Controle de congestionamento do TCP usando o algoritmo K-Means

Tiago da Silva dos Santos R.Vidal e Aline Gesualdi Manhães

***Resumo*—Este trabalho apresenta a elaboração de um classificador automático a fim de distinguir o motivo da ocorrência de perda de pacotes durante uma conexão TCP. A proposta é que este classificador seja capaz de distinguir situações de perda de pacotes, causada por congestionamento na rede ou por elevada taxa de erros no enlace. Utilizando o *software* Mininet-wifi, em conjunto com a técnica de *clustering* do algoritmo K-Means foi construído um classificador com aprendizado não supervisionado, capaz de modificar dinamicamente o algoritmo de controle de congestionamento adotado, obtendo resultados superiores aos algoritmos padrão comparados.**

Palavras-Chave— TCP, Controle de Congestionamento, K-Means, SDN.

***Abstract*— This work presents the elaboration of an automatic classifier capable of discern packet loss failure causes during a TCP connection. The proposal is that the classifier can be able to differentiate packet loss situations, being caused by network congestion or by increased error rate on a link. Using Mininet-wifi software, in-group with clustering technique of K-means algorithm, a classifier with unsupervised learning, it can dynamically modify the adopted congestion control algorithm, obtaining superior results in comparison with standard used algorithms.**

Keywords—TCP, Congestion Control, K-Means, SDN.

I. INTRODUÇÃO

Para fornecer o serviço de transferência confiável de dados, o protocolo TCP possui um mecanismo de controle de congestionamento que regula a quantidade de dados a serem transmitidos, de forma a ser compatível com a largura de banda disponível no meio de comunicação. Esta regulação é feita com o ajuste dinâmico da janela de congestionamento, a partir da detecção de eventos de perdas de pacotes no meio. O TCP reconhece estas perdas pela ocorrência de retransmissão por *timeout* ou pela recepção de mensagens duplicadas de reconhecimento (ACKs duplicados). Uma vez que o TCP não possui nenhum conhecimento sobre a causa da perda de pacotes, ele adota uma forma de ajuste na regulação de dados transmitidos, diminuindo a janela de transmissão, com o objetivo de evitar aumento de congestionamento na rede.

Entretanto, com o uso de diversas tecnologias de comunicação, o ambiente de redes interconectadas tornou-se muito mais heterogêneo, motivando a proposição de vários algoritmos de controle de congestionamento do TCP com propriedades distintas, de forma a superar desafios para se obter melhor desempenho. Um destes desafios está na dificuldade de predizer as características do meio por onde os dados estão trafegando.

Tiago da Silva dos Santos Robaina Vidal e Aline Gesualdi Manhães, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ), Rio de Janeiro-RJ, Brasil, E-mails: tiagots@ymail.com, aline.manhaes@cefet-rj.br.

Considerando o meio de transmissão de redes sem fio, há mais propensão de que as perdas no meio sejam causadas por erro de *bits* no enlace do que por congestionamento. Isto se deve a efeitos como interferências, *fading* e colisões que são fatores inerentes ao ambiente de redes sem fio com características difíceis de predizer com exatidão. Todavia, a redução do tamanho da janela de transmissão não tem efeito nestes casos, gerando apenas a constante redução do tráfego.

A. Sobre a proposta

Desde sua versão original, diversas propostas de algoritmos de controle de congestionamento para o TCP foram desenvolvidas. A proposta de solução aqui apresentada tem por objetivo desenvolver um mecanismo adaptativo para que o lado remetente das conexões TCP seja capaz de detectar as condições do enlace, a partir de dados coletados pelo próprio protocolo. Aplicando técnicas de Inteligência Computacional, propõe-se uma metodologia de classificação automática em grupos, de acordo com a percepção das condições de um enlace em redes sem fio, onde a cada grupo é atribuído um algoritmo distinto de controle de congestionamento do TCP.

Dentre as condições de enlace a serem discriminadas, foi proposta uma classificação em dois grupos. Um para refletir uma condição de perdas de pacotes por congestionamento na rede, outro para refletir perda de pacotes pela presença de elevada taxa de erros. Desta forma, há interesse em compatibilizar as condições do enlace com as características do algoritmo de controle adotado. Para estabelecer de forma automática os grupos, foi realizado um estudo de caso com o algoritmo de *clustering* K-Means, em que, mediante classificação obtida, altera-se dinamicamente o tipo de algoritmo do protocolo TCP utilizado, auxiliando no controle de congestionamento do TCP, com a finalidade de melhorar o desempenho deste importante protocolo.

