10.18605/2175-7275/cereus.v13n4p133-148

ARTIGO ORIGINAL

<< Recebido em: 003/10/2021 Aceito em: 09/11/2021. >>

REVISTA CEREUS ISSN: 2175-7275

Um sistema de Engenharia de Tráfego utilizando Algoritmos Genéticos e Colônias de Formigas

A Traffic Engineering System using Genetic Algorithms and Ant Colonies

Nilton Alves Maia[1], João Paulo Versiani Ladeia[2], Sônia Beatriz de Oliveira e Silva Maia[3], Marilee Patta[4], Renê Rodrigues Veloso[5]

RESUMO

O paradigma de Redes Definidas por Software (SDN, *Software Defined Networks*) se baseia na separação física das funções de controle de encaminhamento de quadros. A idéia chave da separação a programação das funções de controle enquanto o *hardware* especializado para comutar quadros a alta velocidade permanece inalterado. Uma possível aplicação desse paradigma é na Engenharia de Trafego (TE, *Traffic Engineering*). A TE visa otimizar o desempenho de uma rede, analisando, prevendo e regulando o comportamento dos dados transmitidos. Este trabalho tem como objetivo desenvolver um Sistema de TE em SDN alimentado com fontes de tráfego de dados, voz e vídeo. Tal sistema deverá atender às necessidades de uma rede com as características de um Sistema Autônomo da Internet. A obtenção dos caminhos para o encaminhamento do tráfego das aplicações foi realizada com a utilização do algoritmo de Dijkstra e das heurísticas ACO (Ant Colony Optimization) e AG (Algoritmo Genético). Foi utilizado o emulador Mininet para criar as topologias SDN. O uso das soluções obtidas pelo ACO+AG proporcionou um melhor balanceamento de carga e ocupação homogênea dos enlaces. Além disso, os valores das vazões de tráfego não ultrapassaram os limites de largura de banda dos enlaces. As soluções obtidas também atendem ao atraso fim-a-fim máximo exigido pelas aplicações.

Palavras-chave: Engenharia de Tráfego; SDN; Otimização; Algoritmos de Colônia de Formigas; Algoritmos Genéticos.

ABSTRACT

The Software Defined Networks (SDN) paradigm is based on the physical separation of frame forwarding control functions. The key idea of separating the programming from the control functions while the specialized hardware for switching frames at high speed remains unchanged. A possible application of this paradigm is in Traffic Engineering (TE). TE aims to optimize the performance of a network, analyzing, predicting and regulating the behavior of transmitted data. This work aims to develop a TE System in SDN fed with data, voice and video traffic sources. Such a system must meet the needs of a network with the characteristics of an Autonomous Internet System. Obtaining the paths for forwarding the traffic of the applications was performed using the Dijkstra algorithm and the heuristics ACO (Ant Colony Optimization) and AG (Genetic Algorithm). The Mininet emulator was used to create the SDN topologies. The use of the solutions obtained by ACO+AG provided a better load balancing and homogeneous occupation of the links. Furthermore, the values of the traffic flows did not exceed the limits of bandwidth of the links. The solutions obtained also meet the maximum end-to-end delay required by the applications.

Keywords: Traffic Engineering; SDN; Optimization; Ant Colony Algorithms; Genetic Algorithms.

[1] Doutor em Engenharia Elétrica pelo PPGEE/ UFMG.

Universidade Estadual de Montes Claros - Unimontes.

https://orcid.org/0000-0002-2023-2453

E-mail:

nilton.maia@unimontes.br

[2] Mestre em Modelagem Computacional e Sistemas pela Universidade Estadual de Montes Claros - Unimontes.

https://orcid.org/0000-0002-8544-1213

[3] Mestre em Administração pela Faculdade de Estudos Administrativos de Minas Gerais - FEAD.

Universidade Estadual de Montes Claros - Unimontes.

https://orcid.org/0000-0002-4949-2451

[4] Doutora em Geografia pela PUCMG.

Universidade Estadual de Montes Claros - Unimontes.

https://orcid.org/0000-0001-9286-3329

[5] Doutor em Ciências da Computação pela UFMG.

Universidade Estadual de Montes Claros - Unimontes.

https://orcid.org/0000-0001-8989-0529

1. INTRODUÇÃO

A tecnologia de redes de computadores permeia atualmente todos os níveis da sociedade. Grande parte da sociedade depende hoje da Internet em suas atividades do dia-a-dia e, portanto, a estabilidade da rede se tornou uma característica essencial. Isso significa que pesquisas com novos protocolos e tecnologias não são mais possíveis na Internet em geral, devido ao risco de interrupção das atividades que dela dependem. Além disso, a larga adoção das tecnologias já desenvolvidas inviabiliza a inserção de novas tecnologias que dependam de alterações do hardware a ser utilizado (GUEDES et al., 2012).

