Nuwa: A Receiver-driven Congestion Control Framework to Achieve High-throughput and Controlled Delay over Dynamic Wireless Networks

Guanghui Gong, Xianliang Jiang, Yi Xie, Guang Jin, Jianan Zhang, Haiming Chen
Ningbo University
Ningbo, Zhejiang, China
{2011082292, jiangxianliang}@nbu.edu.cn

***Abstract*—In recent years, wireless networks and applications have grown by leaps and bounds, and are converging over a wide range of scenarios. An increasing number of applications require wireless networks that enable high throughput and low latency. However, due to the attenuated propagation of wireless signals, bandwidth change rapidly in a short period. TCP fails to perform well in such an environment and can suffer from low network link utilization and high latency. To solve above problems, this paper proposes a receiver-driven congestion control framework, named Nuwa. Nuwa decouples the congestion avoidance phase of sender-side congestion control and implements it on the receiver side. In addition, Nuwa uses one-way delay to detect network congestion and controls the sending rate of senders through the receive window field in the packet header. We confirm that the throughput degradation caused by network flips can be mitigated by Nuwa. And the throughput of data transmission can be further improved by the design of the receiver's algorithm. The evaluation results show that Nuwa improves the throughput of TCP stream by 10%-23% in most cases and reduces the queuing delay by 29% on average.**

***Index Terms*—TCP, receiver-side, congestion control**

I. Introduction

With the deployment of state-of-the-art network architectures (e.g., 5G cellular networks), the link capacity of networks has been significantly increased [1]. At the same time, Internet-related industries (such as Internet accelerated speed, the integration of new-generation information technology and manufacturing technology in "Made in China 2025", large-scale 5G infrastructure, and Tiansuan Constellation), and the rise of the metaverse, making high demands on users for wireless networks. Achieve high throughput and low latency of data transmission in wireless networks is crucial for the new technologies. However, existing wireless network architectures are inadequate to handle large amounts of data transmission, which can cause serious network congestion problems and degradation of throughput. [2].

Congestion control is critical to TCP data transmission and the primary solution to network congestion. The purpose of congestion control is to match the transmission rate of packets to the available bottleneck bandwidth in the wide-area network. Previously, congestion control mainly used packet delivery rates in the network to infer the channel capacity of wireless networks. Unfortunately, these methods are only effective in specific scenarios, which leads to the fact that traditional congestion control algorithms cannot achieve high throughput and low latency under wireless networks.

For wireless networks, the last hop is most likely to be the bottleneck for data transmission. In particular, the advent of Content Delivery Network (CDN) technology has enabled content providers to place their resources on operators' servers to reduce latency and improve the quality of user experience [3]. This makes data transmission performance dependent on the calculation of last-hop network bandwidth.

However, the vulnerability of wireless transmission to environmental influences causes its bandwidth to be always changing. Variations in wireless bandwidth are affected by a variety of factors (e.g., user location and speed), and anomalies in network metrics can easily lead to rapid drops in throughput. Therefore, sending large amounts of data without awareness of changes in the wireless network may lead to network congestion. We also found that a control message from the receiver side can quickly respond to the change on network state without over-simulating the network. So we propose Nuwa, a receiver-driven congestion control framework. Nuwa decouples the Congestion Avoidance (CA) phase of sender-side congestion control and implements it on the receiver side. We implemented the first receiver-driven algorithm in the Nuwa framework. One-way delay is used to detect network congestion and control the sending rate of sende through the receive window field in the packet header. Network changes are sensed by the receiver side to alleviate network congestion and achieve high throughput and low latency.

We conducted experiments in 5G cellular networks. The results show that Nuwa can accurately estimate the queuing delay of wireless networks. With the adjustment of the utility function, Nuwa can adjust the congestion window in real-time to occupy fully 100% of the available bandwidth. In a real wireless network, Nuwa can effectively improve the performance of data transmission. We set exclusive latency targets for different network environments, enabling Nuwa to share bandwidth with various congestion control protocols in the target network. Specifically, Nuwa can effectively track bandwidth changes under wireless networks and prevent additional retransmissions by adjusting the receive window.

In summary, the main contributions of this paper are as follow:

- In order to achieve high throughput and low latency, we develop a Nuwa framework. Nuwa changes the size of the congestion window by sensing the network bandwidth through one-way delays. This effectively mitigates the problems of throughput degradation and packet loss under wireless networks.
- Nuwa sets different target time delays for each application to meet their respective requirements. The setting of the time delay determines Nuwa's ability to occupy bandwidth. Therefore, setting a reasonable target delay helps Nuwa to get a fair bandwidth. And this approach also provides a solution to the problem of the delay-based approach suffers significant throughput degradation when competing with loss-based algorithms.

