Abaré: Um Framework para Implantação, Monitoramento e Gerenciamento Coordenado e Autônomo para Redes em Malha sem Fio

Billy Anderson Pinheiro[1-3], Vagner de Brito Nascimento[1-3], Eduardo Cerqueira[1-4], Antônio Jorge Gomes Abelém[1-3], Augusto Neto[5]

Grupo de Estudos em Redes de Computadores e Comunicação Multimídia (GERCOM)[1]
Programa de Pós-Graduação em Ciências da Computação (PPGCC)[2]
Universidade Federal do Pará (UFPA)[3]
Faculdade de Engenharia da Computação – UFPA[4]
Universidade Federal de Goiás (UFG)[5]

{billy,vagner,cerqueira,abelem}@ufpa.br, augusto@inf.ufg.br

Abstract. *The Wireless Mesh Networks (WMNs)have been gaining ground as a solution to provide last mile indoors and outdoors Internet access, because of their technical and economic feasibility. However, the existence of open source and proprietary approaches that are not interoperable and the delay in the standardization process make deployment of a large-scale WMN time-consuming and complex. This paper presents an extension of the framework Abaré with autonomic capability and performance evaluation results regarding load balance issues. Abaré defines a set of components and practices in order to optimize the implementation and management of WMN systems, as well as to provide autonomic features in routers to decrease and facilitate the manager workload.*

Resumo. *As redes em malha sem fio vêm se consagrando como solução para o acesso de última milha em ambientes internos e externos, devido sua viabilidade técnica e econômica. No entanto, a existência de soluções de código aberto e proprietárias que não são interoperáveis e a demora no processo de padronização torna a implantação de uma rede em malha sem fio de larga escala uma tarefa demorada e complexa. Este artigo apresenta uma extensão do framework Abaré com capacidade autonômica e resultados de desempenho no que diz respeito ao balanceamento de carga. Abaré define um conjunto de componentes e práticas para otimizar a implantação e gerenciamento de redes em malhas sem fio, bem como prover características autonômicas nos roteadores com o objetivo de reduzir e facilitar o trabalho do administrador .*

1. Introdução

As Redes em Malha Sem Fio (*Wireless Mesh Networks* - WMNs) surgem como uma solução atraente para prover ubiquidade e conectividade à última milha. A cooperação entre os nós permite o uso eficiente da largura de banda e a redução de custos operacionais [Campista *et al* 2008]. No entanto, este tipo da rede ainda sofre com a falta de padronização, o que acarreta em desperdício de recursos e tempo [IEEE draft p802.11s d4.0 2009].

Como forma de aproveitar o crescente mercado das WMNs várias empresas, como a Motorola e Cisco já desenvolveram soluções denominadas *pré-mesh* para facilitar a comunicação nas redes, mas o custo elevado e a falta de interoperabilidade destes equipamentos impedem sua utilização em ambientes de grande escala [Motorola 2009] [Cisco 2009].

Como alternativa às soluções proprietárias, equipamentos com suporte a IEEE 802.11 podem ser usados com o *firmware* modificado, usando uma distribuição Linux [OpenWRT 2009] [DD-WRT 2009]. Este tipo de solução permite a fácil criação de ambientes digitais, possibilitando a distribuição e gerenciamento de novos serviços para equipamentos sem fio e a captação de novos clientes. Vários projetos mostram que o uso desta solução é viável, como o *Vmesh* na Grécia [Vmesh 2009] e o *Remesh* no Brasil [Remesh 2009].

Outra característica esperada pelas WMNs e que será fundamental para a expansão das mesmas é a autonomia. Segundo Khalid [Khalid *et al* 2009] a computação autonômica é um conceito inspirado nos sistemas biológicos, que pretende diminuir a complexidade de administrar grandes sistemas heterogêneos através da capacidade de auto-gerenciamento, minimizando a intervenção humana. Dentre as propriedades que habilitam as capacidades autonômicas de um sistema, podemos destacar a auto-configuração, auto-otimização, auto-recuperação e auto-proteção [Kephart e Chess 2003].

A fim de suprir os requisitos esperados pelas WMNs no que diz respeito à gerência eficiente, auto-organização e interoperabilidade, este artigo estende o *framework* Abaré[1]. Esta solução descreve métodos para auxiliar a implantação e o gerenciamento de WMNs com base em informações coletadas dos roteadores e das condições da rede, provendo facilidades para a gestão por parte do administrador e possibilitando a existência de nós autônomos que possamos tomar decisões sem a necessidade da interferência humana. Avaliação de um protótipo do Abaré foi realizada em ambiente real, sendo executados testes de carga com o intuito de comprovar a viabilidade e comportamento.

Este trabalho está estruturado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados a gerenciamento e WMN. Na seção 3 é apresentado o *framework* Abaré. A Seção 4 apresenta a aplicação do *framework*. Na Seção 5 são apresentados os testes realizados e os resultados obtidos. Finalmente, a Seção 6 apresenta as conclusões gerais e sugere possíveis trabalhos futuros.

2. Trabalhos Relacionados

O *Distributed Architecture for Monitoring Multi-hop Mobile NetworksDistributed Ad-hoc Monitoring Network* (Damon) é um sistema para monitoramento distribuído de redes de sensores *ad-hoc* que foi proposto por [Ramachandran *et al* 2004]. Nele existem agentes usados para coletar informações da rede e enviar os dados para os repositórios. Vale ressaltar que esse algoritmo é dependente do protocolo de roteamento *Ad-hoc On-Demand Distance Vector* (AODV) para sua operação, ou seja, é impossível o uso deste *framework* em uma rede que use OLSR [Aguiar et al. 2007] ou outro protocolo de roteamento, reduzindo desta forma a

[1] Em tupi-guarani: Amigo do Homem.

flexibilidade do sistema.