Os dispositivos de encaminhamento das redes atuais são responsáveis por duas importantes tarefas: a execução do Planos de Dados e Controle. O Plano de Dados faz o encaminhamento dos pacotes entre as interfaces do equipamento de rede. No caso do Switch, o Plano de Dados verifica através do MAC de destino e da tabela de comutação, em qual interface o quadro será encaminhado. No caso do roteador, o procedimento é semelhante, a decisão sobre para qual interface o pacote será encaminhado dependerá do IP de destino e da consulta à tabela de roteamento. O Plano de Controle é responsável pelos protocolos utilizados para preencher as tabelas de encaminhamento dos elementos no Plano de Dados. Além disso, os Planos de Controle dos diferentes dispositivos comutadores podem se comunicar para definir os conteúdos das tabelas de encaminhamento. Essa forma verticalizada de organização atingiu um amadurecimento tal que tornaram as redes atuais pouco flexíveis. Para tentar contornar esse problema, a comunidade de pesquisa em Redes de Computadores tem investido em iniciativas que levem à implantação de redes com maiores recursos de programação, de forma que novas tecnologias possam ser inseridas de forma gradual. Uma forma de abordar a situação consiste em estender o hardware de encaminhamento de pacotes. A ideia é manter essa operação pouco alterada, mas com uma possibilidade de maior controle por parte do administrador da rede (GUEDES et al., 2012). Uma ideia possível, como a proposta em MAIA (2006), é a inclusão de um dispositivo externo que possibilite o controle do encaminhamento, ou seja, determine como os fluxos entrantes em um domínio MPLS (Multi-protocol Label Switching) sejam rotulados e encaminhados, dependendo de suas necessidades de Qualidade de Serviço (QoS, *Quality of Service*). Neste caso, o Plano de Dados e o Plano de Controle serão executados em dispositivos diferentes. Uma das iniciativas mais bem sucedidas nesse sentido foi a definição da

interface e do protocolo OpenFlow (MCKEOWN et al., 2008) .

O conceito de Redes Definidas por Software (SDN, *Software Defined Networks*) nasceu com a criação do protocolo Open-Flow para a programação de comutadores. A arquitetura SDN inclui três camadas: aplicações, dispositivos de controle e comutadores responsáveis pelo encaminhamento de pacotes. O controle da rede, que é executado por um *software* de propósito geral denominado controlador, que pode inspecionar, definir e alterar as entradas da tabela de roteamento dos comutadores. O controlador (ou controladores, no caso de um grande LAN ou WAN) se comunica com os aplicativos da camada de aplicações ou com os comutadores através de interfaces de programação. As interfaces de programação utilizadas para comunicação entre o controlador e os comutadores são denominadas APIs abertas de South-bound. Por outro lado, as APIs abertas de North-bound são as interfaces de programação disponibilizadas para possibilitar a comunicação entre os aplicativos da camada de aplicação e a camada de controle. Esta ultima interface permite o desenvolvimento de aplicativos que ofereçam serviços como segurança, controle de acesso, Gerenciamento da largura de banda, QoS, roteamento e Engenharia de Tráfego (TE - Traffic Engineering).

A TE é a utilização de princípios tecnológicos e científicos para a medição, caracterização, modelagem e controle do tráfego com o objetivo de avaliação e otimização do desempenho das redes IP (AWDUCHE *et al*., 2000). A melhora na QoS prestada pela rede devido a aplicação da TE pode ser observada através de parâmetros de tráfego tais como atraso fim-a-fim, variação do atraso ou perda de pacotes, bem como pela percepção humana, como por exemplo, em aplicações multimídia. As atividades básicas da TE são o controle e a otimização do roteamento com o objetivo de distribuir o tráfego uniformemente pela rede.

Nos problemas de otimização tem-se uma função objetivo que é dependente das variáveis de decisão e essas variáveis são delimitados pelas restrições a elas impostas. Quanto a sua formulação, o problema pode ser de minimização ou de maximização da função objetivo. As soluções podem ser soluções exatas ou aproximadas. Como exemplo de solução exata pode-se citar o Algoritmo de Dijkstra (1959). Ele é um algoritmo para encontrar os caminhos mais curtos entre dois nós em um grafo. Em problemas onde as soluções exatas são de difícil resolução e/ou com alto custo computacional, utilizam-se as heurísticas que são métodos que apresentam soluções aproximadas. As heurísticas não garantem soluções ótimas, mas em geral soluções próximas das ótimas em um tempo

computacional relativamente mais rápido. Como exemplos de heurísticas tem-se o Algoritmo da Colônia de Formigas (ACO - Ant Colony Optimization) e o Algoritmo Genético (AG). O ACO foi desenvolvido por Marco Dorigo, Vitorio Maniezzo e Alberto Colorni (COLORNI et al, 1992). É uma heurística baseada em população, tendo sido inspirada pelo comportamento das formigas na natureza. O AG foi desenvolvido por Holland (1975). È uma heurística inspirada no processo de seleção natural, comumente usados para gerar soluções de alta qualidade para problemas de otimização e busca, confiando em operadores bio-inspirados, como mutação, crossover e seleção.

Dentro deste contexto, o objetivo deste trabalho é desenvolver um sistema de Engenharia de Tráfego em SDNs. É proposto um modelo de otimização utilizando as heurísticas para solução do problema de TE em uma topologia SDN alimentada com fontes mistas (dados, voz e vídeo). A obtenção dos caminhos para o encaminhamento do tráfego das aplicações através da topologia SDN é realizada com a utilização do algorítmo de Dijkstra e das heurísticas Colônia de Formigas e Algoritmos Genéticos.

Este documento está organizado da seguinte forma: a seção 2 trata dos materiais e métodos. Na seção 3, são apresentados os resultados obtidos com a realização do trabalho. Na seção 4, são discutidos resultados obtidos. Na seção 5 são apresentadas as considerações finais. Finalmente são apresentadas as referências.