O desenvolvimento do trabalho proposto está organizado da seguinte forma:

- Na seção II são apresentados os conceitos básicos do controle de congestionamento do TCP, do algoritmo de *clustering* K-Means, assim como o *software* de emulação Mininet que foram utilizados nos experimentos realizados neste trabalho.
- Na seção III, são apresentados os experimentos e os métodos utilizados em um classificador K-Means em redes sem fio emuladas através do *software* Mininet-wifi.
- Na seção IV são apresentadas os resultados obtidos com os experimentos realizados.
- Na seção V são apresentadas as conclusões, destacando a contribuição da metodologia utilizada.

II. CONCEITOS FUNDAMENTAIS

A. Controle de congestionamento TCP

Após a concepção original do TCP [1], diante das constantes mudanças nas tecnologias de rede e estudos buscando melhores resultados no controle de congestionamento do TCP, diversas versões subseqüentes foram propostas, com o foco em melhoria na eficiência dos recursos de rede pelos dispositivos. Estas melhorias propõem alterações no algoritmo do TCP para superar problemas e limitações no protocolo original para ambientes com requisitos específicos, dentre os quais podemos citar redes com fio, redes sem fio, redes de alta velocidade, redes com alto *delay*, dentre outros ambientes. Dentre as propostas elaboradas, se destacam as versões TCP Tahoe [2], o TCP Vegas [3], o TCP CUBIC [4] e o TCP Illinois [5].

B. Algoritmo K-Means

Apurando as técnicas mais utilizadas de aprendizado não-supervisionado e verificado o trabalho de Sooriyabandara et al [6], decidiu-se analisar mais detalhadamente o uso de técnicas de análise de perfis de agrupamento (ou *clustering*), que tem como propósito separar dados analisados em grupos, baseados nas características que estes dados possuem. Um dos métodos de agrupamento mais utilizados é o K-Means. Sua ampla utilização se deve à simplicidade, eficiência e resultados obtidos. Esse método de análise consiste de um agrupamento de dados que busca particionar automaticamente um conjunto de dados em *k* grupos, de maneira que a distância total entre os dados de um grupo e seu respectivo centro seja minimizada. O algoritmo inicia selecionando *k* centros de *cluster*, também chamados de centróides, um para cada *cluster*. De forma iterativa, o algoritmo analisa para qual *cluster* o próximo elemento analisado será designado, considerando a distância euclidiana aos centróides de cada grupo. O algoritmo converge quando não há mais alterações na definição de centróides para os *clusters*. O método K-Means apresenta bom desempenho mesmo em sua simplicidade, sendo então considerado para a aplicação neste trabalho.

C. Emulador Mininet-wifi

No estudo de redes de computadores que envolva análises de protocolos, testes de desempenho de redes e demais pesquisas relacionadas é comum a utilização da emulação de redes. O *software* Mininet [7] é um sistema de emulação de rede, que permite criar de forma virtualizada *hosts*, *switches* e *links* através de um sistema operacional Linux, com suporte a SDN (*Software-Defined Networks)*. Um benefício do Mininet é o fato de ser um programa de código aberto, com ampla colaboração no meio acadêmico. Há grande flexibilidade de programação, através do uso de *scripts* em *Python*, possibilitando a criação de diversos tipos de topologia com agilidade, sem a necessidade de implementação física de cabeamentos e *hardwares* de elementos de rede.

A emulação de rede em ambientes sem fio, diversos fatores presentes nos ambientes físicos reais, como a interferência eletromagnética, as características do canal utilizado, a taxa de transmissão e a mobilidade são condições importantes a serem consideradas e difíceis de serem caracterizadas por modelos programados por *software*. Com este desafio, Fontes et al. [8] elaboraram uma extensão do emulador Mininet, chamado de Mininet-wifi. Este *software* visa prover um ambiente de emulação de redes SDN, incluindo o uso de redes sem fio, permitindo adicionar ao Mininet a emulação de estações com mobilidade e pontos de acesso (AP – *Access Points*), sendo possível emular diferentes tipos de cenários de redes sem fio. Com este recurso, é possível controlar de forma centralizada o ambiente de redes sem fio, mantendo os recursos de redes SDN. Por conta destas características, seu uso está sendo bem aceito em estudos recentes e identificou-se um potencial para utilização neste trabalho.

