços OPES e os despachantes de dados. Se uma política de adaptação é incluída no despachante de dados, então deve existir ao menos um deste no Fluxo OPES para que essas políticas sejam utilizadas. Conceitualmente a arquitetura OPES é independente do protocolo utilizado no fluxo entre o servidor de origem e o usuário, contudo em [2] é utilizado como exemplo o protocolo HTTP (*Hypertext Transfer Protocol*). A Figura 14 ilustra o fluxo de dados da arquitetura OPES.

A política de adaptação do OPES é determinada a partir das *Regras OPES*, que considera os serviços disponíveis e os dados recebidos. Essas regras constituem um conjunto de condições, que devem ser atendidas para que certas adaptações sejam realizadas. Portanto o conjunto de Regras OPES determina quais serviços serão executados sobre o conteúdo recebido. A Figura 15 ilustra o posicionamento e a interação dos diferentes componentes da arquitetura OPES.

Em determinados casos é mais interessante o processador OPES distribuir a responsabilidade de executar os serviços de adaptação, comunicando-se com um ou mais servidores remo-

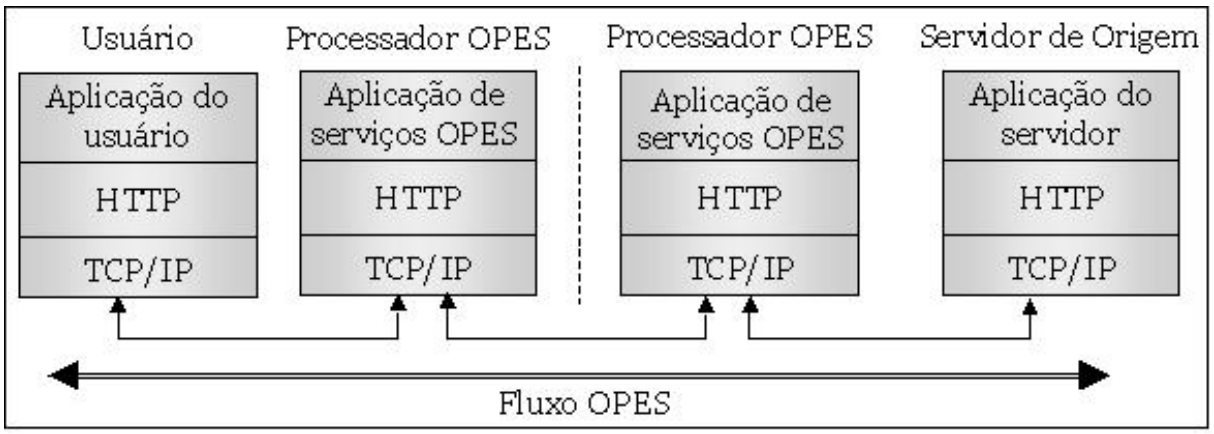


Figura 14: Fluxo OPES

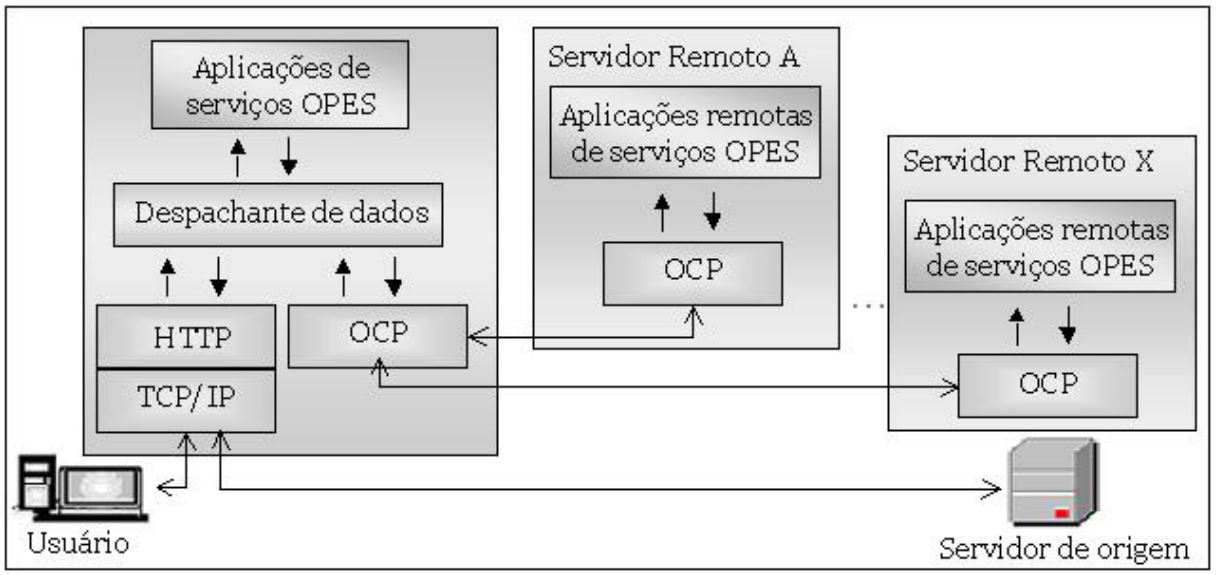


Figura 15: Interação dos componentes da arquitetura OPES

tos. Partindo desse princípio, o despachante de dados invoca o serviço remoto através do OPES *Callout Protocol* (OCP), que realiza a comunicação entre o dispositivo de borda e o servidor de adaptação, conforme apresentado na Figura 15.

Quando existe a necessidade de realizar duas ou mais adaptações de conteúdo, a comunicação entre o processador OPES e os servidores remotos e a realização desses serviços de adaptação podem ser seqüencial ou paralela [12].

Na adaptação **seqüencial**, o processador OPES invoca uma adaptação a um servidor remoto e, somente após a resposta deste, solicita outra adaptação. Assim continua até todas adaptações necessárias serem realizadas. Essa forma de adaptação aumenta a complexidade da política de adaptação a ser implementada na arquitetura, pois a ordem de execução das adaptações de conteúdo pode influenciar no desempenho e na eficiência da mesma.

Apesar da adaptação seqüencial ser mais comum, a adaptação **paralela** pode ser mais adequada para certos tipos de adaptação. Considere o caso de um servidor de origem enviar cópias de uma *stream* de vídeo para vários usuários. Se a política de adaptação decidir por adaptações distintas para cada usuário, então o processador OPES pode solicitar essas adaptações para diferentes servidores remotos ao mesmo tempo. Dessa maneira ao término de cada adaptação o processador OPES entrega o conteúdo adaptado para determinado usuário.

Além das definições anteriores, a arquitetura OPES é estendida através de conceitos como *bypassing*, *tracing* e *notification* [12]. O ***bypassing*** [25] habilita o usuário a indicar que alguns ou todos serviços de adaptação não devem ser executados. Esse conceito permite ao usuário assegurar-se de que o conteúdo é proveniente do servidor de origem. No entanto essa alternativa depende do protocolo de aplicação utilizado pelo usuário. Em [26] define-se uma extensão do protocolo HTTP, permitindo ao usuário (através do *browser*) inserir o cabeçalho "*OPES-Bypass*", que lista os serviços de adaptação que devem ser bloqueados ou indica que nenhum serviço deve ser executado.

OPES *trace* é a informação que armazena quais entidades OPES manipularam o conteúdo durante seu percurso e OPES ***tracing*** é o processo de criar, manipular e interpretar um OPES trace [25]. *Tracing* habilita o administrador da rede a "debugar" o sistema OPES e mostra os serviços de adaptação aplicados sobre o conteúdo. Além disso, esse conceito permite ao usuário ter conhecimento se o conteúdo recebido sofreu um serviço de adaptação.

A utilização do *tracing* no HTTP também é baseada em cabeçalhos de extensão. Em [26] define-se uma extensão do HTTP com o intuito de oferecer o OPES *tracing* através de dois novos cabeçalhos: "OPES-System" e "OPES-via". O primeiro armazena a identificação do sistema OPES que manipulou o conteúdo, enquanto o último oferece informações mais detalhadas, informando os processadores OPES e servidores remotos que processaram o conteúdo.

Notification refere-se à capacidade de informar ao servidor de origem que seu conteúdo disponibilizado sofreu adaptações por sistemas OPES, no entanto esse conceito ainda não foi implementado pelo OPES WG.

3.1.1 Linguagens para Especificação de Regras

Visto que nem todo conteúdo requisitado deve sofrer as adaptações de conteúdo oferecidas pelo dispositivo de borda, é necessário um mecanismo que invoque serviços de adaptação somente se certas condições forem satisfeitas. Esse mecanismo é guiado por regras de adaptação, que definem essas condições e suas adaptações relacionadas.

Partindo desse contexto, o OPES WG elaborou as linguagens IRML (*Intermediary Rule Markup Language*) e "P", que especificam as regras de adaptação, visando prover facilidades e padrões para o uso dessas regras e da política de adaptação.

A linguagem **IRML** foi projetada para oferecer uma simples e eficiente linguagem para expressar uma política de execução de serviços. Essa linguagem, que é baseada em XML, foi criada para refletir os interesses dos dois pontos finais de uma transação de conteúdo: usuário e servidor de origem. Por exemplo, um servidor de origem pode utilizar o IRML para solicitar que suas páginas Web sejam adaptadas a fim de atender pequenos dispositivos [27].

IRML não é direcionada para um protocolo particular da comunicação entre usuário e servidor de origem. Porém, essa linguagem permite o uso de extensões que visam a melhor utilização para protocolos específicos (por exemplo para o HTTP).

Além disso, IRML especifica regras que devem ser analisadas por um processador de regras situado num dispositivo intermediário (dispositivo de borda). O funcionamento dessa linguagem é baseada no *match* de certas condições referentes às propriedades do conteúdo transmitido e às variáveis do ambiente do sistema. Se todos os *matches* de uma regra forem atendidos, a ação especificada nessa regra determina o serviço de adaptação a ser executado [27].

Essa linguagem é constituída por elementos que provêem suporte às informações sobre o contexto da adaptação. No elemento *property* são descritas as condições de determinada adaptação. As ações de cada regra são descritas no elemento *service*. Este é composto pelo atributo *uri*, o qual indica o endereço do servidor de adaptação a ser solicitado, e pelo atributo *parameter*, que descreve os parâmetros enviados ao servidor de adaptação. Esses parâmetros podem ser estáticos (constantes pré-definidas) ou dinâmicos (quando informações obtidas dos perfis são enviadas aos servidores de adaptação, como tamanho de tela).

A Figura 16 mostra um exemplo de especificação de regras de adaptação utilizando a linguagem IRML. Essa regra é aplicada sobre documentos HTML e contém as condições *Content-type = html* (a), *Network-traffic = full* (b) e *Image-user-preference = removal* (c). Uma vez satisfeitas essas condições, a ação *removal-images* (e) é solicitada ao servidor de adaptação de endereço *icap://www.htmladapt.com* (d).

O OPES WG também definiu a linguagem de processamento de mensagens **P** visando descrever as regras de adaptação. P é uma linguagem interpretada e suas características se assemelham às linguagens Smalltalk e C++. Em [28] definiu-se que essa linguagem deve atender as seguintes características: flexibilidade, eficiência, simplicidade, exatidão, segurança e rigidez. Além disso, P é centralizada no conceito de um "objeto" que é similar aos objetos da

```
<rulemodule xmlns="http://...">
... <rule processing-point="4">
     <property name="Content-type" context="res-hdr"
                                          matches="html/">   (a)
       <property name="Network-traffic" context="system"
                                          matches="full">    (b)
         <property name="Image-user-preference"
                  context="system" matches="removal">...(c)
           <service name="HtmlRemoval">
            <uri>icap://www.htmladapt.com</uri>          (d)
            <parameter name="action" type="static">
              <value>removal-images</value> ...          (e)
</rulemodule>
```

Figura 16: Exemplo de especificação de regra de adaptação em IRML

programação orientada a objetos. No caso de P, objeto é essencialmente um pedaço de dado ou informação. Porém a linguagem P não tem facilidades para criação de tipos de objetos e somente os objetos conhecidos pelo interpretador podem ser utilizados.

Os objetos são adicionados ao interpretador através de módulos, sendo que a linguagem P define dois módulos essenciais: *Core* e *Services*. O primeiro é utilizado para manipular objetos do tipo inteiro e *strings*. Já o módulo *Services* define dois métodos: *Services.findOne(URI)*, que busca e retorna o serviço especificado em *URI*, ou retorna falha caso esse serviço não exista; e *Services.applyOne(service, ...)*, que executa o serviço especificado em *service* e opcionalmente fornece parâmetros para relacionados à aplicação do serviço. No entanto pode-se adicionar ao interpretador módulos que possuam outros tipos de objetos, por exemplo um módulo que manipule os cabeçalhos das mensagens HTTP (*Http.message.headers*).

3.2 Protocolos para Adaptação de Conteúdo

A principal contribuição das redes de serviços é a definição dos módulos de adaptação de conteúdo, que se comunicam com os dispositivos de borda para realizar os serviços de adaptação sobre as requisições e respostas de conteúdo. O dispositivo de borda, através de uma política de adaptação implementada, define quais as adaptações necessárias e as solicitam a esses módulos.

Entretanto para solicitar essas adaptações e receber o conteúdo adaptado, deve-se utilizar um protocolo que viabilize a comunicação entre o dispositivo de borda e o módulo de adaptação de conteúdo. Visto que essa comunicação possui certas propriedades específicas, criaram-se os protocolos ICAP e OCP.

3.2.1 Internet Content Adaptation Protocol (ICAP)

A partir do conceito de distribuir as adaptações de conteúdo, um grupo de empresas e indústrias entenderam a necessidade de integrar suas tecnologias com o intuito de criar uma padronização para que diferentes *redes de serviços* consigam trabalhar em conjunto. Com isso seria possível a criação de um sistema unificado para adaptação de conteúdo. Como resultado dessa integração nasceu o ICAP (*Internet Content Adaptation Protocol*).

O protocolo ICAP [3, 21] tem por objetivo possibilitar a comunicação entre dispositivos de borda e servidores de adaptação, visando realizar a adaptação de conteúdo o mais próximo possível do usuário, o que propicia uma melhor personalização desse conteúdo aliada a ganhos de tempo de resposta. O ICAP foi criado através de um projeto iniciado em 1999 e conduzido por um grupo, denominado Fórum ICAP, constituído por aproximadamente 70 empresas [2].

Os principais objetivos desse fórum são: tornar os conteúdos mais flexíveis para os usuários; oferecer um padrão de comunicação comum e aberto para as aplicações dos dispositivos de borda manipularem diferentes serviços de valor agregado; e transferir a execução de serviços de adaptação para servidores dedicados. Para atingir esses objetivos o protocolo ICAP deve ser simples, utilizar uma infra-estrutura existente, usar padrões e métodos de comunicação existentes e ter serviços modularizados, isto é, os serviços devem ser capazes de ser adicionados ou retirados sem afetar a arquitetura interna ou o desempenho do sistema.

Assim, o ICAP é um protocolo desenvolvido para transferir serviços específicos para servidores dedicados, liberando recursos e padronizando a maneira com que esses serviços são implementados. Por exemplo, um servidor dedicado que realiza somente escaneamento de vírus é inerentemente mais eficiente que um servidor de origem, pois este deve realizar várias tarefas adicionais, compartilhando e eventualmente sobrecarregando seu processamento.

Em [21] são apresentados outros benefícios desse protocolo, tais como: ICAP é um protocolo aberto e permite a qualquer servidor ou provedor de aplicação implementá-lo; ICAP é baseado no protocolo HTTP, permitindo seu acesso através de barreiras de segurança; e ICAP simplifica a implementação, confiabilidade e escalabilidade dos serviços de valor agregado.

Na arquitetura ICAP o dispositivo de borda é denominado Cliente ICAP e o servidor de adaptação, Servidor ICAP. Esse protocolo é considerado "leve" com cabeçalhos de requisição e de resposta similares aos cabeçalhos utilizados pelo HTTP (denominados *MIME-Types*). O Cliente ICAP gera uma requisição ICAP através de um *Uniform Resource Identifier* (URI), encapsulando os cabeçalhos e o conteúdo HTTP. No ICAP há uma função *preview* cujo conteúdo

[2] www.i-cap.org

HTTP é removido parcialmente ou totalmente, permitindo que o Servidor ICAP possa desistir da transação antes de ter recebido todo o conteúdo a ser adaptado [21]. Há dois modos de operação definidos para esse protocolo: modificação de requisição (*reqmod*) e modificação de resposta (*respmod*).

No *reqmod*, conforme apresentado na Figura 17, o usuário envia uma requisição ao servidor de origem. Essa requisição é interceptada por um Cliente ICAP, que redireciona essa requisição para um Servidor ICAP. Esse servidor pode então:

- Retornar uma versão modificada da requisição. Em seguida o Cliente ICAP pode executar a requisição modificada contatando o servidor de origem ou encaminhar essa requisição para outro Servidor ICAP realizar uma nova modificação;
- Modificar a requisição de tal forma que esta aponte para uma página contendo uma mensagem de erro ao invés do URI original; ou
- Retornar uma resposta HTTP encapsulada que indica um erro.

O Cliente ICAP deve ser capaz de manusear esses três tipos de respostas. Se o Servidor ICAP falhar, o Cliente ICAP pode retornar um erro ao usuário, ou enviar ao servidor de origem a requisição como esta chegou do usuário, ou ainda reenviar uma solicitação de adaptação ao Servidor ICAP. Uma aplicação para o *reqmod* é a filtragem de conteúdo, na qual o Servidor ICAP pode bloquear requisições de conteúdos "impróprios". A Figura 17 mostra um típico fluxo de dados no *reqmod*, onde:

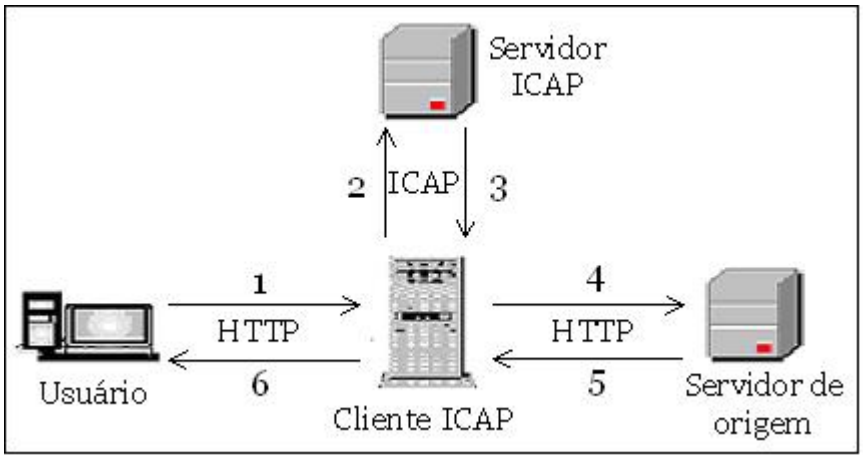


Figura 17: Modificação da requisição do usuário

1. O usuário faz uma requisição ao Cliente ICAP (dispositivo de borda) de um conteúdo do servidor de origem.

2. O Cliente ICAP envia uma requisição de adaptação ao Servidor ICAP.
3. O Servidor ICAP executa o serviço de adaptação e retorna ao Cliente ICAP uma requisição modificada ou uma resposta à requisição original. Se o Servidor ICAP retorna uma resposta, então o fluxo vai diretamente ao passo 6.
4. O Cliente ICAP envia a requisição, possivelmente diferente da original, ao servidor de origem.
5. O servidor de origem responde à requisição.
6. O Cliente ICAP envia a resposta (seja do servidor de origem ou do Servidor ICAP) para o usuário.

Já o modo de operação *respmod* visa um pós-processamento de respostas HTTP antes de entregá-las a um usuário. Nesse modo, a requisição do usuário é processada pelo servidor de origem. A resposta desse servidor é interceptada pelo Cliente ICAP, que a redireciona para um Servidor ICAP. Este pode então:

- Retornar uma versão modificada da resposta.
- Retornar um erro.

