# Redes de Aprendizado Profundo para Classificação do Nível de Congestionamento em Redes TCP/IP

***Resumo*— Atualmente as redes de dados permeiam praticamente todas as áreas de atividade humana. O modelo TCP, que preconiza a manutenção dos módulos de maior complexidade nos terminais da rede, é o grande propulsor dessa rápida escalada das redes digitais, em particular da Internet. Entretanto, essa liberdade de expansão e configuração dos elementos centrais de uma rede TCP, sem qualquer *feedback* para os usuários finais, torna impossível a existência de um mecanismo pré-definido, capaz de proporcionar, a todo instante, o uso mais eficiente de recursos em constante evolução, submetidos aos mais variados perfis de utilização. Tais desafios se refletem principalmente no Controle de Congestionamento (CC), responsável pela dosagem de dados a serem inseridos na infraestrutura da Rede ao longo de uma transmissão. O CC deve ser capaz tanto de poupar a rede, em momentos de sobrecarga, quanto de exigir mais dela, em momentos de subutilização. Buscando um mecanismo de CC flexível, justo e que não precise ultrapassar a capacidade da rede para descobrir a quantidade de banda disponível, este trabalho propõe um mecanismo para construção de uma rede neural de aprendizado profundo, capaz de classificar o nível de congestionamento de uma rede limitada por um roteador de borda. Os resultados mostram que o mecanismo proposto conduz a classificadores, que, em certas condições, são capazes de distinguir com mais de 99% acerto entre momentos de alto e baixo grau de congestionamento ao longo de uma transmissão TCP/IP.**

***Palavras-Chave*— Controle de Congestionamento, Inteligência Artificial (IA), Transmission Control Protocol (TCP).**

## I. Introdução

AS REDES digitais, regidas pela camada de transporte TCP, são praticamente imprescindíveis no dia a dia da sociedade moderna. Tal constatação se deve principalmente à maciça difusão de serviços disponibilizados na internet, que podem ser acessados dos mais diversos tipos de terminais, dotados das mais variadas arquiteturas. Por isso, o acesso à internet passou rapidamente de uma exceção para uma regra, uma vez que está ao alcance de dois terços da população mundial[1].

Por trás da internet, existe um conjunto de convenções pré-definidas pelo protoclo TCP, que viabiliza a comunicação entre os terminais de uma rede IP. Embarcados nos diversos sistemas operacionais, tais convenções, ao serem seguias, permitem às aplicações trocarem dados entre si, acionando um conjunto de serviços organizadas em camadas que dão origem à pilha de protocolos conhecida como TCP/IP.

### A. Controle de Congestionamento na Pilha TCP

Um dos componentes da Pilha TCP é a Camada de Transporte, responsável pelo Controle de Congestionamento (CC). O CC regula a quantidade de dados inserida na infraestrutura de comunicações a cada rodada de transmissão. Dessa forma, constitui parte extremamente relevante nas transmissões TCP, pois é o módulo que efetivamente regula as solicitações dos canais de transmissão por parte dos seus usuários.

O controle de congestionamento nas redes IP é realizado do lado do transmissor e é, em geral, baseado em rodadas de transmissão. No envio de um arquivo, por exemplo, o transmissor divide os bytes desse arquivo em seguimentos. A cada rodada o transmissor insere uma quantidade desses segmentos na rede, baseado numa função que regula o número máximo de seguimentos que podem ser inseridos na rede. Tal quantidade é conhecida como Janela de Congestionamento (*cwnd*). Em geral, a *cwnd* é atualizada a cada confirmação de recebimento por parte do receptor ou na ausência dessa confirmação para pacotes mais recentemente transmitidos.

### B. Classificação por Aprendizado Profundo (Deep Learning)

Matematicamente, classificar significa relacionar os elementos de um conjunto $X$ aos elementos de um outro conjunto $Y$. Mais especificamente, se cada elemento $i$ de $X$ e $j$ de $Y$ puderem ser codificados respectivamente em vetores $x_i$ e $y_j$, a classificação pode ser realizada por meio de uma função $f : X \to Y$, tal que $f(x_i) = y_i$, em que $y_j$ corresponde ao tipo ao qual pertence o elemento representado por $x_i$.