III. DESCRIÇÃO DOS EXPERIMENTOS E METODOLOGIA

A proposta do trabalho consiste em adquirir dados de conexões TCP, de forma a construir uma base de dados para análise do algoritmo K-Means, formando automaticamente agrupamentos (*clusters*) específicos, capazes de discriminar cada tipo de situação de rede, a saber, congestionamento ou perda por erros de *bits*. A partir desta classificação, é possível definir uma ação no TCP, que neste trabalho compreendeu a adoção um tipo de algoritmo de controle de congestionamento do TCP para uso nas conexões subseqüentes. O intuito é ter um controle de congestionamento específico adaptado para cada situação de rede classificada, trazendo assim um ganho de desempenho que pode ser apurado nos resultados de *throughput* médio obtido em medições durante conexões TCP. Para realizar a validação da proposta e construir um cenário de testes, optou-se por utilizar o emulador de rede Mininet-wifi.

Nos experimentos realizados, foram utilizados os recursos de *hardware* e *software* descritos abaixo:

- Ambiente de *Hardware* - No desenvolvimento deste trabalho foi utilizado um microcomputador *notebook* Dell Inspiron N4050 com as seguintes configurações: Intel® Core™ i5-2450M CPU @ 2.50GHz × 4; 4 GB de memória RAM.

- Ambiente de *Software* - As simulações foram feitas sobre a plataforma operacional Linux Ubuntu 16.04 LTS 64 *bits*, além dos controladores e ferramentas: Mininet versão 1.9.r3; Iperf versão 2; TCPTrace versão 6.6.7; Octave versão 4.0.0.

Foi elaborado um cenário de testes dinâmico, de forma a se gerar dados para analisar alterações nas condições de rede. Para tanto, foi proposta uma transferência de dados entre um *host* e um servidor, onde o *host* possui mobilidade, em uma rede sem fio 802.11g, conforme mostrado na figura 1.

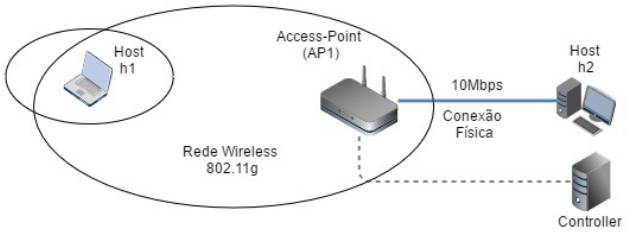

Fig. 1. Topologia Mininet-wifi – Rede emulada para testes

Para a transferência de dados, utilizou-se o *software* IPERF. O IPERF é uma ferramenta utilizada em análise de redes de computadores, pois permite a geração de tráfego e análise da qualidade de um enlace e principalmente a possibilidade de alterar o tipo de algoritmo de controle de congestionamento. Tendo como referência os trabalhos de Geurts, Khayat e Leduc [9], Lim e Jang [10] e Lei [11], foram definidos os dados de RTT (*Round-Trip Time*), Variação do RTT (RTT_dev) e número de mensagens de ACK duplicados (DupAck) para serem os discriminantes de condições de tráfego sujeito a

congestionamento ou à elevada taxa de erros. O parâmetro de *throughput* foi coletado como um verificador de desempenho, após as alterações dinâmicas do tipo de TCP a ser adotado. A coleta dos dados do TCP foi realizada pelo TCPDump e consolidado de forma estruturada pelo TCPTrace, a fim de coletar os dados de RTT médio, variação do RTT e ACKs duplicados em cada conexão TCP. Após ter uma amostra significativa destes dados, podemos utilizar o classificador K-Means para obter os *clusters* relacionados a cada tipo de condição de tráfego. O algoritmo K-Means foi configurado e implementado utilizando o *software* Octave. No algoritmo construído, coleta-se amostras de RTT, RTT_dev e DupAck, normalizam-se estes dados, para formar a base de análise do K-Means e obtenção dos *clusters*. O resumo de funcionamento deste experimento está demonstrado na figura 2.