É no *respmod* que é possível realizar adaptações sobre os conteúdos provenientes dos servidores de origem. Um exemplo de aplicação desse modo é a formatação de uma página HTML para dispositivos que possuam tamanho de tela reduzido. A Figura 18 mostra um típico fluxo de dados no *respmod*, onde:

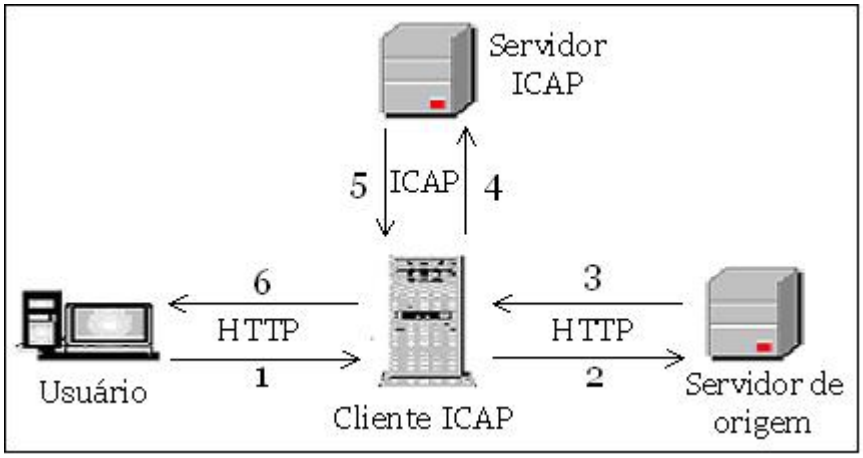


Figura 18: Modificação da reposta do servidor de origem

1. O usuário faz uma requisição ao Cliente ICAP (dispositivo de borda) de algum conteúdo do servidor de origem.
2. O Cliente ICAP envia essa requisição ao servidor de origem.
3. O servidor de origem responde à requisição.
4. O Cliente ICAP envia a resposta do servidor de origem ao Servidor ICAP.
5. O Servidor ICAP executa o serviço de adaptação sobre a resposta do servidor de origem e envia ao Cliente ICAP a resposta possivelmente modificada.
6. O Cliente ICAP envia a resposta possivelmente modificada ao usuário.

A arquitetura ICAP é composta de dois componentes principais: *semântica de transações*, que descreve a comunicação entre o Cliente ICAP e o Servidor ICAP, ou seja como o cliente solicita adaptação ao servidor; e *política de adaptação*, que especifica o tipo de adaptação a ser realizada, quando e a quem solicitá-la. Porém na RFC 3507 [3] esse segundo componente não é definido. Cabe salientar que a semântica de transações, embora absolutamente necessária, torna-se de pouca serventia sem uma política de adaptação definida.

O formato das mensagens trocadas entre o Cliente ICAP e o Servidor ICAP segue a gramática BNF (Backus-Naur *Form*) definida em [3] e apresentada no Apêndice A deste trabalho. No modo de operação *reqmod*, o Cliente ICAP captura o cabeçalho HTTP da requisição do usuário e insere o cabeçalho ICAP, formando assim uma mensagem ICAP de requisição, conforme ilustrada na Figura 19(a).

REQMOD icap://icap-server.ufscar.br/ ICAP/1.0 Host: **icap-server.ufscar.br** Encapsulated: req-hdr=0, null-body=170 GET / HTTP/1.1 Host: **www.origin-server.com** Accept: text/html, text/plain Accept-Encoding: compress Cookie: ff39fk3jur@4ii0e02i If-None-Match: "xyzzy", "r2d2xxxx" (a)	ICAP/1.0 200 OK Date: Mon, 10 Jan 2000 09:55:21 GMT Server: ICAP-Server-Software/1.0 Encapsulated: req-hdr=0, null-body=231 GET / HTTP/1.1 Host: **www.other-server.com** Via: 1.0 icap-server.ufscar.br Accept: text/html, text/plain, image/gif Accept-Encoding: gzip, compress If-None-Match: "xyzzy", "r2d2xxxx" (b)

Figura 19: Reqmod - (a) Mensagem ICAP de requisição; (b) Mensagem ICAP de resposta

Nesse exemplo o conteúdo requisitado pelo usuário é a página Web *www.origin-server.com* e, a partir dessa requisição, o Cliente ICAP solicita uma filtragem de conteúdo ao servidor

icap-server.ufscar.br. A mensagem de resposta desse Servidor ICAP é apresentada na Figura 19(b), sendo que a requisição sofreu modificações e a página Web a ser solicitada é *www.other-server.com*.

No *respmod* o Cliente ICAP captura o cabeçalho HTTP e o conteúdo da resposta do servidor de origem, e adiciona a esses dados o cabeçalho ICAP, obtendo uma mensagem ICAP de requisição, como apresentado na Figura 20(a). Nesse exemplo o Cliente ICAP solicita uma adaptação ao servidor *c-adapter.ufscar.br*, o qual realiza a adaptação sobre o conteúdo e retorna ao Cliente ICAP uma mensagem ICAP de resposta, ilustrada na Figura 20 (b).

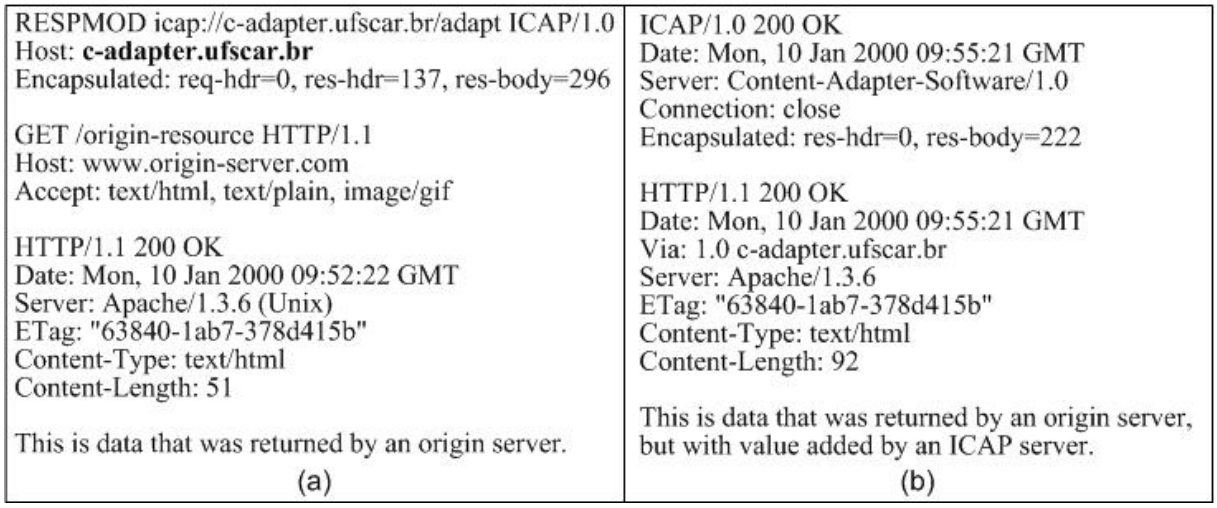


Figura 20: Respmod - (a) Mensagem ICAP de requisição; (b) Mensagem ICAP de resposta

A principal limitação da arquitetura ICAP consiste no fato desta não definir uma política de adaptação. Supondo um sistema que ofereça várias adaptações de conteúdo, se todo conteúdo transmitido ao Cliente ICAP for enviado para todos os Servidores ICAP desse sistema, o processo de adaptação certamente aumentará consideravelmente o tempo de entrega do conteúdo ao usuário. Além disso em [12], Nurmela apresentou outras necessidades de desenvolvimento que deveriam ser adicionadas ou estendidas na arquitetura ICAP:

- O ICAP oferece adaptações somente em dispositivo de borda baseados no protocolo HTTP (*HTTP-proxies*). Entretanto a infra-estrutura da Internet inclui *proxies* baseados em outros protocolos como RTP (*Real-time Transport Protocol*), SOAP (*Simple Object Access Protocol*), SIP (*Session Initiation Protocol*), SMTP (*Simple Mail Transfer Protocol*) e FTP (*File Transfer Protocol*);

- O HTTP provavelmente será usado como "carcaça" (*substrate*) para futuros protocolos da Internet que o ICAP não suportará [29];

- Visando gerenciar operacionalmente a complexidade produzida pelos vários mecanismos de adaptação de conteúdo, a arquitetura do sistema deve suportar *tracing* e *notification*; e

- Visto que essa adaptação interrompe uma comunicação fim a fim (entre usuário e servidor de origem), existe a necessidade de evoluir alguns conceitos "filosóficos", tais como a infração aos direitos reservados do conteúdo a ser adaptado, direito de liberdade de expressão e direito à privacidade, entre outros.

3.2.2 OPES Callout Protocol (OCP)

O protocolo OCP foi desenvolvido pelo grupo OPES WG, visando desempenhar a comunicação entre o dispositivo de borda e o módulo de serviço de adaptação. Inicialmente esse grupo definiu as exigências que deveriam ser atendidas por esse protocolo [30] e recentemente esse mesmo grupo definiu o "núcleo" do OCP [31].

OCP é um protocolo da camada de aplicação e necessita, portanto, de um protocolo de transporte que ordene suas mensagens e que ofereça confiabilidade. Entre as características consideradas no projeto do OCP incluem: codificação dos dados, transporte e formato das mensagens, inicialização da troca de mensagens e acordo das confirmações necessárias [12].

Com o intuito de suportar vários protocolos da camada de aplicação (e.g., HTTP, RTP, FTP e SMTP), o OCP foi dividido em duas camadas, representadas na Figura 21: OCP *Core*, que define suas funcionalidades; e *Protocol Bindings*, que oferece extensões necessárias para as funcionalidades dos protocolos da camada de aplicação. Das várias extensões previstas na camada *Protocol Bindings*, até o presente momento somente a *HTTP-binding* foi definida [26].

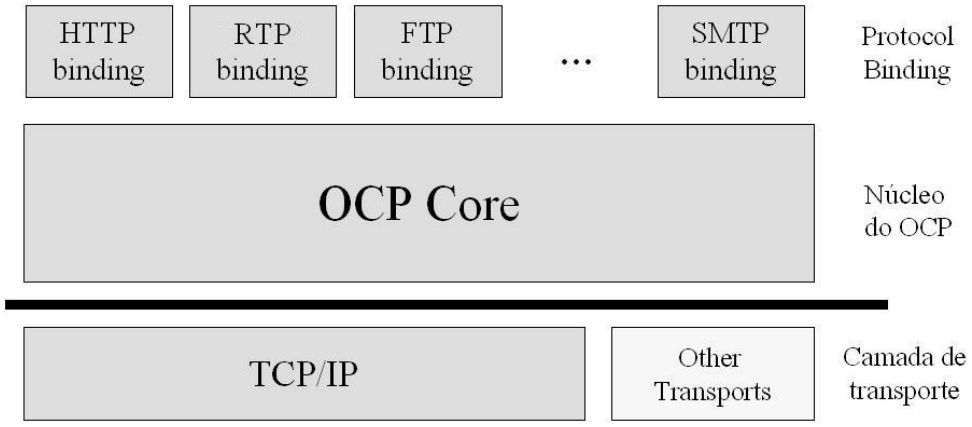


Figura 21: Arquitetura em camadas do OCP

3.3 Trabalhos Correlatos

Em meados da década de 90, com a proliferação do uso da Internet, os primeiros trabalhos surgiram com a preocupação em entregar conteúdos adaptados aos usuários dessa rede. Dentre os possíveis pontos de adaptação de conteúdo, esses trabalhos realizavam os serviços de adaptação no próprio dispositivo de borda.

No trabalho apresentado em [32] são realizadas adaptações sobre imagens, visando entregar esse conteúdo para dispositivos com diferentes características como processamento, armazenamento e dimensões de tela. Esse trabalho realiza uma classificação das imagens e, a partir dessas classes, desempenha a adaptação para cada dispositivo.

Em [33] as adaptações são realizadas para disponibilizar páginas Web a dispositivos móveis. Essas adaptação são executadas por um *proxy*, localizado entre o usuário e o servidor de origem, que foi denominado de *Mowser*.

Entretanto o acúmulo das adaptações de conteúdo pode sobrecarregar o dispositivo de borda. Essa sobrecarga praticamente obriga essas arquiteturas a oferecerem um único tipo de adaptação [32, 33]. Com o surgimento das *redes de serviços*, a tarefa de adaptar um conteúdo é transferida para servidores dedicados (módulos de serviços de adaptação), viabilizando a integração de vários serviços numa mesma arquitetura. Baseados nas redes de serviços, no modelo OPES e no protocolo ICAP, surgiram vários trabalhos com o intuito de prover os serviços de adaptação de conteúdo.

Em [24] definiu-se as *Content Services Networks* (CSNs), que seguem o mesma princípio das redes OPES de introduzir servidores remotos que se comunicam com os dispositivos de borda das CDNs, realizando os serviços de valor agregado. Sendo que nas redes CSNs os servidores de adaptação são denominados *Application Proxy*.

O diferencial das CSNs em relação às redes OPES é que o *Application Proxy* também pode interagir diretamente com servidores de origem e usuários. Para essa comunicação, em [24] foi proposto a criação de um novo protocolo, o *Internet Service Delivery Protocol* (ISDP). A Figura 22 mostra a arquitetura das redes CSNs.

Além do protocolo ICAP, nas redes CSNs especula-se a utilização do protocolo SOAP [34] para realizar a comunicação entre o dispositivo de borda e o servidor de adaptação. O SOAP é um protocolo baseado na linguagem XML que visa a troca de informações num ambiente distribuído. Entretanto, em [35] foi realizada uma comparação entre o SOAP e o ICAP, onde concluiu-se que este último é mais eficiente. Além disso, em [36] foi realizado um estudo para

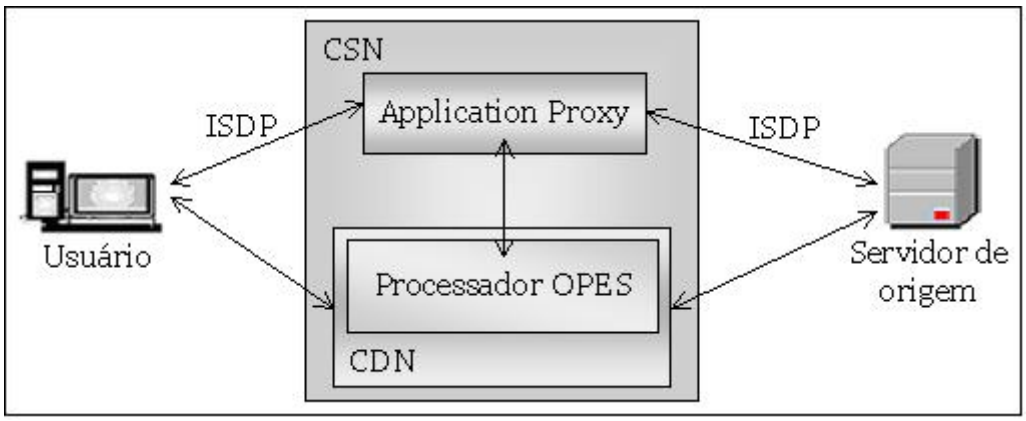


Figura 22: Arquitetura das redes CSNs

avaliar se o ICAP segue as exigências propostas pelo grupo de trabalho OPES, onde concluiu-se que o mesmo preenche a maioria dessas exigências. O ICAP foi desenvolvido especificamente para a adaptação de conteúdo, o que não ocorre com o SOAP.

Em [37] é apresentada uma arquitetura aberta e flexível que permite a adaptação de conteúdo de forma inteligente, ou seja através de uma política de adaptação. Essa arquitetura enfatiza a representação e processamento de regras de adaptação para a invocação dos serviços de adaptação. A Figura 23 apresenta essa arquitetura que é baseada na arquitetura das redes OPES, porém com inserção dos componentes *Ferramenta de Regras* e *Cache*.

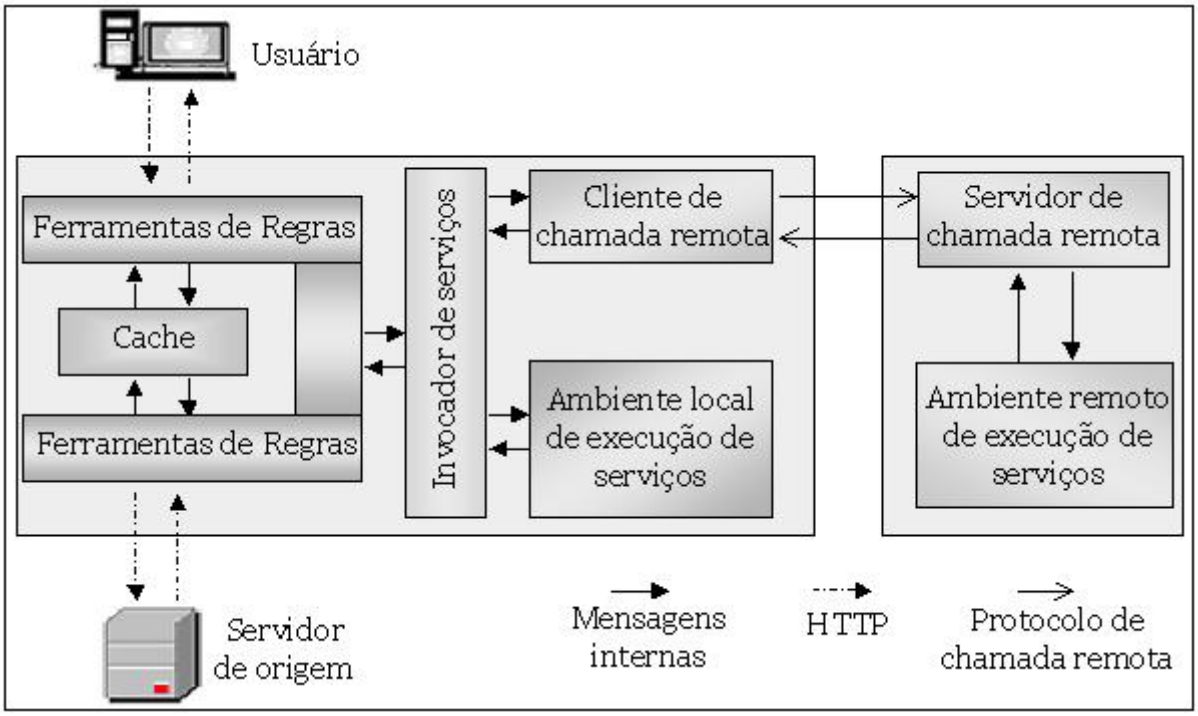


Figura 23: Arquitetura com *Ferramenta de Regras* e *Cache*

O *Cache* armazena temporariamente os conteúdos mais acessados pelos usuários, depen-

dendo da política de *cache* adotada. Esse componente já é bastante utilizado nos *proxies* que não oferecem serviços de adaptação, sendo que o diferencial dessa arquitetura é o componente *Ferramenta de Regras*.

Uma vez que nem todo conteúdo requisitado deve sofrer as adaptações de conteúdo oferecidas pelo dispositivo de borda, é necessário um mecanismo que invoque determinados serviços somente se certas condições forem satisfeitas. Alternativamente, esta decisão poderia ser realizada pelo servidor de adaptação, mas isso exigiria que todas requisições passem por todos os servidores de adaptação disponíveis e conseqüentemente seria introduzido um grande atraso na entrega do conteúdo. Uma solução encontrada é a implementação desse mecanismo no componente *Ferramenta de Regras*, que avalia se o conteúdo satisfaz às regras de adaptações (previamente oferecidas pelos servidores de adaptação) e que solicita as adaptações quando necessárias. Além disso, esse componente deve oferecer uma interface aos servidores de adaptação para que estes possam inserir, modificar ou retirar suas regras de adaptação.

Nessa arquitetura também é utilizado um *Invocador de Serviços* que recebe a adaptação solicitada por *Ferramenta de Regras* e que invoca essa adaptação diretamente ao *Ambiente local de execução de serviços* ou ao *Ambiente remoto de execução de serviços* através de um protocolo de chamada remota.

