Modelo analítico-determinístico para a priorização de tráfego em Redes Definidas por Software com Network Calculus

Michael Prieto Herńandez[1]*, Ricardo Girnis Tombi[1], Samuel Kopp[1]*, Daniel Batista[2], Cíntia Borges Margi[1], Regina Melo Silveira[1]

[1]Departamento de Engenharia de Computação e Sistemas Digitais – Escola Politécnica
Universidade de São Paulo

[2]Departamento de Ciência da Computação – Instituto de Matemática e Estatística
Universidade de São Paulo

{mhernandez,rtombi,samuel,cbmargi,regina}@larc.usp.br,

batista@ime.usp.br

Abstract. *Different classes of packets traffic require specific performance of network resources in terms of Quality of Service metrics. The models proposed to ensure end-to-end QoS have not been well implemented due to their complexity and lack of scalability. The Software Defined Networks feature an innovative paradigm that aims to make the networks programmable. This paradigm allows to know the state of the network and its topology in real time, which makes possible the reconfiguration of routes and allocating resources dynamically. This paper proposes an analytical-deterministic model that allows to prioritize IP packets traffic, taking advantage of SDN and basing the prioritization policies over definitions found in the Network Calculus mathematical theory. The results show the feasibility of using NetCalc as a tool in the packet flows policing in order to ensure QoS on a software defined network.*

Resumo. *Diferentes classes de tráfego requerem desempenho específico dos recursos de rede em termos de Qualidade de Serviço. Modelos propostos para garantir QoS fim-a-fim na literatura não foram implementados de forma satisfatória até hoje devido à sua complexidade e falta de escalabilidade. As Redes Definidas por Software (SDN) apresentam um paradigma inovador que torna as redes programáveis, permitindo conhecer estado e topologia das mesmas em tempo real, além de possibilitar configuração de rotas e alocação de recursos dinamicamente. Este trabalho apresenta um modelo analítico-determinístico para priorização de tráfego aproveitando as vantagens da SDN, baseado em políticas de priorização derivadas da teoria do Network Calculus. Os resultados obtidos comprovam a viabilidade do NetCalc para subsidiar controle de fluxos garantindo QoS em SDN.*

*Esse projeto foi financiado pelo processo nº 2014/50386-5, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

1. Introdução

A distribuição de vídeo através da Internet tem se tornado cada vez mais frequente, e algumas previsões indicam que até o final desta década noventa porcento do tráfego da Internet será constituído por fluxos de vídeo [Cisco 2014]. Considerando as previsões de expansão de consumo desta mídia é necessário desenvolver mecanismos nas redes que permitam viabilizar soluções que apresentem bom desempenho, como robustez e escalabilidade.

Visando garantir QoS fim-a-fim, a Internet Engineering Task Force (IETF) propôs dois modelos, Serviços Integrados (IntServ) [Braden et al. 1994] e Serviços diferenciados (DiffServ) [Blake et al. 1998] os quais não foram satisfatoriamente implementados globalmente até hoje devido à sua complexidade e falta de escalabilidade [Ni 1999]. Embora a IETF tenha criado um mecanismo de transporte (Multi Protocol Label Switching) [Rosen et al. 2001] que possibilita a associação de requisitos de QoS, baseado nos rótulos carregados pelos pacotes, este carece de reconfiguração e adaptabilidade em tempo real [Egilmez and Dane 2012].

Devido às limitações da Internet atual em prover conexão com QoS e, consequentemente, em prover qualidade de experiência (QoE) compatível com as expectativas do usuário, os mecanismos de QoS continuam com grande foco, principalmente em se tratando de mecanismos que atendam as necessidades de sistemas autônomos como as redes dos Provedores de Serviços de Internet (ISP).

Com o surgimento das Redes Definidas por Software (SDN) [ONF 2012], é possível especificar a partir da implementação por linguagem de programação o comportamento de um comutador de pacotes ou *switch*, o que diminui custos e ao mesmo tempo, aumenta a flexibilidade do processo de encaminhamento de pacotes IP. Esta abordagem permite criar mecanismos, que através do monitoramento da rede, forneçam um controle ativo da mesma para superar problemas de limitações de recursos e congestionamentos.

Uma das implementações de maior repercussão de SDN é o protocolo OpenFlow [Mckeown et al. 2008], mantido atualmente pela Fundação Open Networking. Este protocolo é uma implementação de código aberto que tem sido utilizado pela comunidade acadêmica e por corporações para resolver diversos problemas em suas redes. SDN através de OpenFlow centraliza o controle e padroniza a comunicação entre os comutadores e o controlador da rede, diminuindo a complexidade tanto na borda como no núcleo da rede. Do ponto de vista de QoS, SDN elimina limitações chaves que não permitiram que os mecanismos tradicionais, mencionados anteriormente, fossem implantados até hoje. Algumas das características de SDN que facilitam a implementação de mecanismos de QoS são [ONF 2012]:

- Conhecimento da topologia de rede em tempo real;
- Centralização do algoritmo de roteamento que trabalha sobre a topologia;
- Implantação de forma padronizada das políticas de roteamento sobre a topologia de rede;
- Padronização dos atendimentos para diferentes classes de fluxos no percurso entre ilhas SDNs.

Levando em consideração estas características, o objetivo deste trabalho é elaborar, especificar e desenvolver um modelo analítico-determinístico baseado na teoria de

Network Calculus [Cruz 1991b, Cruz 1991a], que permita implantar e operar, através de um módulo de software, uma rede com capacidade de priorização, marcação, filtragem ou policiamento de fluxos de pacotes a fim de garantir QoS usando o paradigma SDN. Esse modelo avança o estado da arte em SDN já que, pelo nosso melhor conhecimento, não há trabalhos na literatura que proponham modelos como este. Os resultados obtidos nos experimentos, comprovam a viabilidade do NetCalc para subsidiar controle de fluxos garantindo QoS em SDN.