Para a composição de escopo estudado, foi necessário definir determinados tipos de algoritmos a serem utilizados nos experimentos e serem comparados os devidos resultados. Sendo assim, definiu-se o uso do TCP Cubic, por ser o controle de congestionamento padrão do sistema operacional Linux, o TCP Vegas, dado o algoritmo ser proposto para melhor responder às situações de congestionamento da rede e o TCP Illinois, pela forma ágil de busca pelo uso de recursos, uma característica importante nas situações de elevada taxa de erros. Atendo-se ao conceito proposto neste trabalho, devem-se aplicar algoritmos de acordo com as situações classificadas como congestionamento e como sujeitas à elevada taxa de erros. Assim, propõem-se rotinas de testes com as associações de cada combinação entre os tipos de TCP definidos (Cubic, Vegas e Illinois). As opções de rotinas de testes a serem analisadas e comparadas foram designadas conforme a tabela I.

TABELA I. ROTINAS DE TESTES PROPOSTOS

		TCP Algoritmo B (Congestionamento)		
		Cubic	Vegas	Illinois
TCP Algoritmo A (Taxa de Erro)	Cubic	Rotina 1	Rotina 2	Rotina 3
	Vegas	Rotina 4	Rotina 5	Rotina 6
	Illinois	Rotina 7	Rotina 8	Rotina 9

Cada rotina de testes compreende a realização de séries de conexões TCP utilizando a combinação de algoritmos definida. A ideia consiste em obter os resultados de *throughput* obtidos em cada uma das rotinas e comparar estes resultados visando identificar os melhores algoritmos a serem aplicados ao cenário proposto.

Com o cenário de uso de Redes sem fio, é importante que sejam analisadas as conexões TCP com a adoção de mobilidade, de forma a variar as distâncias do *host* ao AP, estabelecendo assim uma maneira de serem observadas as condições de enlaces sujeitos a congestionamento (quando mais próximo do AP) ou a taxa de erros (quando mais distante do AP).

O *software* Mininet-wifi permite a configuração de modelos de mobilidade em rotas aleatórias. Entretanto durante a coleta de medições, foi verificado que o uso destes modelos poderia influenciar na comparação de medidas em cada série de testes, uma vez que o fator de distância para o AP influencia no desempenho obtido durante a conexão, prevalecendo sobre a influência do tipo de algoritmo de controle de congestionamento a ser utilizado. Assim, foi definido o uso de pontos previamente definidos para realizar as conexões, de forma a habilitar a comparação, nas mesmas condições de enlace, cada uma das rotinas testadas, reduzindo a influência de fatores externos. Considerando o alcance do AP, propôs-se uma rota linear com intervalos iguais por cada conexão, dispondo 250 pontos fixos por esta rota.