Jardosh [Jardosh *et al* 2008] projetou o SCUBA, um *framework* para visualização interativa de problemas em WMNs de grande escala. Neste *framework*, várias métricas são reunidas em um banco de dados através de um nó *gateway*. Esta informação é usada para gerar uma visão interativa. Uma implementação inicial do *framework* foi testado em uma rede com 15 nós que mostrou a viabilidade do *framework* para fornecer o serviço de visualização. É importante ressaltar que apenas a visualização é fornecida por este *framework,* não fornecendo nenhum mecanismo de gerência.

Riggio [Riggio *et al* 2007] propôs um *framework* distribuído para WMNs chamado JANUS. Os testes realizados foram em uma WMN do tipo cliente [Aggelou *et al* 2009] usando microcomputadores com MCL (*Mesh Connectivity Layer*) instalada[Microsoft 2009]. Apesar dos testes positivos, a proposta atual é restrita à tarefa de monitorar a rede, sendo necessário o uso de outra ferramenta de gerenciamento para configura a rede.

Mesh-Mon é um *framework* proposto e implementado por Nanda e Kotz [Nanda e Kotz 2008] que realiza o monitoramento da rede para auxiliar o administrador com suas tarefas. Este sistema de gerenciamento é definido como sendo escalável e distribuído, capaz de detectar automaticamente e recuperar falhas na rede. No entanto, o *framework* é reduzido às tarefas de monitoramento da rede e executa algumas ações automaticamente se o comportamento da rede é diferente de um padrão que foi definido estaticamente pelo autor.

MobiMESH é uma implementação para WMNs que fornece um abrangente *framework* de análise do comportamento em tempo real , incluindo suporte avançado de roteamento considerando múltiplos rádios, alocação de canais, bem como de gerenciamento [Capone *et al* 2007]. No entanto, este *framework*, como acontece na maioria das outras propostas, não oferece suporte a módulos adicionais que poderiam torná-los adaptáveis à realidade da rede ao longo do tempo.

Após a análise dos trabalhos relacionados, podemos observar que a implantação, monitoramento e gerenciamento de WMNs, tanto de forma autônoma como assistida, são tarefas importantes para o sucesso das mesmas e para atender a esses requisitos, o *framework* Abaré foi proposto e é apresentado na próxima seção.

3. Framework Abaré

O *framework* Abaré [Pinheiro *et al* 2009] tem como objetivo desenvolver um sistema de especificação e padronização para a gerência autonômica de WMNs. Desta forma, o Abaré visa facilitar os processos de implantação e manutenção de WMNs em grande escala. Ele foi concebido utilizando o conceito de OpenMesh, que são WMNs criadas com a utilização de equipamentos IEEE 802.11 convencionais alterando seu *firmware* para uma distribuição Linux embarcada. Além disto, é utilizado um algoritmo de roteamento dinâmico [Moreira *et al* 2007] com o esquema de endereçamento proposto por Tsarmpopoulos [Tsarmpopoulos *et al* 2005] em uma rede em malha seguindo a arquitetura infra-estruturada. Os elementos que compõem esta solução são descritos de maneira detalhada, bem como os passos necessários para a implantação em [Pinheiro et al. 2009].

A primeira versão deste *framework* veio para suprir às necessidades básicas para a implantação e gerência da rede. Porém, foi observado a necessidade de uma maior autonomia por parte do Agente Roteador, que na primeira versão deste framework estava localizado dentro dos roteadores, de maneira a permitir que os roteadores possam tomar decisões sem a necessidade de contato com o Abaré Core.

Para viabilizar este comportamento autônomo, o Agente Roteador foi substituído por um *middleware* chamado Middrouter que irá desempenhar as funções antes oferecidas pelo Agente Roteador, bem como viabilizar a autonomia dos roteadores na tomada de decisão. Para possibilitar a inserção do Middrouter foi necessário estender a arquitetura do Abaré como é apresentado na Figura 1.