Este trabalho apresenta na seção 2 os principais conceitos relacionados com as SDNs e especifica onde, dentro deste contexto, nosso módulo será elaborado. Logo depois, são mencionados os principais mecanismos de QoS existentes até hoje e como nós reutilizamos algumas características destas arquiteturas. Detalha-se depois o conjunto de definições da teoria de NetCalc utilizadas na elaboração do modelo analítico-determinístico proposto. Mais adiante é feita a modelagem de uma rede, fazendo uso desta teoria, com o fim de, na seção 3, detalhar a proposta do modelo analítico-determinístico que permite priorizar fluxos de pacotes através da teoria de NetCalc em uma rede definida por software. São citados, na seção 4, vários trabalhos relacionados com esta pesquisa, ressaltando a nossa contribuição ao estado da arte. Os experimentos realizados, assim como os resultados obtidos, são apresentados na seção 7. As considerações finais são apresentadas na seção 8.

2. Arcabouço teórico

Esta seção resume os principais tópicos necessários para a compreensão do modelo proposto na seção 3.

2.1. Redes Definidas por Software

As redes definidas por software (SDN) [ONF 2012] são um novo paradigma que visa criar facilmente redes dinâmicas, gerenciáveis e adaptáveis. A arquitetura SDN, (Figura 1), define a rede como uma entidade programável e abstrai os serviços e aplicações de rede das camadas inferiores através de padrões abertos como o protocolo *OpenFlow*.

Na arquitetura *OpenFlow*, o controlador da SDN é responsável pelo reconhecimento dos fluxos de pacotes, determinar uma ou várias ações sobre eles e atualizar as tabelas de fluxos nos comutadores. *OpenFlow* fornece um conjunto de estatísticas que permitem que o controlador conheça a topologia e o estado dos nós da rede (comutador *OpenFlow*) em tempo real, possibilitando o tratamento dinâmico de fluxos.

O módulo implementado neste trabalho pode ser considerado como um serviço de rede acoplado a um controlador. Este módulo usa o modelo analítico proposto para, através de *OpenFlow*, aplicar políticas de priorização e garantir atraso máximo fim-a-fim aos fluxos.

2.2. Qualidade de Serviço

A arquitetura dos **Serviços Integrados** é caraterizada pela reserva de recursos. Antes que a transmissão seja efetuada, uma rota é estabelecida e são reservados recursos ao longo do percurso, através do protocolo RSVP [Braden et al. 1994]. Essa reserva de recursos exige que mudanças sejam realizadas tanto no núcleo quanto na borda da rede. Os **Serviços Integrados** propõem duas classes de serviço além do melhor esforço:

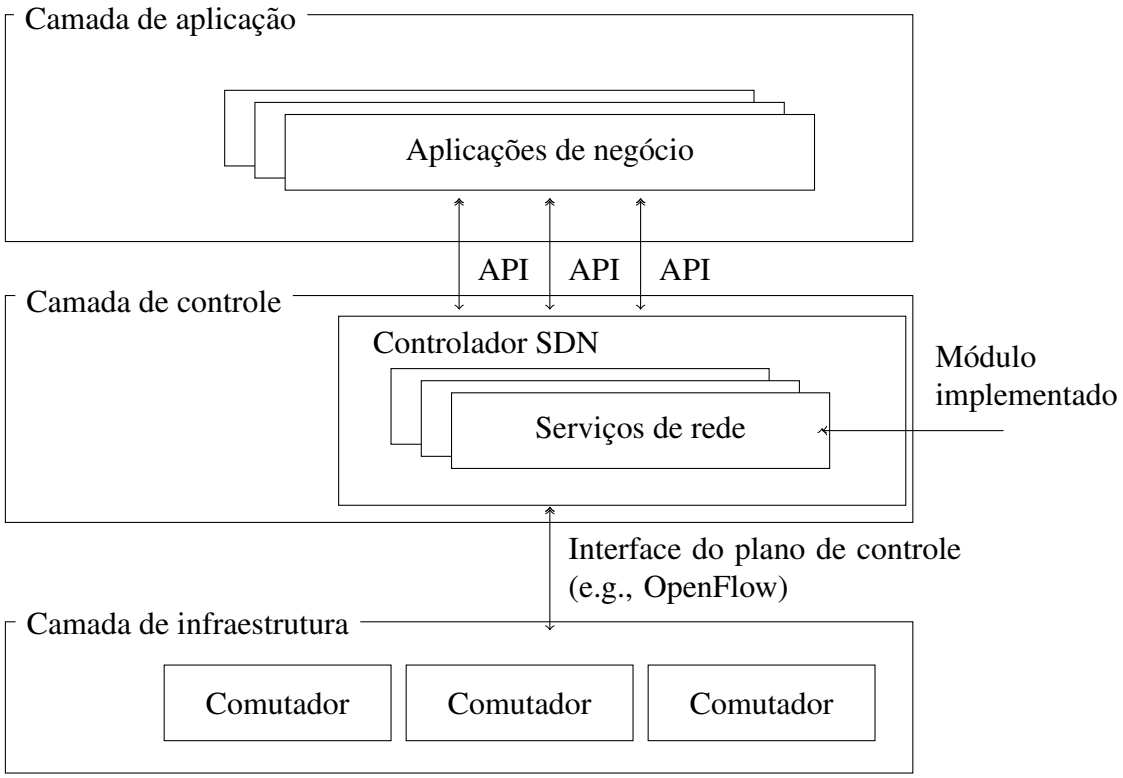


Figura 1. Arquitetura de uma rede definida por software

- Serviço Garantido (*Guaranteed Service*) [Shenker et al. 1997] para aplicações que requerem limites fixos de atraso;
- Serviço de carga controlada (*Controlled-load Service*) [Wroclawski 1997] para aplicações altamente sensíveis a redes congestionadas.