2. MATERIAIS E MÉTODOS

A arquitetura do Sistema de TE proposto é formada pelos módulos responsáveis pelo armazenamento de informações sobre a topologia, capacidades dos enlaces e matriz de tráfego, além do Gerenciador de TE, Controlador SDN e pela topologia do domínio SDN. O gerenciador de TE é responsável pela descoberta dos caminhos para o encaminhamento do tráfego das aplicações entrantes através do domínio SDN. A descoberta dos enlaces mais adequados para escoamento do tráfego das aplicações é feita levando em conta as necessidades das aplicações em termos de vazão e atraso fim-a-fim, atrasos estimados e larguras de banda nos enlaces do domínio SDN. A Figura 1 mostra a arquitetura do sistema de TE.

O Gerenciador de TE é composto do algoritmo de geração de rotas (ACO) e do algorítmo seleção de rotas (AG). Os dois algoritmos trabalham de forma seqüencial. O ACO calcula um conjunto de rotas distintas que podem ou não ser selecionadas como soluções. Em seguida, o AG utiliza o conjunto de rotas encontradas pelo ACO e seleciona

os caminhos que melhor atendem os requisitos das aplicações em termos de vazão, atraso fim-a-fim, prioridade e quantidade de conexões estabelecidas.

O Controlador SDN, utilizando o conjunto de rotas definidas pelo Gerenciador de TE, realiza a implementação da programação, ou seja, atualiza as tabelas de fluxos dos comutadores (swiches) que compõem a topologia SDN.

A topologia SDN, que pode ser observada na Figura 1, é formada por trinta e seis (36) hosts, dezesseis (16) comutadores (switches) e um controlador. Os hosts e os comutadores são interligados através de sessenta e seis (66) enlaces bidirecionais. Os enlaces internos da topologia SDN foram configurados com largura de banda 2 Mbps e atrasos mínimos de 10, 20, 30 e 40 ms. Os enlaces periféricos, responsáveis pela interligação dos nós origens das aplicações à topologia SDN e desta aos nós destinos, foram configurados com 2 Mbps e atrasos mínimos de 1 ms. Durante as simulações foram utilizadas aplicações de voz, dados e vídeo. As características e necessidades das aplicações em termos de vazão mínima e atraso máximo fim-a-fim são apresentadas na Tabela 1.

O encaminhamento do tráfego das aplicações através da topologia SDN foi calculado pelo Gerenciador de TE de duas formas. Na primeira foi utilizando o algoritmo de Dijkstra que calcula as rotas sempre considerando o menor caminho. Nessa forma de encaminhamento não é realizada a TE. Na segunda forma utilizou-se os algoritmos de otimização ACO e AG.

O ambiente de emulação da topologia SDN foi implantado em uma máquina física utilizando o emulador Mininet sob a plataforma Linux. O programa para implementação da topologia SDN foi desenvolvido utilizando a linguagem Python. O controlador Floodlight foi utilizado para realizar o controle da rede através do módulo Static Entry Pusher para alterar as tabelas de roteamento dos seus dispositivos. A geração e a medição do tráfego fim-a-fim na rede foram realizadas com a utilização do software D-ITG.

O Controlador Floodlight utiliza o protocolo OpenFlow para organizar fluxos de tráfego em um ambiente SDN. Ele é responsável por manter todas as regras de funcionamento da rede e fornecer as instruções necessárias sobre como o tráfego deve ser tratado. O Floodlight possui uma interface GUI baseada na web, concebida por meio da API REST do controlador. A GUI possibilita a exibição de informações do estado do controlador, dispositivos de encaminhamento conectados, enlaces entre dispositivos,

hosts, fluxos inseridos nas tabelas de encaminhamento dos dispositivos de encaminhamento e a topologia geral de rede (AKCAY e YILTAS-KAPLAN, 2017).

