Enfoque UTE
ISSN: 1390-6542
enfoque@ute.edu.ec
Universidad UTE
Ecuador

Segarra-Guzman, Edison; Ludeña-González, Patricia
Distributed Congestion Control Based on Utility Function
Enfoque UTE, vol. 15, núm. 2, 2024, -Junio, pp. 9-19
Universidad UTE
Ecuador

Distributed Congestion Control Based on Utility Function

Edison Segarra-Guzman[1], Patricia Ludeña-González[2]

***Abstract*—This paper introduces the Distributed Utility Function Algorithm (D-AFU) as a notable progression in managing and optimizing network traffic within distributed settings. Based on the utility function principle, D-AFU dynamically adjusts data rate in response to ever-changing network demands, with optimal performance and a higher user experience. Contrary to the centralized model, D-AFU employs a distributed, scalable and resilient against failures and system overloads mechanism. Its efficiency was validated using the NS-3 simulator. Three main metrics were used: the data rate allocation, utility per session, and fairness (quantified by the Gini coefficient). D-AFU displays exceptional performance and low latency, particularly vital for real-time applications with high Quality of Service (QoS) requirements.**

***Keywords* - Congestion Control, Utility Function, Real-Time applications, Elastic Applications, Distributed Optimization, Proactive Algorithm.**

***Resumen*—El artículo presenta el Algoritmo de Función de Utilidad Distribuida (D-AFU) como una notable evolución en la gestión y optimización del tráfico de red en entornos distribuidos. Basado en el principio de función de utilidad, D-AFU ajusta dinámicamente la velocidad de datos en respuesta a las demandas cambiantes de la red, con un rendimiento óptimo y una mejor experiencia para el usuario. A diferencia del modelo centralizado, D-AFU emplea un mecanismo distribuido escalable y con mayor resistencia contra fallos y sobrecargas del sistema. Su eficiencia fue validada utilizando el simulador NS-3. Se utilizaron tres métricas principales: la tasa de asignación de transmisión, la utilidad por sesión y la equidad (cuantificada por el coeficiente de Gini). D-AFU mostró un rendimiento excepcional, especialmente vital para aplicaciones en tiempo real que exigen alta Calidad de Servicio (QoS) y baja latencia.**

***Palabras Clave* - Control de congestión, Función de Utilidad, Aplicaciones en Tiempo Real, Aplicaciones Elásticas, Optimización Distribuida, Algoritmo Proactivo.**

I. Introduction

In recent years, data networks users have increased considerably. It causes big information quantity be transfered in and between networks, demanding more links capacity in them. Nowadays, everyone wants to be always connected by shortening distances, thus telecommunications are a critical factor and optimization of available resources is required to guarantee QoS. This mission comes with great challenges. Table I shows the number of devices currently connected to the Internet [1].

Problems such as limited memory resources in routers and bandwidth in links, generate network congestion, according to Kurose [2]. It causes packet loss and delay, therefore, in the literature the occurrence of these phenomena is considered a clear sign of congestion. Congestion causes packets to be retransmitted, affecting network performance. For this reason, throughout the networks evolution, several methods have been developed to control congestion and thus provide an efficient and reliable way to transmit data. Traditional congestion control protocols work by using a window to limit the amount of data a source can transmit to the network. The congestion window is a measure of the amount of data the remitter can send without receiving an acknowledgment from the destination. These protocols implement a flow control mechanism that prevents a source from sending more data than a receiver can process [3]. To reach the optimal congestion window, start with a low transmission rate and slowly increase it until fill the capacity of the network. Although these algorithms are widely used, their application in modern networks is inefficient, since they take a long time to reach full network capacity, so they are being replaced by more sophisticated protocols that avoid congestion without affecting network capacity utilization [4]. This paper provides a detailed and critical analysis of existing congestion control algorithms and presents an innovative approach that improves the efficiency of data transmission in networks. It uses utility function to assign data rate, following Max–Min fairness criterion. Thus, it contributes to the academic debate on congestion control in data networks, and provides a practical and feasible solution. It can serve as a starting point for future research and development in this knowledge area.

II. State of the Art

Congestion control is one of the most crucial issues in the field of networks, due to its direct impact on performance and quality of service. This aspect becomes even more relevant in the face of increasing network traffic demand.

Based on their behavior in response to congestion, the algorithms can be classified into two main categories [5]: - Proactive: They prevent congestion even before it occurs. These algorithms evaluate the state of the network taking measures periodically and determine the optimal transmission rates or window sizes before starting data transmission, anticipate congestion problems.

[1]Edison Segarra-Guzmán. Of Universidad Tecnica Particular de Loja, (e-mail: eesegaarra@utpl.edu.ec). ORCID number 0009-0006-9784-434X.

[2]Patricia Ludeña-González. Of the Departamento de Ciencias de la Computación y Electrónica, Universidad Técnica Particular de Loja, (e-mail: pjludena@utpl.edu.ec). ORCID number 0000-0002-8909-4837.

Manuscript Received: 26/08/2023
Revised: 20/11/2023
Accepted: 18/02/2024
DOI: https://doi.org/10.29019/enfoqueute.994

TABLE I
COMPARISON OF INTERNET USE BETWEEN 2017 AND 2022.

Year	Internet users (billions)	Devices and connections (billions)	Bandwidth	Video applications
2017	3.4	18.0	39.0 Mbps	75%
2022	4.8	28.5	75.4 Mbps	82%

- Reactive: After the congestion occurs these algorithms take necessary measures to counteract it. In large networks, a large number of packets may even be discarded, causing packet retransmission and generating communication delays, before the congestion is detected. It combines the pro-activity in the sense that it dynamically calculates the available bandwidth and adjusts the congestion window and the reactive part. For example, it is like Transmission Control Protocol (TCP) (Tahoe, Reno, NewReno, among other) works. It reacts to variations in the packet loss rate, RTT and throughput to calculate the congestion control factor and adjust the CWND [3].

According to Alizadeh [6], the search for new methods for congestion control for data centers is because they react to congestion after it has already occurred, this can lead to performance degradation and packet loss, on the one hand, small windows of the key features of the two protocols: Slow start when first establishing a TCP connection, the sender starts with a small window size and gradually increases the window as packets are acknowledged. These two mechanisms work together to adjust TCP's transmission rate to network conditions. 'Slow start' allows TCP to quickly 'bootstrap' to a reasonable congestion window size, while Additive Increment, Multiplicative Decrement (AIMD) allows TCP to adjust the sending rate in response to congestion once the connection is up and running. It mitigates congestion by incrementally scaling the volume of traffic transmitted over the network. Upon reaching a predefined threshold, the sender transitions into congestion prevention mode. In this mode, the sender incrementally expands the window size for each acknowledgment received, thereby proactively averting network congestion.