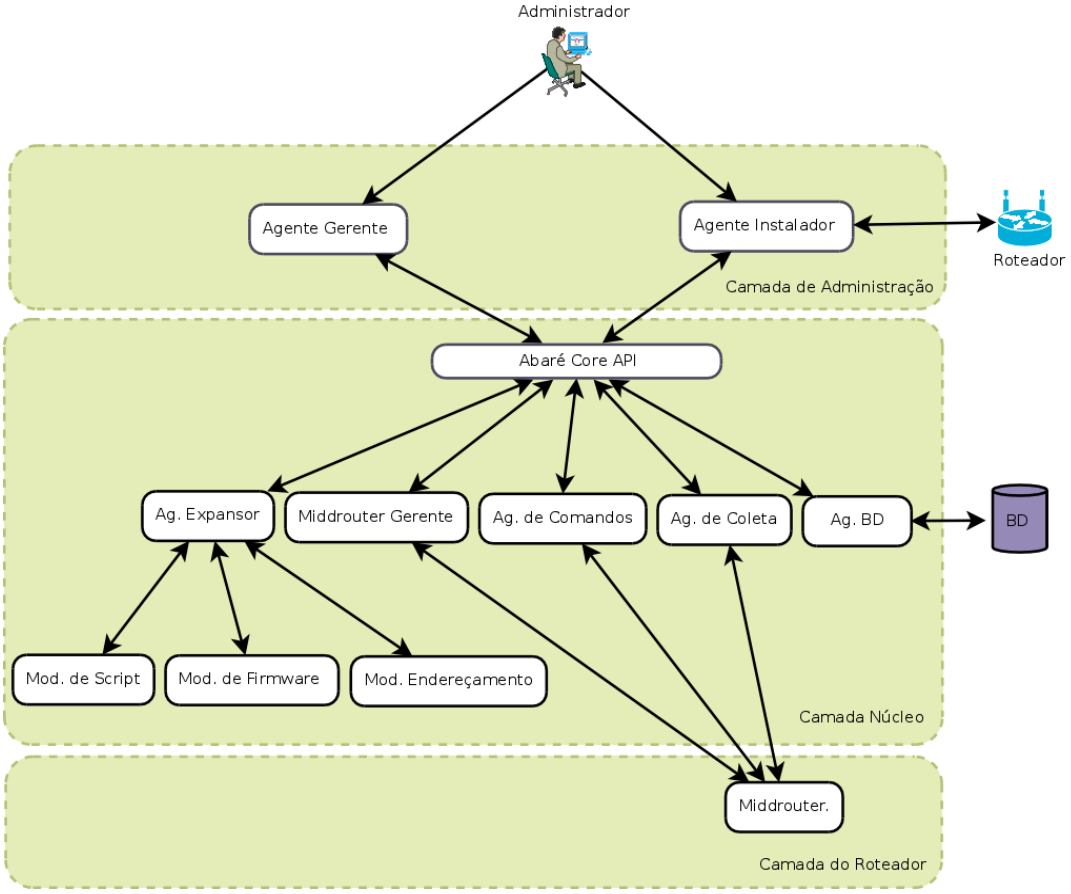

Figura 1: Abaré Framework

O *framework* foi desenvolvido de forma modular e possui três (3) camadas: administração, que é responsável pela interação com o administrador; Núcleo, que representa o centro do sistema onde a parte lógica e o armazenamento das informações estão localizados e a Camada do Roteador, que permite o acesso ao roteador para comunicação direta com o sistema operacional de cada nó da rede e adiciona a característica autonômica ao *framework*. A descrição de cada componente é apresentada a seguir:

- Agente Instalador - É o responsável por fazer as mudanças de *firmware* nos roteadores, executar a configuração inicial destes e armazenar os dados sobre cada roteador no Abaré Core API.

- Agente Gerente - Responsável por fornecer uma interface onde, após a devida autenticação, o administrador pode interagir com o sistema e usar os recursos oferecidos pelo Abaré Core. Em outras palavras, esta é a interface para gestão do sistema e sua implementação deve ser independente do sistema operacional e linguagem de programação.
- Abaré Core API - É o núcleo do sistema, responsável pela obtenção e gestão das informações de todos os componentes do *framework*. Ele deve fornecer funcionalidades, geralmente em formato de *Webservice*, que poderão ser utilizadas pelos Agentes Instalador e Gerente.
- Agente BD - Responsável pela leitura e escrita das informações no banco de dados.
- Agente de Coleta - Requisita informação sobre tráfego, hardware e tabelas de roteamento ao Middrouter e as envia para o Abaré Core API.
- Agente de Comandos - Responsável pelo envio de comandos para o Middrouter. Normalmente, ele é utilizado para tarefas administrativas que precisem de intervenção humana;
- Middrouter Gerente - Este é o responsável por controlar o Middrouter e modificar seus parâmetros. Através dele é possível inserir novos coletores e agentes de decisão, bem como agendar ações a serem tomadas pelos roteadores de forma autônoma.
- Agente Expansor - Permite a extensão do *framework* através da adição de novos módulos, fornecendo abstração suficiente para permitir o desenvolvimento rápido de novos recursos. Estes são módulos que já são definidos por padrão no *framework*:
    - Módulo de Endereçamento: Coordena os IDs dos roteadores e realiza a separação das redes e IPs utilizados;
    - Módulo de Scripts: Gera os scripts com os comandos que são repassados para o Agente de Comandos, que por sua vez, envia os comandos para serem executados nos roteadores.
    - Módulo de Firmware: Obtém os *firmwares* inseridos pelo administrador e fornece-os de acordo com as necessidades do Agente de Instalação.
- Middrouter - É responsável por responder às solicitações do Agente de Coleta, fornecendo as informações solicitadas no formato XML (*eXtensible Markup Language*). É preciso também aceitar os comandos enviados pelo Agente de Comando e executá-los nos roteadores, além de prover a parte autônoma do sistema. Este agente é dividido em seis (6) camadas, como na Figura 2, e suas funcionalidades são descritas a seguir:
    - Entrada e saída: Responsável pela recepção dos pedidos e por encaminhá-los para a camada correta dependendo do tipo de dados recebidos. É também responsável por enviar os resultados dos comandos executados e das métricas coletadas;
    - Parse XML: Recebe informações em XML da camada superior e identifica

qual o tipo da requisição, caso seja de coleta esta é encaminhada para o coletor selecionado, caso seja um comando ele é enviado ao Comandos. Na direção oposta, ele recebe as informações coletadas e as converte em XML para que a camada superior possa enviá-las;

- Comandos: Recebe e executa os comandos enviando uma resposta positiva ou negativa relativa à sua execução;
- Coletores: É um conjunto de pequenos módulos responsáveis pela coleta de informações e o envio para a camada superior. Quando um pedido chega ao Parse XML, apenas o nome do coletor é fornecido para que este seja acionado. Uma vez que o módulo coletor é acionado, ele deve executar esse pedido e responder com a informação solicitada. Assim, é fácil a integração dos novos módulos de coleta, sendo necessário apenas inserir o novo módulo no roteador e que a entidade que precisa usar a métrica fornecida pelo módulo saiba seu nome;

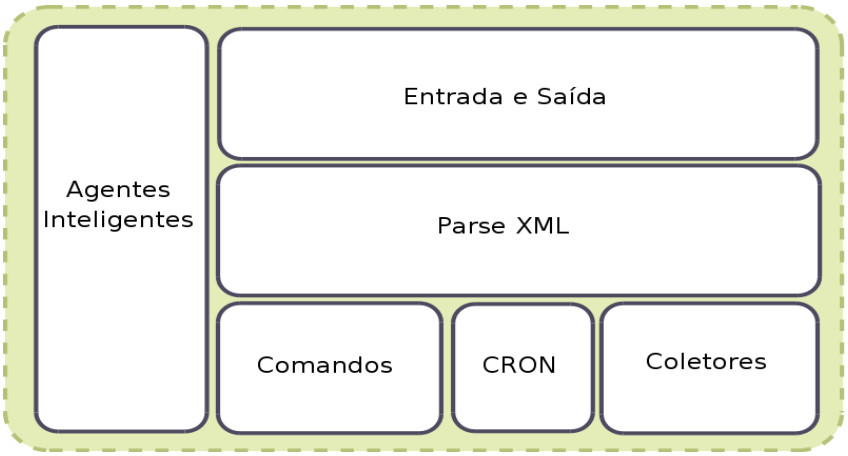

Figura 2: Middrouter

- CRON: Este é o elemento responsável por agendar ações no Middrouter, chamando os elementos de tomada de decisão de acordo com regras temporais estabelecidas.
- Agentes Inteligentes: Estes são os responsáveis por prover a autonomia dos roteadores, permitindo que estes possam tomar decisões com base em informações coletadas e executando comandos de acordo com a análise dos dados recebidos.