Dentre as inúmeras técnicas voltadas para realização de uma classificação eficiente, destaca-se o Aprendizado profundo ou *Deep Learning*[2]. Esta técnica de inteligência artificial se baseia na obtenção de uma função ou modelo $f(x_i) = Wx_i + b$, em que $W$ e $b$ são matrizes capazes de associar de forma eficiente os valores $x \in X$ ao correspondente tipo $y_j$.

As matrizes $W$ e $b$ são obtidas ao longo de um processo de otimização ou treinamento. Supondo que na iteração $n$ tenhamos a matriz $W_n$ e $b_n$, os valores de $f_n(x_i) = W_n x_i + b_n$ calculados para cada $x_i \in X$ são utilizados para obteção de parâmetros que, ao serem multiplicados por uma constante $\alpha$, conhecida como taxa de aprendizado (*Learning Rate*), serão adicionados às matrizes $W_n$ e $b_n$, obtendo-se novas matrizes $W_{n+1}$ e $b_{n+1}$. Em geral, esse processo continuam até que seja atingido um número máximo de iterações, conhecido como épocas, ou até que se obtenha uma $f_n$ que consiga associar os elementos de $X$ ao correspondente tipo, em $Y$, com determinado grau de precisão. Outro fator a ser considerado é que, para se ganhar velocidade durante a fase de treinamento, a cada iteração, os valores de $f_n$ não são calculados para todos os elementos de $X$, mas para um subconjunto de tamanho fixo (*batch size*) formado por elementos aleatoriamente escolhidos de $X$.

Cumpre destacar ainda que, em termos práticos, as matrizes $W$ e $b$ são obtidas pela composição de $k$ outras matrizes, que formam uma estrutura conhecida como Rede Neural de Aprendizado Profundo (*deep learning network*), uma vez que é composta pela concatenação $k$ matrizes, referenciadas por camadas. Dessa forma, o valor de $k$ representa o número de camadas da Rede.

Uma outra questão interessante é que os conjuntos a serem classificados por uma rede neural nem sempre são lineares,

o que raramente ocorre em situações reais. Em contrapartida, a composição de matrizes apresentada levará a modelos que se comportariam bem em modelos de dados lineares. Para contornar este problema, os valores fornecidos pela operação das matrizes (valor linear) devem passar por uma fase de não-linearização. Nesta fase, o valor linear alimenta uma função não-linear, conhecidas como função de ativação, cuja saída é propagada para as demais camadas. Existem inúmeras funções de ativação, entre as quais destacam-se a Sigmoid, Tanh, ReLu e Softmax. Mais detalhes sobre a fase não-linear de uma rede neural e funções de ativação podem ser obtidos em [2]

A eficiência de um modelo não deve ser avaliada apenas no âmbito do conjunto que participou das iterações acima (dados de treinamento). Para ser realmente eficiente, o modelo deve ser aderente a outros conjuntos, além de $X$, que também encontrem correspondência com os elementos de $Y$, o que reflete o grau de generalização do modelo. Em particular, quanto mais precisas e gerais forem as saídas de um modelo que classifique o grau de congestionamento de uma rede entre baixo ou elevado, mais segurança um CC terá para regular sua *cwnd* baseado nestas saídas.

## II. Motivação

Além dos desafios até aqui apresentados, envolvendo redes domésticas e corporativas, no âmbito operacional, há sistemas cuja concepção prevê a alternância constante entre enlaces completamente distintos, como é o caso do Módulo de Telemática Operacional (MTO) [3]. Nestes sistemas, são previstas repentinas variações nas características do enlace, em relação à banda disponível (milhares de bytes a centenas de bits), RTT (centenas de microssegundos a milhares de segundos) e taxa de erro (elevadas em HF). Tal cenário requer que o CC, a cada chaveamento de enlace, realizado de forma completamente aleatória, seja capaz de regular sua taxa de inserção de dados com valores completamente distintos, aderentes às peculiaridades de cada meio de transmissão.