Both are effective in preventing congestion and guaranteeing reliable delivery of data [7] but are not optimal for datacenter deployments because they are designed for long-lived connections with slow startup phases and congestion avoidance.

Modern congestion control algorithms are proactive. However, they are more complex than traditional congestion control algorithms and may not be suitable for all networks. Constant monitoring may generate additional overhead on network resources, which could negatively affect overall performance. Then, proactive algorithms have a higher processing load than reactive algorithms.

According to Ludeña-González, López-Presa and Muñoz [8] proactive congestion control through explicit rate control (ERC) mechanisms has been proposed as a viable alternative to improve efficiency and fairness in communication networks. These algorithms calculate explicit rates to improve convergence time and ensure a fair distribution of resources among competing sessions.

Shiyong Li [9] considers the problem of bandwidth allocation in peer-to-peer (P2P) networks which are a type of decentralized network where users can share resources with each other, a new approach for bandwidth allocation based on utility optimization is proposed. The paper considers two types of services: elastic services and inelastic services. Elastic services are services that can adapt to the available bandwidth and inelastic services are services that must have a constant bandwidth. The bandwidth allocation scheme is based on a gradient-based algorithm. It works by finding the direction of greatest increase (or decrease, depending on the context) in a function, and then updating the parameters (in this case, the bandwidth allocation for different data flows) in that direction [10].

In Bahnasy work [11], a distributed congestion avoidance algorithm that functions at both the Ethernet layer and the TCP layer, is proposed and named Ethernet Congestion Control Zero-tailed Congestion Control Protocol (ECCP). It controls data traffic according to the estimated available bandwidth over a network path and attempts to keep link occupancy below the maximum capacity by a percentage called the Availability Threshold. Each node in the network maintains a link capacity table, when a node receives a packet, it updates its fields for the link on which the packet was received, then the node uses that link data (capacity) to estimate the available network bandwidth. To control the transmission rates of the sessions each node maintains a flow table. When a node receives a packet, it updates the rate table for the flow that sent the packet. Then, the node, once its session table is updated, calculates the maximum transmission rate for each flow.

Adams indicates in his research conclusions that working with active queue management (AQM) is an important technique for congestion control in data networks. [12] AQM algorithms can be used to improve network performance by reducing packet loss, queuing delay, and throughput reduction. Random Early Detection (RED) algorithm is part of the AQM family. It is used to avoid congestion by randomly discarding packets when the queue length exceeds a certain threshold. When congestion occurs, the queue length of a router interface increases which can cause delays for packets entering in a queue order and can even cause packets to be lost. However, it is inefficient in cases where, according to Varma [13], the number of connections passing through the link becomes too small, or the latency and capacity for the connection becomes too great.

Shuihai Hu, et. al., [14] also notes that proactive congestion control algorithms have been proposed to improve the performance of data center networks by explicitly scheduling data transmissions based on network bandwidth availability. However, these algorithms can perform poorly for small flows, which typically have short durations and low bandwidth requirements, so they propose as a solution a new algorithm called Aeolus. It addresses the problem of poor performance for small flows, in turn allows new flows to start at line

rate or the full available link capacity and then selectively discard excess unscheduled packets once congestion occurs, the protocol is evaluated through simulations with realistic workloads, provides support for simulation of a variety of network protocols at various levels, allowed researchers to model and analyze the performance of networks under different conditions [15]. Showing that the algorithm can significantly speed up small flows, for example, offering 55.9 % less 99th percentile completion time, while retaining all the good properties of proactive solutions. It functions by maintaining two separate queues for each flow: a scheduled queue and an unscheduled queue. When a new flow arrives, adds its packets to the scheduled queue and starts sending them at line rate. If congestion occurs, starts discarding packets from the unscheduled queue. When congestion is low, increases the size of the scheduled queue, allowing more packets to be sent at line speed [14]. The paper 'Accurate Congestion Control for RDMA Transfers' by Dimitris Giannopoulos et al. [16] proposes a new congestion control protocol for Remote Direct Memory Access (RDMA) transfers technology that allows two computers to exchange data directly from their memory without involving the operating system or CPU. ACCurate is crafted to embody efficiency and fairness in congestion control. Its precision is achieved through an algorithm dedicated to estimating available bandwidth. The efficiency is notable as it doesn't necessitate the maintenance of per-flow state within the network. Moreover, its commitment to fairness is evident in its assignment of accurate Max–Min fair rates to all flows. The novelty of ACCurate lies in its hardware-based design and implementation for congestion control. Through comprehensive performance evaluations conducted under diverse loads, the results demonstrate that it surpasses TCP-derived protocols and RDMA PAUSA-only in terms of flow completion times and fairness.

In addition, for data centers, Mahmoud Bahnasy and Halima Elbiaze [17] indicates how data centers have evolved from a common Ethernet network to a new era of data-intensive applications such as remote direct memory access, high performance computing and cloud computing, which pose new challenges for researchers, requiring minimal network latency, no packet loss and fairness between flows. Faced with the problem posed the authors propose the congestion control protocol for converged data access (DCB) networks (called HetFlow) is designed to achieve fairness between flows of different packet sizes and different RTTs by using a per-flow delay-based congestion control algorithm to adjust the sending rate using feedback messages in each flow based on the measured delay.

It is important to clarify that this protocol is within the congestion control group at the Ethernet layer, unlike protocols such as the Data-Center TCP (DCTCP) which are within the congestion control protocols of the transport layer and its Functionality leverages explicit congestion notification (ECN) in the network to provide multi-bit feedback to end hosts, when a switch detects congestion, marks packets with the Congestion Experienced (CE) code point. DCTCP hosts observe these markings and reduce their sending rate accordingly, this helps to keep queue occupancy low, which influences low latency and high throughput, it is more resilient to bursts of traffic and is currently one of the most widely used protocols in datacenters. It uses a multiplicative decrementing algorithm of additive increase to recover from congestion [6].