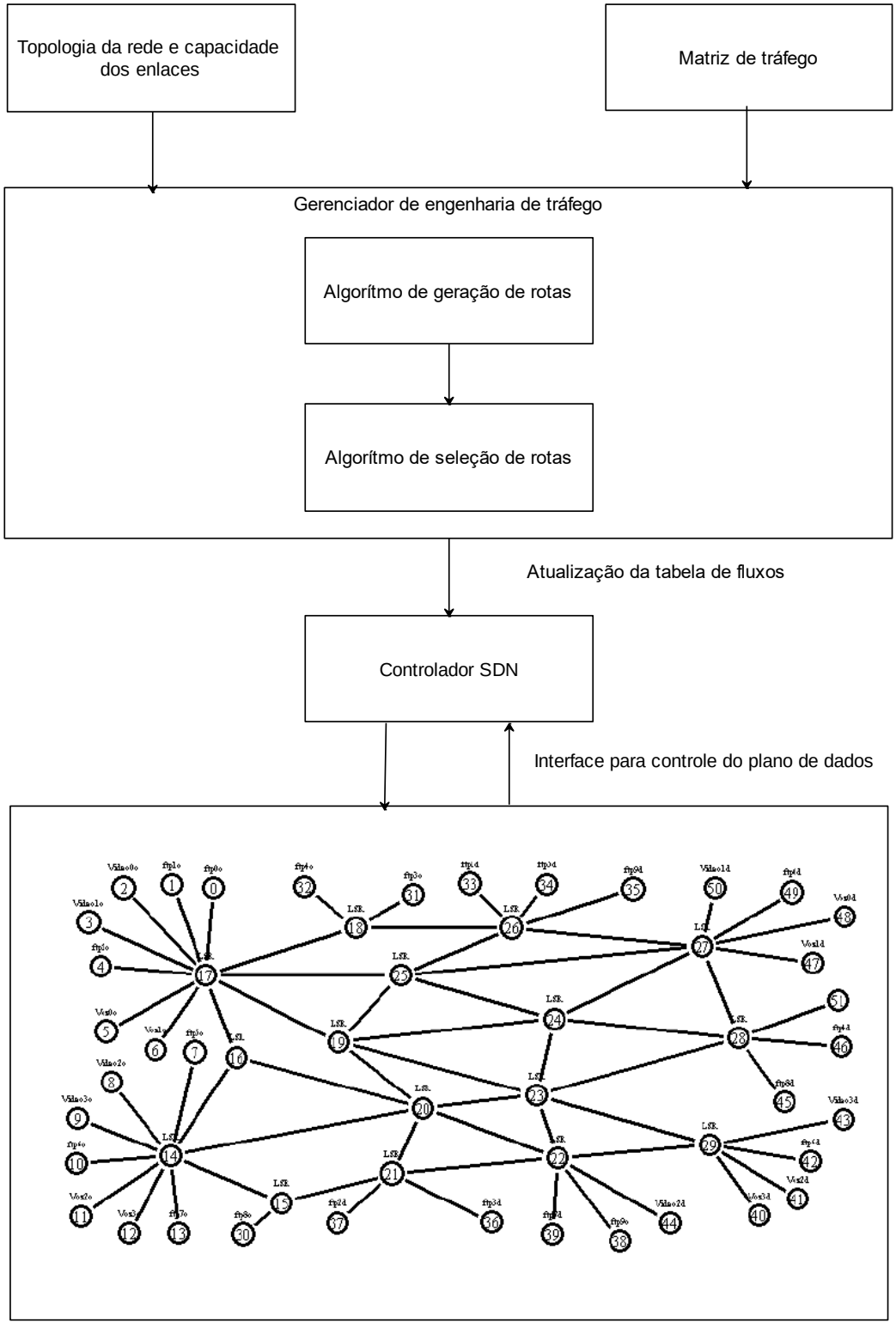

Figura 1. Arquitetura do sistema de Engenharia de Tráfego.

Fonte: Própria

Tabela 1 – Características e necessidades das aplicações de dados, voz e vídeo

APLICAÇÃO	ORIGEM	DESTINO	VAZÃO (MBPS)	ATRASO (MS)	PRIORIDADE
DADOS (FTP0)	52	49	0,256	< 1000	0
DADOS (FTP1)	1	33	0,256	< 1000	0
VÍDEO0	2	51	1,744	< 150	0
VÍDEO1	3	50	1,256	< 150	5
DADOS (FTP2)	4	37	0,256	< 1000	0
DADOS (FTP3)	31	36	0,256	< 1000	0
DADOS (FTP4)	32	46	0,256	< 1000	0
OZ0	5	48	0,512	< 150	5
VOZ1	6	47	0,512	< 150	5
DADOS (FTP5)	7	34	0,256	< 1000	0
VÍDEO2	8	44	1,256	< 150	5
VÍDEO3	9	43	1,256	< 150	0
VOZ2	11	41	0,512	< 150	5
DADOS (FTP6)	10	42	0,256	< 1000	0
VOZ3	12	40	0,512	< 150	5
DADOS (FTP7)	13	39	0,256	< 1000	0
DADOS (FTP8)	30	45	0,256	< 1000	0

A API REST é uma aplicação do controlador Floodlight baseada na arquitetura REST (Representational State Transfer) que utiliza de forma padronizada os recursos e características do protocolo HTTP. Através dessa API pode-se interagir com o controlador, consultando informações e alterando o estado da topologia SDN. O Static Entry Pusher é um módulo do controlador Floodlight, que utilizando a API REST, permite que um usuário insira manualmente fluxos e grupos de fluxos em uma rede que utiliza o protocolo Openflow. O D-ITG (Distributed Internet Traffic Generator) é um software para geração de tráfego no nível de pacote que replica diferentes padrões de tráfego. Os programas que implementam o AG, ACO e Dijkstra, utilizados no trabalho, foram desenvolvidos com a utilização de *Toolboxes* do MatLab. O MatLab integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos.

3. RESULTADOS E DISCUSSÃO

Nesta seção são apresentados os resultados da implementação do sistema de TE.

3.1 Resultados estimados do atraso e vazão considerando as rotas selecionadas pelos algorítmos ACO + AG e Dijkstra

São apresentados, nesta subseção os resultados estimados sobre o uso das rotas encontradas pelos algoritmos ACO+AG e Dijkstra como solução para o encaminhamento do tráfego das aplicações entrantes através dos enlaces da topologia SDN.

3.1.1 Análise do atraso fim-a-fim estimado

A Tabela 2 mostra um conjunto de rotas (caminhos formados por uma seqüência de nós) definido pelos algoritmos ACO+AG como solução otimizada para o encaminhamento do tráfego das aplicações entrantes através dos enlaces da topologia SDN. Na mesma tabela são apresentados o atraso fim-a-fim previsto e o valor aceitável pela aplicação para a cada uma das rotas.