Buscando contribuir com a superação destes desafios, o trabalho segue apresentando alguns trabalhos relacionados ao tema da pesquisa. Em seguida, descreve como os dados de treinamento e testes foram obtidos, por meio do desenvolvimento de uma ferramenta denominada Analisador de Tráfego. A seção V descreve como os dados foram preparados, antes da sua utilização nos testes descritos na seção VI, que descreve o conjunto de experimentos. Os resultados dos experimentos são analisados na seçãoVII, que antecede as conclusões, em que são apresentadas as considerações finais e sinalizadas iniciativas que podem culminar em eventuais trabalhos futuros.

## III. Trabalhos Relacionados

Ainda nos primórdios das Redes TCP, os mecanismos clássicos de CC, tais como o TCP Tahoe, TCP-Reno e NewReno[4] foram alvos de diversas propostas de otimização. Dentre elas, destacam-se o Vegas [5], BIC[6], CUBIC[7], Westwood[8], Fast[9], HighSpeed[10], Hybla[11], Illinois [12], YeAH [13], Compound [14] e, mais recentemente, o BBR, utilizado na plataforma Cloud da Google [15]. Apesar da grande contribuição desses trabalhos, eles apresentam uma abordagem calcada na definição de funções determinísticas para atualizar a (*cwnd*), idealizadas para cenários específicos. A busca por uma Camada de Transporte capaz de adaptar-se às mais diversas condições sazonais de uma rede, levou ao surgimentos de mecanismos de CC, baseados em princípios de Aprendizado de Máquina. Um exemplo típico dessa tendência é o Remmy [16], que se utiliza de uma tabela de ações, modelada para maximizar o desempenho numa rede aderente aos pressupostos assumidos durante a fase de treinamento. Infelizmente, passada a fase de treinamento, as reações às flutuações da rede permanecem fixas, o que faz com que o Remy sofra das mesmas restrições de abordagens que se utilizam de funções fixas para atualização da *cwnd*.

Mais recentemente, inúmeros outros trabalhos se utilizam dos benefícios do Reinforcement Learning [17] na construção de tabelas que mapeiem o estado apresentado por uma rede em uma ação (atualização da *cwnd*), mais promissora. Alguns desses trabalhos se baseiam em treinamento prévio a fim de extrapolarem CC mais flexíveis, tais como [18], [19], [20], outros na mudança *online* de políticas para a escolha de ações, tais como [21], [22] e [23]. Entretanto, métodos baseados em RL sofrem com a natureza contínua tanto do espaço de ações quanto do espaço de estados ao longo de um fluxo de transmissão, podendo até colapsar em alguns casos, como ressaltado em [24].

## IV. Obtenção Dados de Treinamento e Testes - Analisador de tráfego

A proposta do analisador de tráfico é levantar parâmetros de interesse em cada fluxo TCP estabelecido numa rede simulada em ns3. O Analisador pode ser configurado para uma quantidade livre de terminais e roteadores, que podem formar as mais diversas topologias, dentro da qual podem ser estabelecidos fluxos TCP entre quaisquer pares de terminais. A ferramenta é capaz de, para cada fluxo, levantar a cada pacote $ack$ recebido, medidas disponibilizados pelo ns3, que podem ser processados de acordo com o interesse do pesquisador. A figura 1 traz o exemplo de uma topologia *dumbell* na qual a ligação entre as estações 10.0.0.1 e 10.1.0.1 é destacada. Durante o período de simulação, a cada ack recebido por 10.0.0.1, o Analisador de Tráfego cria uma linha em um arquivo csv, cada qual com as seguintes medidas:

- média móvel exponencial ponderada do tempo de chegada entre pacotes $ack(ack\_ewma)$
- média móvel exponencial ponderada do tempo decorrido entre o envio de um seguimento e a chegada do respectivo ack $(send\_ewma)$
- O $RTT$ observado para o ack considerado. Posteriormente, estes RTT serão divididos pelo $RTT_{min}$, dando origem ao $rtt\_ratio$.
- o valor da $cwnd$ no momento da chegada do $ack$.
- média móvel exponencial ponderada do percentual de ocupação do *buffer* do roteador de borda, ao qual está conectado o terminal receptor da transmissão (*last router occupation (lro)*), designada como $lro\_ewma$.