Congestion control in data centers presents challenges due to RTTs expressed in microseconds, the arrival of bursty flows, and a large number of concurrent flows [18]. These factors can force a flow to send at most one packet per RTT or induce a large backlog in the queue. The widespread use of switches with short buffers further exacerbates the problem, as hosts generate multiple flows in bursts. As link speeds increase, algorithms that gradually seek bandwidth take considerable time to reach their fair share. This is why Cho, Jang, and Han [18] propose ExpressPass, a credit-scheduled, end-to-end, delay-scheduled congestion control algorithm for data centers. It uses credit packets to manage congestion, even before sending data packets, leading to bounded delay and fast convergence. This approach handles bursty flow arrivals, for implementation, the results ExpressPass converges up to 80 times faster than DCTCP on 10 Gbps links and the gap increases as link speeds increase. It significantly improves performance under heavy incast workloads and significantly reduces flow completion times [18]. Into wireless networks the authors M. Singh S, et. al, [19] propose the Dynamic TCP (D-TCP) algorithm. It learns the available bandwidth and adjusts the congestion window. It first estimates the available bandwidth using a combination of methods, such as packet loss rate, RTT, and throughput. Once the available bandwidth is estimated, it uses this information to calculate a congestion control factor. Then, this factor is used to adjust the congestion window, which is the amount of data a sender can send before receiving an acknowledgment, D-TCP attempts to adaptively bring the CWND to the previous state with the help of the calculated bandwidth (based on learning). This helps to efficiently control CWND for better network utilization, especially under conditions of high packet loss and high delay bandwidth product.

Machine Learning (ML) is a branch of artificial intelligence that focuses on the development of algorithms and statistical models that allow computer systems to learn and improve their performance based on data and past experience, rather than being explicitly programmed [20]. Within trend analysis in congestion control there are works that use ML for congestion control, the work of Ning Li, et. al., [9], presents AdaBoost-TCP in a satellite network, where congestion control in highly dynamic networks represents a significant challenge due to the frequent switching of satellite links. In the context of this paper 'boost' refers to the process of converting a set of weak learners into a strong learner, and 'adaptive' refers to how AdaBoost adjusts the misclassification weights to guide the learning of weak learners [21]. Switching in satellite networks can result in connection instability and increase packet loss, thus reducing network efficiency. TCP fails to effectively distinguish packet loss types, leading to network underutilization [22]. The sender adopts adaptive congestion control measures based on the type of packet lost, which allows for greater efficiency in congestion management. The results, when the packet loss rate is between 10^{-5} and 10^{-4},

the AdaBoost-TCP strategy can increase throughput by 10 % compared to other congestion control algorithms, such as Hybla [23] which was designed to improve TCP throughput over links with long RTT. In addition, AdaBoost-TCP shows good fairness in comparison to NewReno.

Mozo, López-Presa and Fernandez Anta [24] presents an algorithm based on Max–Min fairness, named B-Neck. It is a distributed, quiescent and proactive protocol. It calculates the optimal data rates for each session without any information about routers flows. B-Neck converges to the optimal solution very fast and keeps the queues short. It is quiescente because stop to transmit packets if reach the optimal data rate, thus it saved energy and bandwidth.

The work of Ludeña-González, López-Presa, and Muñoz [8] proposes a solution to achieve maximum-to-minimum fairness (UMMF) in high-speed multipath networks. A centralized algorithm called c-SLEN (Saturation Level Explicit Notification) is presented and is based on the saturation level of the links to compute the fair rates of flows without affecting the throughput due to link capacity. In addition, a distributed version called d-SLEN is developed, which is the first of its kind and is characterized by its convergence speed independent of network capacity. Simulation results show that c-SLEN achieves session rates similar to other UMMF algorithms, but without saturating links, which improves network utilization. In addition, d-SLEN exhibits faster and more stable convergence than other distributed approaches.

Finally, as part of the study in the field of network congestion control, a novel approach has been observed through the Utility Function Algorithm (UFA). This algorithm, a variant of the B-Neck method, has incorporated Utility Functions with the objective of calculating the performance of the application as a function of the type of traffic generated by the sessions [25].The Max–Min fairness criterion has been applied by the AFU for bandwidth allocation, demonstrating significant effectiveness in network congestion control management. The AFU algorithm has been evaluated in three different scenarios through Matlab, providing a series of performance metrics including the allocated transmission rate, the utility achieved per session and the fairness measured through the Gini coefficient. The results obtained indicate that real-time applications, particularly those transmitting voice and video, experience better performance when using the AFU algorithm compared to other congestion control strategies. This finding suggests that the AFU could be particularly useful in network environments where real-time traffic is predominant.

Importantly, this work offers promising insights for congestion control in networks and marks a way forward in the investigation of control mechanisms that can efficiently handle the demands of different types of traffic. However, today real-time applications demand increases, so more work is needed to explore and refine strategies such as AFU and to investigate how these techniques can be adapted and optimized for different network conditions.

Research in the area of congestion control in networks has provided a number of innovative algorithms that have informed and enriched this evolving field. In particular, works such as B-Neck and the Algorithm Utility Function (AFU) have proven to be influential, providing novel and valuable insights for the efficient management of network traffic. It is necessary to emphasize the relevance of these fundamental works. B-Neck and AFU have established novel criteria, compiling resource allocation with fairness, but also based on a utility function, in this area of research, setting a benchmark for future research and algorithm development.

III. Methodology: Congestion Control and Optimization strategy

A. Fair Max–Min Allocation Strategy

Congestion control algorithms and performance metrics are tailored to the needs of the applications. There is no ideal state in delay- and loss-based algorithms. It is complicated to guarantee an equitable distribution of network capacity, especially with different algorithms and routes, to measure the fairness of resource allocation in networks [26].

Equation 1

$$Fairness = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{1}$$

This is the formula for the Jain equity index. In this equation, x_i represents the throughput of the i-th flow, and n is the total number of flows.

Max–Min fairness principle seeks a fair bandwidth distribution for all applications, regardless of their criticality. Initially, it assumes that all rates are 0 and increase at the same rate until capacity limits are reached. This process is repeated until the rate cannot be increased any further in one session [27]. However, despite its apparent fairness, Max–Min fairness may not lead to optimal network utilization.

In proportional allocation, resources are distributed according to a metric, such as user demand. This could result in better resource utilization, but may be less fair if some users have a high demand for resources.

It is emphasized that fairness in congestion control seeks a fair distribution of network resources among different data flows. This aspect is critical in the design of congestion control mechanisms, as it contributes to the quality and perception of a better service for users.

B. Utility functions

Customer satisfaction in the consumption of goods and services is measured through a utility function. In the network context, it reflects a combination of objectives, such as efficiency, fairness, and quality of experience, and may incorporate factors such as the relative importance of different flows or sessions, so it is a measure of Quality of Service (QoS). Two types of traffic are distinguished in this work: best-effort traffic, which has no QoS requirements, and traffic with QoS requirements that needs specific resources [28], [29].