Outra arquitetura para adaptação foi descrita [38], que insere o conceito de preferências do usuário e de perfil de dispositivo, além de considerar a possível existência de vários serviços num mesmo servidor de adaptação. Ainda em [38], foi desenvolvido um modelo de análise de desempenho para adaptações de imagens utilizando servidores dedicados, que será explorado no capítulo 6.

Outro trabalho equivalente é encontrado em [35], cuja arquitetura também contém componentes de *cache* e tratamento das regras de adaptação. Entretanto, nessa arquitetura é inserido um componente que gerencia quais serviços estão correntemente disponíveis, assegurando que as adaptações serão solicitadas a servidores de adaptação realmente ativos.

Em [39] é apresentado um *framework* para gerenciamento de personalização dos conteúdos. Esse *framework* foi construído baseado no modelo OPES, com a adição dos componentes *Gerenciador de Serviços* e *Servidor de Autorização*, representados na Figura 24.

Quando a requisição de conteúdo chega ao dispositivo de borda, este envia uma requisição de autorização ao *Gerenciador de Serviços* que a repassa ao *Servidor de Autorização*. Este por sua vez consulta o repositório de perfis para determinar os serviços que o usuário está subscrito e autorizado a receber e retorna uma resposta de autorização ao *Gerenciador de Serviços*. Nessa

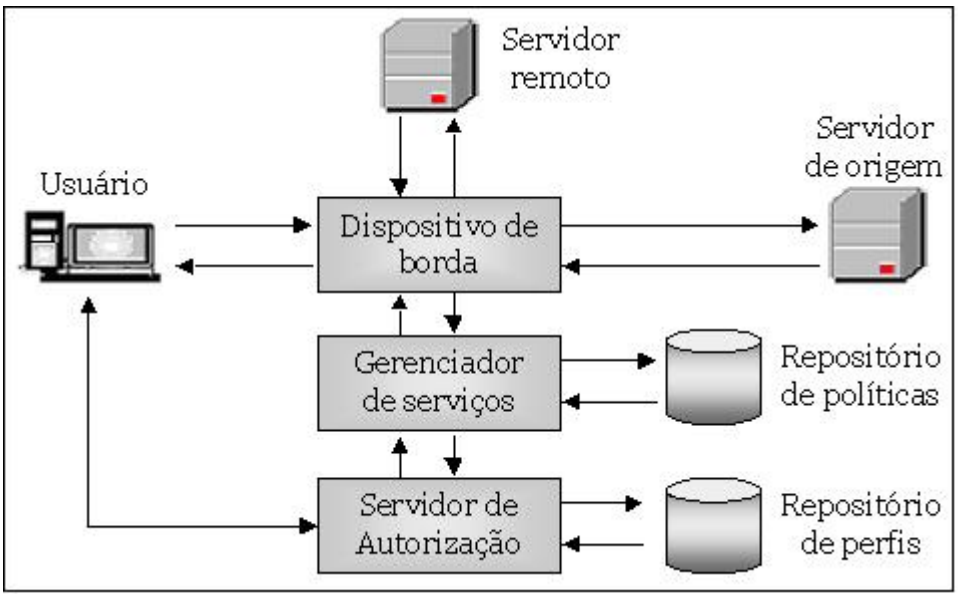


Figura 24: Arquitetura com o *Gerenciador de Serviços* e o *Servidor de Autorização*

resposta são indicados os serviços do usuário, suas preferências e o perfil do seu dispositivo. O *Gerenciador de Serviços* combina as preferências do usuário, o perfil do dispositivo, as informações do conteúdo, a política de serviço do servidor de origem e os serviços disponíveis com o intuito de gerar uma política de regras para execução dos serviços de adaptação. Em seguida, o *Gerenciador de Serviços* envia uma resposta de autorização ao dispositivo de borda contendo essa política de regras.

O dispositivo de borda recupera o conteúdo do servidor de origem e analisa esse conteúdo através da política de adaptação, invocando um ou mais serviços de adaptação se necessário. Após o último servidor de adaptação retornar o conteúdo modificado, este é entregue ao usuário.

Outro trabalho dessa área foi descrito em [40], que apresenta adaptações direcionadas apenas a dispositivos móveis e que oferece adaptações de imagem, vídeo e compressão de textos. Nesse trabalho também foram utilizadas informações sobre a rede de acesso para decidir qual a melhor adaptação a ser realizada. Nesse trabalho os perfis de usuário e dispositivo são armazenados nos próprios dispositivos móveis e isso pode sobrecarregá-los, visto que esses dispositivos têm recursos limitados.

4 *Arquitetura para Adaptação de Conteúdo na Internet*

Apesar dos avanços dos modelos de redes que visam oferecer uma estrutura de adaptação de conteúdo na Internet, esses modelos ainda contêm limitações principalmente no âmbito da política de adaptação. Um aspecto que deve ser considerado é a definição das informações utilizadas no processo de tomada de decisão sobre a adaptação a ser realizada. Contudo essa arquitetura visa suprir as necessidades detectadas nos trabalhos anteriores e estimular o uso dos serviços de adaptação.

A arquitetura proposta neste trabalho é baseada no modelo OPES e utiliza o ICAP para realizar a comunicação entre o dispositivo de borda e o servidor de adaptação. Essa arquitetura pode ser considerada genérica, visto que a mesma é implementada para suportar e desempenhar serviços para quaisquer tipos de dispositivos, redes de acesso e usuários. A flexibilidade da arquitetura é garantida, pois os serviços de adaptação podem ser adicionados ou removidos da arquitetura sem que esta mude sua estrutura.

A arquitetura proposta, resumida na Figura 25, é constituída de dois componentes principais: o *Adaptation Proxy*, que realiza as funções do *processador OPES*; e o *Adaptation Server*, que realiza as funções do *módulo de serviços de adaptação*.

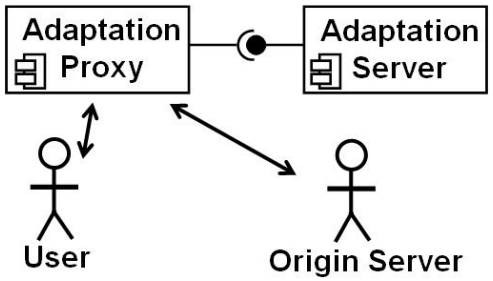


Figura 25: Arquitetura para adaptação de conteúdo na Internet

Essa arquitetura suporta adaptações nos dois modos de operação do ICAP: *reqmod* e *respmod* e também aceita a inclusão de vários *Adaptation Servers*, aumentando o escopo de adaptações disponíveis ao usuário sem sobrecarregar o dispositivo de borda.

A grande carência encontrada nos trabalhos anteriores foi a implementação de uma política de adaptação. Portanto essa arquitetura, através do *Adaptation Proxy*, define uma política de adaptação orientada pelas regras de adaptação e baseada nos perfis de usuário, de dispositivo e de contrato de serviços, além das informações sobre a rede de acesso e o conteúdo.

A modelagem e o desenvolvimento da arquitetura basearam-se em componentes de software, visando criar padrões que facilitam seu reuso em outras arquiteturas semelhantes. Embora se saiba que a construção de componentes genéricos para um domínio de aplicações não é uma tarefa simples, aceitou-se o desafio de pesquisar seu emprego na arquitetura proposta. Como resultado, além de gerar maior coesão do código, obteve-se uma primeira versão de componentes para o domínio de arquiteturas que suportam adaptação de conteúdo.

4.1 Componentes da Arquitetura

Além dos componentes principais, definiu-se outros componentes que estão contidos no *Adaptation Proxy* e no *Adaptation Server*. O ***Adaptation Proxy***, que é constituído pelos componentes apresentados na Figura 26, intercepta as requisições do usuário e as respostas do servidor de origem e, em função de uma política de adaptação, solicita as adaptações pertinentes.

O componente ***Content Transfer Protocol*** é responsável pela comunicação com o usuário e com o servidor de origem, sendo genérico para suportar os protocolos de aplicação utilizados na Internet para transmissão de conteúdo, tais como HTTP, RTP, SOAP e FTP.

Aas solicitações de adaptação são feitas pelo *Adaptation Proxy* ao *Adaptation Server* através de um protocolo que realiza a comunicação entre o dispositivo de borda e o módulo de serviços. Assim, foi definido o componente ***Callout Protocol Client*** que realiza essa chamada remota de adaptação. Esse componente suporta diferentes protocolos, assim como os protocolos ICAP [3] e OCP [31], criados especificamente para esse tipo de comunicação.

Outro componente é o ***Cache***, que foi construído visando melhorar o desempenho da arquitetura. Esse componente armazena temporariamente os conteúdos requisitados da Internet (e.g., páginas Web, vídeos ou imagens). Antes de solicitar um conteúdo ao servidor de origem, o *Adaptation Proxy* verifica se o mesmo está em *Cache*. Caso positivo, a arquitetura evita uma requisição ao servidor de origem e o conteúdo é entregue mais rapidamente ao usuário.

Apesar de utilizar os módulos de serviços de adaptação, a arquitetura proposta oferece suporte à execução de adaptações no próprio dispositivo de borda, através do componente ***Local Adapter***. Este utiliza os recursos desse dispositivo, logo o uso em grande escala do mesmo

afetará o desempenho do *Adaptation Proxy*, retardando a entrega do conteúdo.

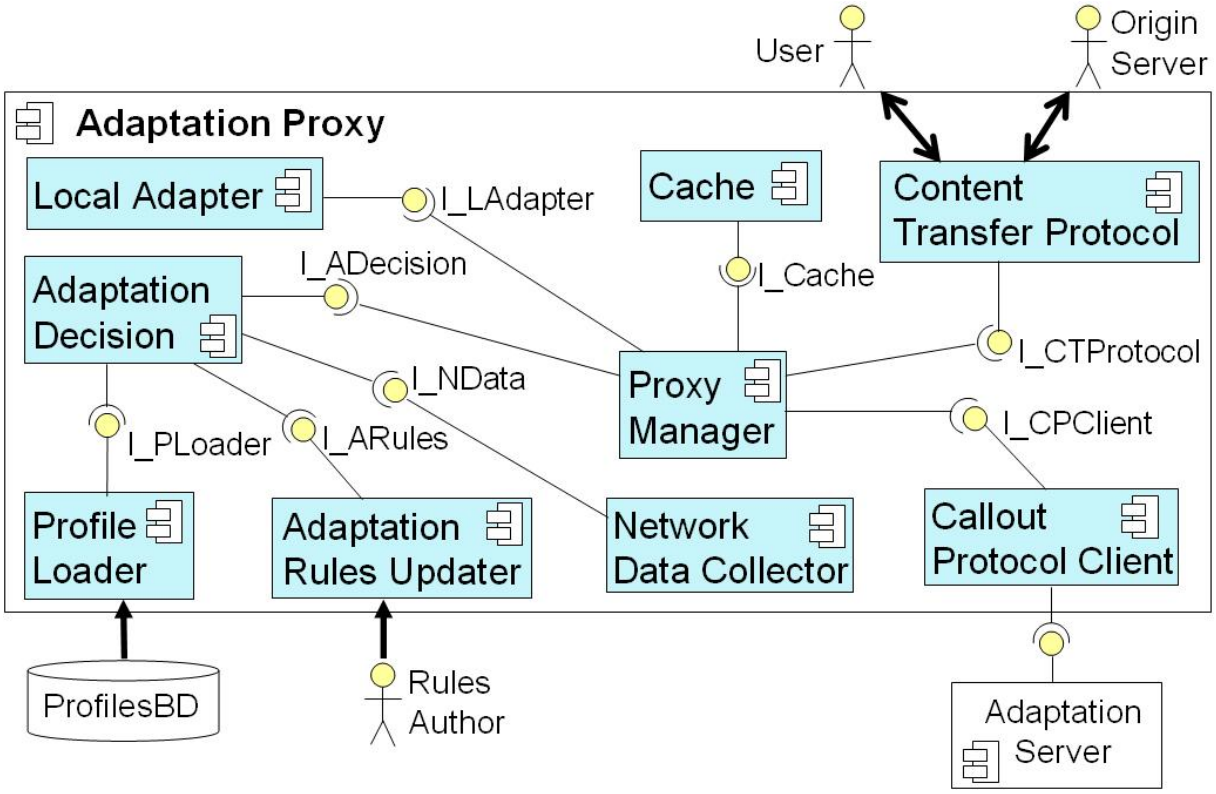


Figura 26: Componentes do *Adaptation Proxy*

O componente ***Proxy Manager*** gerencia o fluxo de dados do *Adaptation Proxy*. Através do *Content Transfer Protocol*, esse componente troca mensagens com o usuário e com o servidor de origem. De posse do conteúdo, o *Proxy Manager* requisita uma análise de adaptação ao componente *Adaptation Decision*. E, caso uma adaptação seja necessária, o *Proxy Manager* invoca esse serviço remotamente através do componente *Callout Protocol Client* ou localmente através do componente *Local Adapter*. Além disso, antes de requisitar um conteúdo, o *Proxy Manager* verifica se o mesmo está presente no *Cache*, visando eliminar requisições desnecessárias.

Um importante requisito da adaptação de conteúdo é a definição da política de adaptação. Na arquitetura deste trabalho, essa política é orientada por regras de adaptação e baseada nas informações do usuário, dispositivo, rede de acesso, conteúdo e contrato de serviços. A política de adaptação dessa arquitetura é implementada através dos componentes *Adaptation Decision*, *Profile Loader*, *Adaptation Rules Updater* e *Network Data Coletor*.

O componente ***Profile Loader*** busca os perfis de usuário, de dispositivo e de contrato de serviços armazenados na base de dados *ProfilesDB*. Esses perfis permitem à arquitetura satisfazer as preferências e os interesses dos usuários, além de atender às capacidades e limitações dos dispositivos e determinar quais os serviços que o usuário está habilitado a receber. Para

inserir os perfis de usuário e de dispositivo na base *ProfilesDB*, deve ser disponibilizada uma interface ao usuário da Internet. Essa interface pode ser uma página Web que contenha um formulário, onde os perfis são preenchidos em campos pré-estabelecidos. Já o perfil de contrato de serviços é inserido pelo *administrador* do sistema, também através de uma interface.

O componente ***Network Data Coletor*** monitora e coleta as informações da rede de acesso. Esse componente oferece uma interface, pela qual o *Adaptation Decision* solicita as modificações que ocorreram nas características da rede de acesso.

O controle da política de adaptação da arquitetura é realizado pelo ***Adaptation Decision***. Esse componente, após receber o conteúdo a ser analisado através do *Proxy Manager*, verifica as modificações na rede de acesso e busca os perfis de usuário, dispositivo e de contrato de serviços na base de perfis. De posse dessas informações, o *Adaptation Decision* utiliza as regras de adaptação para decidir quais adaptações deverão ser executadas, qual servidor (seja local ou remoto) que realizará cada adaptação e, caso seja necessário desempenhar várias adaptações, determina a ordem de execução das mesmas.

Para assegurar que regras desatualizadas ou inválidas não sejam processadas, o ***Adaptation Rules Updater*** oferece as atualizações dessas regras. O autor das regras (*Rules Author* pode inserir, remover ou modificar as regras de adaptação no componente *Adaptation Rules Updater*. Esse autor de regras pode ser um servidor de adaptação, servidor de origem ou administrador do dispositivo de borda.

Seguindo todo o processo descrito acima, o *Adaptation Proxy* pode solicitar adaptações remotas ao outro componente principal da arquitetura, o ***Adaptation Server***. Considerando que os servidores de adaptação podem realizar diferentes tipos de serviços, cada *Adaptation Server* poderá ter uma estrutura interna diferente. Contudo baseado nas idéias de componentes de software, foi definida uma estrutura genérica para os diversos *Adaptation Servers*, composta pelos componentes apresentados na Figura 27.

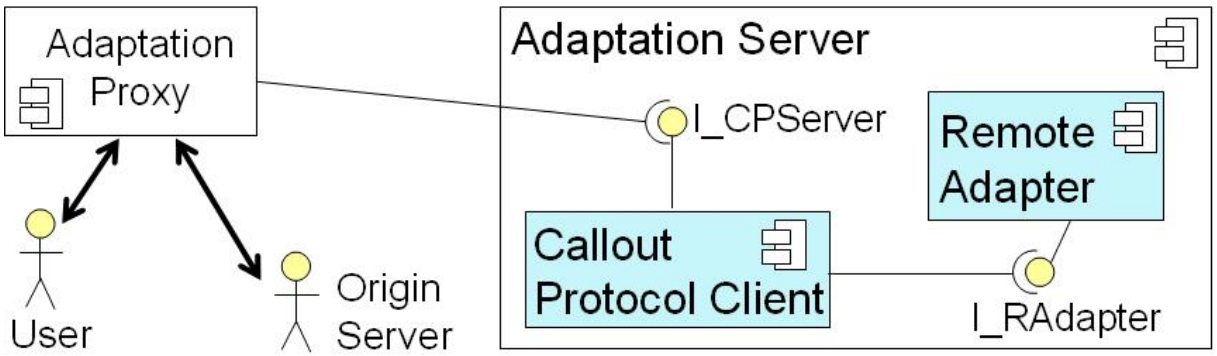


Figura 27: Componentes do *Adaptation Server*

O componente ***Callout Protocol Server*** viabiliza a comunicação com o *Adaptation Proxy*. Esse componente recebe e analisa a solicitação de adaptação do *Callout Protocol Client*, obtendo a ação a ser realizada e seus parâmetros. A partir desses dados o *Callout Protocol Server* invoca a adaptação solicitada ao componente *Remote Adapter*. Esse componente também suporta diferentes protocolos como ICAP e OCP.

O ***Remote Adapter*** é responsável pela execução das adaptações de conteúdo. A estrutura interna desse componente suporta variações conforme o tipo de adaptação oferecido pelo *Adaptation Server*, sendo que um exemplo de modelagem desse componente pode ser encontrado em [41]. O *Remote Adapter* realiza as mesmas funções do *Local Adapter*, porém não utiliza os recursos de processamento do *Adaptation Proxy*. A decisão de adaptar o conteúdo localmente ou remotamente deve considerar o tempo gasto com o *Callout Protocol* em relação ao tempo gasto com a adaptação do conteúdo. Quanto menor essa relação, mais favorável será a adaptação remota.

4.2 Comportamento da Arquitetura

Seguindo as definições de [3], a arquitetura deste trabalho foi construída para suportar dois modos de operação: modificação da requisição (*reqmod*) e modificação da resposta (*respmod*). No *reqmod* a adaptação é executada sobre a requisição que o usuário envia ao servidor de origem. Já o *respmod* visa o processamento das respostas dos servidores espalhados pela Internet antes de entregá-las ao usuário.

Durante o fluxo de requisições e respostas dos conteúdos, existem quatro pontos onde a política de adaptação pode ser processada, conforme mostra a Figura 28 [42]. Nos pontos 1 e 2 a política é processada sobre a requisição do usuário antes ou após a busca do conteúdo em *Cache* respectivamente. Nos pontos 3 e 4, executa-se a política sobre a resposta do servidor de origem, respectivamente, antes ou após seu armazenamento em *Cache*. A definição do ponto de processamento da política depende do serviço de adaptação a ser realizado. Por exemplo, um serviço de antivírus deve ser executado no ponto 3, evitando assim que um conteúdo contaminado seja armazenado em *Cache*.

Figura 28: Pontos de processamento da política de adaptação

O componente *Adaptation Decision* exporta em sua interface o método *policy.request*, que contém um parâmetro que indica o ponto de processamento da análise de adaptação. A Figura 29 mostra um modelo de seqüência do caso de uso que realiza uma adaptação da resposta do servidor de origem (*respmod*). Os casos de uso da arquitetura são apresentados no Apêndice B.