Para $ack\_ewma$ e $send\_ewma$ foram atribuídos o peso de $1/8$ para as novas medidas. Já para $lro\_ewma$ foi adotado o valor de $1/10$. Além disso, visando um treinamento mais preciso da rede, o analisador foi configurado para registrar dados apenas em situações em que $lro\_ewma < 60\%$ ou $lro\_ewma > 70\%$. Ou seja, na classificação do estado da rede, situações em que a rede apresenta ocupação de buffer

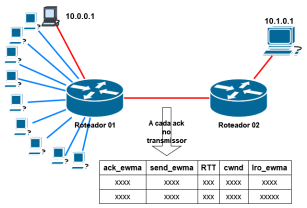



Fig. 1: Fluxo rastreado pelo Analisador de Tráfego

entre 60 e 70% foram consideradas como *don´t care*. Esse design visa deixar o modelo mais robusto na classificação de situações extremas, de subutilização ($< 60\%$) ou de sobrecarga ($> 70\%$).

## V. Preparação dos Dados, Estrutura da Rede de Aprendizado e Treinamento

Antes da inserção dos dados na rede neural de aprendizado profundo, eles passaram por um processo de normalização. Na primeira fase (vertical) de preparação, as *features* $ack\_ewma$, $send\_ewma$, $rtt\_ratio$ e $cwnd$ são divididas, respectivamente, por $(ack\_ewma\_max)$, $(send\_ewma\_max)$ $rtt\_ratio\_max$ e $cwnd\_max$, que correspondem aos valores máximos obtidos para cada uma delas ao longo de um período de simulação no qual o analisador de tráfego estava ativo. Posteriormente, cada vetor obtido pelo processo anterior é dividido pela respectiva norma, dando origem ao vetor $(ack\_ewma\_nor, send\_ewma\_nor, rtt\_ratio\_nor, cwnd\_nor)$.

Para realização do treinamento, os vetores acima descritos são acompanhados de 1, caso a $lro\_ewma$ esteja abaixo de 60% ou 2, caso esteja acima de 70%. Retomando o viés matemático do processo de classificação, cada $x_i$ seria a quádrupla $(ack\_ewma\_nor, send\_ewma\_nor, rtt\_ratio\_nor, cwnd\_nor)$ e os elementos de $Y$ o conjunto $\{1, 2\}$

A estrutura da rede neural utilizada é composta por quatro camadas. As funções de ativação dessas camadas são ReLU. Após essas quatro camadas é acoplada uma camada de saída *softmax*. Este modelo se utiliza de uma taxa de aprendizado de 0.001, (512) de *batch* size e 1000 épocas. A implementação dessa rede está disponível em `https://github.com/reisdout/FederatedColab/blob/main/neural_network_model.ipynb`

## VI. Experimentos

Para cada um dos experimentos descritos a seguir, foram realizados os seguintes passos:

- Passo 01: 05 simulações (sim01, sim02, sim03...sim05) ns3 nas quais são levantadas as *features* $ack\_ewma$, $send\_ewma$, $rtt\_ratio$, $cwnd$ e $lro\_ewma$.
- Passo 02: Treinamento da rede neural com os dados da sim01
- Passo 03: Generalização do modelo treinado pela sim01 aos demais fluxos.

A partir de agora, a menção aos termos (sim01, sim02, sim03...sim05) se referem ao contexto dos passos acima apresentados.

### A. Apenas um Fluxo longo

Neste cenário temos uma estação ligada ao roteador 01 por um enlace de 1Gbps e 10ms de RTT, que busca transmitir 4GB para o Servidor 01 (2).