Problemas de escalabilidade, complexidade e custo de implementação são conhecidos e detalhados em [Ni 1999]. Porém, nós reutilizamos o conceito de serviço garantido, a fim de avaliar os fluxos que serão aceitos na rede e fornecer rotas com o menor atraso possível. Este trabalho propõe alocação de recursos não por fluxos individuais e sim por fluxos agregados de uma mesma classificação.

Em **Serviços Diferenciados**, os pacotes são marcados diferentemente visando criar várias classes de fluxos. Os pacotes em classes diferentes recebem serviços diferenciados. Nesta arquitetura a qualidade de serviço é garantida por meio de filas com prioridade [Ni 1999].

Ao contrário dos Serviços Integrados, os Serviços Diferenciados só precisam de mudanças nas bordas da rede. Os roteadores de borda devem classificar os pacotes e os roteadores do núcleo da rede encaminham estes pacotes segundo as prioridades.

Nesta pesquisa a identificação, classificação e a decisão de enfileiramento de fluxos é feita no controlador da rede. Cada fluxo é cadastrado inicialmente no controlador SDN pela fonte transmissora (e.g., instituição fornecedora de fluxos) e deve conter as informações necessárias para a identificação do fluxo bem como os parâmetros de QoS que caracterizam o perfil de tráfego a ser tratado.

2.3. Network Calculus

Network Calculus (NetCalc) é um tópico que fornece um conjunto de resultados matemáticos visando descrever problemas de fluxos encontrados em redes [Cruz 1991b, Cruz 1991a]. NetCalc viabiliza o entendimento de propriedades fundamentais dos serviços integrados, agendamento, alocação de recursos e dimensionamento de atraso, onde, com o fim de fornecer garantias para os fluxos de dados, algumas caracterizações dos tráfegos e serviços da rede são necessários.

Com esse propósito, NetCalc define os conceitos de **curvas de chegadas** e **curvas de serviço**. O subconjunto de definições do arcabouço do NetCalc utilizadas neste trabalho são as seguintes:

- **Notação**
  $R(t)$: função acumulativa de entrada no sistema, estritamente crescente, que determina a quantidade de unidades (bits) em um tempo t
  $R'(t)$: função acumulativa de saída do sistema, estritamente crescente, que determina a quantidade de unidades (bits) atendidos em um tempo t
  $\otimes$: Operador que indica a convolução de duas funções
  $\oslash$: Operador que indica a deconvolução de duas funções
  $\wedge$: Operador que indica o ínfimo de um conjunto de valores

- **Curva de chegada**: Dada uma função crescente α definida para $t \geq 0$ podemos dizer que um fluxo R é restrito por α se e somente se para todo $t \leq s$:

$$R(t) \text{ - } R(s) \leq \alpha(t \text{ - } s), \forall s, t \textit{ onde } 0 \leq s \leq t \tag{1}$$

  Pode-se dizer que R tem α como uma representação abstrata.
- **Curva de chegada remanescente**: Considere um fluxo delimitado por uma curva de chegada α, que atravessa um sistema que oferece uma curva de serviço β, o fluxo de saída é delimitado pela curva de chegada através da operação de deconvolução dada por:

$$\alpha^* = \alpha \oslash \beta \tag{2}$$

- **Curva de serviço**: Considere um sistema S e um fluxo que passa através de S com funções de entrada e saída R e R'. Pode-se dizer que S oferece uma curva de serviço β para o fluxo se e somente se β é estritamente crescente, $\beta(0) = 0$ e:

$$R'(s) - R'(t) \geq \beta(s - t), \forall s, t \textit{ onde } 0 \leq s \leq t \tag{3}$$

  ou

$$R'(t) \geq (R \otimes \beta)(t) \tag{4}$$

- **Atraso**: Menor intervalo de tempo $A(t)$ tal que todas as tarefas iniciadas no instante t sejam completadas para o instante $t + A(t)$ e pode ser especificada da seguinte forma:

$$A(t) = \min\{T : T \geq 0 \wedge R(t) \leq R'(t + T)\} \tag{5}$$

  O atraso no instante t é o atraso que seria experimentado por um *bit* que chega no tempo t se todos os *bits* recebidos antes dele já fossem servidos. Para calcular o limite do atraso o conceito de distância horizontal deve ser utilizado.

- **Distância horizontal**: Para duas funções α e β estritamente crescentes o desvio horizontal $h(\alpha, \beta)$ é dado por:

$$h(\alpha, \beta) = \max_{s \geq 0}\{\min\{T : T \geq 0 \wedge \alpha(s) \leq \beta(s+T)\}\} \tag{6}$$

  A distância horizontal é o valor utilizado para determinar o limite do **atraso** que pode ser descrito como:
- **Limite de atraso**: Para um fluxo restrito por uma curva de chegada α com instantes diferentes e consecutivos de início, sendo servido por uma política de *first-in-first-out* e um sistema S que garante uma curva de serviço β, o atraso para todo t satisfaz:

$$A(t) \leq h(\alpha, \beta), \forall t \geq 0 \tag{7}$$

- **Concatenação**: Considere um fluxo que passa através dos sistemas S_1 e S_2 em sequência. Assuma que S_i oferece uma curva de serviço β_i,
  $i = 1, 2$ ao fluxo. Então a concatenação dos dois sistemas oferece uma curva de serviço $\beta_1 \otimes \beta_2$ ao fluxo.