Figure 1 shows utility fuctions used in this work. Elastic applications are a subcategory of best-effort traffic. They vary the transmission rate according to the congestion signal and benefit from higher bandwidth. Theses applications

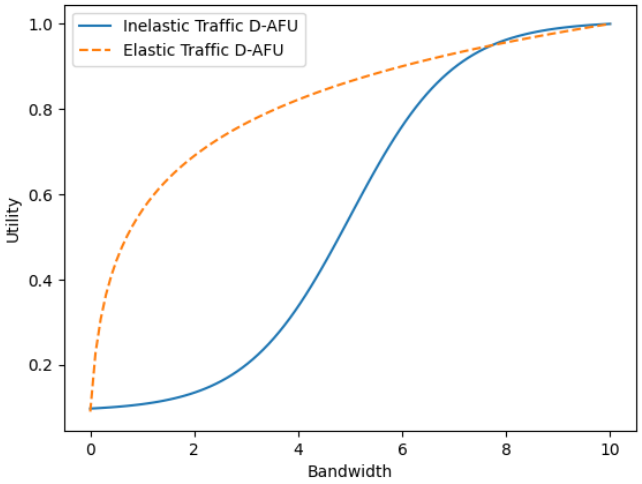

Fig. 1. Utility function used by D-AFU.

or services can adjust to different bandwidth levels. Their utility function is a logarithmic function, showing that utility increases as bandwidth increases, but at a decreasing rate [30]. The visual representation provided in aligns with the theoretical framework outlined by Equations 2 and 3, offering a graphical insight into the key principles presented in this study Logarithmic form of this function suggests that there is a decreasing benefit to increasing bandwidth. Examples of this function are web browsing, where increasing bandwidth improves the experience up to a certain point, but then further increases have a minor impact.

Equation 2

$$U^s(y) = w(\log(ay + b) + d) \tag{2}$$

U(y): Represents the utility, or user's perceived satisfaction, associated with the bandwidth *y*, *w*: A constant that scales the entire utility function, reflecting the importance or weight assigned to the utility, *a*: A parameter that influences the rate at which utility increases with bandwidth. *b*: An offset parameter that affects the baseline utility, possibly representing a minimum utility even at low bandwidth. *d*: An additional constant that could introduce an offset or baseline utility.

In this utility function the natural logarithm *(log)* is commonly used to represent diminishing marginal returns, implying that the utility increases at a decreasing rate as bandwidth increases. The parameter *a* affects how quickly the utility increases as bandwidth improves. The constant *b* could represent a baseline utility even with low bandwidth. The constant *d* might introduce an overall offset or baseline utility. The increasing nature of the utility function aligns with the description that as bandwidth increases, the user's perceived satisfaction or utility also increases. This type of utility function is consistent with the idea that additional bandwidth contributes positively to the user experience, and the rate of increase may be subject to diminishing returns. Consider an application such as file transfer via File Transfer Protocol (FTP). FTP is an elastic application that is it can vary its transmission rate depending on the amount of available bandwidth. If bandwidth is limited, FTP can adapt by reducing its transfer rate. However, if more bandwidth is available, FTP can increase its transfer rate, resulting in faster file transfer.

The utility function for FTP is an increasing function, meaning that as bandwidth increases, so does the user's perceived utility [27]. However, this function is typically logarithmic or concave, meaning that there are diminishing returns: perceived utility increases with bandwidth, but at a decreasing rate [31].

In contrast, inelastic traffic, which includes applications such as audio streaming and VoIP, does not easily adapt to changes in delay and throughput. If the allocated bandwidth is insufficient, throughput is significantly affected [31] because these applications or services has rigid bandwidth requirements. Its utility function is a sigmoidal function, showing a rapid initial increase in utility followed by a deceleration [27]. Specifically, during the very initial phase, the rate of increase is slower compared to an inelastic response. In other words, the utility experiences a gradual ascent during its early stages before entering a phase of accelerated growth. This sigmoidal behavior is indicative of the system's sensitivity to changes, with a more measured initial response that transitions into a steeper incline as the input or conditions vary. There is a point at which additional bandwidth increases no longer add much value (saturation). This could represent, for example, a live video stream where, beyond a certain point, increasing bandwidth does not significantly improve video quality.

Equation 3

$$U^r(y) = w\left(\frac{1}{1 + e^{-a(y-b)}} + d\right) \tag{3}$$

In the context of congestion control and network resource allocation, utility functions play a crucial role in evaluating the performance and fairness of resource distribution. The utility functions are typically used to model the satisfaction or benefit that users derive from the allocated resources, represents the utility associated with a particular resource allocation. The parameters *a, b, w* and *d* are essential in shaping the characteristics of the utility function. *y:* represents a variable associated with the resource allocation, which could be, for example, network bandwidth or throughput. *Logistic Function Models* the non-linear relationship between the resource allocation and user satisfaction. The parameters *a and b* control the shape of the curve, determining how quickly user satisfaction increases with resource allocation. *w*: represents a weight or importance factor, scaling the entire utility function. It could reflect the relative importance of user satisfaction in the overall network management objectives. *d*: Represents an additional constant that could introduce an offset or baseline utility.

C. Algorithms

Distributed control algorithms are important for architecture design and performance engineering of the communication

network. This allow the network to scale more easily because the work is divided among many nodes [32]. This is particularly useful in large networks where a centralized approach can be difficult to manage and could cause bottlenecks: if one node fails, the overall system operation does not stop. The other nodes can continue to function independently, which improves the resilience of the network.

Max–Min fairness principle focuses on fair resource allocation. Under this approach, all data flows start with the same transfer rate. As demand increases, the transfer rate increases proportionally for all flows until one of them reaches its maximum capacity [24].At this point, the transfer rate of that particular flow remains constant, while the other flows that still have additional capacity can continue to increase their transfer rate. This process is repeated until all transfer rates have been maximized, ensuring that no flow receives less than what would be given to any other.

The utility function describes the relationship between resource allocation and user satisfaction. For example, for elastic applications, which can adjust their transfer rate as a function of bandwidth availability, user utility or satisfaction increases as more bandwidth is allocated to the application, but at a decreasing rate [25].

In practice, these two concepts can be combined to develop resource allocation algorithms that balance fairness and maximization of user satisfaction. One possible approach would be to use the Max–Min fairness principle to determine an initial resource allocation and then adjust it based on the Utility Functions of different applications.

In one scenario, it could start by allocating bandwidth equally among all applications. Then, it could examine the Utility Functions and reallocate bandwidth from applications with decreasing marginal utilities (those that derive less benefit from additional increases in bandwidth) to applications with increasing marginal utilities (those that derive more benefit from additional increases in bandwidth).

By distributing decision making, each node can make adjustments and decisions based on local information, which can lead to more efficient and effective resource management. In the context of congestion control, distributed algorithms can respond to local traffic conditions, which helps to avoid congestion before it becomes a network-level problem [24].