4. Aplicação do Framework

O uso do *framework* Abaré visa tornar os passos de implantação e manutenção de uma rede WMN uma tarefa sistemática e auxiliada por *software*, ou seja, criar uma modelagem que possa ser facilmente executada com a ajuda de uma aplicação que atenda aos requisitos do *framework*. A aplicação do Abaré em um ambiente real foi realizadaseguindo os princípios OpenMesh como apresentada na Figura 3.

A Figura 3 exibe os Middrouters nos roteadores e os computadores usados para a implantação e gestão da rede. É possível identificar no *backbone* da WMN a presença

do Middrouter que são incorporados em cada roteador *mesh*. O Abaré Core está localizado fora do *backbone* e conectado a WMN através dos *gateways mesh*.

O Abaré Core está configurado no mesmo equipamento que hospeda o banco de dados e o servidor de autenticação, mas isso não é obrigatório. A única exigência é que eles devem estar na mesma sub-rede, de preferência, conectados por meio cabeado, para evitar problemas de segurança e disponibilidade. A equipe de suporte da Figura 3 representa os usuários dos Agentes Instalador e Gerente que podem ser instalados em terminais fixos ou móveis já que o Abaré Core é concebido como um *Web Service* que fornece independência de plataforma e de linguagem de programação [Park e Lee 2003].

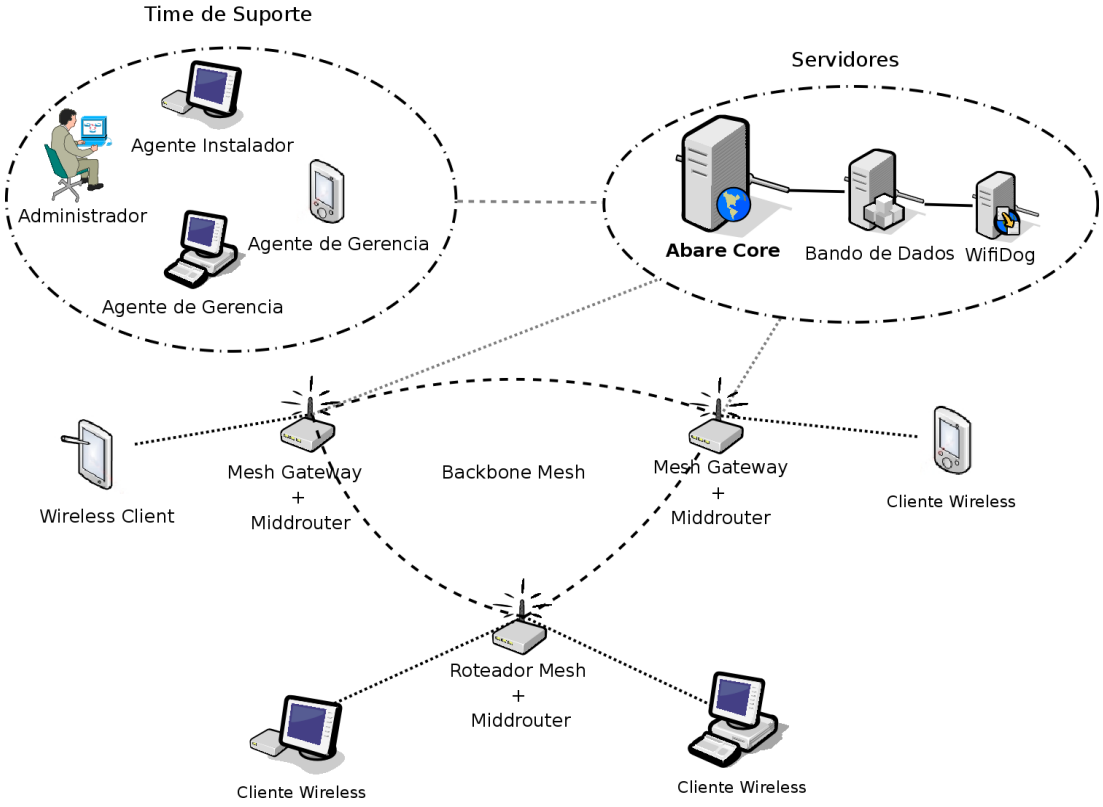

Figura 3: Aplicação do Abaré

Para avaliar o *framework*, foram desenvolvidos protótipos contendo todas as funcionalidades do Agente Instalador, os módulos de *firmware*, Controle e Scripts, um protótipo do Agente Gerente com as interfaces para acessar os módulos do core e o Middrouter. Quase todos os protótipos empregados nas avaliações foram desenvolvidos em linguagem Python[2], com a tecnologia XMLRPC[3] e OpenSSL[4] para a troca segura de mensagens, suporte para autenticação e a utilização de GTK[5] (GIMP toolkit) para a interface gráfica do usuário. A única exceção foi no Middrouter, pois as limitações de hardware impostas pelo equipamento impossibilitavam a utilização de um Web Service

[2] http://www.python.org
[3] http://www.xmlrpc.com
[4] http://www.openssl.org/
[5] http://www.gtk.org/

normal, com isso, foi necessário o desenvolvimento de um Web Service embarcado, feito sob medida para os dispositivos utilizados nos testes, sendo preciso implementar as bibliotecas necessárias para prover um Web Service.