Tabela 2 – Conjunto de rotas definidas pelos algoritmos ACO+AG

N°	Rotas Selecionadas (Nós)										Qtd Enlaces	Atraso Previsto (ms)	Atraso Máx (ms)
1	52	17	18	26	27	49	-	-	-	-	6	32	1000
2	1	17	25	27	26	33	-	-	-	-	6	32	1000
3	2	17	25	24	28	51	-	-	-	-	6	52	150
4	3	17	16	20	19	23	24	27	50	-	9	122	150
5	4	17	18	26	27	28	23	22	21	37	10	172	1000
6	31	18	26	27	24	23	22	21	36	-	9	102	1000
7	32	18	17	19	23	28	46	-	-	-	7	82	1000
8	5	17	18	26	27	48	-	-	-	-	6	32	1000
9	6	17	19	23	24	27	47	-	-	-	7	92	150
10	7	14	20	19	17	18	26	34	-	-	8	102	1000
11	8	14	20	22	44	-	-	-	-	-	5	42	150
12	9	14	15	21	22	29	43	-	-	-	7	82	150
13	11	14	16	20	23	29	41	-	-	-	7	72	150
14	10	14	20	23	22	29	42	-	-	-	7	72	1000
15	12	14	15	21	20	23	29	40	-	-	8	102	1000
16	13	14	20	21	22	39	-	-	-	-	6	72	150
17	30	15	21	20	19	24	28	45	-	-	8	92	150

Conforme pode ser observado na Tabela 2, os valores de atraso se mantêm dentro da faixa permitida. O cálculo do atraso estimado é realizado pela soma dos atrasos associados a cada mudança de nó. Por exemplo, o atraso previsto para a rota 5 é

calculado através do somatório dos atrasos associados a comunicação entre os enlaces entre os nós 4, 17, 18, 26, 27, 28, 23, 22, 21 e 37 nesta ordem. Para esta rota, utilizando a Tabela 2, tem-se a seguinte soma de atrasos: 1 + 10 + 10 + 10 + 80 + 20 + 20 + 20 + 1 = 172 milissegundos (ms).

A Tabela 3 apresenta um conjunto de rotas definido pelo algoritmo de Dijkstra como solução para o encaminhamento do tráfego das aplicações entrantes através dos enlaces da topologia SDN. São apresentados também na tabela o atraso fim-a-fim previsto e o valor aceitável pela aplicação para a cada uma destas rotas. É possível verificar na Tabela 3 que os valores de atraso foram menores que a solução obtida pelos algoritmos ACO+AG. Isso se deve ao fato que o algoritmo de Dijkstra obtêm as rotas mais curtas entre dois nós e conseqüentemente o atraso-fim-a-fim estimado entre os enlaces também é menor.

Tabela 3 – Conjunto de rotas definidas pelo algoritmo Dijkstra

N°	Rotas Selecionadas (Nós)							Qtd Enlaces	Atraso Previsto (ms)	Atraso Máx (ms)
1	52	17	25	27	49	-	-	5	22	1000
2	1	17	18	26	33	-	-	5	22	1000
3	2	17	25	24	28	51	-	6	52	150
4	3	17	25	27	50	-	-	5	22	150
5	4	17	19	20	21	37	-	6	72	1000
6	31	18	17	19	20	21	36	7	82	1000
7	32	18	26	27	28	46	-	6	102	1000
8	5	17	25	27	48	-	-	5	22	1000
9	6	17	25	27	47	-	-	5	22	150
10	7	14	20	19	25	26	34	7	122	1000
11	8	14	15	21	22	44	-	6	72	150
12	9	14	15	21	22	29	43	7	82	150
13	11	14	15	21	22	29	41	7	82	150
14	10	14	15	21	22	29	42	7	82	1000
15	12	14	15	21	22	29	40	7	82	1000
16	13	14	15	21	22	39	-	6	72	150
17	30	15	21	22	23	28	45	7	82	150

3.1.2 Análise da vazão estimada

Na Tabela 4 é apresentada a relação de enlaces internos da topologia SDN e suas respectivas capacidades máximas de largura de banda, bem como o percentual de utilização dos enlaces proporcionado pela solução otimizada com os algoritmos ACO+AG e a solução com o algorítmo Dijkstra. É possível perceber que a utilização dos enlaces na topologia SDN otimizada é distribuída de forma mais homogênea e balanceada. A

capacidade máxima de largura de banda não é ultrapassada e apenas dois enlaces não são utilizados (19-25 e 25-26). Por outro lado, a solução obtida com a utilização do algoritmo de Dijkstra provoca congestionamento ao proporcionar que a vazão de tráfego ultrapasse a capacidade máxima de largura de banda em seis (6) enlaces (14-15, 15-21, 17-25, 21-22, 22-29 e 25-27). Além disso, um total de dez (10) enlaces (14-16, 16-17, 16-20, 19-23, 19-24, 20-22, 20-23, 23-24, 23-29 e 24-27.) não são utilizados.