Defining the part of the distributed algorithm has been taken as a source designed by Mozo, Lopez and Fernandez [24], where they define the following tasks according to the network segment in question, which the authors call them tasks, each of them is described below, the router tasks and the source node 1 are responsible for processing the majority of messages referring to the algorithm, while the destination node III-C tasks are limited to receiving messages to know when a session has joined, a test has started, or there is a message.

Router Link: Receives packets from source nodes and forwards them to other routers. Maintains a table of the current forwarding rates of all sessions traversing the router. Periodically updates the forwarding rates of all sessions based on the B-Neck algorithm. Measures the current bandwidth available on the link. Provides this information to the routers connecting to the link.

Algorithm 1
Task SourceNode(s, e).

procedure STARTPROBECYCLE $F_e \leftarrow \emptyset$; $R_e \leftarrow \{s\}$ pending_probe$_s \leftarrow$ FALSE bneck_rcv$_s \leftarrow$ FALSE $\mu_s^e \leftarrow$ WAITING_RESPONSE Send downstream Probe(s, D_s, e)
end procedure
while received API.Join(s, r) **do** $F_e \leftarrow \emptyset$; $R_e \leftarrow \{s\}$ $D_s \leftarrow \min(r, C_e)$ pending_probe$_s \leftarrow$ FALSE pending_leave$_s \leftarrow$ FALSE bneck_rcv$_s \leftarrow$ FALSE $\mu_s^e \leftarrow$ WAITING_RESPONSE Send downstream Join(s, D_s, e)
  while received API.Leave(s) **do**
    if $\mu_s^e =$ IDLE **then** $F_e \leftarrow \emptyset$; $R_e \leftarrow \emptyset$ send downstream Leave(s)
    else pending_leave$_s \leftarrow$ TRUE

    while received Update(s) **do**
      if $\mu_s^e =$ IDLE **then** StartProbeCycle()

      while received Bottleneck(s) **do**
        if $\mu_s^e =$ IDLE $\wedge$ bneck_rcv$_s$ **then** bneck_rcv$_s \leftarrow$ TRUE API.Rate(s, λ_s^e)
          if $D_s > \lambda_s^e$ **then** $F_e \leftarrow \{s\}$; $R_e \leftarrow \emptyset$ send downstream SetBottleneck($s, D_s = \lambda_s^e$)

        while received Response(s, τ, λ, η) **do**
          if pending_leave$_s$ **then** $F_e \leftarrow \emptyset$; $R_e \leftarrow \emptyset$ send downstream Leave(s)
          else if $\tau =$ UPDATE $\vee$ pending_probe$_s$ **then** StartProbeCycle()
          else if $\tau =$ BOTTLENECK **then** $\lambda_s^e \leftarrow \lambda$ $\mu_s^e \leftarrow$ IDLE bneck_rcv$_s \leftarrow$ TRUE API.Rate(s, λ_s^e)
            if $D_s = \lambda_s^e$ **then** bneck_rcv$_s \leftarrow$ TRUE API.Rate(s, λ_s^e) send downstream SetBottleneck(s, TRUE)
            else send downstream SetBottleneck(s, FALSE)

Source node: Sends packets to the router. Receives updates from the router on the current sending rates of all sessions. Adjusts its own sending rate based on updates from the router.

Destination Node: Receives packets from router. Delivers packets to application. Provides feedback to router.

Algorithm 2
Task DestinationNode(s).

while received SetBottleneck(s, β) **do**
  if $\neg\beta$ **then** Send upstream Update(s)
  end if
end while
while received Join(s, λ, η) **do** Send upstream Response(s, RESPONSE, λ, η)
end while
while received Probe(s, λ, η) **do** Send upstream Response(s, RESPONSE, λ, η)
end while

It should be noted that, once stable, the algorithm remains idle until a new session occurs or resources are released.

The Algorithm Utility Function (AFU) is a sophisticated method designed for the optimal allocation of bandwidth in a network, with the objective of controlling network congestion efficiently. AFU incorporates the Max–Min fairness criterion, which means that it seeks to maximize the minimum bandwidth allocation to any node in the network. The key to AFU's efficiency is its use of Utility Functions. These functions quantify the 'value' or 'utility' of a given bandwidth allocation for a particular application or type of traffic. Different applications and traffic types may have different Utility Functions or depending on their bandwidth requirements, their sensitivity to latency [26]. As can be seen the pseudocode of the 3 algorithm proposes bandwidth allocation according to the utility function.

Algorithm 3
Centralized AFU

$e \in E$
if $e \neq E$ **then**
$R_e \leftarrow S_e^*$
end if

$L \leftarrow \{e \in E \mid R_e \neq 0\}$
while $L \neq \emptyset$ **do** $e \in E$
if $e \neq E$ **then** $s \in e$
$U_s^*(y) \leftarrow U[1, 0]$
$B_s \leftarrow \frac{1}{U_s^*(y)} \cdot \left(C_e - \sum_{s' \in F_e} \lambda_{s'}^*\right)$
$B_e \leftarrow \{B_s\}$

end if

$B \leftarrow \min_{e \in L}\{B_e\}$
$L' \leftarrow \{e \in L \mid B_e = B\}$
$X \leftarrow \cup_{e \in L'} R_e \quad s \in X$
if $s \neq 0$ **then**
$\lambda_s^* \leftarrow B$
end if
$e \in L \setminus L'$
if $e \neq 0$ **then**
$F_e \leftarrow F_e \cup (R_e \cap X)$
$R_e \leftarrow R_e \setminus F_e$
end if

$L \leftarrow \{e \in (L \setminus L') \mid R_e \neq 0\}$
end while=0

Initialization: Resources are initially allocated equitably among all users, each user is associated with a utility function representing their satisfaction with the resource allocation. Max-Min: Users with lower utility (or less allocation) have the opportunity to obtain additional resources until their utility matches that of others. Utility Function Evaluation: The utility function for each user is evaluated based on their current resource allocation. The shape and parameters of the utility function determine the user's satisfaction in relation to the allocated resources. Iterative adjustments: The algorithm iterates to dynamically adjust resource allocations, users with lower utility are given priority for resource increments, promoting fairness. Dynamic Adaptation: The algorithm dynamically adapts to changes in network conditions, such as fluctuations in available bandwidth or the introduction of new users.