Figura 29: Modelo de seqüência do *Adaptation Proxy* numa adaptação *respmod*

Nesse modelo o componente *Proxy Manager* solicita ao *Content Transfer Protocol*, através do método *cTP.receiveRequest*, que aguarde uma requisição do usuário (*User*). Quando essa requisição é realizada, o *Proxy Manager* solicita um processamento da política de adaptação (*policy.request*) ao *Adaptation Decision*, enviando os identificadores do usuário, dispositivo e da rede de acesso, além de fornecer o conteúdo, que nesse caso é a requisição do usuário. Baseado nesses dados o componente *Adaptation Decision* busca os perfis (*profile.find*) e determina se será realizado um serviço de adaptação. No caso mostrado na Figura 29, não existe

essa necessidade. Dessa forma o *Proxy Manager* verifica sem sucesso se o conteúdo original está armazenado em *Cache* (*cache.search*), solicitando em seguida outra análise de adaptação (ponto 2). Uma vez que não é necessário adaptar a requisição do usuário, o conteúdo original é solicitado ao servidor de origem através de *cTP.sendRequest* e sua resposta é coletada através de *cTP.receiveResponse*.

Em seguida o *Proxy Manager* invoca uma análise de adaptação referente ao ponto de processamento 3. Nesse caso *Adaptation Decision* retorna uma adaptação remota a ser executada. Então o *Proxy Manager* solicita essa adaptação ao componente *Callout Protocol Client* (*cPC.invoke*), indicando o endereço do servidor de adaptação (*url*), a ação a ser executada (*action*) e seus parâmetros (*parameters*), além do próprio conteúdo (*content*). O conteúdo adaptado é armazenado pelo *Proxy Manager* em *Cache* (*cache.write*). Em seguida o mesmo solicita outra análise de adaptação (*policy.request* - ponto 4), que nesse caso não retorna uma adaptação a ser realizada. Finalmente o conteúdo adaptado é retornado ao usuário (*cTP.sendResponse*).

No modelo de seqüência da Figura 30, o elemento *Adaptation Proxy* solicita um serviço (*serviceRequest*) ao componente *Callout Protocol Server*, que a recebe através do método *calloutProtocol.receive*, juntamente com o cabeçalho do protocolo e o conteúdo a ser adaptado. O *Callout Protocol Server* analisa esse cabeçalho, extraindo deste a ação a ser executada e seus parâmetros, e invoca (*service.invoke*) a adaptação ao componente *Remote Adapter*. Após a realização da adaptação, o conteúdo modificado é encapsulado numa mensagem de resposta e retornado ao *Adaptation Proxy* através do método *calloutProtocol.send*.

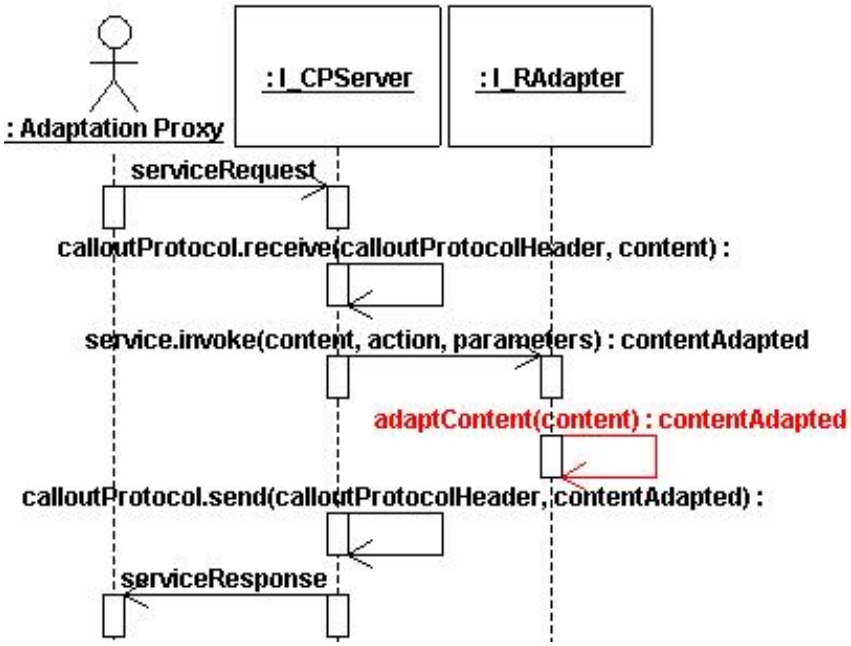


Figura 30: Modelo de seqüência do *Adaptation Server*

O modelo de seqüência da Figura 31 representa o comportamento da arquitetura quando o conteúdo requisitado está armazenado em *Cache* e a adaptação é realizada localmente pelo *Local Adapter*. Nesse modelo o *Proxy Manager* requisita o método *cache.search* ao *Cache*, que retorna o conteúdo solicitado pelo usuário, evitando uma requisição ao servidor de origem.

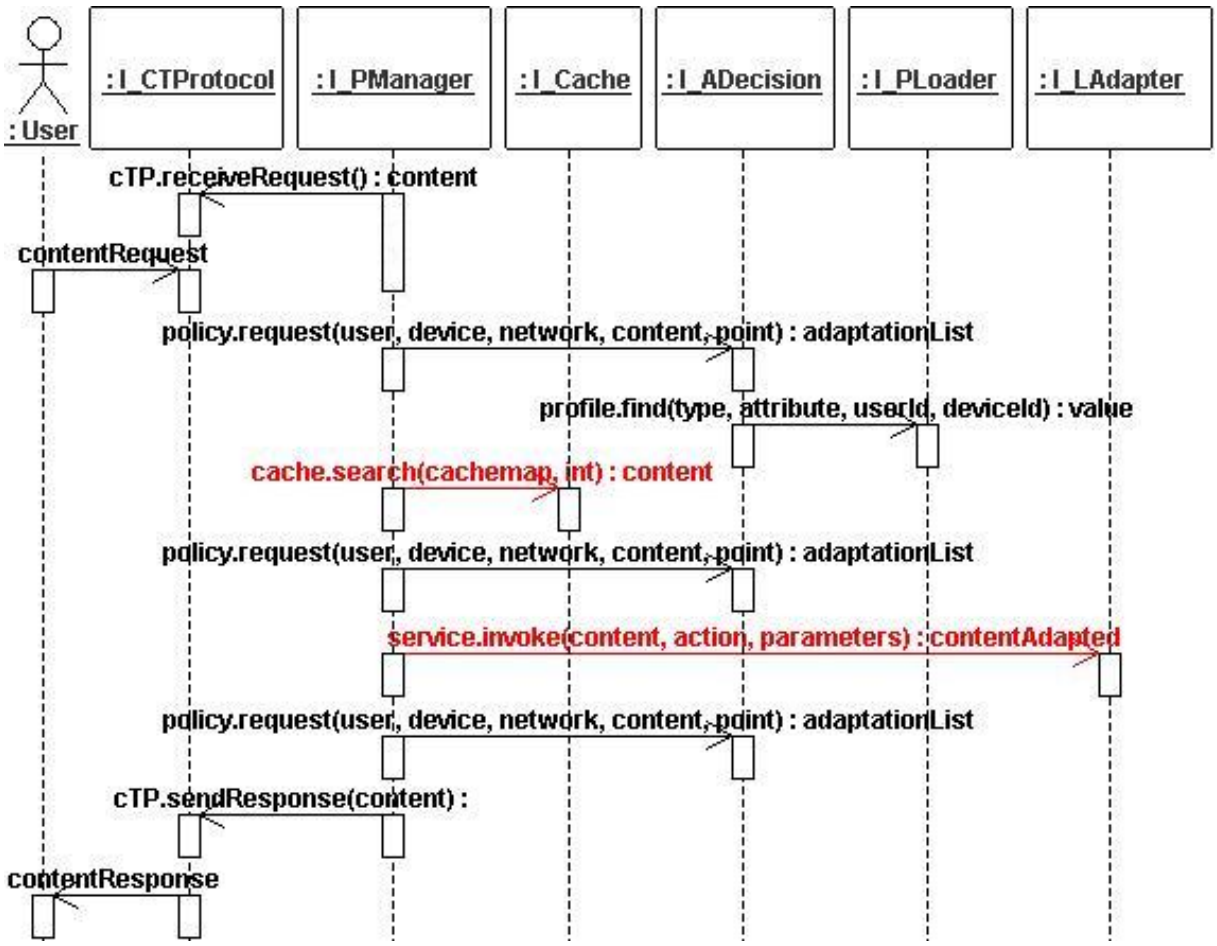


Figura 31: Modelo de seqüência do sucesso em Cache e adaptação local

Após essa busca em *Cache*, o *Proxy Manager* solicita uma análise de adaptação ao *Adaptation Decision* através do método *policy.request*. Essa solicitação se refere ao ponto de processamento 3, visto que o conteúdo requisitado pelo usuário foi encontrado em *Cache*. Essa análise de adaptação retorna um serviço a ser realizado localmente, então o *Proxy Manager* solicita esse serviço diretamente ao componente *Local Adapter* através do método *service.invoke*.

Percebe-se nesse modelo que o fluxo de dados se restringiu à comunicação entre o *User* e o *Adaptation Proxy*. O *Origin Server* não foi solicitado pois o conteúdo original estava presente em *Cache* e o *Adaptation Server* não foi requisitado porque o serviço de adaptação foi realizado localmente.

4.3 Política de Adaptação

A política de adaptação especificada nessa arquitetura compreende a definição dos serviços de adaptação que serão executados, dos adaptadores locais ou remotos que realizarão essas adaptações, de quando as adaptações poderão ser solicitadas e da ordem de suas execuções.

Para que essas decisões sejam tomadas com eficiência é necessário conhecer as informações relativas ao ambiente de adaptação, assim como as preferências e os dados pessoais dos usuários, as características e capacidades de seus dispositivos, as características dos conteúdos requisitados, as condições e capacidades da rede de acesso e o contrato de serviços entre o usuário e o provedor de serviços.

4.3.1 Perfis

A adaptação de conteúdo é orientada por informações que visam atender às reais necessidades de adaptação. Um meio de disponibilizar essas informações é através de perfis. Neste trabalho foram definidos os perfis de usuário, dispositivo e de contrato de serviços.

O **perfil de usuário** contém dados pessoais do usuário e suas preferências de adaptação. A Tabela 4 apresenta o perfil de usuário adotado na arquitetura, contendo seus atributos e uma descrição para cada atributo. Com base nesse perfil, o processo de adaptação é direcionado para que seu resultado esteja em conformidade com as predileções do usuário, visto que diferentes usuários podem requerer diferentes adaptações para um mesmo conteúdo. Adaptações que não se baseiam nas preferências do usuário podem ser inconvenientes ou indesejadas.

Essas informações do perfil de usuário são importantes para manter a satisfação do mesmo. Por exemplo numa transmissão de vídeo onde a largura de banda da rede de acesso se encontra estreita, se o usuário habilita a redução da resolução do vídeo, a arquitetura realizará essa adaptação, garantindo uma melhor entrega do conteúdo e satisfazendo a preferência do usuário.

No **perfil de dispositivo** estão contidas características e capacidades de hardware e software do dispositivo. O conhecimento dessas capacidades é fundamental para o processo de adaptação, que considera as limitações de cada dispositivo. A Tabela 5 apresenta o perfil de dispositivo utilizado na arquitetura, contendo os atributos e suas descrições.

Um exemplo de adaptação determinado pelo perfil de dispositivo pode ser descrito como: se o conteúdo requisitado é uma página HTML e o dispositivo somente suporta páginas WML, caso típico de certos telefones celulares, então a arquitetura solicitará uma adaptação da página HTML para WML, caso esse serviço esteja disponível.

Tabela 4: Perfil de usuário

Atributo	Descrição
Dados Pessoais	
name	Nome do usuário
email	Endereço eletrônico do usuário
age	Idade do usuário
occupation	Profissão do usuário
city, state e country	Localização geográfica do usuário
Adaptação de Imagens (o usuário pode habilitar ou desabilitar as adaptações)	
imageGrayScale	Redução das cores da imagem para escala de cinza
imageReduction	Redução do tamanho da imagem
imageDownResolution	Redução da resolução da imagem
imageRemoval	Remoção da imagem
Adaptação de Vídeos	
videoGrayScale	Redução das cores do vídeo para escala de cinza
videoReduction	Redução do tamanho do vídeo
videoDownResolution	Redução da resolução do vídeo
videoRemoval	Remoção do vídeo
Adaptação de Som	
soundDownQuality	Redução da qualidade do som
soundRemoval	Remoção do som
Adaptação de e-mails	
attachmentRemoval	Remoção dos arquivos anexos no e-mail
firstLines	Apresenta apenas as primeiras linhas do e-mail
justSubject	Apresenta apenas o assunto do e-mail
justSender	Apresenta apenas o remetente do e-mail
Adaptação de páginas Web	
backgroundRemoval	Remove o plano de fundo de uma página
languageTranslation	Traduz a página para o idioma local
Outras adaptações	
scanVirus	Predileção por escaneamento de vírus
adInsertion	Permissão para inserção de propaganda

Em se tratando dos perfis de dispositivos móveis, foi criado um repositório denominado UAProf [1], onde são armazenados vários perfis desses dispositivos e disponibilizados na Internet. O objetivo desse repositório é incentivar os fabricantes a criarem os perfis dos seus dispositivos para que estes sejam utilizados pelos sistemas de adaptação.

No **perfil de contrato de serviços** constam os serviços oferecidos pela arquitetura para cada usuário. Caso esses serviços sejam cobrados através de taxas, o provedor de serviços pode oferecer diferentes planos de serviços, conforme as necessidades e os recursos de cada usuário. Os serviços que estão disponíveis na arquitetura podem ser bloqueados ao usuário, caso estes não estejam contidos no seu perfil de contrato de serviços. A Tabela 6 mostra o perfil de contrato

[1] http://w3development.de/rdf/uaprof_repository/

Tabela 5: Perfil de dispositivo

Atributo	Descrição
Capacidades de Hardware	
deviceType	Tipo do dispositivo
cpu	Capacidade de processamento do dispositivo
memory	Memória do dispositivo
displayWidth	Largura da tela do dispositivo
displayHeight	Altura da tela do dispositivo
color	Configuração de cores suportadas pelo dispositivo
inSound	Suporte à entrada de som
outSound	Suporte à saída de som
model	Modelo do dispositivo
vendor	Fabricante do dispositivo
Capacidades de Software	
systemOperational	Sistema operacional do dispositivo
browser	Modelo do navegador do dispositivo
imageCapable	Capacidade de exibir imagem
frameCapable	Capacidade de exibir *frames*
tableCapable	Capacidade de exibir tabelas
codecsVideo	Formatos de vídeos suportados pelo dispositivo
codecsAudio	Formatos de áudios suportados pelo dispositivo
javaAppletEnabled	Suporte à *applets* Java
htmlEnable	Suporte ao HTML
wmlEnable	Suporte ao WML

de serviços adotado na arquitetura proposta.

O perfil de contrato de serviços pode ser útil para habilitar ou desabilitar as adaptações de conteúdo. Por exemplo, se em determinada ocasião um conteúdo deve sofrer uma verificação de vírus, porém o usuário não paga a taxa referente a esse serviço, então o escaneamento não será realizado.

- **Linguagem para representação dos perfis**

Com o objetivo de padronizar a representação desses perfis, foi utilizado a linguagem CC/PP (*Composite Capabilities/Preferences Profile*) [43]. A utilização de um padrão facilita a coleta desses perfis e a sua migração entre entidades diferentes. O CC/PP é baseado em XML e criado pela W3C (*World Wide Web Consortium*) para descrever capacidades de dispositivos e preferências do usuário. A escolha de uma linguagem baseada em XML oferece facilidades como flexibilidade, independência de plataforma, grande aceitação para armazenamento, recuperação e troca de informações.

O CC/PP define um *framework* que deve ser preenchido com informações que descrevem

Tabela 6: Perfil de contrato de serviços

Atributo	Descrição
planChoice	Plano de serviços
bandwidth	Largura de banda contratada
dataFlow	Fluxo de dados contratado
videoAdaptation	Suporte à adaptação de vídeo
soundAdaptation	Suporte à adaptação de som
languageTranslation	Suporte à tradução de idioma
imageadaptation	Suporte à adaptação de imagem
emailAdaptation	Suporte à adaptação de e-mail
virusScanning	Suporte à escaneamento de vírus
contentFiltering	Suporte à filtragem de conteúdo
adsInsertion	Suporte à inserção de propaganda

os perfis. Essas informações são definidas através de um vocabulário. Os vocabulários são documentos que definem nomes, domínios, tipos de valores e outras informações sintáticas e semânticas relacionadas aos termos particulares de uma aplicação. Um vocabulário é utilizado para definir os termos particulares de cada perfil (e.g., *memory* para o perfil de dispositivo).