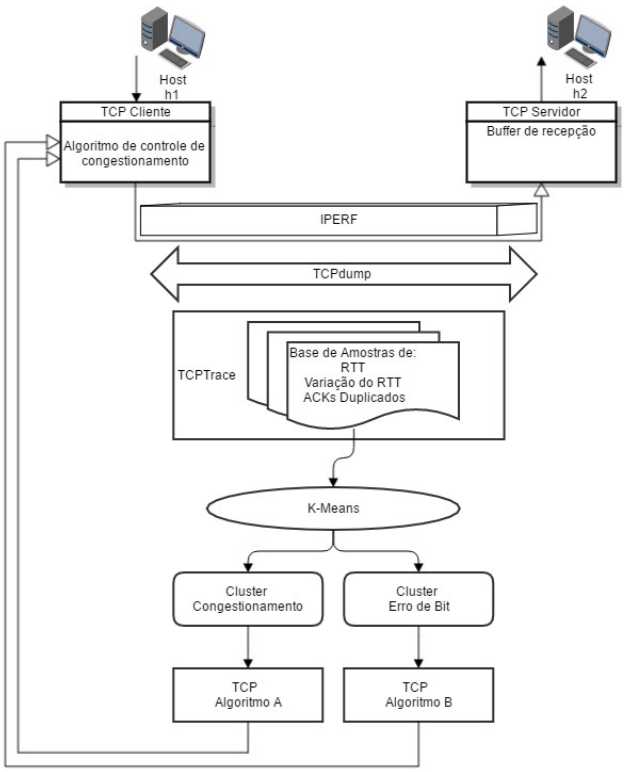

Fig. 2. Modelo do Algoritmo proposto com K-Means

O *software* Mininet-wifi possui recursos gráficos para visualizar a condição de mobilidade dos dispositivos. Na figura 3, demonstram-se os passos de movimentação do *host* h1, exemplificando pontos de transferência utilizados no experimento. Mesmo em pontos pré-definidos, as situações emuladas do ambiente sem fio possuem componentes aleatórios de perda devido ao canal que podem influenciar nos resultados. Para reduzir a influência destes fatores, propôs-se então a repetição de 10 séries com 250 conexões com mobilidade em cada rotina, onde cada conexão é realizada em um ponto previamente definido. Durante a execução das séries de testes, a cada uma das 250 conexões TCP são realizadas medições de amostras de dados que servirão de base para a execução do algoritmo.

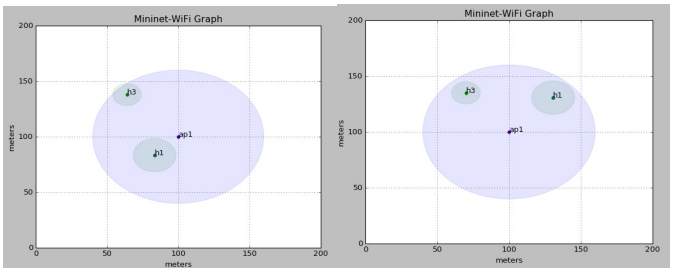

Fig. 3. Rota de mobilidade do *host* h1. Posições 55(esquerda) e 240(direita)

A partir de uma base inicial composta de 60 valores pode-se analisar de forma iterativa as conexões TCP com o algoritmo K-Means. A iteratividade de testes se justifica pela

necessidade de respostas adaptadas às mudanças nas condições do canal, com ênfase nos dados mais recentes. Estabeleceu-se uma métrica de se alterar continuamente a base de amostras analisada pelo K-Means, através do acréscimo de amostras mais recentes e descarte das amostras com mais tempo na base.

É importante que os *clusters* possam ser discriminados entre situação de congestionamento ou de taxa de erros. Para a identificação dos *clusters*, foi analisado o trabalho de Sooriyabandara et al [6], que identifica cada classe de *clusters* de acordo com os valores de RTT. Samaraweera [12] correlaciona o RTT com os dados de *throughput* obtido durante o fluxo de dados. Ele assume que uma perda de pacote acima de certo limite é usualmente correlacionada a congestionamento, devido ao uso de *buffers* de recepção. De forma oposta, se o valor de RTT for baixo, os recursos de rede estão subutilizados, indicando uma perda aleatória devido à taxa de erros. Assim, foi adotado o parâmetro de referência do valor de RTT para discriminar cada *cluster* em separado, analisando o posicionamento do centróide de cada *cluster*. O *cluster* com centróide de valor mais alto será referenciada ao *cluster* de situação de rede com congestionamento. O *cluster* com centróide de valor mais baixo está relacionado à situação de rede com taxa de erro. Um exemplo da distribuição gráfica dos *clusters* está na figura 4.

A partir desta parametrização, o algoritmo realiza uma lógica de testes específica. Com a base amostral preparada, K - Means é executado com os dados que integram esta base e os dois *clusters* são definidos. Daí então o algoritmo recebe as três últimas amostras coletadas e realiza testes para mensurar a proximidade de cada uma destas amostras ao centróide das duas classes, através de distância euclidiana. Se, destes valores medidos, a maioria das amostras estiver mais próximas do centróide de valor mais elevado de RTT, define-se que o atual momento da conexão estará mais sujeita à situação de congestionamento. De forma similar, caso a maioria destas três amostras esteja próxima do centróide de menor valor de RTT, classifica-se a condição de taxa de erro.