D. Simulation settings

In the simulation, the proposed protocol is tested on various network configurations to explore how different conditions may affect the performance of the strategies. This includes variations in the number of nodes, the amount of available resources, the demand for resources. The simulated network for this study considers two main types of traffic: Voice over Internet Protocol (VoIP) and File Transfer Protocol (FTP). These two types of traffic have been selected because of their divergent characteristics and common usage in today's networks. VoIP traffic is real-time and latency sensitive. It requires constant and relatively small bandwidth, and prioritizes low latency over absolute data integrity. Dropped or delayed packets can result in a noticeable degradation of call quality. On the other hand, FTP traffic is not real-time and is less sensitive to latency. It requires high volumes of bandwidth and prioritizes absolute data integrity over low latency. FTP transfers can occupy much of the available bandwidth, but can tolerate higher latencies.

E. Metrics

The network topology in simulations can vary depending on the number of routers to be incorporated, with several hops for a packet to reach its destination or within a single domain. Each router is configured to handle both VoIP and FTP traffic, and allocates bandwidth between these services using the Utility Function Algorithm (UFA), which applies the Max–Min fairness criterion to control network congestion, The topology depicted in Figure III-D is constructed using routers and hosts connected by links with fixed bandwidth. Hosts in this network play specific roles, serving as both traffic senders and receivers. The fixed bandwidth of the links establishes the capacity for data transfer between network elements. This configuration allows for the simulation and evaluation of network traffic scenarios, facilitating the assessment of the performance and adaptability of congestion control algorithms under controlled conditions. The role distinction of hosts as traffic sources and destinations contributes to a comprehensive evaluation of the algorithm's efficiency in handling bidirectional communication.

It has measured a number of metrics to evaluate the performance of the proposed strategies and algorithms. These include resource allocation fairness, resource usage efficiency, delay and packet loss. These metrics allow to quantify and compare the results in an objective manner.

In our experiment within a consistent network topology, we aimed to validate the effectiveness of a congestion control algorithm by diversifying traffic types and routing. Deploying distinct traffic categories, including voice, video, data, and routing them through various paths allowed us to assess the algorithm's adaptability and performance under diverse conditions. By measuring key parameters such as latency, packet loss, and bandwidth utilization, we gained insights into the

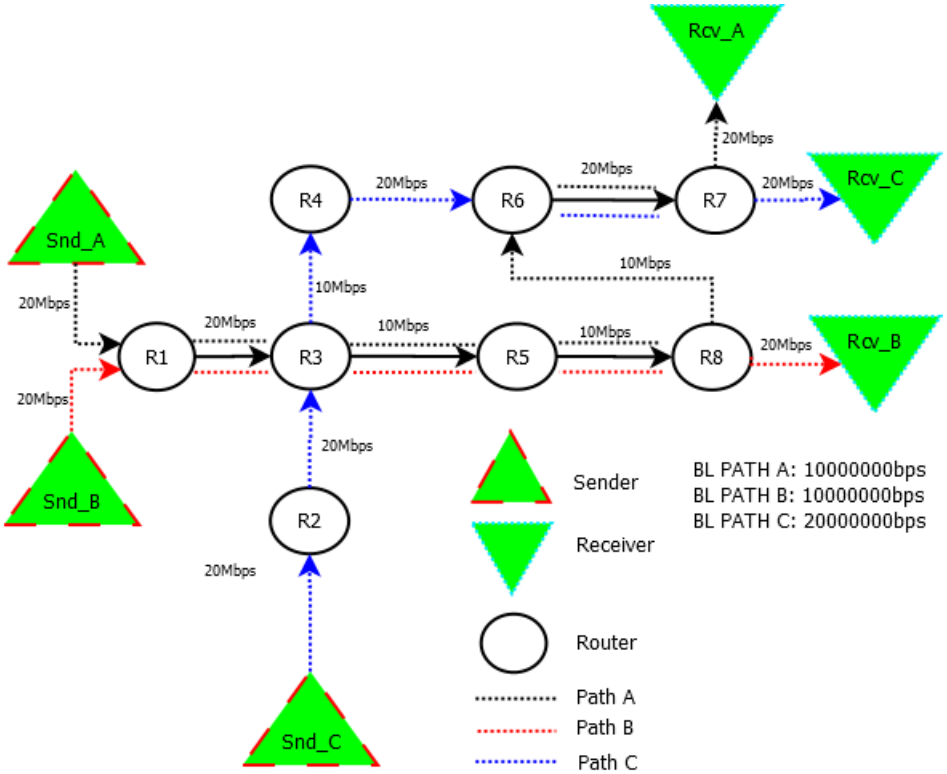

Fig. 2. Network topology used in simulations.

algorithm's responsiveness and efficiency. The experiment's findings provide valuable confirmation of the algorithm's validity, demonstrating its ability to handle varied network scenarios and optimize resource allocation based on the nature of the transmitted data.

F. Lost packages and queues

Packet loss refers to the number or percentage of packets that are sent from a source but do not reach their intended destination. This loss can be due to errors in network devices, congestion and collisions. Packet loss can have a significant impact on real-time applications, such as VoIP or online gaming, while network queuing refers to the temporary accumulation of packets at a network point, such as a router or switch, before they are processed or forwarded. Metrics related to queues can include queue length (number of packets in queue), queue delay (time a packet waits in queue) and discard due to full queues [33].

G. End-to-end packet delay

End-to-end packet delay, commonly referred to as latency, refers to the time it takes for a data packet to travel from a source to a destination over a network [34]. This delay can be caused by a variety of factors, including propagation time, transmission time, processing speed of intermediate devices, and queuing time on network devices. It is important to note that on larger or more congested networks latency can vary considerably. Latency variability is known as 'jitter' and can be problematic for time-sensitive applications.

The RTT is a crucial metric in networking that quantifies the time required for a data packet to traverse from the sender to the receiver and then return to the sender. RTT is a fundamental parameter in assessing the responsiveness and efficiency of network communication. It directly influences the perceived delay in data transmission, making it a significant consideration for real-time applications, such as video conferencing and online gaming

H. Fairness in resource distribution

Gini coefficient in the context of congestion control is a measure that quantifies the unequal allocation of resources in a network. Its value varies between 0 and 1, where 0 represents a completely equal distribution of resources and a completely unequal allocation [35]. The Lorenz curve, which plots the distribution of resources in the network, shows the cumulative proportion of bandwidth allocated in relation to the cumulative proportion of sessions or flows in the network [36].

IV. Results

All experiments have been performed in the NS3 network simulator. It is a discrete event simulator that is widely used in network research to model the behavior of computer networks [37]. This simulator has allowed to accurately and controlled recreate the requiered network conditions, and has provided a platform for D-AFU implementing and testing.

In this work, we proposed a novel congestion control algorithm with the aim of enhancing network performance in specific scenarios. To assess the effectiveness and efficiency

of our algorithm, we conducted a comprehensive comparison with TCP Reno, a widely recognized and utilized congestion control algorithm in networking environments.