The rest of this paper is structured as follows. Section II presents the related work. In Section III, we present the background and motivation for the design of the receiver algorithm. Section IV describes our receiver algorithm in detail. Section V evaluates the scheme. We conclude in Section VI.

II. Related Works

Server-side oriented congestion control protocols: Most of the data in the network environment is transmitted base on TCP, so congestion control of TCP has been one of the research topics of great interest. Congestion control algorithms are critical to the performance of network data transmission. Based on different congestion feedback, server-side congestion control algorithms can be classified into three categories: loss-based algorithms, delay-based algorithms, and delay-based combined loss algorithms.

TCP Reno [4], and TCP NewReno [5] were among the early approaches that were loss-based congestion control algorithms. HSTCP [6] and CUBIC [7] modify the window growth model to achieve high network utilization quickly. CUBIC is the default congestion control algorithm in the Linux kernel.

Delay-based protocols (e.g., TCPVegas [8]) detect network congestion and adjust cwnd based on RTT, which can react to network conditions faster than packet-drop-based algorithms. However, the delay-based approach suffers significant throughput degradation when competing with loss-based algorithms such as TCP-Reno.

In addition, CTCP [9] combines delay-based components into a loss-based TCP congestion avoidance algorithm. TCP BBR [10] estimates the bottleneck bandwidth and RTT latency and uses distributed control loops to achieve an optimal state that exploits the network while adequately maintaining small queues. Recently, some new algorithms have also been proposed, such as LEDBAT [11]. However, these mechanisms are mainly designed for wired networks and are not applicable to highly variable cellular networks.

Receiver-oriented congestion control protocols: The receiver-based congestion control algorithm is mainly used to solve the TCP performance problem by tuning the reception window. In DRWA [13], the receiver increases rwnd as the current RTT approaches the minimum RTT, and increases rwnd as the RTT becomes more extensive, bringing the RTT closer to its minimum RTT. in DFCSD [15], the application's need for high throughput and low latency is addressed by controlling the cellular network's RTAC [14] integrates AQM into the loss-based congestion algorithm and implements it at the receiver side.

III. Motivation

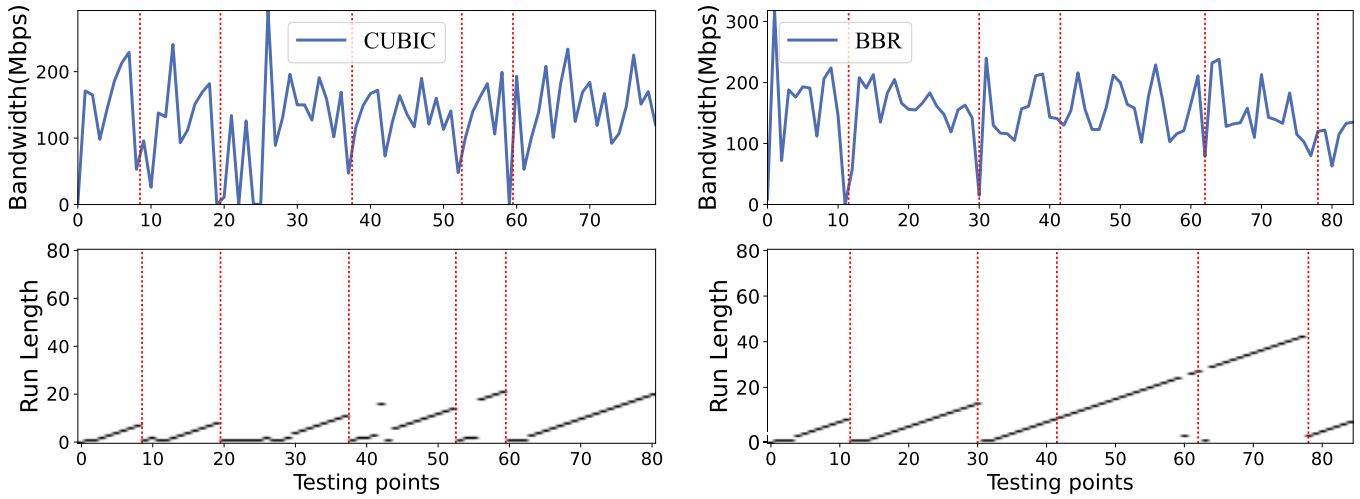
In this section, we first discuss the development of 5G wireless network technology. Then the reasons why wireless networks do not provide satisfactory performance are explored.