Desta forma, o Middrouter foi desenvolvido utilizando a linguagem C. Devido à impossibilidade de utilização de bibliotecas XMLRPC ou SOAP convencionais, desenvolvemos um pequeno servidor HTTP para atender as requisições dos clientes sob o protocolo HTTP 1.0 [Berners-Lee *et al* 1996]. Além disto, foi implementada a biblioteca XML para realizar o tratamento das requisições XMLRPC. Estas implementações juntas, deram origem ao Middrouter com 32KB de tamanho, compilado para a arquitetura MIPS (*Microprocessor without Interlocked Pipeline Stages*), atendendo aos padrões de comunicação XMLRPC.

Para validar a utilização da nova arquitetura do Abaré, foram realizados testes tendo como foco o Middrouter. O cenário de testes foi a rede em malha da Universidade Federal do Pará (UFPA), que foi implantada em uma área que possui prédios com altura média de oito metros, com predominância de árvores de grande porte e copas largas, típicas da região amazônica. Possui também altos índices pluviométricos. A Figura 4 apresenta o *backbone* instalado da rede em malha da UFPA.

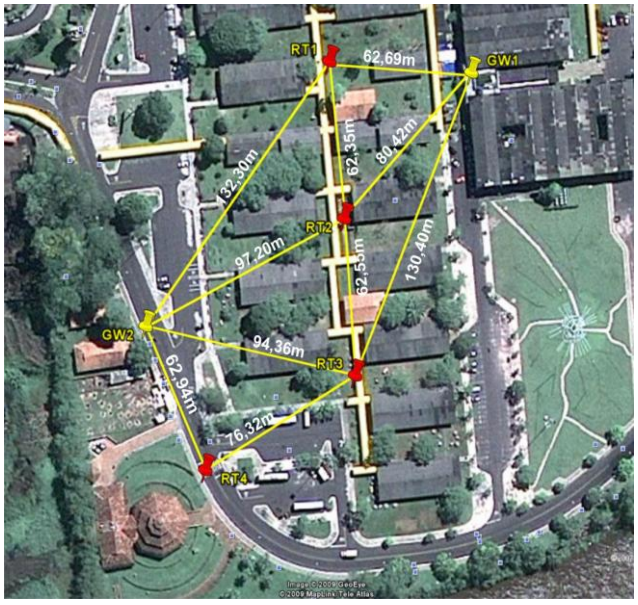

Figura 4: Backbone da rede em malha da UFPA

A rede em malha localizada na UFPA possui seis kits para redes em malha sem fio. Esses kits são compostos de caixa hermética para comportar os roteadores sem fio e uma antena omnidirecional de 18.5dBi de ganho. Os roteadores sem fio utilizados na rede são da marca Linksys e modelo WRT54GL. Estes possuem as configurações necessárias, para a utilização do *firmware* OpenWRT, sendo o mesmo baseado no sistema operacional Linux para sistemas embarcados. Segue abaixo, as configurações dos dispositivos.

- Arquitetura MIPS;
- Chipset Broadcom 5352EKPB;
- CPU speed: 200MHz;

- Memória Flash: 4MB;
- Memória RAM: 16MB;
- Interface Wireless: Broadcom BCM43xx – 802.11b/g;

5. Experimentos e Resultados

5.1 Teste de Carga

Para validar a parte de coletas do Middrouter e verificar seu comportamento em um sistema real foi realizado um teste de carga. Nele foram realizadas sucessivas requisições para um dos coletores presentes no Middrouter. O coletor em questão é o *get_mem_used*, que retorna a quantidade de memória utilizada. A intenção do teste era verificar a variação do tempo de resposta de acordo com o aumento do numero de requisições. Foram realizadas 10 execuções com um intervalo de confiança de 95% para cada número de requisições, no caso: 1, 2, 3, 4, 5, 10, 20 e 40.

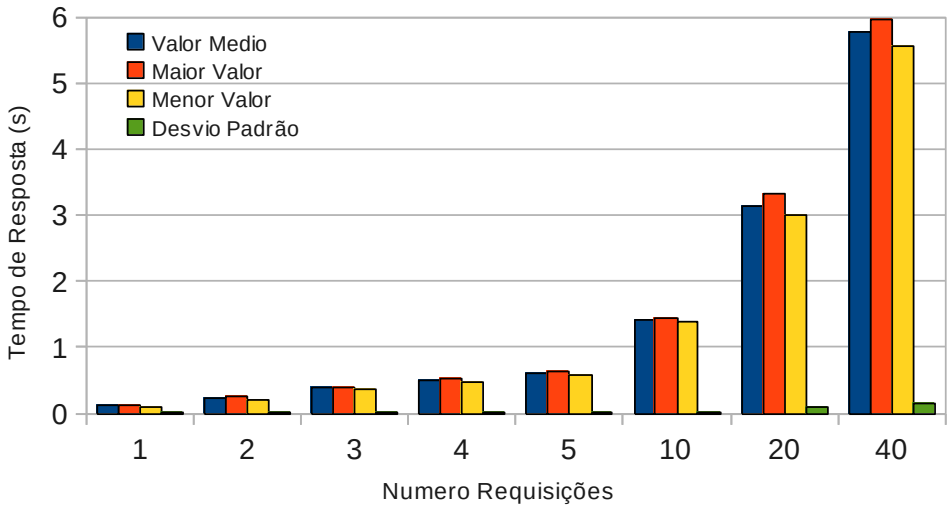

Figura 5: Tempo de Resposta do Middrouter

Na Figura 5 é possível perceber que o Middrouter pode responder as 40 requisições simultâneas em um tempo inferior a 6 segundos. É possível observar também um crescimento proporcional do tempo de resposta, uma vez que todos os pedidos são atendidos dentro dos limites de processamento do sistema.