TCP Reno has stood out as one of the most commonly implemented and studied congestion control algorithms in the networking community. Its approach of slow start, congestion avoidance, and fast recovery has served as a foundational framework for numerous developments in this field.

Throughout our testing and comparative analyses, we assessed the performance of our algorithm across diverse scenarios, taking into account factors such as bandwidth utilization efficiency, adaptability to network changes, and tolerance to packet loss. The comparison with TCP Reno provides a robust benchmark for understanding how our algorithm fares in relation to a well-established standard within the congestion control domain.

1) Lost packages and queues: From the results of a simulation of the network topology, where it plots the packets in the queue on the interface according to the flows sent, a larger queue is an indicator of congestion and packet loss occurs due to the timers in the applications. Figure 3 shows D-AFU keeps short queue all-time. It can be concluded by applying D-AFU, the queue decreases substantially, there is no packet loss and no packet retransmissions.
The absence of lost packets implies that all packets sent by the sources reach their intended destinations, which in turn may indicate that the network congestion control system is effectively functioning to prevent network saturation.

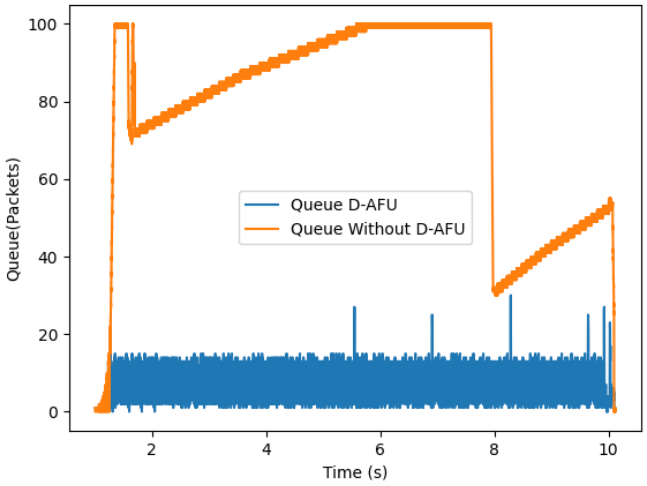

Fig. 3. Queues in intermediate nodes.

2) Data rates for each session: The results of the simulation applying elastic and inelastic traffic (real time) can be seen in the Figure 4. It shows that when applying D-AFU the values that the application needs are reached in a shorter time than the same resources obtained without applying the algorithm. In the elastic traffic instead it adapts to the available resources, which is directly related to the size of the queue. Vertical lines have been traced where alert a change made in the traffic. The application in real time obtains the bandwidth requirement in a shorter time than without the algorithm strategy. For the elastic traffic instead when sharing resources on a link this adjusts its bandwidth requirement based on the availability of: first the type of application with which it shares and second with the total bandwidth available, which can be noted from time six in the graph and at time 8 they manage to adjust a fair distribution of resources.

TABLE II
COMPARATIVE METRICS FOR THE NETWORK.

Strategy	Pkt. Send	Pkt. Recv	Avg. RTT (ms)	Freq. (sec)
D-AFU	300	300	61.5	0.3
Without D-AFU	300	250	718.6	0.3

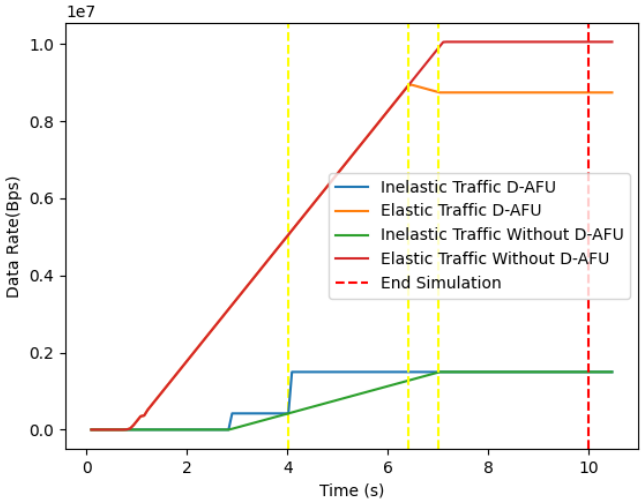

Fig. 4. Data rates for different traffic.

In addition, to analyze the effectiveness of congestion control, performance parameters were obtained using the D-AFU approach and a traditional method without D-AFU. The results presented in the Table II indicate a clear superiority of the D-AFU approach in terms of packets received. Both methods sent the same number of packets (300 packets in 10 seconds). The D-AFU method managed to deliver all packets successfully, while the non-D-AFU method delivered only 250 of the 300 packets. That noted, while the results of this particular simulation are positive, it is important to remember that simulations are simplifications of the real world and their results are dependent on user-specified conditions. In addition, having zero dropped and lost packets does not necessarily mean that the network is perfect. Other factors, such as latency or jitter, could still affect network quality. In Table II it can see the average RTT values. With D-AFU the lattency is slower that without D-AFU because, it proves to be almost ten times faster than the non-D-AFU method.

These results suggest that D-AFU congestion control is more efficient in saturated environments, providing better quality of service and faster response times. It improves the network performance.

3) Fair distribution of resources
Figure 5 plots Gini coefficient and Lorenz curve for the allocation data rate in the network. For D-AFU algorithm Gini coefficient is 0.52 while without applying the proposed algorithm is 0.81. This means that D-AFU, considering the utility function, makes a more fair distribution of network resources.

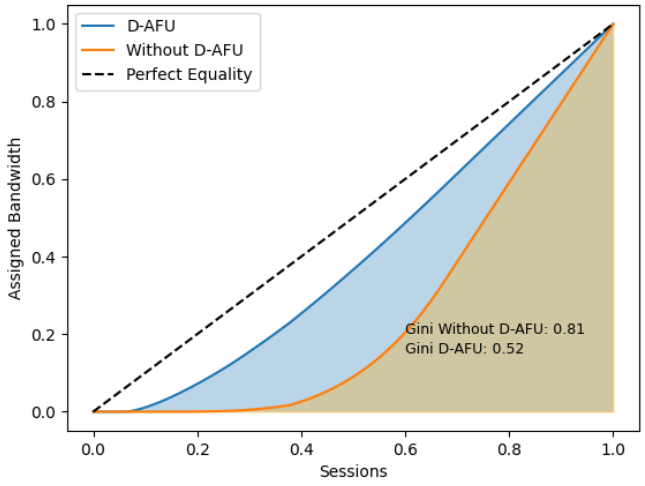

Fig. 5. Fairness measure.