Um perfil CC/PP é estruturado como uma árvore de dois níveis, sendo que o primeiro nível é composto pelos componentes e o segundo nível, pelos atributos que os descrevem. A Figura 32 mostra um perfil CC/PP que faz uma pequena descrição de um dispositivo do tipo *desktop*, o qual possui os atributos *cpu*, *displayWidth* e *displayHeight*, que possuem respectivamente os valores *200 MHz*, *768 pixels* e *1024 pixels*.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:dev="http://www.example.com/schema#">
  <rdf:Description rdf:about="http://www.example.com/profile#MyDevProfile">
    <ccpp:component>
      <rdf:Description rdf:about="http://www.example.com/profile#TerminalHardware">
  (1) <rdf:type rdf:about="http://www.example.com/schema#Desktop"/>
  (2) <dev:cpu> 200 </dev:cpu>
  (3) <dev:displayWidth> 1024 </dev:displayWidth>
  (4) <dev:displayHeight> 768 </dev:displayHeight>
      </rdf:Description>
    </ccpp:component>
  </rdf:Description>
</rdf:RDF>
```

Figura 32: Perfil de dispositivo em CC/PP

A utilização da linguagem CC/PP nessa arquitetura também é valorizada devido à pos-

sível integração com o repositório UAProf, onde os perfis de dispositivos são representados em CC/PP. Assim a arquitetura apresentada pode se valer do uso desses perfis que são gratuitamente disponibilizados na Internet.

Além dos perfis, a política de adaptação dessa arquitetura analisa as informações dinâmicas, que são coletadas em cada processo de decisão sobre a adaptação a ser realizada, ou seja, a cada requisição e resposta de conteúdo. Visto que essas informações sofrem alterações freqüentemente, torna-se desvantajoso o armazenamento das mesmas numa base de dados e sua representação numa linguagem, como é feito com os perfis.

4.3.2 Rede de Acesso e Conteúdo

As informações dinâmicas utilizadas nessa arquitetura se referem à rede de acesso do usuário e ao conteúdo. A **rede de acesso** compreende o meio de comunicação entre o usuário e o dispositivo de borda. Visto que o usuário pode utilizar diferentes meios de acesso (e.g., sem fio, a cabo, ADSL, via fibra ótica), faz-se necessário coletar as informações dessa rede, visando adequar a entrega do conteúdo requisitado às suas capacidades.

A decisão sobre certas adaptações de conteúdo depende do estado corrente das capacidades da rede de acesso, as quais podem ser alteradas ao decorrer do tempo, por isso a importância de desenvolver um sistema de coleta dinâmica dessas informações. Na arquitetura apresentada essa função é realizada pelo componente *Network Data Coletor*.

A análise da rede de acesso pode ser bastante útil principalmente em redes de acesso com taxas de transmissão instáveis, como as redes sem fio. As informações sobre essa rede podem evitar uma alta taxa de erros em transmissões de vídeo, como demonstrado em [40]. A Tabela 7 apresenta os parâmetros da rede de acesso analisados na política de adaptação dessa arquitetura.

Tabela 7: Informações sobre a rede de acesso

Parâmetro	Descrição
bandwidth	Largura de banda
trasmitionRate	Taxa de transmissão
numberOfUsers	Números de usuários da rede
latency	Tempo de latência da rede

Além da rede de acesso, as informações sobre o **conteúdo** podem ser utilizadas na análise da adaptação. Cada conteúdo traz consigo detalhes que podem ser importantes no processo de adaptação. Como esses dados se alteram a cada conteúdo requisitado, faz-se necessário a coleta dinâmica dos mesmos. Se a comunicação entre o servidor de origem e o *Adaptation Proxy* for realizada através do HTTP, então algumas informações referentes ao conteúdo podem ser

encontradas no cabeçalho desse protocolo e outras podem ser retiradas dos seus *metadados*, assim como os dados "<HEAD>" das páginas HTML ou o cabeçalho das imagens e vídeos. A Tabela 8 mostra as características de conteúdo utilizadas na arquitetura apresentada.

Tabela 8: Informações sobre o conteúdo

Característica	Descrição
Conteúdo de requisição	
server	Servidor de origem
url	Endereço do destino
Conteúdo de resposta	
author	Criador do conteúdo (servidor de origem)
language	Idioma do conteúdo
size	Tamanho do conteúdo
format	Formato do conteúdo
dimensions	Dimensões para imagem ou vídeo
color	Escala de cores para imagem ou vídeo

4.4 Uma Experiência de Reuso da Arquitetura

A arquitetura deste trabalho foi desenvolvida com o objetivo de fornecer uma estrutura básica para o desenvolvimento das aplicações de adaptação de conteúdo na Internet através do reuso de seus componentes. Esta seção apresenta uma experiência de reuso da arquitetura proposta, representada no modelo de componentes da Figura 33. Nesse modelo têm-se os componentes reutilizados através da **instanciação direta**: *Local Adapter*, *Remote Adapter*, *Proxy Manager*, *Cache*, *Adaptation Rules Updater*, *Network Data Coletor* e *Profile Loader*.

Nessa experiência de reutilização da arquitetura, três componentes foram adaptados com implementações específicas de suas interfaces, caracterizando o reuso através de **especialização**. Assim o componente *Content Transfer Protocol* foi especializado para efetuar a transmissão das requisições e respostas de conteúdo através do *HTTP*. Esse protocolo é bastante utilizado na Internet para transmissão de páginas Web, imagens, arquivos de vídeos, páginas dinâmicas ou aplicações. O componente modificado contém um *parser* HTTP, que identifica as ações semânticas de cada mensagem recebida.

Os componentes *Callout Protocol Client* e *Callout Protocol Server* foram especializados para viabilizar a comunicação entre o dispositivo de borda e o servidor de adaptação através do *ICAP*. Ao receber uma solicitação de serviço, o componente *Callout Protocol Client* encapsula a adaptação, seus parâmetros e o conteúdo numa mensagem ICAP de requisição e a envia ao *Callout Protocol Server*. Este faz uma análise semântica dessa mensagem, retirando as

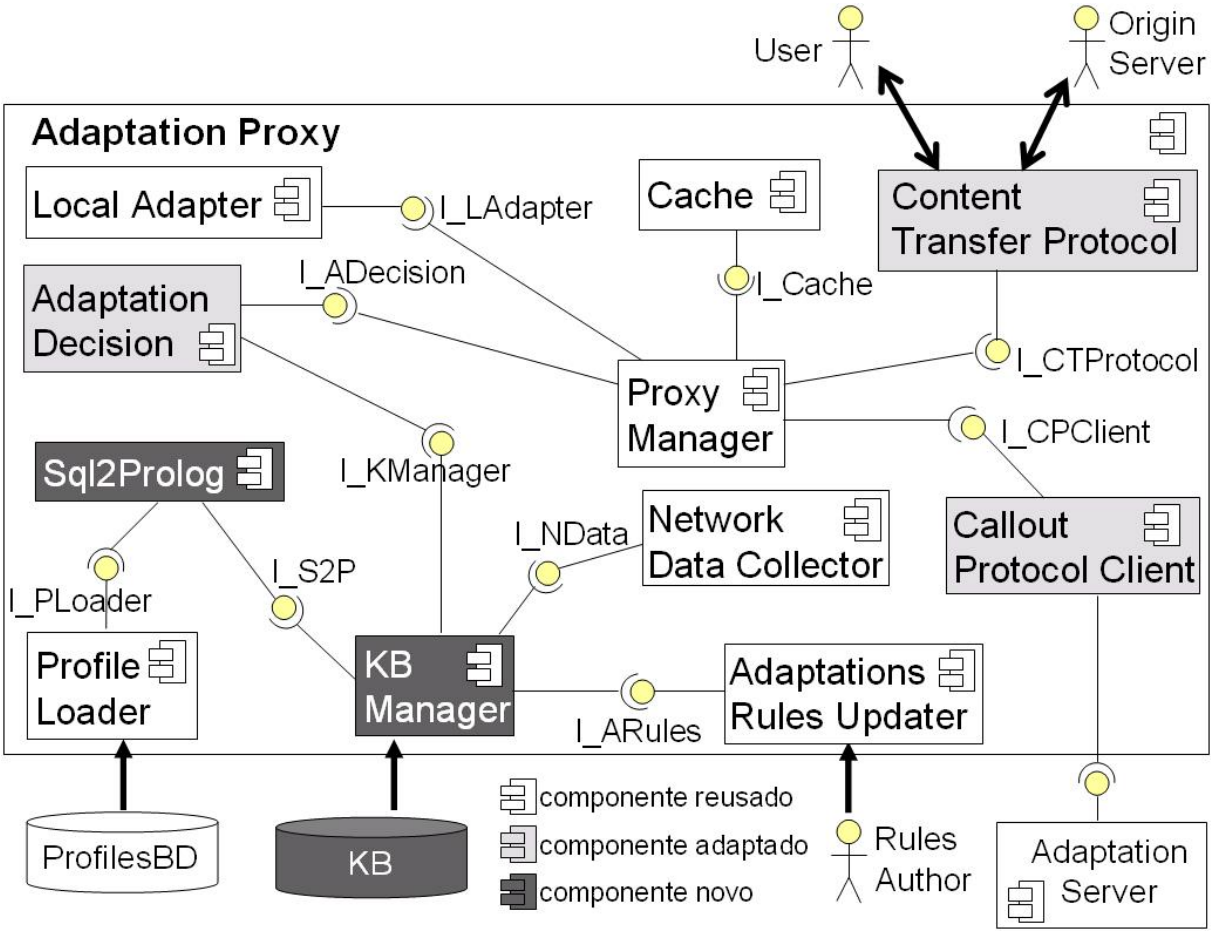


Figura 33: Reutilização da arquitetura proposta

informações necessárias e invoca o serviço solicitado ao *Remote Adapter*. O conteúdo, após ser adaptado, é encapsulado pelo *Callout Protocol Server* numa mensagem ICAP de resposta e retornado ao *Adaptation Proxy*.

Uma vez que a implementação da política de adaptação é a principal dificuldade encontrada nos sistemas de adaptação, essa arquitetura define que sua política deve considerar as regras de adaptação e as informações relativas ao ambiente de adaptação. Contudo, existem diversas formas de realizar esse processo de decisão, inclusive através da implementação de um algoritmo em linguagem procedimental. Assim, para atender este requisito específico da aplicação alguns componentes foram adicionados e outros modificados, sendo que nessa experiência o processo da política de adaptação é controlado por um mecanismo de inferência.

4.4.1 Inserção do Mecanismo de Inferência

A decisão da política de adaptação da arquitetura é de responsabilidade do componente *Adaptation Decision*. Nessa experiência este foi adaptado para trabalhar integrado com um

mecanismo de inferência que, baseado nas regras de adaptação e nas informações do ambiente, define quais serviços de adaptação devem ser realizados. Logo o *Adaptation Decision* define apenas a ordem de execução desses serviços, caso existam múltiplas adaptações, e determina o adaptador (local ou remoto) que realizará essas adaptações.

Nessa reutilização da arquitetura, o mecanismo de inferência foi introduzido através de uma base de conhecimento (*Knowledge Base* - KB), implementada em linguagem Prolog. Esse mecanismo fornece uma maior flexibilidade à política de adaptação da arquitetura e uma inteligência no processo de tomada de decisão.

Além da base de conhecimento, para o funcionamento desse mecanismo, foram adicionados dois novos componentes: *KB Manager* e *Sql2Prolog*. O ***KB Manager*** gerencia o funcionamento da base de conhecimento, realizando as seguintes funções: recebe do componente *Adaptation Rules Updater* as atualizações das regras de adaptação, convertendo-as para a linguagem Prolog e as inserindo na *KB*; recebe do componente *Network Coletor Data* as modificações das condições da rede de acesso, inserindo-as na *KB*; e quando solicitado pelo *Adaptation Decision* faz uma consulta de adaptação à *KB*, informando o usuário, dispositivo, conteúdo e rede de acesso que devem ser considerados.

Para responder a essa consulta, a *KB* utiliza os perfis de usuário, dispositivo e de contrato de serviços, além das informações sobre a rede de acesso e o conteúdo. Considerando que a *KB* está implementada em Prolog e que os perfis estão na base *ProfilesDB*, a qual utiliza a linguagem de consulta SQL, existe uma incompatibilidade entre essas bases. Dessa forma foi desenvolvido o componente ***Sql2Prolog***, que viabiliza a troca de informações entre essas bases, permitindo à *KB* receber e analisar os perfis armazenados na base *ProfilesDB*.

4.4.2 Regras de Adaptação

Além de utilizar as informações relativas ao usuário, dispositivo, contrato de serviços, rede de acesso e ao conteúdo, a política de adaptação da arquitetura deste trabalho é guiada por regras de adaptação, que indicam as condições a serem atendidas para que determinada adaptação seja realizada. Nessa experiência de reuso da arquitetura, as regras de adaptação foram implementadas como cláusulas na *KB* e utilizam o mecanismo de inferência do Prolog para deduzir as ações adotadas em função das condições a serem satisfeitas.

A consulta das regras de adaptação na *KB* é realizada pelo *KB Manager*. Como este está implementado em C/C++ e a *KB* é implementada em Prolog, foi criada uma interface C++/Prolog. Essa interface está localizada no próprio *KB Manager* e foi implementada baseada no manual

do GNU Prolog [44].

A consulta às regras de adaptação pode ser efetuada nos quatro pontos de processamento da arquitetura, conforme definido em [42]. Cada ponto de processamento é representado de forma diferente na base de regras, permitindo ao *KB Manager* solicitar a análise de adaptação à *KB* num ponto específico. Por exemplo a Figura 34 mostra um exemplo de implementação de regra de adaptação a ser invocada no ponto 3 (*point_three*). Nessa regra são fornecidos o identificador do usuário (*UserID*), o identificador do dispositivo (*DeviceID*), o contrato entre usuário e provedor (*ContractID*) e o tipo do conteúdo (*Content*). O parâmetro *Ret* devolve o resultado (a adaptação a ser executada) ao *KB Manager*.