Tabela 4– Percentual de utilização da Largura de Banda nos enlaces

N°	enlaces	Largura banda Max	Solução otimizada com ACO + AG			Solução utilizando Dijkstra		
			N° Conexões	Vazão Utilizada	Percentual de utilização (%)	N° Conexões	Vazão Utilizada	Percentual de utilização (%)
1	14-15	2	2	1,712	86%	6	3,936	197%
2	14-16	2	1	0,512	26%	0	0	0%
3	14-20	2	4	1,968	98%	1	0,256	13%
4	15-21	2	3	1,968	98%	7	4,192	210%
5	16-17	2	1	1,2	60%	0	0	0%
6	16-20	2	2	1,712	86%	0	0	0%
7	17-18	2	5	1,536	77%	2	0,512	26%
8	17-19	2	3	1,024	51%	2	0,512	26%
9	17-25	2	2	2	100%	5	4,224	211%
10	18-26	2	5	1,536	77%	2	0,512	26%
11	19-20	2	3	1,712	86%	3	0,768	38%
12	19-23	2	3	1,968	98%	0	0	0%
13	19-24	2	1	0,256	13%	0	0	0%
14	19-25	2	0	0	0%	1	0,256	13%
15	20-21	2	3	1,024	51%	2	0,512	26%
16	20-22	2	1	1,2	60%	0	0	0%
17	20-23	2	3	1,28	64%	0	0	0%
18	21-22	2	4	1,968	98%	7	4,192	210%
19	22-23	2	3	0,768	38%	1	0,256	13%
20	22-29	2	2	1,456	73%	4	2,48	124%
21	23-24	2	3	1,968	98%	0	0	0%
22	23-28	2	2	0,512	26%	1	0,256	13%
23	23-29	2	2	1,024	51%	0	0	0%
24	24-25	2	1	1,744	87%	1	1,744	87%
25	24-27	2	3	1,968	98%	0	0	0%
26	24-28	2	2	2	100%	1	1,744	87%
27	25-26	2	0	0	0%	1	0,256	13%
28	25-27	2	1	0,256	13%	4	2,48	124%
29	26-27	2	5	1,536	77%	1	0,256	13%
30	27-28	2	1	0,256	13%	1	0,256	13%

3.2 Resultados da QoS oferecida às aplicações entrantes na topologia SDN

São apresentados, nesta subseção, os resultados dos testes para avaliação da QoS oferecida às aplicações entrantes após a implementação na topologia SDN dos conjuntos de rotas encontrados pelos algoritmos ACO+AG e pelo Dijkstra. Optou-se pela realização de cinco (5) testes, tanto para a solução otimizada, quanto para a solução com Dijkstra. Os resultados médios da vazão, atraso fim-a-fim, jitter e perda de pacotes obtidos após a realização dos testes na topologia SDN, são apresentados na Figura 9.

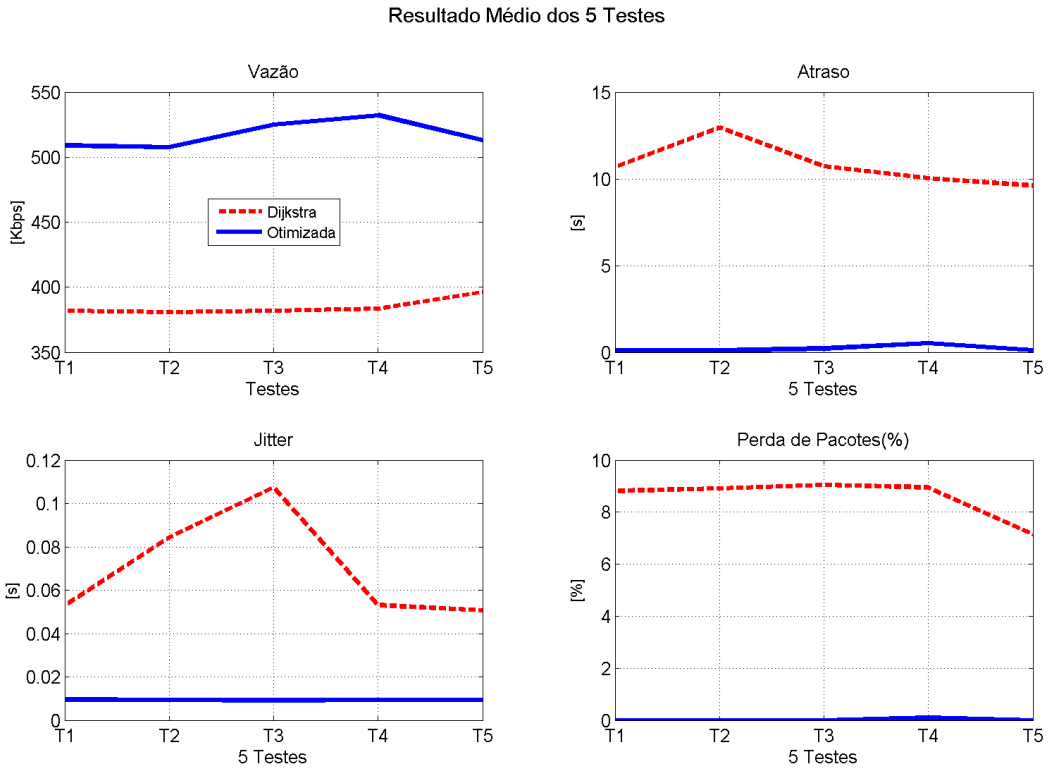

Figura 9 – Resultados médios da vazão, atraso fim-a-fim, jitter e perda de pacotes.

Os resultados apresentados na Figura 9 mostram que a solução otimizada apresentou resultados superiores. A vazão média obtida na solução otimizada é superior (acima de 500 kbps). Além disso, o atraso fim-a-fim médio na solução otimizada é inferior. O jitter médio na solução otimizada também é inferior (próximo de 0,01 segundo). Finalmente, a média de perda de pacotes na solução otimizada também é inferior (próximo de 0 %). Na Tabela 5 são apresentados os resultados médios gerais da comparação entre a solução otimizada com ACO + AG e a solução usando Dijkstra.