5.2 Teste com o Agente Inteligente.

Uma das características mais importantes do Middrouter é a existência de agentes inteligentes que possibilitam a tomada de decisão por parte do roteador, ou seja, tornam possível a autonomia do roteador para coletar, analisar e tomar uma ação de forma autônoma.

Para validar este agente, foi implementado um simples agente de balanceamento de carga para demonstrar o funcionamento da parte autônoma do Middrouter. Os seguintes elementos foram desenvolvidos, utilizando os princípios propostos pelo Abaré:

- **ifstat** – É um coletor que foi implementado para monitorar a vazão;

- **ch_gw** – É um agente que troca o *gateway* do roteador hospedeiro caso ele receba uma ordem vinda de um dos *gateways*.
- **lb_gw** – É um Agente Inteligente, ele faz o uso do ifstat para monitorar a vazão, caso o número ultrapasse um limiar estabelecido ele envia uma ordem para alguns roteadores trocarem sua tabela de rota, alterando seu *gateway*, evitando uma sobrecarga em um determinado *gateway*.

No roteadores RT1, RT2 e RT3 foram implantados o **ch_gw** e no *gateway* GW1 o **ifstat** e o **lb_gw**. O CRON do Middrouter foi utilizado para agendar a execução do agente **lb_gw** em intervalos de tempo de 3 segundos. Para estes testes os roteadores RT4 e o GW2 não foram usados, sendo este último apenas usado como *gateway* alternativo em um dos testes. Todos os testes apresentados a seguir foram realizados 10 vezes, para gerar uma melhor confiabilidade nos testes.

Primeiramente foi realizado um teste com cada roteador isoladamente, para verificar a vazão máxima de cada um. Foi utilizada a ferramenta *Iperf* para gerar 8MB (Mega Byte) de dados que foram enviados usando o Protocolo de Controle de Transporte (*Transport Control Protocol* - TCP). O servidor ficou localizado na rede externa, ligado aos *gateway*s via Ethernet 100Mbits, o cliente estava dentro de cada roteador. A Figura 5 mostra os resultados obtidos.

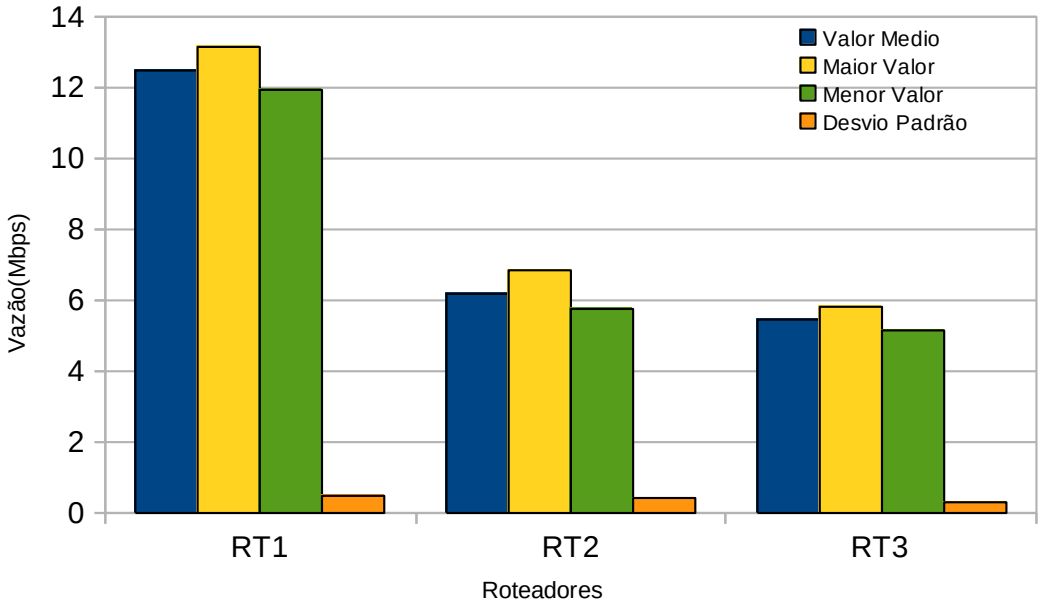

Figura 6: Vazão Individual dos Roteadores

É possível verificar os valores de 12.44Mbps, 6.18Mbps e 5.42Mbps para os roteadores RT1,RT2 e RT3 respectivamente. Atribuímos estas diferenças a fatores como distância e obstáculos que precisam ser vencidos pelo sinal de cada roteador. A partir dos dados coletados vimos que 12Mbps por segundo é o limite que estes roteadores conseguem alcançar através do GW2.

Com essas informações parametrizamos o **lb_gw** para alterar as rotas dos outros roteadores caso o valor da vazão do GW2 alcançasse 7Mbps, para ter uma margem que possa sustentar a comunicação com os demais roteadores que continuaram dependentes desse *gateway*.

Foram os três roteadores que de maneira concorrente fizeram requisições de um arquivo de 8MB para o servidor *iperf*, que está na rede externa como descrito anteriormente. Diante do trafego gerado o **lb_gw** foi ativado desencadeando a troca do *gateway* de um dos roteadores. A Figura 6 mostra os resultados obtidos:

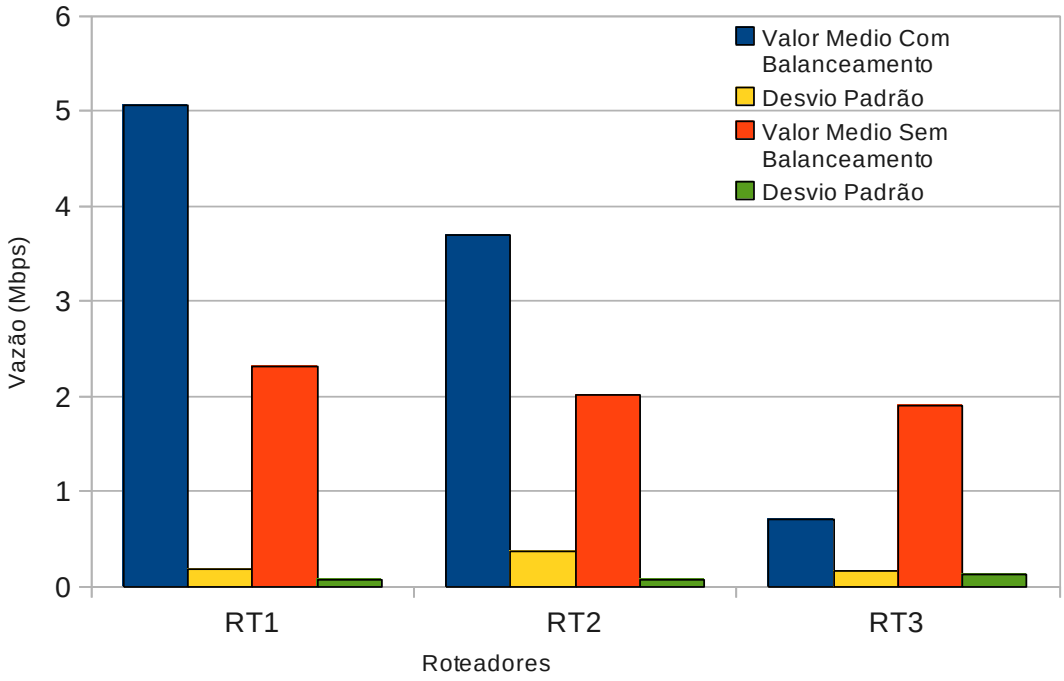

Figura 7: Vazão com os fluxos começando simultaneamente

É possível identificar, com o uso do **lb_gw**, uma melhora de 45.63% e 54.35% nos tráfegos do RT1 e RT2, respectivamente, porém o RT3 sofreu uma degradação em sua vazão. Isto ocorreu, pois o Agente **lb_gw** alterou a tabela de rotas do RT3 redirecionando-o para outro *gateway*, no entanto a sessão TCP foi perdida visto que a rede não implementa um tratamento para esta troca de *gateways* [Ito *et al* 2009].

Diante destas informações foi realizado um terceiro teste com os mesmo parâmetros do segundo, porém iniciando o tráfego do RT3, três segundos após os demais, desta forma a mudança de roteamento não quebraria a sessão TCP, pois ela ocorreria antes da sessão ser iniciada.

Na Figura 8 é possível perceber uma melhoria de 56.56% no trafego do RT3, uma vez que o TCP não sofre a quebra de sessão, promovendo o balanceamento de carga, ainda de maneira simples, pois esta não é a proposta do artigo. Com estes teste foi possível mostrar as facilidades proporcionadas pelo *framework*, que viabilizam a criação de outros módulos e agentes que resolvam problemas específicos, como balanceamento de carga, gerência de mobilidade entre outros.

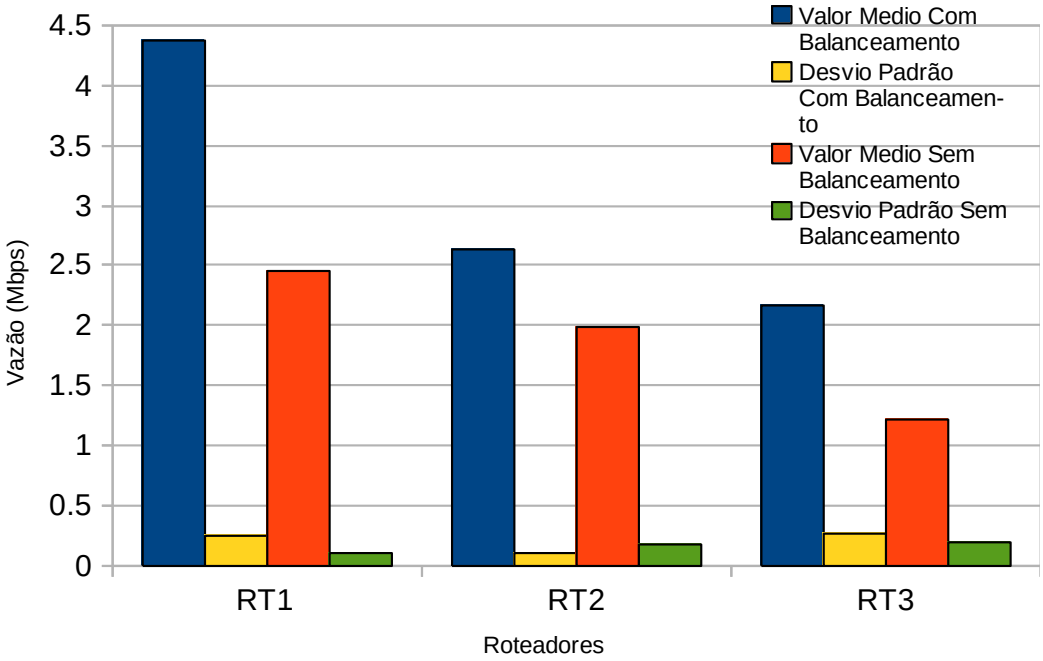

Figura 8: Vazão com os fluxos começando em tempos diferentes

É importante ressaltar que toda a comunicação entre os roteadores já é oferecida pelas camadas de entrada e saída e pelo parse XML, que abstraem a complexidade envolvida, deixando o desenvolvedor focado apenas na solução do problema principal.

6. Conclusões e Trabalhos Futuros

A implantação e gerenciamento são tarefas importantes para WMNs. Apesar de sua importância, apenas o monitoramento e poucas tarefas de configuração têm sido realizadas pelos *framewoks* existentes. Neste artigo, apresentamos uma nova arquitetura para o Abaré, levando em conta os aspectos de implantação e gerencia, bem como propiciando características autonômicas aos roteadores.