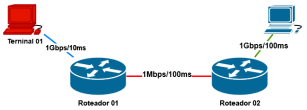



Fig. 2: Topologia para 1 longo.

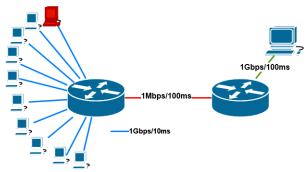



Fig. 3: Topologia 10 longos

*1) Treinamento da rede neural com os dados da sim01 - 01 fluxo longo:* Os resultados dos testes com dados fornecidos pela sim01, utilizada para treinar o modelo, podem ser observados por meio da matriz de confusão da figura 4-a:

*2) Generalização do modelo a outros fluxos - 01 fluxo longo:* O modelo gerado na fase anterior foi utilizado para prever o nível de congestionamento de outros quatro fluxos, que, reforçando, não forneceram dados para o treinamento. Os resultados são expressos pelas matrizes de confusão da figura 5.

### B. Dez fluxos longos

A segunda fase de testes foi realizada com 10 fluxos longos, cinco médios (arquivos de 100KB) e outros quatro curtos (arquivos de 10KB). Há um fluxo longo que perdura por toda simulação, do qual são extraídos os dados de treinamento e testes (terminal em vermelho na figura 3). Para evitar sincronização, os outros fluxos longos entram após decorridos 30 segundos de simulação, espaçados de 0.01s. Os fluxos médios e curtos entram em tempos aleatórios no terço final da simulação e servem para inserção de um tráfego de fundo. Uma vez estabelecidos, todos os terminais ligados ao roteador 01 estabelecem conexões TCP com o Servidor 01.

*1) Treinamento da rede neural com os dados da sim01 - 10 fluxos longos:* Assim como no experimento anterior, os dados fornecidos pela sim01 foram utilizados para treinar o modelo. O resultado dos testes com dados extraídos deste conjunto de treinamento podem ser observados por meio da matriz de confusão da figura 4-b:

*2) Generalização do modelo a outros fluxos - 10 fluxos longos:* A generalização do modelo aos dados dos fluxos oriundos das demais simulações é dada pela figura6.

### C. Vinte Fluxos Longos

Esses testes seguiram os mesmos moldes daquele com 10 fluxos longos. A diferença é que foram estabelecidos 20 fluxos longos, 10 médios e 9 curtos. Os resultados dos testes realizados com dados de treinamento estão na figura 4-c, enquanto que a aderência a fluxos que não participaram do treinamento estão na figura 7.

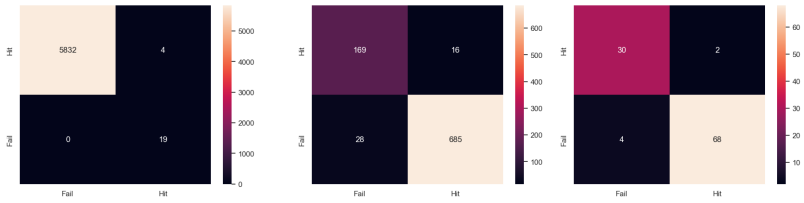
## VII. Análise dos Resultados

A capacidade de classificação do modelo se mostrou extremamente eficiente em todos os experimentos. Para testes realizados com os dados de treinamento, todos os percentuais de acerto ficaram acima dos 94%, atingindo seu pico para o experimento realizado com 1 fluxo longo (tabela I). Os testes realizados sobre dados que não participaram do treinamento também revelaram um elevado percentual de acerto do modelo, novamente com destaque para as simulações com 1 fluxo longo (tabelaII).