2.4. Representação do modelo da rede

Neste trabalho a topologia da rede é tratada como um grafo conectado e ponderado onde cada um dos nós oferece uma curva de serviço β. O peso das arestas é dado pelo atraso que experimenta um fluxo caracterizado por uma curva de chegada ao longo das possíveis rotas desde a fonte até o destino, como pode ser observado na Figura 2.

Formalizando matematicamente a rede (Figura 2), para um fluxo que passa através dos nós 1,2,3 temos que:
Seja $G = (V, E)$ uma rede onde V é o subconjunto de nós que compõem a rede, representados pelas suas respetivas curvas de serviço (β),

- V:$\{\beta_1, \beta_2, \beta_3\}$

E é o subconjunto de segmentos de caminhos, formados por enlaces pelos quais o fluxo irá passar da origem X ao destino Y, as quais tem associado um atraso (h), considerando o fluxo de entrada (α) e seu atendimento pelo nó através do serviço (β).

- E:$\{h_{x->y}(\alpha, \beta_1 \otimes \beta_2 \otimes \beta_3)\}$

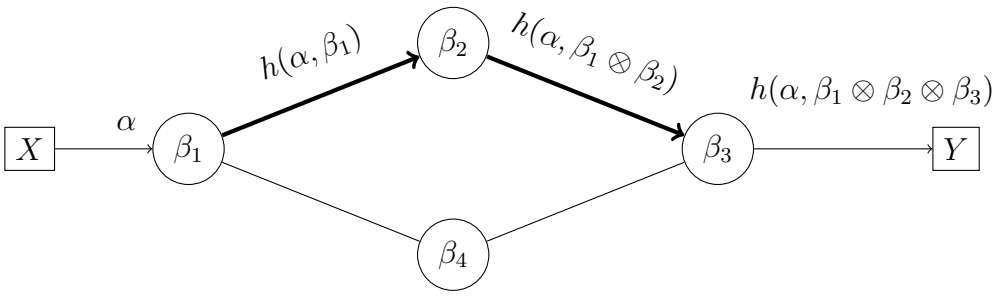


Figura 2. Modelo de rede através de NetCalc

Sendo assim, ao escolher um caminho P qualquer, desde uma fonte transmissora até o destino que requisita a transmissão, o atraso máximo A_{max}, experimentado pelo

fluxo α através de P será:

$$A_{\max} \leq h(\alpha, (\beta_i \otimes \beta_{i+1} ..., \otimes \beta_m)) \quad (8)$$

Observe que no grafo não é considerado nem o enlace inicial de X para o primeiro nó, nem o atraso associado. A explicação desta consideração é que o controlador da SDN somente possui informação sobre os comutadores SDN e não dos equipamentos (e.g., *hosts*) conectados em cada um deles.

3. Modelo para priorização de tráfego com Network Calculus

O grau de sensibilidade dos fluxos priorizados às métricas de QoS como o atraso, a variação de atraso, a largura de banda e a perda de pacotes, vai depender do tipo de fluxo. Para vídeos interativos (vídeo conferência) [Hattingh, Christina and Szigeti 2004] recomenda uma perda de pacotes menor que 1%, atraso menor que 150 ms e uma variação de atraso não maior que 30 ms. Por sua vez, para *streaming* de vídeo, o mesmo autor sugere uma perda menor que 5% e um atraso não maior que 5 segundos. Para este caso não há valores significativos para a variação de atraso.

Visando fornecer atendimento diferenciado para distintas classes de fluxos, nós precisamos que os nós da rede ofereçam mais de uma curva de serviço. Sem perda de generalidade, neste modelo é proposta a criação de três classes de fluxos: **C1, C2, C3** sendo C1 a classe com maior prioridade e C3 com a menor prioridade. Cada classe tem associada a sua respectiva fila de atendimento as quais, neste trabalho, são do tipo *first-in-first-out* (FIFO), como é observado na Figura 3.

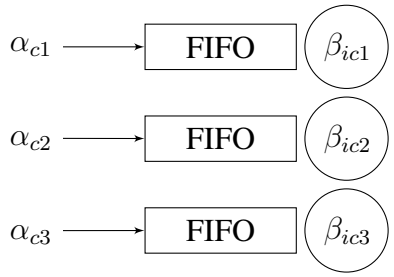


Figura 3. Atendimento diferenciado nos canais de saída dos nós da rede

Uma nova rede pode ser definida como:
Seja $G = (V, E)$ uma rede onde V é o subconjunto de nós que compõem a rede, representados pelas suas respetivas curvas de serviço específicas para cada classe de fluxo ($\beta_{c1,c2,c3}$). E é o subconjunto de segmentos de caminhos, formados por enlaces pelos quais o fluxo irá passar da origem X ao destino Y, os quais tem associado um atraso (h), considerando o fluxo de entrada (α) e seu atendimento **diferenciado** pelo nó através do serviço específico requisitado ($\beta_{c1|c2|c3}$). Ver (Figura 4) :

- V:$\{(\beta_{1c1}, \beta_{1c2}, \beta_{1c3}), (\beta_{2c1}, \beta_{2c2}, \beta_{2c3}), (\beta_{3c1}, \beta_{3c2}, \beta_{3c3})\}$;
- E:$\{h_{X->Y}(\alpha_{cm}, \beta_{1cm} \otimes \beta_{2cm} \otimes \beta_{3cm})\}$

Novamente escolhendo um caminho P, desde uma fonte transmissora até o destino que requisita a transmissão, o atraso máximo $A_{\max}$, experimentado pelo fluxo α com atendimento diferenciado Cm através de P será:

$$A_{\max} \leq h(\alpha_{cm}, (\beta_{icm} \otimes \beta_{(i+1)cm} ..., \otimes \beta_{jcm})), \text{ } \textit{onde } i, j \in \mathbb{N}, j = |P| \quad (9)$$