Este *framework* visa incentivar o desenvolvimento de ferramentas de gerência, uma vez que fornece a base teórica necessária para a sua criação, por especificar e padronizar os elementos de gerencia de uma WMN, dado que um dos maiores obstáculos para implantação em larga escala de Redes OpenMesh é a falta de ferramentas que possam auxiliar neste processo. A flexibilidade do Abaré permite que a inclusão de serviços através do Agente Expansor.

Os testes de carga com o Agente de Coleta mostraram que o sistema pode suportar um numero de 40 requisições sem gerar sobrecarga no sistema. Já os testes com os Agentes Inteligentes comprovaram a viabilidade do Middrouter para prover inteligência e autonomia da rede.

Como trabalhos futuros, pretendemos ampliar os módulos já propostos incluindo módulos de gerenciamento de usuários, *Qualidade de Serviço* (QoS) e *Qualidade de Experiência* (QoE).

Agradecimentos

Este trabalho foi financiado pela FAPESPA, FADESP, PROPESP – UFPA e CNPq (476202/2009-4.)

Referências

Aggelou, G. Wireless Mesh Networking With 802.16, 802.11, and ZigBEE.2008.

Aguiar, E. ; Bittencourt, P. ; Moreira, W. ; Abelém, A . Estudo comparativo de protocolos de roteamento para redes Mesh na região Amazônica. In: 25º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Belém/PA, Brazil. , 2007. v. 2. p. 1105-1110.

Berners-Lee,T. Fielding, R. e Frystyk, H., Hypertext Transfer Protocol - HTTP/1.0, RFC 1945, May 1996.

Campista, M. E. M. and at All, "Routing metrics and protocols for wireless mesh networks," Network, IEEE, vol. 22, no. 1, pp. 6–12, 2008.

Capone, A., Cesana, M., Napoli, S. e Pollastro, A. (2007) "MobiMESH: A Complete Solution for Wireless Mesh Networking", IEEE MASS 2007 (4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems) Pisa, Italy.

Cisco, "Cisco mesh products", ultimo Acesso, Março 2010. Disponível: <http://www.cisco.com/en/US/products/ps8368/index.html>

DD-WRT, "DD-WRT", Disponivel em: <http://www.ddwrt>. ultimo acesso, Dezembro 2010.

IEEE , "Ieee draft p802.11s d4.0." IEEE Unapproved Draft Std P802.11s/D4.0, Março 2010.

Ito, M., Shikama, T., e Watanabe, A. 2009. Proposal and evaluation of multiple gateways distribution method for wireless mesh network. 3rd international Conference on Ubiquitous information Management and Communication.

Jardosh, A. P. Suwannatat, P. Hollerer, T. Belding E. M. , and Almeroth, K. C. , "Scuba: Focus and context for real-time mesh network health diagnosis." in PAM, ser. Lecture Notes in Computer Science, M. Claypool and S. Uhlig, Eds., vol. 4979. Springer, 2008, pp. 162–171.

Kephart, J. O. e Chess, D. M.The vision of autonomic computing. Computer, 36(1):41–50, 2003.

Khalid, A., Haye, M. A., Khan, M. J., and Shamail, S. 2009. Survey of Frameworks, Architectures and Techniques in Autonomic Computing.In Proceedings of the 2009 Fifth international Conference on Autonomic and Autonomous Systems.

Microsoft, "MCL.". Disponivel em, <http://research.microsoft.com/mesh/>, Acessado em Março de 2010.

Moreira, W.; Aguiar, E.; Abelém, A. J. G.; Stanton, M.Using Multiple Metrics with the Optimized Link State Routing Protocol fo Wireless Mesh Networks. 26° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC 2008. Maio 2008.

Motorola, "MotoMesh", Acessado em Março de 2010. Disponível em http://developer.motorola.com/products/twowayradios/motomesh/ .

Nanda, S.e Kotz D. , "Mesh-mon: A multi-radio mesh monitoring and management system," Comput. Commun., vol. 31, no. 8, pp. 1588–1601, 2008.

Openwrt, "Openwrt wireless fredom," Disponivel em: <http://openwrt.org/> .Ultimo

acesso, Março 2010.

Park, N. e Lee, G. "Agent-based Web services middleware," Global Telecommunications Conference, 2003. GLOBECOM '03.IEEE , vol.6, no., pp. 3186-3190 vol.6, 1-5 2003.

Passos, D. e Albuquerque, C., “Uma Abordagem Unificada para Métricas de Roteamento e Adaptação Automática de Taxa em Redes em Malha Sem Fio”, *Simpósio Brasileiro de Redes de Computadores(SBRC 2009)*, Recife, maio de 2009.

Pinheiro, B. A. ; Nascimento, V. B. ; Moreira, W. ; Abelém, A . Abaré: A Deployment and Management Framework for Wireless Mesh Network. In: IEEE Latin-American Conference on Communications (IEEE LatinCom 2009), 2009, Medellín, Colombia.

Ramachandran, K. Belding-Royer E., e Aimeroth K., “Damon: a distributed architecture for monitoring multi-hop mobile networks,” Oct. 2004, pp. 601–609.

ReMesh, “ReMesh”. Disponivel em: <http://vmesh.inf.uth.gr/>.Ultimo acesso, Março 2010.

Riggio, R. Scalabrino, N. Miorandi, D. e Chlamtac, I. “Janus: A framework for distributed management of wireless mesh networks,” TridentCom 2007. 3rd International Conference on, 2007.

Tsarmpopoulos, N., Kalavros I. e Lalis S. (2005) A Low-cost and Simple-to-Deploy Peer-to-Peer Wireless Network based on Open Source Linux Routers.

Vmesh, “Vmesh - wireless network testbed”, Disponivel em: <http://vmesh.inf.uth.gr/>, ultimo acesso, Março 2010.