## VIII. Conclusão

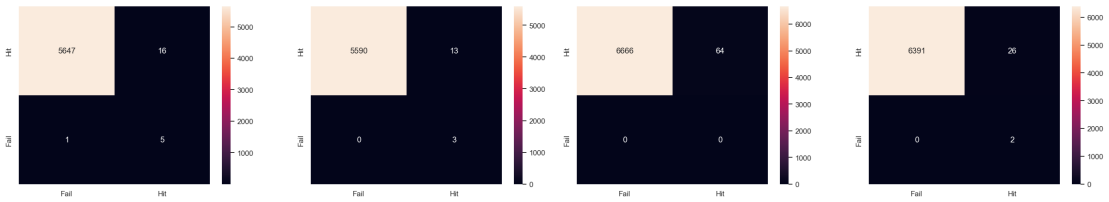
A técnica apresentada conduziu a modelos extremamente eficientes na classificação do nível de congestionamento. Os classificadores foram capazes de prever com precisão o nível de congestionamento da rede, chegando a mais de 99% de acerto para cenários em que há apenas 1 fluxo estabelecido. Mesmo para cenários mais complexos, os modelos se destacam, batendo 96% e 88 % de acerto para 10 e 20 fluxos, respectivamente. A precisão apresentada nos testes habilita a proposta apresentada como meio auxiliar para construção de mecanismos de CC.

Outro fator a considerar é a relação entre a complexidade na obtenção do nível de congestionamento e a simplicidade da solução



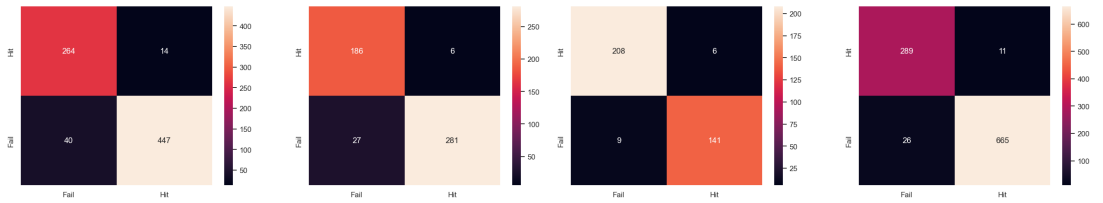

(a) sim 01 - 01 Longo (b) sim 01 - 10 Longos (c) sim 01 - 20 Longos

Fig. 4: Testes - Dados Treino



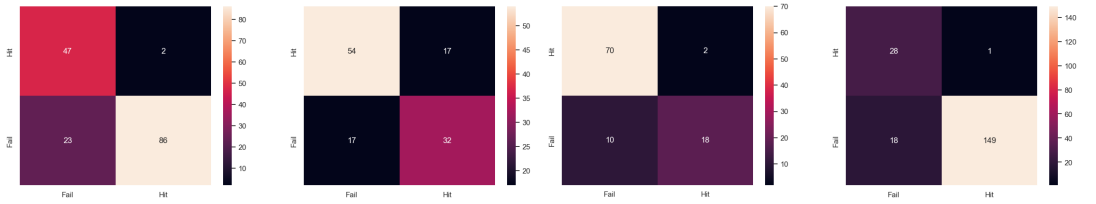

(a) sim 02 (b) sim 03 (c) sim 04 (d) sim 05

Fig. 5: Generalização - 01 Longo




(a) sim 02 (b) sim 03 (c) sim 04 (d) sim 05

Fig. 6: Generalização - 10 Longo




(a) sim 02 (b) sim 03 (c) sim 04 (d) sim 05

Fig. 7: Generalização - 20 Longos

TABELA I: Eficiência dos Modelos Gerados - Dados Treino

| Fluxos Longos | Acerto (%) |
|---|---|
| 1 | 99% |
| 10 | 95% |
| 20 | 94% |

TABELA II: Eficiência dos Modelos Gerados - Dados não treino

| Fluxos Longos | Simulação | Acerto (%) |
|---|---|---|
| 1 | sim02 | 99,68% |
| | sim03 | 99,76% |
| | sim04 | 99,05% |
| | sim05 | 99,59% |
| 10 | sim02 | 92,94% |
| | sim03 | 93,40% |
| | sim04 | 95,88% |
| | sim05 | 96,27% |
| 20 | sim02 | 94,23% |
| | sim03 | 84,18% |
| | sim04 | 71,67% |
| | sim05 | 88,00% |

proposta. A rede Neural apresentada é bem simples e foi treinada com um número bem modesto de épocas. Obviamente, as topologias trabalhadas trazem uma quantidade de terminais limitada, abrindo espaço para busca de modelos mais gerais, eficientes em ambientes mais complexos.