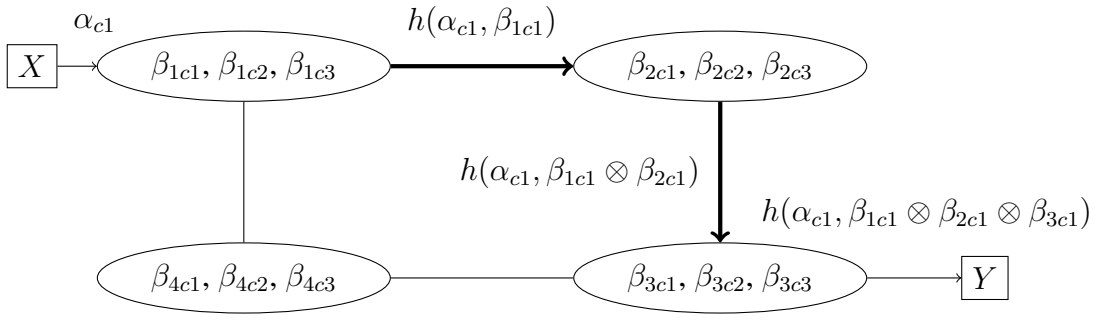


Figura 4. Modelo de rede com atendimento diferenciado

3.1. Agregação de fluxos

Até o momento nosso modelo permite atender três fluxos, se e somente se, eles precisarem de classes de serviços diferentes (i.e., um fluxo por classe). Com o fim de agregar fluxos da mesma classe de atendimento, nosso modelo deve possuir uma forma de somar os fluxos admitidos no sistema, visando dar como resultado o atraso máximo experimentado, desta vez não por um fluxo, e sim pelos fluxos agregados de uma mesma classe.

Para um conjunto de fluxos da forma α que passam pelo caminho P, o atraso máximo experimentado pelo fluxo resultante da agregação é dado pela expressão:

$$A_{\max} \leq h(\alpha_{1cm} + \alpha_{2cm} + \alpha_{3cm}..., +\alpha_{ncm}, (\beta_{icm} \otimes \beta_{(i+1)cm}..., \otimes\beta_{jcm})) \qquad (10)$$

Em teoria n pode chegar até o infinito, na prática só serão aceitos no sistema um conjunto de fluxos, onde $A_{\max}$ experimentado pela agregação, seja menor ou igual ao menor dos atrasos especificados por cada fluxo cadastrado no **controlador** da rede.

3.2. Melhor caminho fim-a-fim

Múltiplos algoritmos para encontrar o caminho mais curto entre os vértices de um grafo foram propostos ao longo do tempo. Um dos mais utilizados nas redes (e.g., IS-IS [Smit and Li 2004], OSPF [J.Moy 1998]) é o algoritmo de Dijkstra [Dijkstra 1959].

Qualquer algoritmo, que para encontrar o caminho mais curto leve em consideração o valor associado a cada aresta, pode utilizar nosso modelo analítico. Neste trabalho utilizaremos Dijkstra, sendo este relativamente simples e eficiente (i.e., $O(n \log(n))$), onde n é a quantidade de vértices do grafo.

Um aspecto a ser considerado na computação do melhor caminho fim-a-fim é que, segundo foi demostrado na teoria de *Network Calculus*, a soma dos atrasos experimentados em cada nó da rede (i.e., $h(\alpha, \beta_1) + h(\alpha^*, \beta_2) + h(\alpha^*, \beta_3)... + h(\alpha^*, \beta_n)$), é maior do que a convolução das curvas de serviço oferecidas pelos nós no caminho (i.e., $h(\alpha, \beta_1 \otimes \beta_2 \otimes \beta_3... \otimes \beta_n)$). Portanto devemos adaptar o algoritmo de Dijkstra visando ser coerente com as inequações (8)(9) e (10), como é mostrado no Algoritmo 1. Considere que quando se acessa a curva de serviço β, em um comutador, ela corresponde ao atendimento diferenciado solicitado pelo fluxo, caraterizado por α.

Algoritmo 1: Dijkstra utilizando NetCalc

```
Data: rede, fonte, destino, alpha
Result: melhor_rota, atraso_máximo
begin
    atraso[fonte] := 0 ;                                  // Atraso para chegar em [comutador]
    alpha_agregada := fonte.alpha + alpha ;                                  // Agregação, (10)
    atraso_máximo := 0;
    for comutador ∈ rede do
        if comutador ≠ fonte then
            atraso[comutador] := INFINITO;
            anterior[comutador] := INDEFINIDO;
        end
        adiciona comutador em nao_visitados
    end
    while nao_visitados ≠ ∅ do
        /* c é a fonte na 1ra iteração                                                    */
        c := comutador ∈ nao_visitados com menor atraso[c];
        remover c de nao_visitados;
        for vizinho v de c do
            /* Atraso para chegar em v, (8)(9)                                          */
            a := h(alpha_agregada, c.beta_{c->v});
            if a < atraso[v] then
                /* v.betas, curvas de serviço de todas as portas do comutador
                   */
                v.betas = c.beta_{c->v} ⊗ v.betas ;                          // Concatenação
                atraso[v] := a ;                                  // Atualização do atraso
                anterior[v] := c ;                                  // Construção da rota
            end
            if v = destino then
                atraso_máximo := h(alpha_agregada, convolução(anterior[].betas));
                /* Rota fonte->destino                                                  */
                return anterior[].invertido(), atraso_máximo ;
            end
        end
    end
end
```

4. Trabalhos relacionados

Vários trabalhos apresentam o uso do *OpenFlow* para a otimização da transmissão de fluxos, onde o melhor esforço da Internet não atende os requisitos mínimos em termos de qualidade de serviço. Uma destas propostas implementa um módulo que é denominado OpenQoS integrado ao controlador *OpenFlow*, que utiliza um mecanismo de diferenciação de tráfego. Este módulo permite tratar dinamicamente, de forma diferenciada, os fluxos de vídeo escolhendo a rota que mais se adequa aos parâmetros de QoS requisitados [Egilmez and Dane 2012].