Observando-se a Tabela 5, pode-se verificar que a vazão média da solução otimizada é trinta e quatro por cento (34%) maior que o Dijkstra. O valor do atraso fim-a-fim médio da solução otimizada é aproximadamente cinqüenta e uma (51) vezes inferior. O valor do jitter médio da solução otimizada é aproximadamente sete vezes e meio (7,5) vezes inferior. Por fim, o valor médio da perda de pacotes da solução otimizada é aproximadamente quinhentos e dez (510) vezes inferior ao da rede usando Dijkstra.

Tabela 5 – Resultados médios gerais da solução otimizada e com o uso de Dijkstra.

PARÂMETROS DE QOS	SOLUÇÃO OTIMIZADA COM ACO + AG	SOLUÇÃO COM USO DO DIJKSTRA
Vazão Média (Kbit/s)	**517.3307**	**384.7603**
Atraso Médio (s)	**0.2109**	**10.7962**
Jitter Médio (s)	**0.0093**	**0.0697**
Perda de Pacotes (%)	**0.0168**	**8.5747**

Na Tabela 6 são apresentados os resultados de vazão e perda de pacotes após a aplicação das soluções otimizada e com Dijkstra. Algumas aplicações (2, 3, 5, 6, 7 e 10) na solução otimizada apresentam desempenho inferior, em termos de vazão. Para o restante das aplicações (1,4,8,9,11,12,13,14,15,16 e 17) a solução otimizada apresenta um desempenho superior. Em termos de perdas de pacotes, a solução otimizada apresenta um desempenho superior ao Dijkstra em 16 das 17 aplicações.

Tabela 6 – Resultados da vazão e perda de pacotes usando soluções otimizada e com Dijkstra.

N°	Aplicação	Origem	Destino	Vazão de referência	Vazão usando solução otimizada	Vazão usando solução com Dijkstra	Perda de Pacotes usando solução otimizada (%)	Perda de Pacotes usando solução com Dijkstra(%)
1	Dados	0	49	256	250,31	222,56	0,0000	0,00
2	Dados	1	33	256	250,37	251,86	0,0000	0,00
3	Vídeo	2	51	1744	1405,01	1440,20	0,0000	0,00
4	Vídeo	3	50	1256	1072,53	822,81	0,0000	0,00
5	Dados	4	37	256	250,33	251,77	0,0000	0,00
6	Dados	31	36	256	250,25	251,87	0,0000	0,00
7	Dados	32	46	256	250,18	251,83	0,0000	0,00
8	Voz	5	48	512	482,21	465,14	0,0000	3,38
9	Voz	6	47	512	482,73	401,26	0,0000	0,00
10	Dados	7	34	256	249,85	251,88	0,1054	0,00
11	Vídeo	8	44	1256	1066,53	688,52	0,0865	36,63
12	Vídeo	9	43	1256	1070,25	707,21	0,0000	34,44
13	Voz	11	41	512	482,45	306,19	0,0000	36,26
14	Dados	10	42	256	249,66	10,71	0,0000	0,07

15	Voz	12	40	512	482,15	21,31	0,0000	0,00
16	Dados	13	39	256	249,55	161,43	0,0935	34,97
17	Dados	30	45	256	250,28	34,38	0,0000	0,02
			Média	580,2353	517,3307	384,7603	0,0168	8,5747
			Desvio Padrão	465,8391	371,8967	349,3084	0,0364	15,0033

Na Tabela 7 são apresentados os resultados de atraso fim-a-fim e jitter para as aplicações entrantes na topologia SDN após a aplicação das soluções otimizada e com o Dijkstra. Pode-se notar que os valores de atraso fim-a-fim e jitter para a solução otimizada foram, em sua maioria, inferiores. Além disso, os valores de atraso fim-a-fim em quatorze (14) de um total de dezessete (17) aplicações da topologia SDN com solução otimizada foram inferiores aos valores máximos de referência. Por outro lado, os valores de atraso fim-a-fim da solução usando Dijkstra ultrapassam os valores máximos de referência para a maioria das aplicações.

Tabela 7– Resultados do atraso fim-a-fim e jitter na soluções otimizada e com Dijkstra.

N°	Aplicação	Origem	Destino	Atraso fim-a-fim de referência (s)	Atraso fim-a-fim usando solução otimizada (s)	Atraso fim-a-fim usando solução com Dijkstra (s)	Jitter usando solução otimizada (s)	Jitter usando solução com Dijkstra (s)
1	Dados	0	49	1	0,0539	5,4526	0,0154	0,0292
2	Dados	1	33	1	0,0537	0,0427	0,0154	0,0167
3	Vídeo	2	51	0,150	0,0546	0,0644	0,0012	0,0012
4	Vídeo	3	50	0,150	0,1354	7,1648	0,0049	0,0085
5	Dados	4	37	1	0,2057	0,1021	0,0205	0,0196
6	Dados	31	36	1	0,1354	0,1544	0,0209	0,0204
7	Dados	32	46	1	0,0881	0,1040	0,0031	0,0017
8	Voz	5	48	0,150	0,0345	3,5509	0,0019	0,0049
9	Voz	6	47	0,150	0,1128	5,7557	0,0107	0,0164
10	Dados	7	34	1	0,5306	0,1244	0,0038	0,0019
11	Vídeo	8	44	0,150	0,4724	4,4476	0,0021	0,0017
12	Vídeo	9	43	0,150	0,1534	4,4079	0,0023	0,0018
13	Voz	11	41	0,150	0,0747	4,3821	0,0019	0,0024
14	Dados	10	42	1	0,6078	61,7454	0,0206	0,6113
15	Voz	12	40	0,150	0,1876	49,9390	0,0107	0,2819
16	Dados	13	39	1	0,4927	4,2379	0,0032	0,0027
17	Dados	30	45	1	0,1911	31,8598	0,0203	0,1617
			Média	0,6	0,2109	10,7962	0,0093	0,0697
			Desvio Padrão	0,424264	0,1833355	18,05741	0,007619	0,153235

5. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi desenvolver um sistema de TE em Redes Definidas por Softwares. Foi proposto um modelo de otimização utilizando heurísticas para solução do problema de TE em uma topologia SDN alimentada com fontes de dados, voz e vídeo. A obtenção dos caminhos para o encaminhamento do tráfego das aplicações através da topologia SDN foi realizada com a utilização do algoritmo de Dijkstra e das heurísticas ACO e AG. Foi utilizado o emulador Mininet para criar as topologias SDN.

O uso das soluções obtidas pelo ACO+AG proporcionou um melhor balanceamento de carga e ocupação homogênea dos enlaces. Além disso, os valores das vazões de tráfego não ultrapassaram os limites de largura de banda em todos os enlaces. A análise do atraso fim-a-fim estimado indica também que as soluções obtidas pelo ACO+AG atendem ao atraso fim-a-fim máximo exigido pelas aplicações.

Os resultados obtidos, em termos dos requisitos de QoS vazão, atraso fim-a-fim, jitter e perda de pacotes mostraram que a topologia otimizada com soluções calculadas pelo ACO+AG, apresentou desempenho consideravelmente superior ao Dijkstra. A topologia otimizada obteve os seguintes resultados em relação ao Dijkstra: vazão média 34% maior, atraso fim-a-fim médio 51 vezes menor, jitter médio 7,5 vezes menor e perda de pacotes média 510 vezes menor.

A partir dos resultados obtidos pode-se observar que a implementação das heurísticas de otimização ACO e AG foi realizada de forma satisfatória e cumpriu adequadamente a tarefa de roteamento para o sistema de TE no domínio SDN. Como continuação pode-se sugerir os seguintes trabalhos futuros:

a) A implementação do sistema de TE com outras topologias de domínio SDN.
b) Realização de um estudo sobre o dimensionamento dos enlaces do domínio SDN.
c) Desenvolvimento de aplicativo para monitorar o tráfego nos enlaces do domínio SDN.

REFERÊNCIAS

AKCAY, H.; YILTAS-KAPLAN, D. Web-Based User Interface for the Floodlight SDN Controller. Int. **J. Advanced Networking and Applications**, 2017. Disponível em: http://oaji.net/articles/2017/2698-1493631900.pdf. Acesso em: 05 abr. 2021.

AWDUCHE, D. O.; CHIU, A.; ELWALID, A.; WIDJAJA, I.; XIAO, X. A Framework for Internet Traffic Engineering. **Internet-Draft**, 2000. Disponível em: http://www3.ietf.org/proceedings/01aug/I-D/draft-ietf-tewgframework-05.txt. Acesso em: 05 jun. 2021.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. An Investigation of some Properties of an" Ant Algorithm". **In: PPSN**. 1992. Disponível em: https://www.researchgate.net/publication/220701582_An_Investigation_of_some_Properties_of_an_Ant_Algorithm". Acesso em: 05 jun. 2021.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische mathematik**, 1959. Disponível em: https://link.springer.com/article/10.1007/BF01386390. Acesso em: 08 de Jun. 2021

GUEDES, Dorgival; VIEIRA, Luiz Filipe Menezes; VIEIRA, Marcos Menezes; RODRIGUES, Henrique; NUNES, Rogério Vinhal. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. XXX **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos,** SBRC 2012. Disponível em: https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/minicurso-sdn.pdf. Acesso em: 18 de Jun. 2021.

HOLLAND, J. Adaptation in natural and artificial systems: an introductory analysis with application to biology, Control and artificial intelligence. **MIT Press**, 1975.

KAUR, K; SINGH, J.; GHUMMAN, N. S. Mininet as software defined networking testing platform. I**n: International Conference on Communication, Computing & Systems (ICCCS**). 2014. Disponível em: https://www.researchgate.net/publication/287216738_Mininet_as_Software_Defined_Networking_Testing_Platform. Acesso em: 15 de Jun. 2021.

LANTZ, Bob; HELLER, Brandon; MCKEOWN, N.. A network in a laptop: rapid prototyping for software-defined networks. In: **Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks**, New York, 2010. Disponível em: https://dl.acm.org/doi/abs/10.1145/1868447.1868466. Acesso em: 10 de Jun. 2021.

MAIA, N. A. Engenharia de Tráfego em Domínio MPLS Utilizando Técnicas de Inteligência Computacional. Tese (Doutorado). **Universidade Federal de Minas Gerais**, 2006. Disponível em: < http://www.cpdee.ufmg.br/documentos/Defesas/654/TeseDoutorado-Nilton-TextoFinal.pdf>. Acesso em: 05 abr. 2021.

MCKEOWN, N.; ANDERSON, T.;Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.;Shenker, S.; TURNER, J. Openflow: enabling innovation in campus networks. ACM **SIGCOMM Comput. Commun Review,** 2008. Disponível em: https://dl.acm.org/doi/10.1145/1355734.1355746. Acesso em: 02/07/2020.

MININET. Mininet: an instant virtual network on your laptop (or other pc). Disponível em < http://mininet.org/ >. Acesso em: 02/08/2020.