```
point_three(Ret,UserID,DeviceID,ContractID,Content):-
      contentIsText(Content),
      userPayforFilter(ContractID),
      userIsChild(UserID),
      =(Ret,' content-filter.gsdr.dc.ufscar.br filter').
...
userIsChild(UserID):-
       sql_query(UserID,List),
       search(5,List,Res),
       <(Age,18).
```

Figura 34: Regra de adaptação para filtragem de conteúdo

Na regra *point_three* têm-se três condições: o conteúdo deve ser texto (*contentIsText*), o usuário deve pagar pela filtragem de conteúdo (*userPayforFilter*) e o usuário deve ter idade inferior a 18 anos (*userIsChild*). Se essas condições forem satisfeitas a ação a ser executada é armazenada na variável *Ret*. Esta recebe o valores *content-filter.gsdr.dc.ufscar.br* e *filter* que indicam, respectivamente, o *Adaptation Server* e o serviço de adaptação.

Na Figura 34 também é demonstrado como uma condição é analisada. Por exemplo para verificar se o usuário tem menos que 18 anos (*userIsChild*), realiza-se uma consulta no perfil de usuário através da regra *sql_query*, sendo seu identificador fornecido através de *UserID* e o retorno da busca armazenado em *List*. Em seguida, é obtido dessa lista o valor do parâmetro idade (através de *search*) e então verifica-se se esse valor é menor que 18, *<(Age,18)*.

O Apêndice C mostra parte das regras de adaptação que foram implementadas na arquitetura. Cabe salientar que essas regras se tornam flexíveis visto que são implementadas em Prolog. Essa flexibilidade viabiliza um aprimoramento constante da política de adaptação da arquitetura, além de facilitar a manutenção dessa política quando novos serviços de adaptação forem adicionados à arquitetura.

4.4.3 Comportamento da Política de Adaptação

Com a inserção de novos componentes nessa experiência de reuso da arquitetura, o comportamento da sua política de adaptação sofreu alterações, uma vez que em seu processo de análise foi introduzido um mecanismo de inferência.

A nova interação entre os componentes dessa política é descrita no modelo de seqüência da Figura 35. Nesse modelo o *Proxy Manager* continua solicitando a análise de adaptação através do método *policy.request*, pertencente ao componente *Adaptation Decision*. Este por sua vez solicita ao *KB Manager* a decisão sobre quais as adaptações que deverão ser efetuadas através do método *analyse*. Em seguida, o *KB Manager* busca na base de conhecimento (*KB*) quais as adaptações são necessárias, indicando por parâmetro as informações sobre a rede de acesso e sobre o conteúdo, além dos identificadores de usuário e dispositivo.

A partir desses dados, a *KB* consulta os perfis de usuário, dispositivo e contrato de serviços usando o método *sql_query* do componente *Sql2Prolog*. Este solicita essa consulta ao *Profile Loader* através de *profile.find*, transformando os resultados da consulta SQL para a linguagem Prolog de forma que a *KB* possa manipulá-los. Em seguida, a *KB* envia o resultado da análise ao *KB Manager*, que retorna as adaptações necessárias ao *Adaptation Decision*. Este por sua vez finaliza a análise retornando as adaptações necessárias ao *Proxy Manager*.

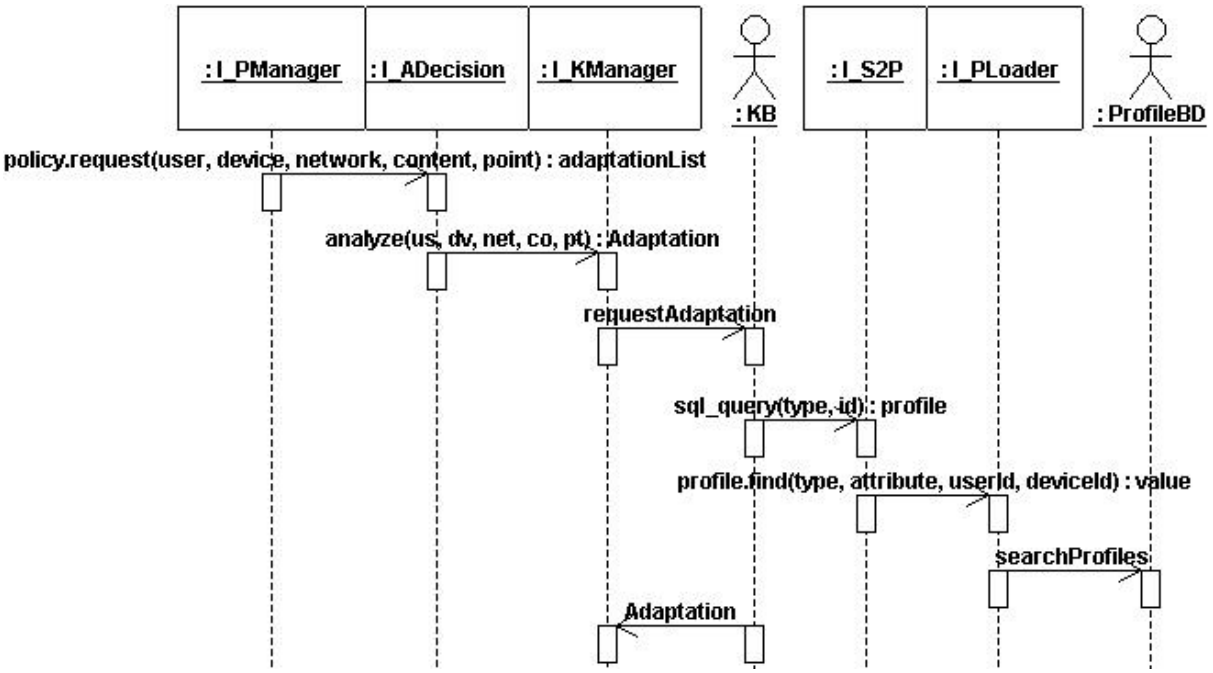


Figura 35: Modelo de seqüência da requisição da análise de adaptação

5 *Avaliação da Arquitetura*

Com o intuito de validar a arquitetura proposta, este trabalho também descreve uma avaliação de sua implementação. Nessa avaliação foram implementados os componentes definidos na reutilização da arquitetura, sendo utilizados portanto o protocolo ICAP para a comunicação entre o *Adaptation Proxy* e os *Adaptation Servers*, o protocolo HTTP para que o *Adaptation Proxy* se comunique com o usuário e com o servidor de origem e o mecanismo de inferência para implementar as regras de adaptação e processar a política de adaptação.

5.1 Estudo de Caso

Nos testes executados, objetivou-se coletar os tempos da entrega do conteúdo nas adaptações de páginas HTML e imagens. Esse estudo também avalia o *overhead* gerado pela política de adaptação e pelas próprias adaptações de conteúdo. Além disso foi realizado um teste de carga visando avaliar a escalabilidade da arquitetura.

A implementação do *Adaptation Proxy* foi baseada no Proxy Shweby [1] e executada na plataforma Linux. Shweby é um proxy HTTP de código aberto com suporte ao ICAP e sua principal desvantagem se refere a sua escalabilidade. Este não foi projetado para suportar um grande número de usuários, sendo que seu limite gira em torno de 20 requisições por segundo.

Nesse estudo foram implementados, em C/C++, três *Adaptation Servers*: o *Virus Scan* (VS), o *Content Filter* (CF) e o *Image Adapter* (IA). Os servidores VS e IA basearam-se no ICAP Server da Network Appliance [2], e o servidor CF foi desenvolvido em outro projeto do mesmo grupo deste trabalho [41]. Os servidores VS e CF foram testados no sistema operacional Windows e o servidor IA, no Linux.

O servidor *Virus Scan* suporta o escaneamento de conteúdos, visando encontrar arquivos contaminados. Quando é detectado um vírus no conteúdo, a resposta ICAP desse servidor

[1] http://shweby.sourceforge.net/
[2] http://www.i-cap.org/docs/

conterá uma página Web com uma mensagem informando o vírus encontrado. Para realizar a varredura nos conteúdos e detectar os vírus, o servidor VS utiliza a ferramenta ClamWin [3].

O servidor *Image Adapter*, realiza a remoção de imagens e redução nas dimensões das imagens. A primeira é necessária quando o dispositivo não suporta determinado tipo de imagem e a segunda, quando o dispositivo possui dimensões de tela reduzidas. A remoção das imagens é realizada sobre documentos HTML, de onde são retirados os *links* de suas imagens, evitando que o *browser* do usuário requisite essas imagens. Já as adaptações nas dimensões das imagens são executadas através da ferramenta Image Magick [4].

O servidor *Content Filter* analisa o conteúdo através da busca de palavras impróprias e verifica se a URL (*Unified Resource Locator*) do conteúdo pertence a uma classe de *sites* impróprios. O bloqueio do conteúdo baseia-se em informações contidas numa base de dados com palavras e URL's proibidas. Caso o conteúdo deva ser bloqueado, uma página Web é enviada ao usuário, informando o motivo do bloqueio.

Visando avaliar a arquitetura apresentada e as adaptações implementadas, executaram-se inicialmente serviços de adaptação sobre páginas HTML. A página *www.folha.com.br* (sem imagens), com tamanho de 23.991 bytes, foi utilizada para execução de 8 testes: sem adaptação (*No Adaptation* - NA); utilizando as adaptações dos *Adaptation Servers* disponíveis (CF, IA e VS); combinando dois desses servidores (VS+IA, VS+CF e CF+IA); e combinando os três servidores (CF+VS+IA). Para cada teste realizou-se 1000 requisições dessa página, extraindo-se a média e o desvio padrão do tempo da entrega desse conteúdo ao usuário. Esses resultados são apresentados na Figura 36.

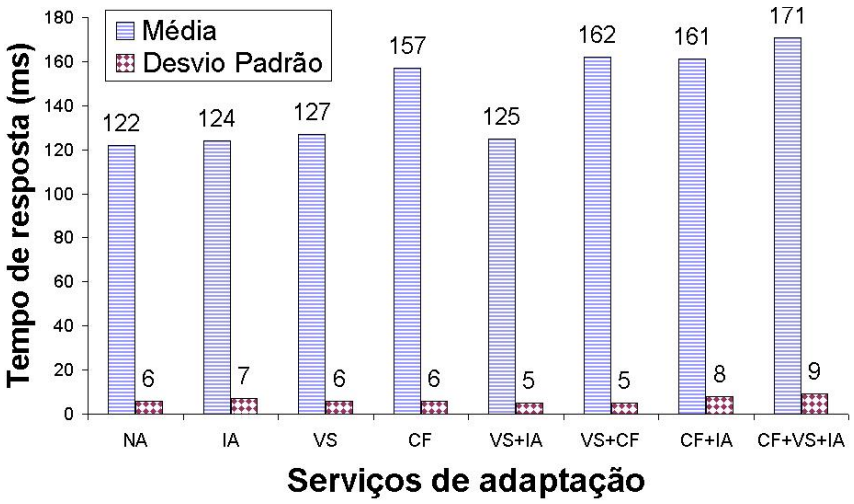


Figura 36: Adaptações de páginas HTML

[3]http://www.clamwin.com
[4]http://www.imagemagick.org

Adaptações de imagens também foram realizadas para avaliar a arquitetura. Nessas avaliações requisitou-se uma imagem da Internet com tamanho de 222.510 bytes e sobre esta foram realizados 5 testes: sem adaptação (NA); utilizando os *Adaptation Servers* (IA e VS); e combinando esses dois servidores, em ordens alternadas (VS+IA e IA+VS). A ordem das adaptações foi variada para verificar as diferenças de seus resultados no tempo final. Conforme a Figura 37, sobre a imagem foram realizadas 100 requisições obtendo-se a média e o desvio padrão do tempo total da entrega do conteúdo.

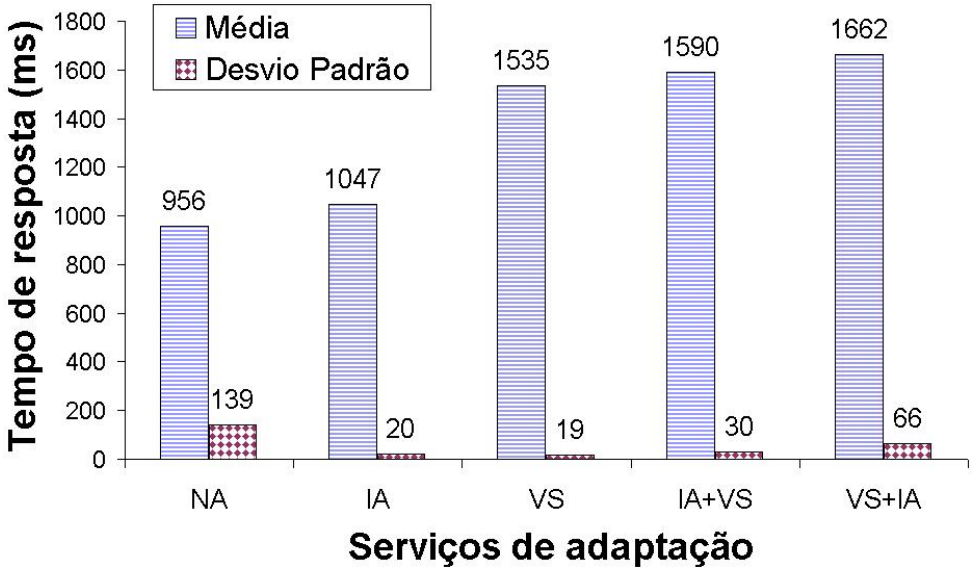


Figura 37: Adaptações de imagens

Visando expandir os casos analisados nesse estudo, foram realizados testes sobre outra página HTML. Neste caso foi selecionada página *www.google.com.br*, que possui um tamanho de 2.460 bytes. Além disso, nessa avaliação é exposto um histórico do tempo de entrega do conteúdo ao longo de 100 requisições da página, conforme mostra a Figura 38. Nesse estudo foram executados quatro testes: sem adaptação (NA); e utilizando as adaptações disponíveis (IA, VS e CF). As médias do tempo de entrega obtida nesses testes foram: 272 ms (milissegundos), 336 ms, 419 ms e 1672 ms, respectivamente.

Visto que este trabalho define uma política de adaptação, esta deve ocupar pouco tempo da entrega do conteúdo. Deve-se considerar também o tempo gasto com o protocolo ICAP e com a adaptação de conteúdo. Logo esse estudo realiza a avaliação do tempo gasto na adaptação de conteúdo e na política de adaptação, compreendendo o mecanismo de inferência adicionado.

Para avaliar o desempenho dos componentes dessa arquitetura e aferir o tempo gasto na adaptação de conteúdo e na política de adaptação, definiram-se certos pontos para medição dos tempos gastos num fluxo de requisição e resposta de conteúdo. A Figura 39 apresenta esses

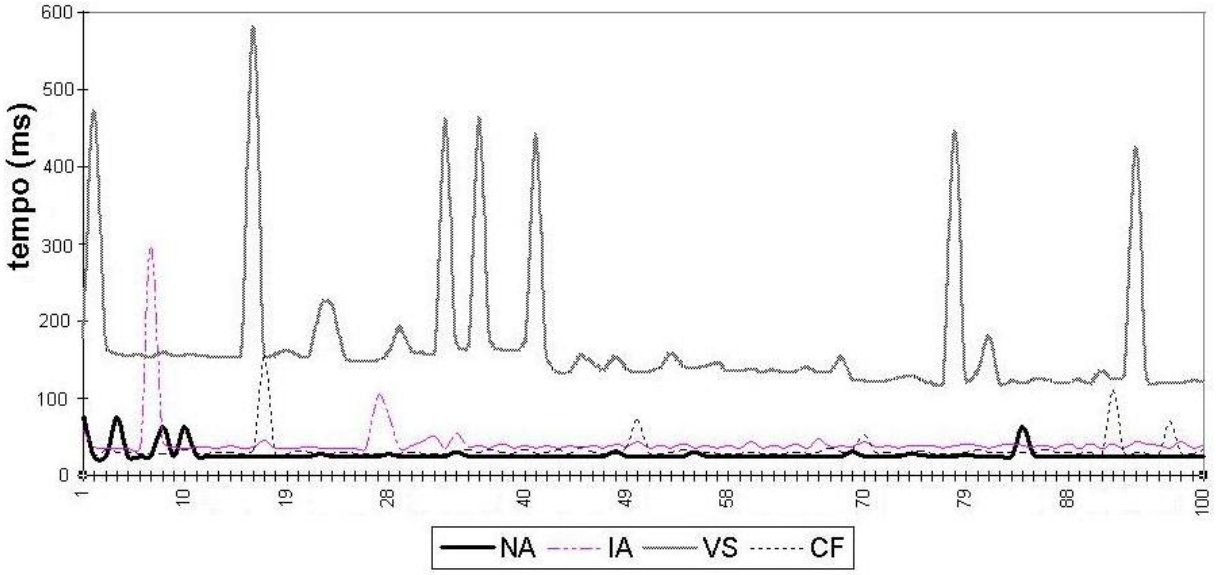


Figura 38: Histórico das adaptações de uma página HTML

pontos que fornecem os seguintes tempos: *T(Origin Server)*, que mede o tempo de resposta do servidor de origem ao *Adaptation Proxy* ; *T(Analysis)*, referente ao tempo gasto com o processo de análise da política de adaptação; *T(Adaptation)*, que representa o tempo gasto pelo protocolo ICAP e pelas adaptações do conteúdo; e *T(Delivery)*, que mede o tempo que o conteúdo demora a chegar ao usuário após todas adaptações.

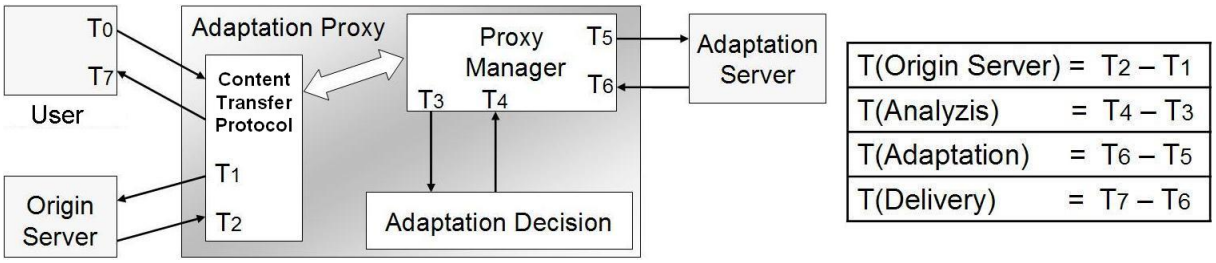


Figura 39: Tempos utilizados na avaliação da arquitetura

A partir das definições desses tempos, é possível examinar o *overhead* ocasionado pela política de adaptação e pelas adaptações de conteúdo, respectivamente, através dos tempos *T(Analysis)* e *T(Adaptation)*. Para realizar essa avaliação, foram executadas 1000 requisições do endereço Web *www.folha.com.br*, adaptando-se somente sua página HTML sem as imagens. Nessa avaliação foram executados cinco testes diferentes: sem adaptação (NA); utilizando os três *Adaptation Servers* disponíveis (CF, VS e IA); e combinando-se esses três servidores (CF+VS+IA). A Figura 40 apresenta a média dos tempos *T(Origin Server)*, *T(Analysis)*, *T(Adaptation)* e *T(Delivery)* coletados nos testes realizados.

Nesses testes pode-se verificar que o maior atraso ocorreu na espera da resposta do servidor

de origem e o tempo gasto no processo da política de adaptação foi semelhante em todas as adaptações testadas. Nos testes realizados sem a adaptação (NA), o tempo total da entrega do conteúdo teve média de 121 ms, sendo 103 ms referentes à resposta do servidor de origem, 17 ms referentes à política de adaptação e apenas 1 ms referente à entrega do conteúdo ao usuário. Dessa maneira, percebe-se que o atraso ocasionado pelo mecanismo de inferência (17 ms) é relativamente pequeno e satisfatório para a aplicação.

Utilizando os servidores VS e IA, o tempo médio gasto nas adaptações de conteúdo foi de 5,3 ms e 2,5 ms respectivamente. Logo esses servidores se mostraram eficazes, provocando um atraso pouco significativo na entrega do conteúdo.

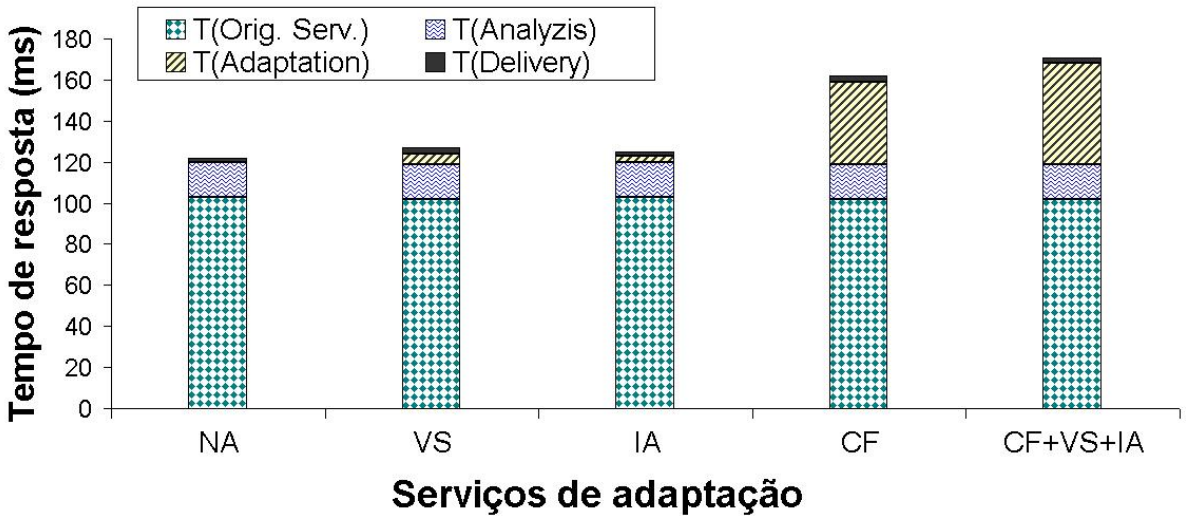


Figura 40: *Overheads* da arquitetura

Esse estudo também apresenta os testes de carga, que têm o intuito de aferir a escalabilidade da arquitetura. Essa avaliação foi realizada através da ferramenta *Webserver Stress Tool v6.0 - Enterprise Edition*. Cada teste consistiu em acessar cinco *links* pré-definidos, sendo que foi aplicada uma carga progressiva de usuários, na qual um usuário era adicionado a cada 1,5 s até o limite de 100 usuários, com cada usuário clicando num *link* a cada 5 s. A Figura 41 mostra o comportamento da arquitetura quando acessada por vários usuários, realizando-se as seguintes testes de adaptação: sem adaptação (NA), adaptação de imagem (IA), filtragem de conteúdo (CF) e escaneamento de vírus (VS).

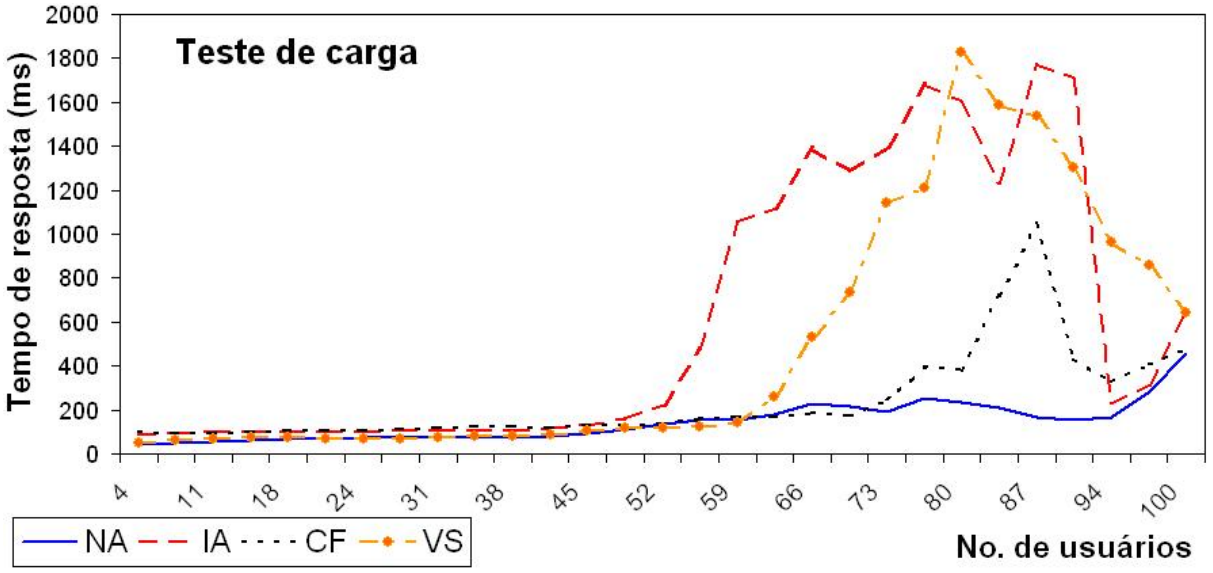


Figura 41: Testes de carga

5.2 Adaptação de Conteúdo em Ambientes de Computação Ubíqua

A necessidade de adaptar conteúdos foi principalmente impulsionada pelo surgimento dos ambientes de computação ubíqua. Esse novo cenário da computação proporcionou o acesso à Internet através de dispositivos com capacidades limitadas de hardware e software e que geralmente utilizam redes de acesso instáveis e/ou com baixa largura de banda. Partindo desse contexto, este trabalho apresenta um estudo de caso que visa demonstrar o uso de adaptação de conteúdo nos ambientes de computação ubíqua. Para simular esses ambientes utilizou-se emuladores de *palmtop* e de telefone celular.

Inicialmente testaram-se as adaptações de conteúdo realizadas pelos servidores de filtragem de conteúdo (CF) e adaptação de imagens (IA), descritos na seção anterior. Para simular o usuário dessa aplicação foi utilizado o *Wapaka*, um *micro-browser* escrito em Java que emula a navegação através de *palmtops* [5].

Na adaptação de imagens foi acessada a página Web cujo endereço é *shweby.sourceforge.net*, sendo realizado uma remoção de imagens através do servidor IA e inserido um *link* para essa imagem, conforme mostra a Figura 42 (a). Essa adaptação foi executada com o intuito de diminuir a largura de banda utilizada da rede de acesso do usuário e melhorar a apresentação da página no dispositivo. Utilizando o mesmo emulador de *palmtop*, foi realizado uma filtragem de conteúdo, através do servidor CF, para demonstrar seu uso no bloqueio de uma página im-

[5] http://www.wapaka.com

própria, admitindo que o usuário tenha idade inferior à 18 anos. Essa adaptação é apresentada na Figura 42 (b).

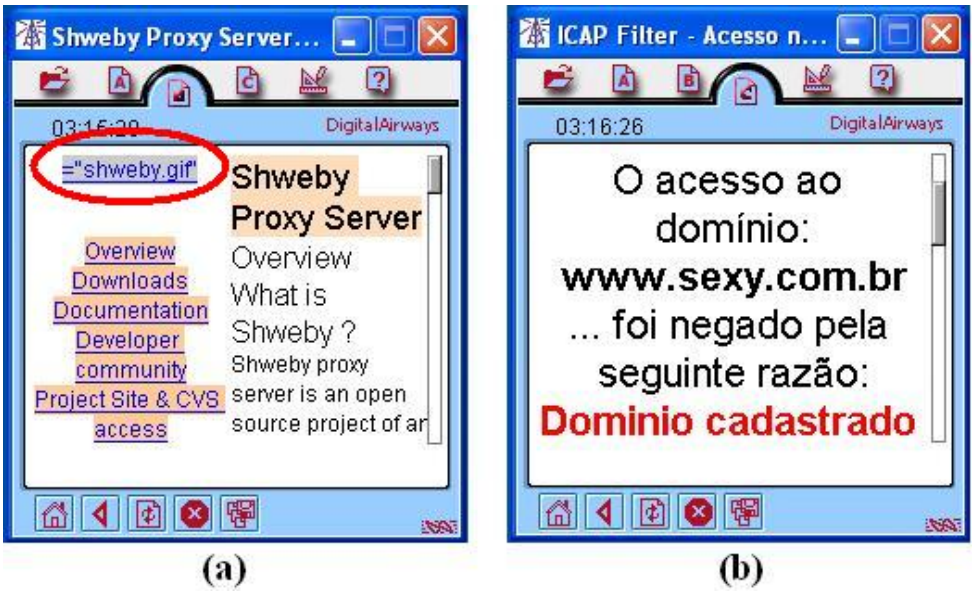


Figura 42: Adaptações de imagens e filtragem de conteúdo no *palmtop*

Outro exemplo de adaptação de conteúdo foi realizado buscando disponibilizar imagens para diferentes dispositivos. Nesse estudo a arquitetura proposta adaptou a mesma imagem, cujo formato é JPEG, viabilizando a sua entrega para *palmtop* e telefone celular, os quais foram simulados através de emuladores. O resultado desses serviços são expostos na Figura 43.

No primeiro caso a adaptação somente reduziu as dimensões da imagem, oferecendo ao *palmtop* uma melhor visualização desta e reduzindo a largura de banda utilizada na rede de acesso. No segundo caso simulou-se o uso de um telefone celular que obtinha acesso à Internet através do protocolo WAP. Dessa forma esse dispositivo somente suporta imagens no formato WBMP. Porém a arquitetura tornou possível a visualização dessa imagem no telefone celular, transformando o formato da imagem original para WBMP.

5.3 Modelo de Análise de Desempenho

Com a inserção dos serviços de valor agregado na entrega de conteúdos aos usuários, o desempenho desses serviços torna-se um ponto crítico na implementação da arquitetura proposta. Assim a avaliação de desempenho da mesma torna-se uma questão importante a ser considerada, pois a execução desses serviços tende a elevar o tempo de entrega do conteúdo. Como a arquitetura visa dentre outros propósitos a satisfação do usuário, a adaptação não deve comprometer a entrega do conteúdo as usuário.

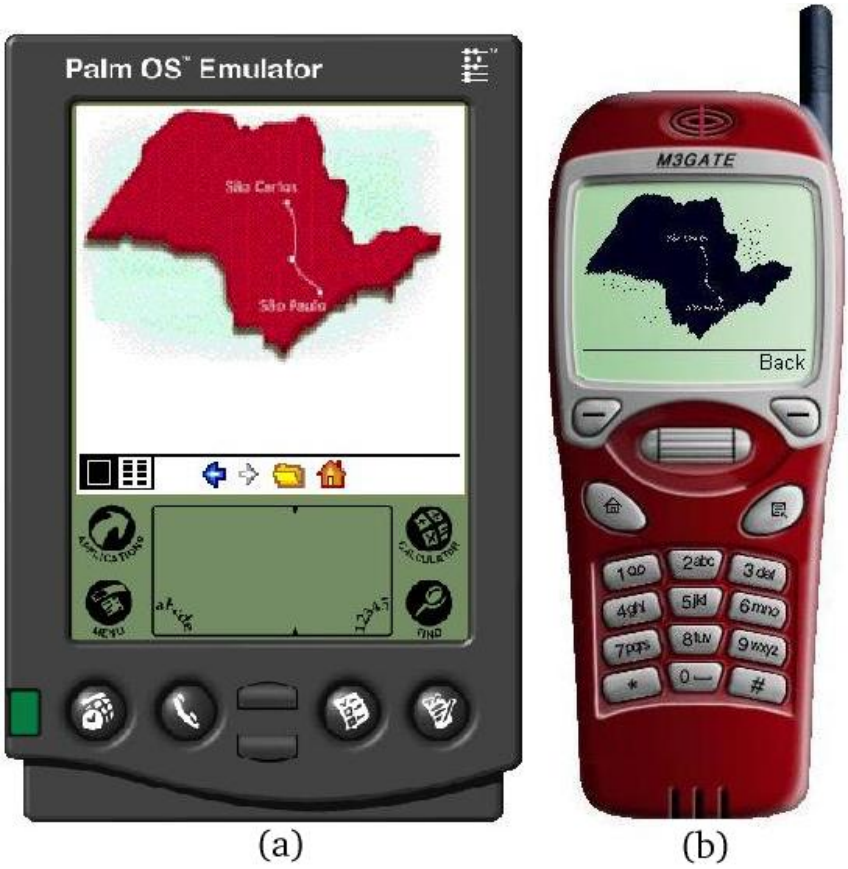


Figura 43: Adaptações de imagens para *palmtop* e celular

Em [38] foi desenvolvido um modelo de análise de desempenho para adaptação de conteúdo. Nesse modelo, ilustrado na Figura 44, *LB* é a largura de banda (*bandwidth*), *TTR* é o tempo de transmissão da rede (*roundtrip time latency*), *C* é o tamanho do conteúdo em *bytes* e *A* é o atraso provocado pela adaptação de conteúdo. Os termos *O* e *M* se referem ao conteúdo *original* e *modificado* respectivamente.

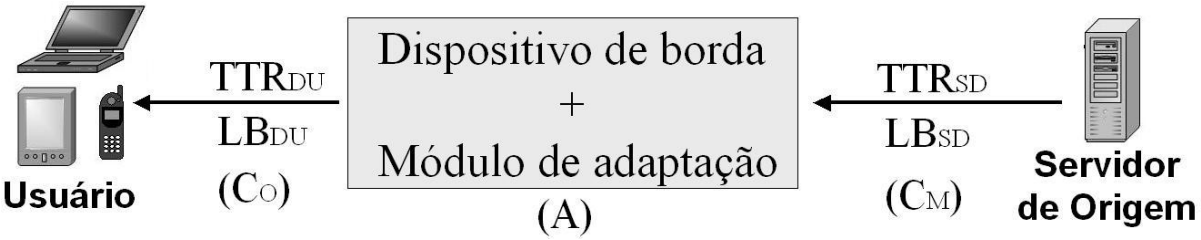


Figura 44: Modelo de análise de desempenho da arquitetura

O tempo de latência fim a fim (*L*), ou seja tempo de entrega do conteúdo, consiste em cinco variações e cada uma destas pode representar sua latência através de uma expressão matemática, a qual utiliza os parâmetros definidos na Figura 44. Considerando que nos dois primeiros casos o conteúdo não é adaptado e nos três últimos uma adaptação é realizada, têm-se as seguintes

alternativas dos tempos de latência:

1. No primeiro caso o conteúdo original não está armazenado no *cache*, então o tempo de transmissão entre o dispositivo de borda e o usuário (TTR_{DU}) e tempo de transmissão entre o servidor de origem e o dispositivo de borda (TTR_{SD}) serão somados ao tempo total. Como a adaptação de conteúdo não é realizada, o tamanho do conteúdo não sofre alteração e o tempo de latência fim a fim é representado por:

   $L_0 \ = \ 2TTR_{SD} \ + \ 2TTR_{DU} \ + \ \frac{C_O}{min(LB_{SD} + LB_{DU})}$

2. Nesse caso o conteúdo está armazenado no *cache*, assim o tempo gasto na transmissão entre o servidor de origem e o dispositivo de borda (TTR_{SD}) e largura de banda dessa comunicação (LB_{SD}) serão desconsiderados. Como não ocorre adaptação, o tamanho do conteúdo não é alterado e o tempo de latência fim a fim é representado por:

   $L_1 \ = \ 2TTR_{DU} \ + \ \frac{C_O}{LB_{DU}}$

3. Nessa alternativa o conteúdo original sofre adaptação de conteúdo, assim seu tamanho será modificado e o tempo de atraso da adaptação (*A*) será adicionado à equação. Considerando que nesse caso o conteúdo original não esteja armazenado no *cache*, o tempo de transmissão TTR_{SD} também será adicionado ao tempo de latência fim a fim, que é representado por:

   $L_2 \ = \ 2TTR_{SD} \ + \ 2TTR_{DU} \ + \ A \ + \ \frac{C_O}{LB_{SD}} \ + \ \frac{C_M}{LB_{DU}}$

4. Nesse caso também ocorre adaptação de conteúdo e, admitindo que o conteúdo original esteja armazenado no *cache*, então o tempo de transmissão TTR_{SD} e largura de banda LB_{SD} serão desconsiderados no cálculo do tempo de latência fim a fim, que segue a seguinte expressão:

   $L_3 \ = \ 2TTR_{DU} \ + \ A \ + \ \frac{C_M}{LB_{DU}}$

5. No último caso ocorre a adaptação e tanto o conteúdo original quanto o modificado estão armazenados no *cache*. Assim o tempo de atraso de adaptação não será adicionado no tempo de latência fim a fim, nem o tempo de transmissão TTR_{SD} e a largura de banda LB_{SD}. Logo a expressão do tempo de latência fim a fim desse caso é definida por:

   $L_4 \ = \ 2TTR_{DU} \ + \frac{C_M}{LB_{DU}}$

Através das expressões definidas acima e considerando que a busca do conteúdo em *cache* reduz o atraso da entrega, pode-se concluir que $L_0 > L_1$ e $L_2 > L_3 > L_4$. Admitindo um fluxo normal de requisição e resposta de conteúdo (isto é, sem a busca do conteúdo em *cache*), para

que a adaptação de conteúdo reduza o tempo de latência então L_2 deve ser menor que L_0. Nesse caso o atraso provocado pela adaptação de conteúdo deverá obedecer à expressão:

(I) $A < \frac{C_O - C_M}{LB_{DU}} - \frac{C_O}{LB_{SD}}$

A probabilidade dessa expressão ser verdadeira é grande quando se tem uma banda larga entre o servidor de origem e o dispositivo de borda (LB_{SD}) e uma banda estreita entre o usuário e o dispositivo de borda (LB_{DU}), caso típico em ambientes de computação ubíqua, no qual essa arquitetura pode ser muito bem aproveitada.

Partindo desse modelo de análise de desempenho, este trabalho mostra um estudo realizado com adaptações de vídeo e, através da expressão (I), demonstra na prática a viabilidade de realizar essas adaptações nos ambientes de computação ubíqua.

Nesse estudo a adaptação foi realizada sobre um arquivo de vídeo cujo tamanho original é 3,31 MB e após a execução da adaptação o conteúdo modificado obteve o tamanho de 0,48 MB. Essa adaptação foi realizada 200 vezes e a média do tempo gasto nessa adaptação foi de 4,1 segundos. Admitindo que esse vídeo seja acessado por um telefone celular, esse estudo considera quatro redes de acesso diferentes, pelas quais o usuário se conecta ao dispositivo de borda, descritas na Tabela 9. Essa tabela apresenta a largura de banda prevista na especificação dessas redes.

Tabela 9: Largura de banda das redes de acesso analisadas

Rede de Acesso	Largura de Banda
CDMA (Code-Division Multiple Access)	14,4 Kbps
CDMA 2000	144 Kbps
GPRS (General Packet Radio Service)	171,2 Kbps
EDGE (Enhanced Data Rates for Global Evolution)	473,6 Kbps

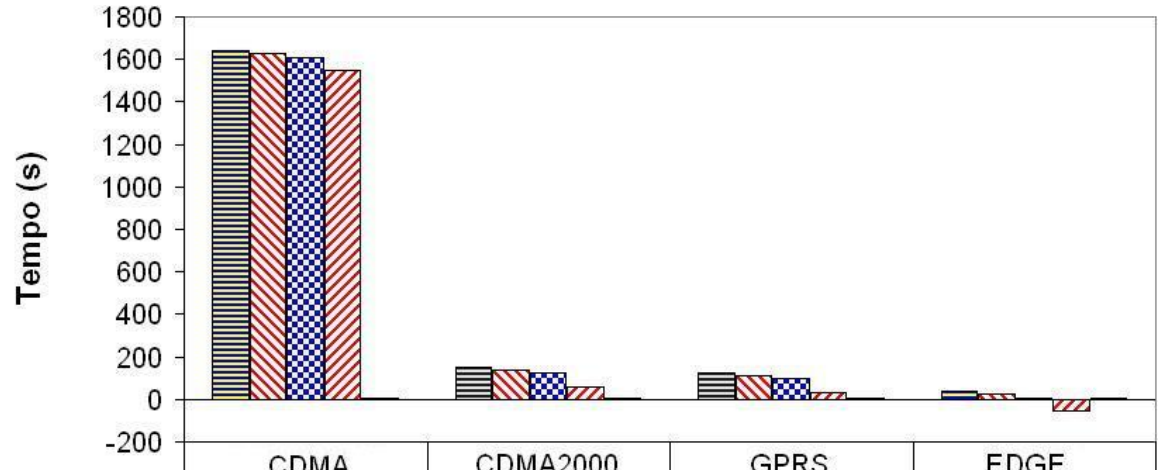
A partir da expressão (I), pode-se encontrar qual o tempo máximo que a adaptação deve gastar para que esta não atrase a entrega do conteúdo. Para definir o "atraso aceitável" desse estudo, considera-se que $C_O = 3,31MB$ e $C_M = 0,48MB$. A largura de banda entre o dispositivo de borda e o usuário (LB_{DU}) recebe os valores definidos na tabela 9. Visto que a largura de banda entre o servidor de origem e o dispositivo de borda (LB_{SD}) pode variar, nesse estudo foram definidos quatro servidores (S1, S2, S3 e S4) que possuem largura de banda de 2 Mbps, 1 Mbps, 640 Kbps e 256 Kbps respectivamente.

Definidos esses valores, a Figura 45 apresenta a comparação entre o "atraso aceitável" para a adaptação do vídeo de cada servidor (S1, S2, S3 e S4) utilizando cada rede de acesso do usuário (CDMA, CDMA2000, GPRS e EDGE). Nessa figura pode-se notar que a adaptação é bastante proveitosa quando o usuário utiliza a rede de acesso CDMA, neste caso o atraso real

equivale cerca de 0,25% do atraso aceitável. A adaptação também é bastante viável quando se acessa o servidor de origem S1, o qual possui largura de banda de 2MB.

Essa figura também mostra um atraso aceitável de valor negativo quando o usuário acessa o servidor S4, com largura de banda de 256 Kpbs, através da rede de acesso EDGE, com largura de banda de 473.6 Kbps. Isso ocorre porque a largura de banda entre o dispositivo de borda e o servidor de origem é menor que a largura de banda entre o usuário e o dispositivo de borda, logo nesse caso a adaptação é inviável.

Entretanto o tempo gasto pela adaptação foi menor que o atraso aceitável na maioria dos cenários avaliados, obtendo assim um ganho de desempenho e demonstrando a viabilidade do uso da arquitetura nos ambientes de computação ubíqua.



	CDMA	CDMA2000	GPRS	EDGE
S1	1639	151,9	125,6	36,9
S2	1625,8	138,7	112,4	23,7
S3	1609,8	122,7	96,4	7,7
S4	1546,1	59	32,7	-56
ATRASO REAL	4,1	4,1	4,1	4,1

Redes de Acesso

Figura 45: Comparação entre os atrasos aceitáveis e o atraso real

6 Conclusões

A evolução da Internet e os avanços de suas tecnologias de acesso ocorridos em razão do surgimento da computação ubíqua, gerou a necessidade de adaptar os conteúdos dessa rede. A diversidade e heterogeneidade encontrada atualmente na Internet, incluindo usuários, dispositivos, redes de acesso e conteúdos, formam uma quantidade enorme de possíveis combinações a cada entrega de determinado conteúdo. Logo se torna uma tarefa complexa ao servidor de origem criar e disponibilizar seu conteúdo de maneira que atenda às diversas combinações desses elementos da Internet.

Partindo desse contexto, este trabalho apresentou uma arquitetura baseada em componentes focada na criação de diversas apresentações de um mesmo conteúdo e na realização de serviços de personalização dos conteúdos da Internet, visando satisfazer às preferências de seus usuários, atender às capacidades de seus dispositivos e otimizar o uso das redes de acesso.

Diferentemente dos trabalhos relacionados, a arquitetura apresentada neste trabalho mostra o detalhamento da política de adaptação, que é baseada nos perfis de usuário, de dispositivo e de contrato de serviços, além de considerar as informações sobre a rede de acesso e o conteúdo. Neste trabalho os perfis são especificados utilizando o padrão CC/PP, permitindo a troca dos mesmos entre vários dispositivos de borda.

A arquitetura proposta foi modelada e desenvolvida baseando-se em componentes de software visando criar padrões que facilitem seu reuso em outras arquiteturas semelhantes. Para demonstrar a vantagem dessa escolha, este trabalho expôs uma experiência de reuso da arquitetura, onde a mesma foi instanciada para utilizar os protocolos ICAP e HTTP, além de ser inserido um mecanismo de inferência que controla a política de adaptação.

O uso do ICAP possibilita a integração da arquitetura com diversos servidores de adaptação, visto que esse protocolo é um padrão aberto e pode ser implementado por qualquer instituição ou empresa, oferecendo assim suporte à inserção de vários serviços de adaptação.

Na arquitetura foi utilizado um mecanismo de inferência, baseado na linguagem Prolog, para implementar as regras de adaptação, as quais determinam o comportamento da política

de adaptação. A utilização desse mecanismo facilita a manutenção dessas regras e oferece uma flexibilidade ao processo de decisão sobre quais adaptações devem ser realizadas, além de adicionar certa "inteligência" a esse processo.

Neste trabalho foram expostos testes da implementação da arquitetura, demonstrando o uso dos serviços de adaptação na arquitetura, assim como foi apresentado que o mecanismo de inferência processa a política de adaptação sem afetar o desempenho da entrega do conteúdo. Este trabalho também expõe um estudo que demonstra a viabilidade do uso da arquitetura em ambientes de computação ubíqua, sendo relatado que o atraso gasto na adaptação atende ao tempo aceitável estimado através de um modelo de análise de desempenho.

6.1 Trabalhos Futuros

Espera-se com o reuso da arquitetura em novas experiências, que seus componentes sejam refinados visando atender um maior número de aplicações. Acredita-se que com uma maior generalização da arquitetura seja possível definir padrões de software para o domínio de adaptação de conteúdo na Internet.

Visando o aprimoramento da arquitetura, um padrão para o gerenciamento das regras de adaptação pode ser integrado à mesma, por exemplo através da linguagem IRML, que oferece mecanismos para especificação das regras de adaptação. Outro trabalho que pode ser explorado é a implementação de um processo de coleta das informações sobre a rede de acesso a fim de colaborar na análise da política de adaptação. Além disso um estudo comparativo pode ser realizado para definir as diferenças e vantagens entre os protocolos ICAP e OCP.

Outra limitação da arquitetura que pode ser aprimorada em trabalhos futuros é a construção das interfaces para inserção dos perfis e das regras de adaptação, assim como também pode ser incorporado à arquitetura um mecanismo de autorização e autenticação de seus usuários.

Referências

1 HANSMANN, U. et al. *Pervasive Computing*. 2003. Second edition, Springer-Verlag.

2 BARBIR, A. et al. *An Architecture for Open Pluggable Edge Services*. 2002. IETF Internet Drafts. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-opes-architecture-04.txt>.

3 ELSON, J.; CERPA, A. *Internet Content Adaptation Protocol*. 2003. IETF Request for Comments 3507. http://www.ietf.org/rfc.html. Disponível em: <http://www.ietf.org/rfc.html>.

4 LEINER, B. et al. *A brief history of the internet*. 2000. Disponível em: <http://www.isoc.org/internet/history/brief.shtml>.

5 TOMMLINSON, e. a. G. *A Model for Open Pluggable Edge Services*. 2001. IETF Internet Draft, Work In Progress. http://www.ietf.org/internet-drafts/drafttomlinson -opes-model-01.txt.

6 CARPENTER, B. *Architecture Principles of the Internet*. 1996. IETF Request for Comments 1958. Disponível em: <http://www.rfc-editor.org/rfc/rfc1958.txt>.

7 NUA SCOPE COMMUNICATIONS GROUP. *How Many Online*. 2002. Disponível em: <http://www.nua.ie/surveys/how_many_online/index.html>.

8 INTERNET WORLD STATS. *Internet Usage Statistics - The Big Picture*. Disponível em: <http://www.internetworldstats.com/stats.htm>.

9 HOBBES, R. *Hobbes' Internet Timeline v8.0*. Disponível em: <http://www.zakon.org/robert/internet/timeline/#Growth>.

10 AKAMAI TECHNOLOGIES, INC. *Internet Bottlenecks the Case for Edge Delivery Services*. 2000. Disponível em: <http://www.akamai.com/en/resources/pdf/BottlenecksWhitepaper1.pdf>.

11 DAY, M. et al. *A Model for Content Internetworking (CDI)*. 2003. IETF Request for Comments 3466. Disponível em: <http://www.ietf.org/rfc/rfc3466.txt>.

12 NURMELA, T. *Analysis of Open Pluggable Edge Services*. 2004. Disponível em: <www.cs.helsinki.fi/u/kraatika/Courses/IPsem04s/OPES_final_nurmela.pdf>.

13 VAKALI, A.; PALLIS, G. *Content Delivery Networks: Status and Trends*. 2003. IEEE Internet Computing, Vol. 7, No. 6, pp. 68-74.

14 LAZAR, I.; TERRILL, W. *Exploring Content Delivery Networking*. 2001. IEEE IT Professional, Vol. 3, No. 4, pp. 47-49.

15 KRISHNAMURTHY, B.; WILLS, C.; ZHANG, Y. *On the Use and Performance of Content Distribution Networks*. 2001. Anais do International Internet Measurement Workshop, ACM Press, pp. 169-182. Disponível em: <citeseer.nj.nec.com/krishnamurthy01use.html>.

16 IPSOS-INSIGHT. *The Face of the Web*. 2004. Disponível em: <http://www.ipsos-insight.com/industryfocus/techandcomm/FOW.aspx>.

17 BUCHHOLZ, S.; SCHILL, A. *Adaptation-aware web caching: Caching in the future pervasive web*. 2003. Disponível em: <citeseer.ist.psu.edu/buchholz03adaptationaware.html>.

18 LEI, Z.; GEORGANAS, N. *Context Based Media Adaptation in Pervasive Computing*. 2001. Anais do IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Disponível em: <http://www.discover.uottawa.ca/ leizj/Lei-CCECE01.pdf>.

19 BRITTON, K. H. et al. *Transcoding: Extending e-business to new environments*. 2001. IBM Systems Journal, Vol. 40, No. 1, pp. 153-178. Disponível em: <http://researchweb.watson.ibm.com/journal/sj/401/britton.html>.

20 MOHAN, R.; SMITH, J.; LI, C. *Adapting Multimedia Internet Content for Universal Access*. IEEE Transactions on Multimedia, Vol. 1, No. 1, p. 104-114. 1999. Disponível em: <citeseer.nj.nec.com/mohan99adapting.html>.

21 NETWORK APPLIANCE. *Internet Content Adaptation Protocol (ICAP): Version 1.01*. 2001. Disponível em: <http://www.i-cap.org/docs/>.

22 NETWORK TECHNOLOGIES LAB. *Internet Content Adaptation Protocol (ICAP): White Paper*. India HCL Technologies Ltd. 2003. Disponível em: <http://www.hcltech.com/cdn>.

23 BECK, A.; HOFMANN, M.; CONDRY, M. *Example Services for Network Edge Proxies*. 2000. IETF Internet Draft. Disponível em: <http://standards.nortelnetworks.com/opes/non-wg-doc/draft-beck-opes-esfnep-01.txt>.

24 MA, W.; SHEN, B.; BRASSIL, J. *Content Services Networks: The Architecture and Protocol*. 2001. Anais do International Workshop on Web Caching and Content Distribution (WCW'01). Disponível em: <citeseer.ist.psu.edu/ma01content.html>.

25 BARBIR, A. *OPES Entities and End Points Communication*. 2004. IETF Request for Comments 3897. Disponível em: <http://www.ietf.org/rfc/rfc3897.txt>.

26 ROUSSKOV, A.; STECHER, M. *HTTP adaptation with OPES*. 2004. IETF Internet Draft. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-opes-http-02.txt>.

27 HOFMANN, M.; BECK, A. *IRML: A Rule Specification Language for Intermediary Services*. 2001. IETF Internet Draft. Disponível em: <http://standards.nortelnetworks.com/opes/non-wg-doc/draft-beck-opes-irml-02.txt>.

28 BECK, A.; ROUSSKOV, A. *P: Message Processing Language*. 2003. IETF Internet Draft. Disponível em: <http://www.measurement-factory.com/tmp/opes/>.

29 MOORE, K. *On the use of HTTP as a Substrate*. 2002. IETF Request for Coments 3205. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc3205.txt>.

30 BECK, A. et al. *Requirements for OPES Callout Protocols*. 2002. IETF Internet Drafts. http://www.ietf.org/internet-drafts/draft-ietf-opes-protocol-reqs-03.txt. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-opes-protocol-reqs-03.txt>.

31 ROUSSKOV, A. *OPES Callout Protocol Core*. 2005. IETF Request for Comments 4037. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc4037.txt>.

32 SMITH, J.; MOHAN, R.; LI, C. *Content-based Transcoding of Images in the Internet*. Anais da IEEE International Conference on Image Processing (ICIP), p. 7-11. 1998.

33 BHARADVAJ, H.; JOSHI, A.; AUEPHANWIRIYAKUL, S. *An Active Transcoding Proxy to Support Mobile Web Access*. Anais do IEEE Symposium on Reliable Distributed Systems, p. 118-123. 1998.

34 WORLD WIDE WEB CONSORTION. *Simple Object Access Protoco (SOAP) 1.2*. 2003. Disponível em: <http://www.w3.org/TR/SOAP/>.

35 MASTOLI, V.; DESAI, V.; SHI, W. *SEE: A Service Execution Environment for Edge Services*. 3o. IEEE Workshop on Internet Applications, p. 61-65, EUA. 2003. Disponível em: <141.217.16.163/papers/mastoli03-see.pdf>.

36 STECHER, M. *Evaluating the ICAP protocol regarding the OPES callout protocol requirements*. 2002. IETF Internet Drafts. http://www.ietf.org/internet-drafts/draft-stecher-opes-icap-eval-00.txt. Disponível em: <http://www.ietf.org/internet-drafts/draft-stecher-opes-icap-eval-00.txt>.

37 BECK, A.; HOFMANN, M. *Enabling the Internet to Deliver Content-Oriented Services*. Sixth International Workshop on Web Caching and Content Distribution, EUA. 2001. Disponível em: <hofmann.us/markus/papers/wcw-2001.pdf>.

38 TAM, W. L. et al. *ICAP Solution to Internet Content Adaptation for Pervasive Computing*. Multimedia Computing and Networking 2003, IS&T/SPIE's 15th Annual Symposium Electronic Imaging Science and Technology. 2003. Disponível em: <http://icap.ie.cuhk.edu.hk/>.

39 RAVINDRAN, G.; JASEEMUDIN, M.; RAYHAN, A. *A Management Framework for Service Personalization*. Anais do 5º IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), EUA. 2002. Disponível em: <http://www.ee.ryerson.ca/j̃aseem/>.

40 MARQUES, M.; LOUREIRO, A. *Adaptation in Mobile Computing*. 23o. Simpósio Brasileiro de Redes de Computadores (SBRC), 2004.

41 FORTE, M.; SOUZA, W. de; PRADO, A. do. *Servidor de classificação e filtragem de conteúdo com suporte ao protocolo ICAP*. 2004. Workshop Cooperação UFSCar - FSA em Ciência da Computação.

42 BABIR, A. et al. *Policy, Authorization, and Enforcement Requirements of the Open Pluggable Edge Services (OPES)*. 2004. IETF Request for Comments 3507. http://www.ietf.org/rfc.html.

43 WORLD WIDE WEB CONSORTIUM. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*. 2003. Disponível em: <http://www.w3.org/TR/2003/WD-CCPP-struct-vocab-20030325/>.

44 DIAZ, D. *GNU PROLOG - A Native Prolog Compiler with Constraint Solving over Finite Domains*. 2002. Edition 1.7, version 1.2.16. Disponível em: <http://pauillac.inria.fr/ diaz/gnu-prolog/manual/>.

APÊNDICE A – Gramática BNF do ICAP

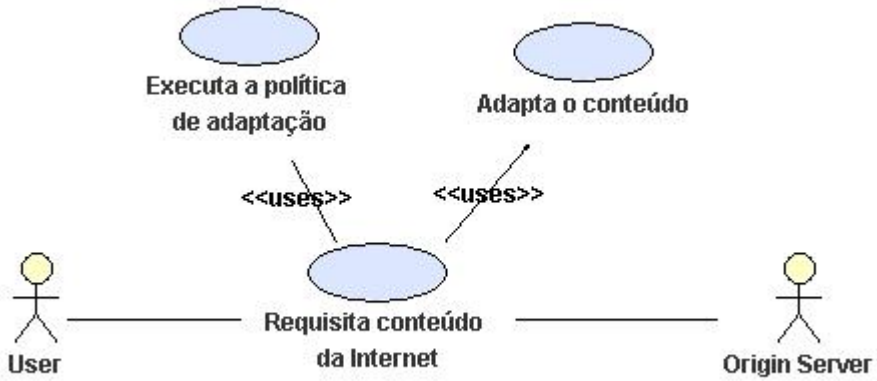
```
ICAP-Version =          "ICAP/1.0"
ICAP-Message =          Request | Response
Request =               Request-Line
                        *(Request-Header CRLF)
                        CRLF
                        [ Request-Body ]
Request-Line =          Method SP ICAP_URI SP ICAP-Version CRLF
Method =                "REQMOD" ;
                        | "RESPMOD" ;
                        | "OPTIONS" ;
                        | Extension-Method ;
Extension-Method =      token
ICAP_URI =              Scheme ":" Net_Path [ "?" Query ] ;
Scheme =                "icap"
Net_Path =              "//" Authority [ Abs_Path ]
Authority =             [ userinfo "@" ] host [ ":" port ]
Request-Header =        Request-Fields ":" [ Generic-Field-Value ]
Request-Fields =        Request-Field-Name
                        | Common-Field-Name
Request-Field-Name =    "Authorization" ;
                        | "Allow" ;
                        | "From" ;
                        | "Host" ;
                        | "Referer" ;
                        | "User-Agent" ;
                        | "Preview" ;
```