A partir desta parametrização, o algoritmo realiza uma lógica de testes específica. Com a base amostral preparada, o K-Means é executado com os dados que integram esta base e os dois *clusters* são definidos. Daí então o algoritmo recebe as três últimas amostras coletadas e realiza testes para mensurar a proximidade de cada uma destas amostras ao centróide das duas classes, através de distância euclidiana. Se, destes valores medidos, a maioria das amostras estiverem mais próximas do centróide de valor mais elevado de RTT, define-se que o atual momento da conexão estará mais sujeita à situação de congestionamento. Caso contrário, adota-se a condição de taxa de erro como prevalecente.

Este ajuste do tipo de TCP a ser adotado nas próximas conexões é realizado alterando-se o nome do arquivo que o *script* faz a leitura deste parâmetro no *software* IPERF. De maneira análoga, no funcionamento real de um sistema Unix, esta lógica poderia ser adotada, alterando o controle de congestionamento do TCP diretamente no Kernel Linux.

IV. RESULTADOS

A partir da metodologia descrita nos tópicos anteriores, foram realizados testes utilizando variações dos algoritmos de controle de congestionamento do TCP. Em cada conexão, além das amostras de dados para a composição dos *clusters*, foram coletados também os dados de *throughput* obtidos. Os dados de *throughput* servem para aferir o desempenho das conexões TCP. O valor mais elevado de *throughput* revela que o desempenho da conexão foi melhor, uma vez que mais dados foram transferidos em um menor tempo. Para exemplificar este cenário, pode-se observar a figura 5, que mostra o *throughput* durante cada uma das 250 conexões TCP em uma determinada série medida. O eixo horizontal representa a distância entre o *host* e o AP. O valor zero se refere ao posicionamento onde o *host* h1 se encontra no mesmo posicionamento do AP.

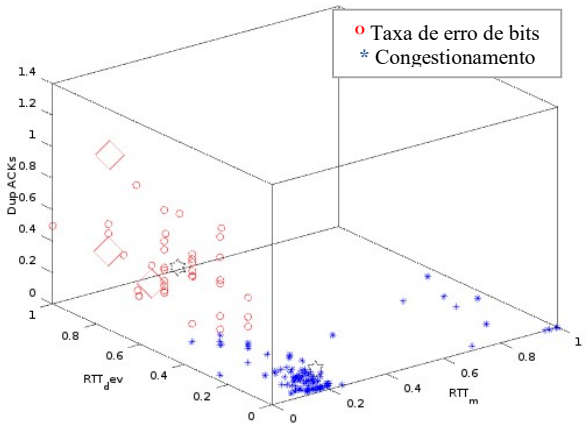

Fig. 4. Distribuição gráfica dos *clusters* na amostra 230

Quando o *host* está mais distante do AP, observa-se que o *throughput* é baixo. Isto se deve à baixa potência de recepção devido às perdas no meio, gerando elevada taxa de erros de *bits*. Esta taxa de erros de *bits* resulta em sucessivas perdas de pacotes, cabendo ao TCP identificar necessidades de retransmissão pela ocorrência de *timeout* e o tratamento a ser dado ao reduzir a janela de transmissão. São esperados então melhores resultados quando o algoritmo TCP adotado reduzir a janela de transmissão adequadamente.

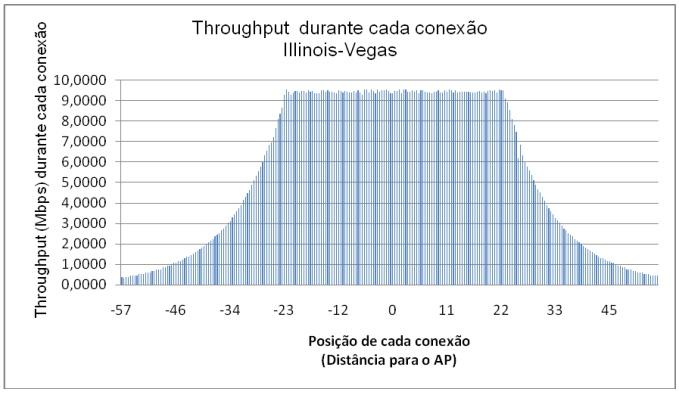

Fig. 5. *Throughput* obtido em uma série de conexões (Illinois-Vegas).