Em outro trabalho Egilmez propõe a utilização de codificação de vídeo escalável criando com isso diferentes camadas de qualidade de serviço e priorização, que serão usadas de acordo com as condições da rede [Egilmez et al. 2011]. A priorização é feita estaticamente no controlador no instante de associar uma rota a um fluxo e não existe uma alocação de recursos no *switch*.

Outro trabalho [Owens II and Durresi 2013] propõe ainda a implantação de um módulo no controlador *OpenFlow* que manipula métricas de QoS. O protocolo proposto visa garantir o entendimento mútuo entre o transmissor e o receptor na hora da solicitação de um vídeo e a procura de uma rota com qualidade de serviço. Basicamente é uma troca de mensagens que resulta no estabelecimento ou não de uma sessão. O autor propõe modificações ao protocolo *OpenFlow* para implementar a sua solução, acrescentado métricas de Qualidade de Serviço nas descrições das filas dos comutadores *OpenFlow*.

Em um trabalho mais recente Qin [Qin et al. 2014] utiliza NetCalc e Algoritmo Genético para definir dinamicamente diferentes oportunidades e agendamento de atendi-

mento de fluxos por uma estrutura SDN para cenários de Internet das Coisas (IoT – Internet of Things) em que são consideradas redes sem fio heterogêneas. O sistema implementado (uma extensão do *Multinetwork INformation Architecture* (MINA) ao ser integrado a veículos elétricos, demonstrou ganhos de eficiência na capacidade de atendimento em transmissões de dados gerados em cenário de IoT baseada em redes heterogêneas.

A contribuição de nosso trabalho é explorar o NetCalc como uma ferramenta para que o controlador SDN possa fazer a seleção de rotas para o atendimento a diferentes classes de fluxos, levando em consideração requisitos de QoS.

5. Experimentos e resultados

Visando validar os aspectos teóricos deste trabalho, descritos na seção 3, foi elaborado um ambiente virtual composto pelo controlador de rede Floodlight [Shah et al. 2013] e uma topologia de comutadores e estações (i.e., $ht1, ht2, ht3, ht4$), criada através da ferramenta Mininet [Lantz et al. 2010], como pode ser observado na Figura 5. Este ambiente permite a simulação de uma rede definida por software que opera através do protocolo OpenFlow.

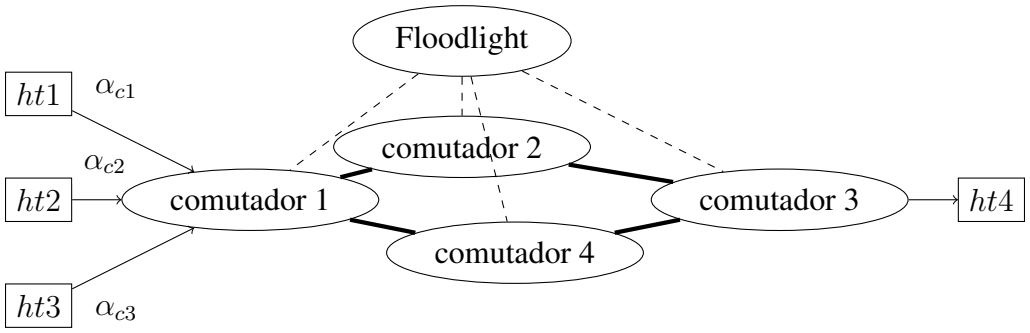


Figura 5. Rede virtual SDN.

5.1. Módulo para o controlador SDN

Foi desenvolvido um **módulo** dentro do controlador da rede que permite cadastrar as especificações dos fluxos de dados a serem transmitidos. Os parâmetros a seguir foram utilizados para a caracterização do tráfego:

- **ipsrc**: Endereço da fonte transmissora;
- **psrc**: Porta utilizada na transmissão;
- **ac**: Função envelope que carateriza o tráfego;
- **class**: Classe de fluxo que especifica o tipo de atendimento (C1, C2, C3);
- **delay**: Atraso máximo fim-a-fim suportado.

Uma vez que chega um pacote de rede ao controlador, este módulo compara os parâmetros cadastrados com os atributos do pacote (i.e., ipsrc, psrc). No caso em que o pacote seja identificado positivamente e o atraso virtual máximo da rota fim-a-fim, calculado através do algoritmo 1, seja menor ou igual ao atraso máximo cadastrado, o fluxo é aceito na rede. Logo depois é definida a rota com a priorização requisitada nas tabelas de fluxos nos comutadores.

O módulo é responsável pela distribuição dos recursos disponíveis nas portas de saída dos comutadores *Openvswitch* [Pfaff et al. 2009]. Esta distribuição é correspondente ao grau de priorização dos fluxos, e é realizada utilizando filas *Hierarchical Token*

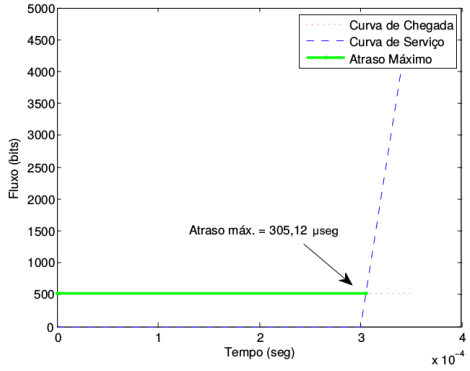
Bucket [Hubert and Others 2002], através do protocolo *The Open vSwitch Database Management Protocol* [Pfaff and Davie 2013]. Isto permite que fluxos com classes diferentes experimentem atrasos de acordo com a utilização de sua fila de atendimento.