```
Common-Field-Name =     "Cache-Control" ;
                        | "Connection" ;
                        | "Date" ;
                        | "Expires" ;
                        | "Pragma" ;
                        | "Trailer" ;
                        | "Upgrade" ;
                        | "Encapsulated" ;
                        | Extension-Field-Name ;
Extension-Field-Name =  "X-" token
Generic-Field-Value =   *( Generic-Field-Content | LWS )
Generic-Field-Content = <the OCTETs making up the field-value
                        and consisting of either *TEXT or
                        combinations of token, separators,
                        and quoted-string>
Request-Body =          *OCTET ;
Response =              Status-Line
                        *(Response-Header CRLF)
                        CRLF
                        [ Response-Body ]
Status-Line =           ICAP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code =           "100" ;
                        | "101" ;
                        | "200" ;
                        | "201" ;
                        | "202" ;
                        | "203" ;
                        | "204" ;
                        | "205" ;
                        | "206" ;
                        | "300" ;
                        | "301" ;
                        | "302" ;
                        | "303" ;
                        | "304" ;
                        | "305" ;
                        | "306" ;
```

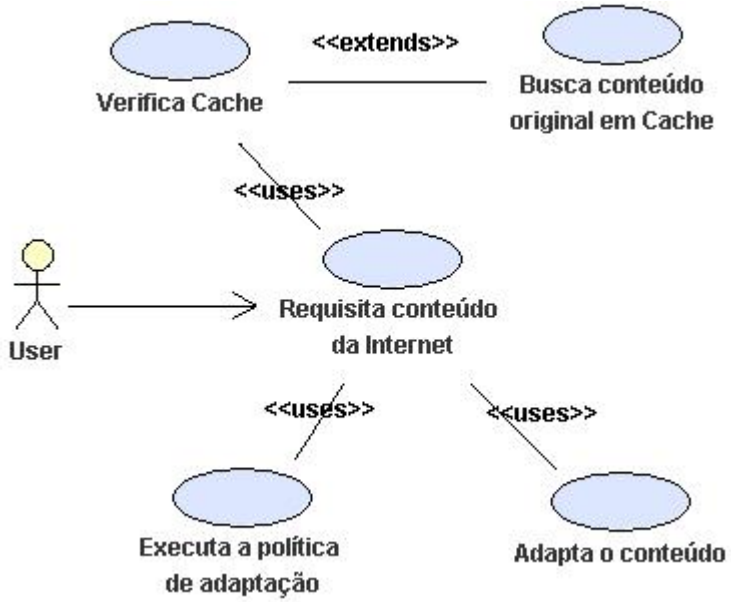
```
                        | "307" ;
                        | "400" ;
                        | "401" ;
                        | "402" ;
                        | "403" ;
                        | "404" ;
                        | "405" ;
                        | "406" ;
                        | "407" ;
                        | "408" ;
                        | "409" ;
                        | "410" ;
                        | "411" ;
                        | "412" ;
                        | "413" ;
                        | "414" ;
                        | "415" ;
                        | "416" ;
                        | "417" ;
                        | "500" ;
                        | "501" ;
                        | "502" ;
                        | "503" ;
                        | "504" ;
                        | "505" ;
                        | Extension-Code
Extension-Code =        3DIGIT
Reason-Phrase =         *<TEXT, excluding CR, LF>
Response-Header =       Response-Fields ":" [ Generic-Field-Value ]
Response-Fields =       Response-Field-Name
                        | Common-Field-Name
Response-Field-Name =   "Server" ;
                        | "ISTag" ;
Response-Body =         *OCTET ;
```

APÊNDICE B – Diagramas de casos de uso da Arquitetura

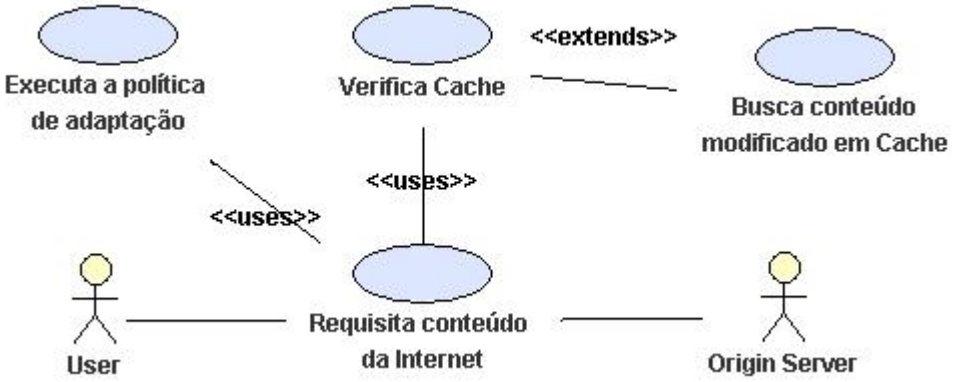
Caso de uso de um fluxo normal de requisição de conteúdo



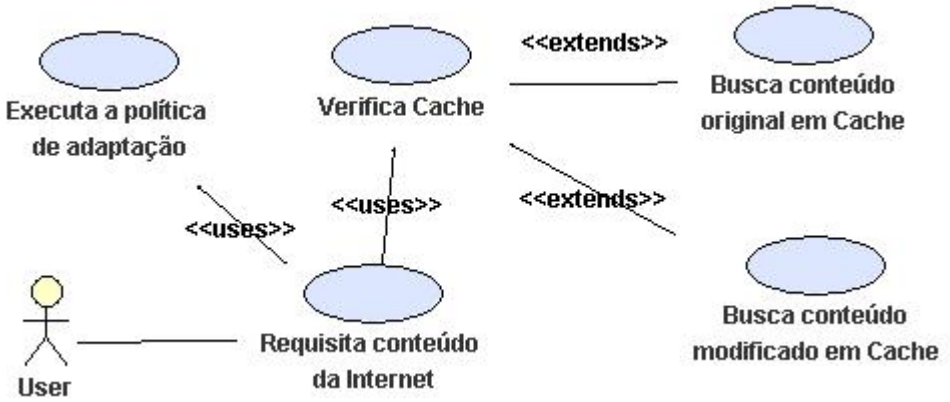
Caso de uso quando o conteúdo original se encontra em Cache



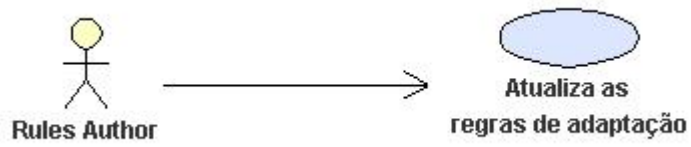
Caso de uso quando o conteúdo modificado se encontra em Cache



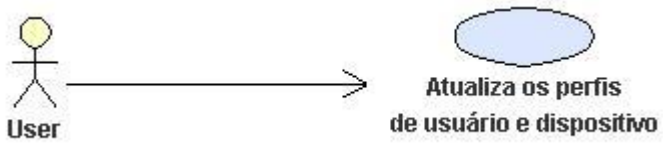
Caso de uso quando os conteúdos original e modificado se encontram em Cache



Caso e uso de remoção, inserção e modificação das regras de adaptação



Caso e uso de remoção, inserção e modificação dos perfis de usuário e de dispositivo



APÊNDICE C – Regras de Adaptação