A. Base station switching in 5G

Enhanced Mobile Broadband (eMBB) is one of the 5G application scenarios defined by the International Telecommunication Union (ITU). It focuses on the explosive growth of mobile Internet traffic and provides users with more stable services in poorer network environments. The 5G performance metrics under eMBB are more diverse to supply the demands of different application scenarios. Peak rates of 10-20 Gbit/s are expected to be needed for 5G to accommodate data transmissions such as HD video and virtual reality. And air interface latency is as low as 1ms to accommodate real-time applications such as autonomous driving and telemedicine. However, the limited coverage of wireless technology allows 5G and other wireless communication technologies still have performance issues, such as reduced throughput during base station switching.

To address these issues, 5G technology proposes Dual Active Protocol Stack Switching (DAPS). The emergence of DAPS mainly solves the communication interruption problem during switching. In 4G networks, when a UE switches to connect to a base station, it disconnects from the old base station before connecting to the new one, which causes a link disruption of about 200ms. DAPS adopts a new base station switching method, which will maintain the link and data communication with the old base station before switching to the new one, thus solving the problem of communication interruption during base station switching.

Base station switching is the result of the combined effect of various factors (e.g., signal fading, surge of connected users). DAPS is proposed to solve the problem of throughput degradation during base station switching. However, when the base station switching is caused by signal quality degradation, the throughput of data transmission is also reduced. It is because the quality of 5G channel is varied with the different equipment locations. And data cannot be adjusted timely by the congestion control algorithm due to the rapid change of network. Therefore, we propose to focus the problem on how to address the impact of signal quality changes on data transmission in wireless networks. In the next subsection, we will discuss the impact of network signal quality fluctuations on data transmission and present the possibility of using receiver solutions to solve the problem.



(a) Trace in CUBIC

(b) Trace in BBR

Fig. 1. The relationship between network fluctuations and base station switching is detected from the collected bandwidth variations. The trace is segmented using a Bayesian online change point detection technique. The bottom plot shows the current run length (or count of monitored points since the last change point) for each point in the network trace. Run lengths are reset when there are significant changes in the trace.

B. Limitations of TCP in 5G scenario

This subsection exposes the problem of TCP and mobile wireless networks through file download experiments. The experimental topology used in our experiments is shown in Figure 2. The user equipment (UE) is connected to a 5G base station through a wireless mobile radio network. The base station network is connected to the WAN through an associated node. The server is deployed in the cloud with a base bandwidth of 5 Mbps. The UE downloads files from the server located on the wired side, and the cloud files are large enough to accommodate 80 seconds of file downloads. The default buffer size of the base station is 50 packets. Our experiments were conducted on a sealed road with six base stations deployed around this road. The tester is constantly moving on the road to cause multiple base station switching in a short period of time.

Fig. 2. The topology of UE Experiment.

To demonstrate the impact of network bandwidth variation on data transmission, we count packet loss, bandwidth variation and base station switching information when network packets are transmitted under 5G networks. Considering the impact of congestion control algorithms, we used two classical congestion control algorithms, BBR and CUIBIC, for comparison. The relationship between network bandwidth variation and base station switching is experimentally explored.

Oboe [12] is a video block acquisition scheme based on network state identification that proposes piecewise-stationary model for network throughput. Oboe shows that the traces of network throughput can be segmented into pieces of time stationary series and uses them for network state modeling. We use this scheme for the analysis of wireless network performance. Meanwhile, we use the Bayesian change point detection from Oboe to analyze the traces of bandwidth.

As shown in Figure 1, we show the split of the throughput traces for the CUBIC and BBR. The top half of the image shows the bandwidth variation, and the red line indicates the base station switching. The bottom half of the image shows the run length from the last change point, where the values represent the number of monitoring points. The results show that the trace under the wireless network is a non-smooth segmented trajectory. There is a positive correlation between the switching of the base station and the throughput. However, different results are found in the trace of the BBR algorithm. In Figure 1(b), the effect of BBR is not significant during the third and fourth base station switching.

We believe that network fluctuations become the main factor affecting data transmission under 5G networks. Data transmission is related to algorithms, and the impact of signal quality is more significant for algorithms based on packet loss, such as the CUBIC. Meanwhile, BBR can better handle the throughput degradation in some specific situations. So, we believe that the algorithm improvement can effectively avoid the data transmission fading problem in 5G era.