Visando obter os resultados fornecidos pela teoria de NetCalc o módulo implementado utiliza a biblioteca DiscoDNC[1]. Os principais motivos para a escolha desta biblioteca, depois da análise de várias [Boyer 2010], foram a linguagem de programação em que é escrita (i.e., Java), que é a mesma do controlador Floodlight, e o fato do DiscoDNC fornecer classes prontas que representam as definições apresentadas na seção 2.3.

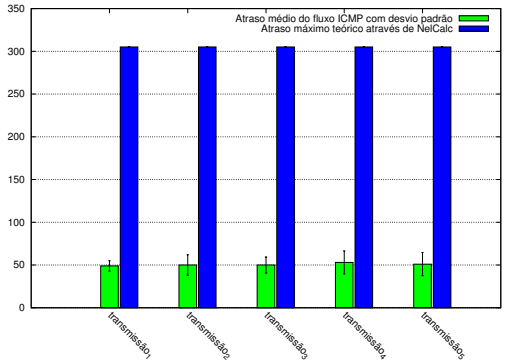
5.2. Descrição do experimento

O primeiro objetivo do experimento foi validar o correto funcionamento do ambiente simulado. Para isso foi modelado um fluxo com origem-destino $ht1 - ht4$ e prioridade alta *C1*. A curva de chegada deste fluxo foi considerada como $\alpha = \gamma_{r,b} = \gamma_{64,64}$, isto é uma taxa de 64 bytes com rajadas também de 64 bytes. Inicialmente todas as portas dos comutadores da rede possuem uma capacidade de $100Mbps$.

Várias análises têm sido realizadas com relação ao desempenho dos comutadores *Openvswitch* em termos de atraso de atendimento [Jarschel et al. 2011]. Esta métrica varia entre 35 e 100 microssegundos e vai depender fundamentalmente da dimensão da tabela de fluxos e do tamanho dos pacotes a serem processados. Neste experimento foi adotado um atraso de $T = 0.0001s$, sendo que as curvas de serviço das filas de saída dos comutadores estão dadas por $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta_{R,T} = \beta_{12500000,0.0001}$. A rota fim-a-fim foi inserida manualmente através dos comutadores 1,2 e 3. Aplicando a definição (8) o atraso máximo teórico é de 305.12 microssegundos como pode ser observado na Figura 6(a).



(a) Atraso máximo entre $\gamma_{64,64}$ e $\beta_{12500000,0.0003}$



(b) Atrasos dos fluxos ICMP e atraso máximo teórico com NetCalc

Figura 6. Comparação entre os atrasos teóricos através de NetCalc e reais no ambiente virtual.

Visando realizar uma comparação entre os atrasos que o fluxo modelado experimentou teoricamente e um fluxo real com as mesmas características, foram feitas 5 transmissões ICMP de 60 s desde $ht1$ até $ht4$. O resultado desta comparação está refletida na Figura 6(b), onde o atraso máximo do fluxo ICMP foi de 66.5 microssegundos. Então, aplicando (8) novamente temos que: $A_{\max} \leq \frac{b}{\min(R1,R2,R3)} + \sum_{n=1}^{3} T_n$

[1]The Disco Deterministic NetCalc, http://disco.cs.uni-kl.de/index.php/projects/disco-dnc

Tabela 1. Fluxos cadastrados no controlador SDN

	ipsrc	psrc	class
Fluxo 1	ht1	5001	C1
Fluxo 2	ht2	5002	C2
Fluxo 3	ht3	5003	C3

Logo, $66.5 \leq 5.12 + 3T$, então, $T \geq 20.46$ microssegundos. Este resultado significa que o atraso de atendimento T, dos comutadores de nosso experimento está 14.54 microssegundos abaixo do limite inferior da faixa de 35 a 100 microssegundos obtida em [Jarschel et al. 2011]. Tal comportamento pode ser justificado por diversos fatores, dentre eles, as melhorias com relação ao desempenho na implementação do *Openvswitch*, desde a data em que foi feito o trabalho referenciado até hoje.

O principal objetivo da elaboração deste experimento é aplicar o modelo analítico-determinístico proposto na seção 3, sobre a rede virtual SDN elaborada. Para isso faremos uso do módulo desenvolvido no controlador SDN, explicado anteriormente. Inicialmente foi diminuída a capacidade dos enlaces do comutador 4 para $10Mbits/seg$. Esta modificação permite comprovar se o Algoritmo 1 escolhe a melhor rota fim-a-fim. As capacidades dos enlaces foram distribuídas de acordo ao grau de priorização, sendo que, $C1 = 50\%, C2 = 30\%, C3 = 20\%$. Foram cadastrados no controlador 3 fluxos com perfis de tráfego diferentes como se mostra na Tabela 1. Logo depois, 3 transmissões simultâneas TCP de 100MBytes foram feitas com destino $h4$.