Quando o *host* está mais próximo do AP, podemos observar uma região de saturação, onde o tráfego está limitado à capacidade de banda da conexão entre o *host* h2 e o AP, definida em 10Mbps. A maior parte das perdas neste trecho estará relacionada ao congestionamento e são esperados melhores resultados com o uso de um algoritmo que consiga adaptar a esta condição, mantendo o sistema perto da plena utilização em ambientes menos suscetíveis a taxa de erros. Os valores de *throughput* médio foram coletados em cada série de conexões e assim obtidos os insumos para se calcular o *throughput* médio de cada rotina. Os dados de *throughput* obtidos em cada rotina de testes estão descritos na tabela II.

TABELA II. COMPARAÇÃO DOS VALORES DE *THROUGHPUT* MÉDIOS

	TCP A (Taxa de erro)	TCP B (Congestiona-mento)	*Throughput* Médio
Rotina 1	Cubic	Cubic	5,3386 Mbps
Rotina 2	Cubic	Vegas	5,2812 Mbps
Rotina 3	Cubic	Illinois	5,3389 Mbps
Rotina 4	Vegas	Cubic	5,2389 Mbps
Rotina 5	Vegas	Vegas	5,3691 Mbps
Rotina 6	Vegas	Illinois	5,3745 Mbps
Rotina 7	Illinois	Cubic	5,4794 Mbps
Rotina 8	Illinois	Vegas	5,7824 Mbps
Rotina 9	Illinois	Illinois	5,5757 Mbps

Verifica-se melhor desempenho quando utilizado o TCP Illinois, para situações classificadas como sujeitas a perdas por taxa de erros e o TCP Vegas para situações classificadas como congestionamento. O melhor desempenho utilizando o algoritmo Illinois, quando há classificação de taxa de erro, resulta principalmente da característica de crescimento da janela de congestionamento e pela baixa redução da janela em uma ocorrência de perda não relacionada a congestionamento.

Destaca-se o resultado comparativo de *throughput* obtido com o uso da combinação Illinois-Vegas, com o TCP Cubic, algoritmo utilizado por padrão no kernel Linux, assim como a comparação no uso somente com o TCP Illinois e com o TCP Vegas, demonstrado na figura 6. Estabeleceu-se um teste de médias utilizando o teste T de Student, onde se validou a hipótese que há diferença estatística entre as médias das demais amostras com a combinação Illinois-Vegas, em um intervalo de confiança de 5%.

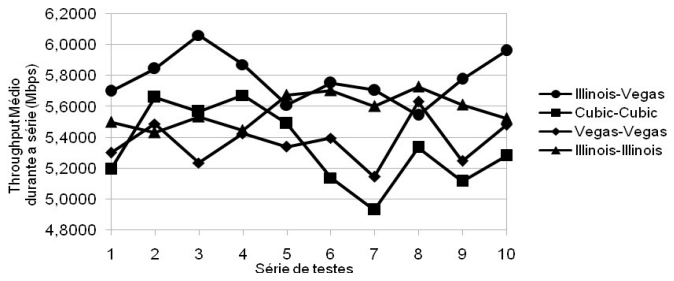

Fig. 6. *Throughput* Médio obtido em cada Série

V. CONCLUSÕES

Neste trabalho foi apresentada uma proposta de solução que tem por objetivo desenvolver um mecanismo adaptativo para detectar as condições do enlace a partir de dados coletados pelo próprio protocolo TCP, aplicando técnicas de Inteligência Computacional como ferramenta de auxílio ao controle de congestionamento do protocolo TCP. Foi proposto e implementado uma classificação automática em grupos, de acordo com a percepção das condições de um enlace em redes sem fio, onde a cada grupo foi atribuído um algoritmo distinto de controle de congestionamento do TCP. Especialmente aplicado a ambientes com mobilidade, onde deve ser considerada a ocorrência de erros na transmissão, que acaba se refletindo em perda de pacotes e assim impactando de forma significativa o desempenho percebido pelos usuários de dispositivos móveis, onde a partir do lado do usuário final é possível obter dados da rede e reagir adequadamente às variações ocorridas na rede.