```
%ooooooooooooooooooooooooooooooooooo%
% ponto de processamento das regras %
%ooooooooooooooooooooooooooooooooooo%

point_one(Ret,UserID,DeviceID,Contract,Content):-
        sql_connect("localhost","root","123456","profiles"),
        contentIsText(Content),
        userPayforFilter(Contract),
        userIsChild(UserID),
        =(Ret,"filter_content f-request").

point_two(Ret,UserID,DeviceID,Contract,Content):-
        sql_connect("localhost","root","123456","profiles"),
        contentIsText(Content),
        userPayforFilter(Contract),
        =(Ret,"filter_content f-request").

point_three(Ret,UserID,DeviceID,Contract,Content):-
        sql_connect("localhost","root","123456","profiles"),
        contentCanBeVirus(Content),
        userPayforScanVirus(Contract),
        userIsChild(UserID),
        =(Ret,"virus scan-and-alert").

point_three(Ret,UserID,DeviceID,Contract,Content):-
        contentIsText(Content),
        userPayforImageAdapt(Contract),
```

```
        deviceNotSupportImage(DeviceID),
        =(Ret,"image_adapter removal").

point_three(Ret,UserID,DeviceID,Contract,Content):-
        contentIsImage(Content),
        userPayforImageAdapt(Contract),
        resolutionIsHigh(Content),
        =(Ret,"image_adapter resolution").

point_three(Ret,UserID,DeviceID,Contract,Content):-
        contentIsText(Content),
        userPayforFilter(Contract),
        userIsChild(UserID),
        =(Ret,"filter_content f-response").

point_three(Ret,UserID,DeviceID,Contract,Content):-
        contentIsVideo(Content),
        userPayforVideoAdapt(Contract),
        resolutionIsHigh(Content),
        =(Ret,’video_adapter resolution’).

point_four(Ret,UserID,DeviceID,Contract,Content):-
        sql_connect("localhost","root","123456","profiles"),
        contentIsText(Content),
        userPayforFilter(Contract),
        =(Ret,"filter_content f-response").

%ooooooooooooooooooo%
% regras do usuário %
%ooooooooooooooooooo%

userIsChild(X):-
        sql_query(X,List),
        search(5,List,Res),
        string_to_int(Age,Res),
```

```
        <(Age,18).

%ooooooooooooooooooooooooooooooo%
% regras do contrato de serviços %
%ooooooooooooooooooooooooooooooo%

userPayforFilter(X):-
       sql_query(X,List),
       search(14,List,Res),
       ==(Res,"yes").

userPayforImageAdapt(X):-
       sql_query(X,List),
       search(10,List,Res),
       ==(Res,"yes").

userPayforVideoAdapt(X):-
       sql_query(X,List),
       search(7,List,Res),
       ==(Res,"yes").

userPayforScanVirus(X):-
       sql_query(X,List),
       search(13,List,Res),
       ==(Res,"yes").

%ooooooooooooooooooooo%
% regras do dispositivo %
%ooooooooooooooooooooo%

deviceNotSupportImage(X):-
       sql_query(X,List),
       search(5,List,Res),
       ==(Res,"no").
```

```
deviceNotSupportColor(X):-
       sql_query(X,List),
       search(4,List,Res),
       ==(Res,"no").

deviceNotSupportAudio(X):-
       sql_query(X,List),
       search(13,List,Res),
       ==(Res,"no").

%ooooooooooooooooooooo%
% regras do conteúdo  %
%ooooooooooooooooooooo%

contentIsVideo(video).
contentIsImage(image).
contentIsText(text).
contentCanBeVirus(text).
contentCanBeVirus(image).
contentCanBeVirus(application).

%oooooooooooooooooooooooo%
% Busca elemento em lista %
%oooooooooooooooooooooooo%
search(1,[X|_],X):- !.
search(N,[X|Y],K):-
        Z is N - 1,
        search(Z,Y,K).

%ooooooooooooooooooo%
% Chamadas externas %
%ooooooooooooooooooo%
:- foreign(sql_connect(+codes, +codes, +codes, +codes)).
:- foreign(sql_query(+codes, term), [choice_size(1)]).
:- foreign(string_to_int(-integer,+codes)).
```