## REFERÊNCIAS

[1] "Global Connectivity Report 2022." [Online]. Available: https://www.itu.int/itu-d/reports/statistics/2022/05/29/gcr-chapter-1

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Nov. 2016.

[3] M. Hinago and F. P. Piurcosky, "A capacitação no projeto SISFRON: as lições aprendidas do projeto piloto e as perspectivas para o prosseguimento das próximas fases." pp. 285–320, Dec. 2021. [Online]. Available: https://ojs.ufgd.edu.br/index.php/moncoes/article/view/14387

[4] B. Singh, "A Comparative Study of Different TCP Variants in

Networks," *International Journal of Computer Trends and Technology*, vol. 4, no. 8, 2013.
[5] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance."
[6] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *IEEE INFOCOM 2004*, vol. 4, Mar. 2004.
[7] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: https://dl.acm.org/doi/10.1145/1400097.1400105
[8] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks."
[9] S. Hegde, D. Lapsley, B. Wydrowski, J. Lindheim, D. Wei, C. Jin, S. Low, and H. Newman, "FAST TCP in High-Speed Networks: An Experimental Study."
[10] S. Floyd, "HighSpeed TCP for Large Congestion Windows," Internet Engineering Task Force, Request for Comments RFC 3649, Dec. 2003. [Online]. Available: https://datatracker.ietf.org/doc/rfc3649
[11] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, Sep. 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/sat.799
[12] S. Liu, T. Başar, and R. Srikant, "TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks," *Performance Evaluation*, vol. 65, no. 6-7, pp. 417–440, Jun. 2008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0166531607001307
[13] A. Baiocchi, A. P. Castellani, and F. Vacirca, "YeAH-TCP: Yet Another Highspeed TCP."
[14] K. T. J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks."
[15] "TCP BBR congestion control comes to GCP – your Internet just got faster." [Online]. Available: https://cloud.google.com/blog/products/networking/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster
[16] K. Winstein and H. Balakrishnan, "TCP ex Machina: Computer-Generated Congestion Control."
[17] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction."
[18] N. Jay, N. H. Rotman, P. B. Godfrey, M. Schapira, and A. Tamar, "A Deep Reinforcement Learning Perspective on Internet Congestion Control."
[19] W. Li, F. Zhou, W. Meleis, and K. Chowdhury, "Learning-Based and Data-Driven TCP Design for Memory-Constrained IoT," in *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Washington, DC, USA: IEEE, May 2016, pp. 199–205. [Online]. Available: http://ieeexplore.ieee.org/document/7536338/
[20] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic Meets Modern: a Pragmatic Learning-Based Congestion Control for the Internet," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. Virtual Event USA: ACM, Jul. 2020, pp. 632–647. [Online]. Available: https://dl.acm.org/doi/10.1145/3387514.3405892
[21] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 445–458, Jul. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8357943/
[22] V. Badarla and C. S. R. Murthy, "Learning-TCP: A stochastic approach for efficient update in TCP congestion window in ad hoc wireless networks," *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 863–878, Jun. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0743731510002819
[23] B. Ramana, B. Manoj, and C. Murthy, "Learning-TCP: a novel learning automata based reliable transport protocol for ad hoc wireless networks," in *2nd International Conference on Broadband Networks, 2005.* Boston, MA: IEEE, 2005, pp. 521–530. [Online]. Available: http://ieeexplore.ieee.org/document/1589652/
[24] S. Emara, B. Li, and Y. Chen, "Eagle: Refining Congestion Control by Learning from the Experts," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. Toronto, ON, Canada: IEEE, Jul. 2020, pp. 676–685. [Online]. Available: https://ieeexplore.ieee.org/document/9155250/