Pelos experimentos realizados, verifica-se que o objetivo de melhorar o desempenho do TCP foi alcançado, uma vez que o modelo exposto obteve um ganho de *throughput* comparado às versões típicas do TCP, alcançando um ganho de cerca de 8% entre o uso dinâmico das versões Illinois e Vegas, em comparação com o TCP Cubic, algoritmo padrão em sistemas operacionais Linux. Este ganho de desempenho pode vir a melhorar a percepção de usuários e a velocidade de transferência de dados entre aplicações. Diante disso, é possível esperar que este algoritmo seja capaz de ser usado em grande escala em dispositivos móveis, como *smartphones* e *notebooks* em redes sem fio, fazendo desse trabalho relevante onde se propõe a fornecer melhoria de desempenho utilizando recursos de *software*, sem necessitar de grandes alterações na compilação de algoritmos TCP disponíveis nos dispositivos.

REFERÊNCIAS

[1] INFORMATION SCIENCES INSTITUTE. RFC:793: Transmission control protocol protocol specification. California (EUA): University Of Southern California, 1981. 90 p. Disponível em: < https://tools.ietf.org/html/rfc793>. Acesso em: 06 fev. 2015.

[2] JACOBSON, Van L. Congestion avoidance and control. ACM SIGCOMM Computer Communication Review: SIGCOMM, Nova York (EUA), v. 18, n. 4, p.314-329, ago. 1988.

[3] BRAKMO, Lawrence S.; PETERSON, Larry L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE: Journal On Selected Areas In Communications, [Estados Unidos], v. 13, n. 8, p.1465-1480, out. 1995. Disponível em: <http://www.cs.toronto.edu/syslab/courses/csc2209/06au/papers/vegas.pdf>. Acesso em: 11 fev. 2016.

[4] HA, Sangtae; RHEE, Injong; XU, Lisong. CUBIC: a new TCP-friendly high-speed TCP variant. ACM Sigops Operating Systems Review: SIGOPS, Nova York (EUA), v. 42, n. 5, p.64-74, jul. 2008. Disponível em: <http://netsrv.csc.ncsu.edu/export/cubic_a_new_tcp_2008.pdf>. Acesso em: 2 jan. 2016.

[5] LIU, Shao; BAşAR, Tamer; SRIKANT, R.. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. Performance Evaluation, University Of Illinois (EUA), v. 65, n. 6-7, p.417-440, jun. 2008. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166531607001307>. Acesso em: 2 fev. 2016.

[6] SOORIYABANDARA, Mahesh et al. Experiences with discriminating TCP loss using K-Means clustering. 2010 International Conference On Information And Communication Technology Convergence (ictc), Reino Unido, p.352-357, nov. 2010. Disponível em: <http://ieeexplore.ieee.org/abstract/document/5674698/authors>. Acesso em: 01 set. 2017.

[7] LANTZ, Bob; HELLER, Brandon; MCKEOWN, Nick. A network in a laptop: rapid prototyping for software-defined networks. Proceedings Of The Ninth Acm Sigcomm Workshop On Hot Topics In Networks - Hotnets '10, Nova York (eua), n. 19, p.19, out. 2010. Disponível em: <http://dl.acm.org/citation.cfm?id=1868466>. Acesso em: 23 jan. 2017.

[8] FONTES, Ramon R. et al. Mininet-WiFi: Emulating software-defined wireless networks. 2015 11th International Conference On Network And Service Management (cnsm), São Paulo, p. 384-389, nov. 2015.

[9] GEURTS, P.; KHAYAT, I. El; LEDUC, G. A Machine Learning Approach to Improve Congestion Control over Wireless Computer Networks. Fourth IEEE International Conference On Data Mining (icdm'04), [s.l.], 2004.

[10] LIM, Chang-hyeon; JANG, Ju-wook. An Adaptive End-to-End Loss Differentiation Scheme for TCP over Wired/Wireless Networks. IJCSNS International Journal Of Computer Science And Network Security, [s.l], v. 7, n. 3, p.72-83, mar. 2007.

[11] LEI, Niu. Applying the Linear Neural Network to TCP Congestion Control. Proceedings Of The 5th International Conference On Advanced Design And Manufacturing Engineering, China, p.1-5, jan. 2015.

[12] SAMARAWEERA, N.k.g.. Non-congestion packet loss detection for TCP error recovery using wireless links. Iee Proceedings - Communications, Eastleigh (Reino Unido), v. 146, n. 4, p.222-230, ago. 1999. Disponível em: <http://digital-library.theiet.org/content/journals/10.1049/ip-com_19990597>. Acesso em: 30 dez. 2016.