V. Conclusions

Distributed Utility Function Algorithm (D-UFA) is an algorithm that achieves Max–Min fairness. This implies that it ensures that all sessions receive a fair share of the network bandwidth, a factor especially relevant in contexts with significant variations in traffic types, for example VoIP and FTP applications. It is a distributed algorithm, meaning that it does not require any central coordination. This allows D-AFU to efficiently adapt to network variations and respond locally to congestion. It is efficient and scalable, which means that it can be used in large networks. It is a quiescent algorithm because it stops generating traffic once it has converged to optimal sending rates. This is an important feature to avoid adding additional congestion to the network once the optimal state is reached. The results show D-AFU is a promising approach for network congestion control.

VI. References

References

[1] Cisco, "Cisco Annual Internet Report (2018–2023)," 2020. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.docx/_jcr_content/renditions/white-paper-c11-741490_0.png

[2] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-down Approach*. Pearson, 2017. [Online]. Available: https://books.google.com.ec/books?id=OljpOAAACAAJ

[3] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated: The Protocols*, ser. Addison-Wesley professional computing series. Addison-Wesley, 1994. [Online]. Available: https://books.google.com.ec/books?id=-btNds68w84C

[4] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in *Proceedings - 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*. IEEE, 2008, pp. 5–13.

[5] P. Ludeña, "Fairness and Proactive Congestion Control in Multipath Networks," Ph.D. dissertation, Universidad Politécnica de Madrid, 2021.

[6] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, 2010. [Online]. Available: https://doi.org/10.1145/1851275.1851192

[7] K. R. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno and SACK TCP," *Comput. Commun. Rev.*, vol. 26, pp. 5–21, 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:7459148

[8] P. Ludeña-González, J. L. López-Presa, and F. D. Muñoz, "Upward Max-Min Fairness in Multipath High-Speed Networks," *IEEE Access*, 2021.

[9] N. Li, Z. Deng, Q. Zhu, and Q. Du, "AdaBoost-TCP: A Machine Learning-Based Congestion Control Method for Satellite Networks," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 2019, pp. 1126–1129.

[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: https://books.google.com.ec/books?id=IUZdAAAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[11] M. Bahnasy, H. Elbiaze, and B. Boughzala, "Zero-Queue Ethernet Congestion Control Protocol Based on Available Bandwidth Estimation," *Journal of Network and Computer Applications*, vol. 105, pp. 1–20, 2018.

[12] R. Adams, "Active Queue Management: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[13] S. Varma, *Internet Congestion Control*. Elsevier Science, 2015. [Online]. Available: https://books.google.com.ec/books?id=gbPoBgAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[14] S. Hu, W. Bai, B. Qiao, K. Chen, and K. Tan, "Augmenting Proactive Congestion Control with AEOLUs," in *ACM International Conference Proceeding Series*, 2018, pp. 22–28.

[15] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. Springer US, 2009.

[16] D. Giannopoulos, N. Chrysos, E. Mageiropoulos, G. Vardas, L. Tzanakis, and M. Katevenis, "Accurate Congestion Control for RDMA Transfers," *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, 2018.

[17] M. Bahnasy and H. Elbiaze, "Fair Congestion Control Protocol for Data Center Bridging," *IEEE Systems Journal*, vol. 13, no. 4, pp. 4134–4145, 2019.

[18] I. Cho, K. Jang, and D. Han, "Credit-Scheduled Delay-bounded Congestion Control for Datacenters," in *SIGCOMM 2017 - Proceedings of the 2017 Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 239–252.

[19] M. R. Kanagarathinam, S. Singh, I. Sandeep, A. Roy, and N. Saxena, "D-TCP: Dynamic TCP Congestion Control Algorithm for Next Generation Mobile Networks," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2018, pp. 1–6.

[20] T. M. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions. McGraw-Hill, 1997.

[21] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X

[22] C. D. Maciel and C. M. Ritter, "TCP/IP Networking in Process Control Plants," *Computers & Industrial Engineering*, vol. 35, no. 3, pp. 611–614, 1998. [Online]. Available: https://doi.org/10.1016/S0360-8352(98)00171-5

[23] 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, 2004.

[24] A. Mozo, J. L. López-Presa, and A. Fernández Anta, "A distributed and Quiescent Max-Min Fair Algorithm for Network Congestion Control," *Expert Systems with Applications*, vol. 91, pp. 492–512, 2018.

[25] J. R. Carrion Torres, "Aplicabilidad de Funciones de Utilidad Para el Control de Congestión en Redes de Computadores," Master's thesis, Universidad Técnica Particular de Loja, Loja, 2020.
[26] R. Al-Saadi, G. Armitage, J. But, and P. Branch, "A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3609–3638, 2019.
[27] M. Welzl, *Network Congestion Control: Managing Internet Traffic*. J. Wiley, 2005.
[28] J. Jin, W.-H. Wang, and M. Palaniswami, "Utility Max–Min Fair Resource Allocation for Communication Networks with Multipath Routing," *Computer Communications*, vol. 32, no. 17, pp. 1802–1809, 2009. [Online]. Available: https://doi.org/10.1016/j.comcom.2009.06.014
[29] L. Chen, B. Wang, X. Chen, X. Zhang, and D. Yang, *Utility-Based Resource Allocation for Mixed Traffic in Wireless Networks*. Institute of Electrical and Electronics Engineers, 2011.
[30] S. Li, Y. Zhang, Y. Wang, and W. Sun, "Utility Optimization-Based Bandwidth Allocation for Elastic and Inelastic Services in Peer-to-Peer Networks," *International Journal of Applied Mathematics and Computer Science*, vol. 29, no. 1, pp. 111–123, 2019.
[31] Q. V. Pham and W. J. Hwang, "Network Utility Maximization-Based Congestion Control over Wireless Networks: A Survey and Potential Directives," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 1173–1200, 2017.
[32] M. Chiang, "Distributed Network Control Through Sum Product Algorithm on Graphs," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 3, 2002, pp. 2395–2399 vol.3.
[33] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Prentice Hall, 2011.
[34] C. Demichelis and P. Chimento. (2002) IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). Network Working Group, RFC 3393.
[35] C. Gini, *Variabilità e mutabilità*, 1912, vol. 5, no. 20.
[36] M. O. Lorenz, "Methods of Measuring the Concentration of Wealth," *Publications of the American Statistical Association*, vol. 9, no. 70, pp. 209–219, 1905.
[37] G. F. Riley and T. R. Henderson, "The NS-3 Network Simulator." in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Günes, and J. Gross, Eds. Springer, 2010, pp. 15–34. [Online]. Available: http://dblp.uni-trier.de/db/books/collections/Wehrle2010.html#RileyH10