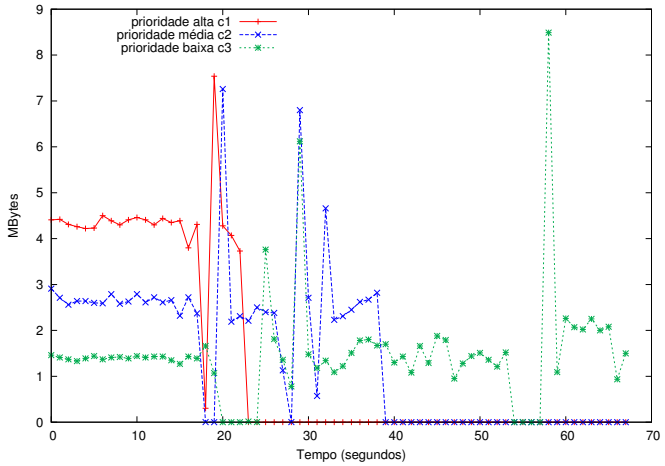


Figura 7. Transmissão simultânea de 3 fluxos com diferentes graus de priorização

No resultado destas transmissões, observou-se na Figura 7, como são atendidos os fluxos de acordo com o grau de priorização de cada um. O fluxo com prioridade alta manteve uma taxa média aproximada de 34 Mbps e finalizou em 23 s. Por sua vez o fluxo com prioridade média foi atendido de forma a manter uma taxa média de transmissão de 20.4 Mbps. Este fluxo transferiu os 100MBytes em 39 s.

Por último o fluxo de menor prioridade finalizou a transmissão em 68.4 s com uma taxa média aproximada de 11.68 Mbps. Evidentemente o Algoritmo 1, teve sucesso na procura da rota com melhor QoS em termos de atraso fim-a-fim e foi garantida a

priorização dos fluxos.

6. Considerações finais

Neste trabalho foi criado um modelo analítico-determinístico baseado na teoria de *Network Calculus*, permitindo implementar mecanismos de QoS em uma rede definida por software. Visando validar este modelo, foi desenvolvido um módulo acoplado a um controlador de rede que mostrou a viabilidade de utilização deste tipo de abordagem em um ambiente onde é feito um controle centralizado da rede.

O modelo elaborado faz uso das vantagens de programabilidade da rede que oferecem as SDNs. Considera-se que os resultados obtidos nesta pesquisa podem contribuir no aprimoramento dos mecanismos de QoS existentes até hoje, assim como as teorias desenvolvidas para a engenharia de tráfego. O gerenciamento de redes pode ser beneficiado também com o modelo analítico elaborado, pois este não precisa do monitoramento contínuo sobre a infraestrutura de rede para obter métricas de QoS.

Em estudos futuros pretende-se ampliar o modelo proposto de forma a incorporar outras métricas de QoS. Além disso será realizado um aprofundamento na teoria de Netcal, visando refletir modelos de redes mais complexos, onde sejam levados em consideração aspectos fundamentais como a capacidade das filas nas portas dos comutadores, assim como variabilidade do tamanho dos pacotes de um fluxo com igual grau de priorização.

Referências

Blake, S., Microsystems, S., and Wang, Z. (1998). An Architecture for Differentiated Services. *RFC*, pages 1–37.

Boyer, M. (2010). NC-maude: A rewriting tool to play with network calculus. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6415 LNCS, pages 137–151.

Braden, R., Clark, D., and Shenker, S. . (1994). Integrated Services in the Internet Architecture: an Overview. *IETF RFC 1633, July*, pages 1–28.

Cisco (2014). The Zettabyte Era: Trends and Analysis, White Paper. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html`. Acessado em 31 de Março de 2015.

Cruz, R. (1991a). A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131.

Cruz, R. (1991b). A calculus for network delay. II. Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271.

Egilmez, H. and Dane, S. (2012). OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. *Signal & Information ...*, pages 1–8.

Egilmez, H. E., Gorkemli, B., Tekalp, A. M., and Civanlar, S. (2011). Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. *2011 18th IEEE International Conference on Image Processing*, pages 2241–2244.

Hattingh, Christina and Szigeti, T. (2004). End-to-end QoS network design: quality of service in LANs, WANs, and VPNs.

Hubert, B. and Others (2002). Linux advanced routing & traffic control HOWTO.

Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., and Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. *2011 23rd International Teletraffic Congress (ITC)*, pages 1–7.

J.Moy (1998). OSPF Version 2. *IETF RFC 2328, April*, pages 1–245.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. . . . *Workshop on Hot Topics in Networks*, pages 1–6.

Mckeown, N., Anderson, T., Balakrishnan, H., Parulkar, G. M., Peterson, L. L., Rexford, J., Shenker, S., Turner, J. S., and Louis, S. (2008). OpenFlow: enabling innovation in campus networks. *Computer Communication Review*, 38:69–74.

Ni, L. (1999). Internet QoS: a big picture. *IEEE Network*, 13(2):8–18.

ONF (2012). Software-Defined Networking : The New Norm for Networks. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`. Acessado em 31 de Março de 2015.

Owens II, H. and Durresi, A. (2013). Video over Software-Defined Networking (VSDN). In *2013 16th International Conference on Network-Based Information Systems*, pages 44–51. IEEE.

Pfaff, B. and Davie, B. (2013). The Open vSwitch Database Management Protocol. Technical report, RFC 7047, IETF, December.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. (2009). Extending Networking into the Virtualization Layer. In *8th ACM Workshop on Hot Topics inNetworks*, volume VIII, page 6.

Qin, Z., Denker, G., Giannelli, C., Bellavista, P., and Venkatasubramanian, N. (2014). A Software Defined Networking architecture for the Internet-of-Things. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9. IEEE.

Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol label switching architecture. *Vasa*, pages 1–62.

Shah, S. A., Faiz, J., Farooq, M., Shafi, A., and Mehdi, S. A. (2013). An architectural evaluation of SDN controllers. In *2013 IEEE International Conference on Communications (ICC)*, pages 3504–3508. IEEE.

Shenker, S., Partridge, C., and Guerin, R. (1997). RFC-2212: Specification of Guaranteed Quality of Service.

Smit, H. and Li, T. (2004). Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE). *RFC 3784*.

Wroclawski, J. (1997). RCF-2211: Specification of the Controlled-Load Network Element Service.