IV. The Proposed Nuwa Algorithm

This section discusses the problem of adapting the receive window in the current Linux kernel, and discusses the feasibility of the Nowa.

A. RWND Adjustment in Linux Kernel

The receive window in the TCP protocol is designed to control the sending rate of data to prevent it from exceeding the limited buffer space of the receiver. It reflects the amount of data that the receiver can handle so that the sender will not send packets more than receiver can accommodate. This is also known as TCP flow control, and it is different from TCP congestion control, whose goal is to prevent network overload. The transmit window of the network is subject to the effect of the receive window and the congestion window, the transmit window is the minimum of them.

With the development of storage technology, it is common for mobile devices to be loaded with large caches. Therefore, the current cache space at the receiving side is not a bottleneck for the data transmission. To improve the efficiency of the transmission, a technique for automatic adjustment of the receive buffer, dynamic adjustment (DRS) [16], has been proposed. In DRS, by dynamically adjusting the size of the receive buffer to fit the requirements of the connection, DRS estimates the size of the sender's send window in each RTT and then sets the local receive window to twice than the send window. This prevents TCP data transfers from becoming smaller due to receive window limitations.

Linux has integrated DRS into the kernel since 2.4.27 and uses this technology as a receive buffer auto-tuning scheme. With the continuous upgrading of the Linux kernel, the receive window and receive buffer size are now dynamically resized most of the time. However, this adjustment is unidirectional: DRS increases the receive window only when it is possible to limit the growth of the congestion window [13].

B. Nuwa Design

Network flips cause a drop in throughput during transmission because the Linux kernel does not use an appropriate policy for congestion windows. So, in order to make the change of congestion window consistent with the user network state, we propose a receiver-driven congestion control based framework, named Nuwa. The architecture of Nuwa's design is shown in Fig. 3, which consists of two main parts: 1- Receiver part and 2- Sender part. The receiver part mainly includes *Queue Detector*, *Fitting Trend*, and *Action Enforce*, where the receiver part of the algorithm can be replaced by the dynamic loading module of the kernel. The sender part uses the ack header information to assign a value to the congestion window. We discuss the algorithm in the context of a typical wireless access network-based scenario, where users connect to the Internet over a 5G network.

Nuwa calculates the one-way queue delay of a link in *Queue Detector*. The one-way queue delay indicates the congestion in the link, which allows Nuwa to get a sense of the network. Since this value is not directly available, we use Eq. 1 to calculate the one-way queuing delay.

$$\mathrm{D}_c = (S_t - S_m) - (R_t - R_m) \tag{1}$$

Where S_t is the current packet send time (value of timestamp), R_t is the current packet receive the time. S_m and R_m are the send time and receive time of the packet with minimum one-way delay (receive time minus send time) in the sliding window respectively.D_c is used to ensure that the algorithm has an accurate picture of the state of the network. We use Kalman Filter to process D_c to remove noise and interference from the data.

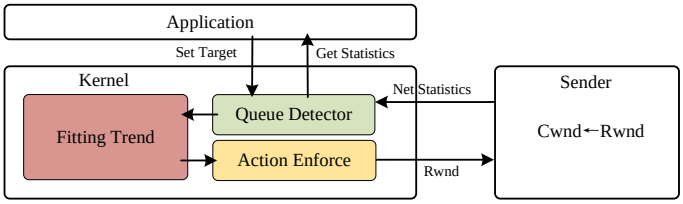


Fig. 3. A picture of Nuwa's design.

After sensing network change, Nuwa uses *Fitting Trend* to perform calculations on the trend of the window changes. In Nuwa, we set different target optimized delay T_d for different applications to achieve better data transmission. Thus, T_d-D_c represents the direction of the window change and the magnitude of the adjustment.

To improve the flexibility of the algorithm to network changes, we use the $tanh$ function to calculate the size of rwnd, and we set θ to $tanh(x)$. In which, we use $\mathrm{x} = \frac{T_d - D_c}{\rho}$ as the input to $tanh$. ρ is the sensitivity coefficient of the queuing delay fluctuation, which is used to map the experimental network variation into the $tanh$ function to represent the network variation. The $tanh$ equation is shown in Eq. 2, and the trend of $tanh$ is shown in Fig. 4. When x gets closer to 0, the function changes by a larger amount. Therefore, when the network fluctuates, $tanh$ can quickly change the window.

$$\tanh(x) = \left(e^{x} - e^{-x}\right) / \left(e^{x} + e^{-x}\right) \tag{2}$$

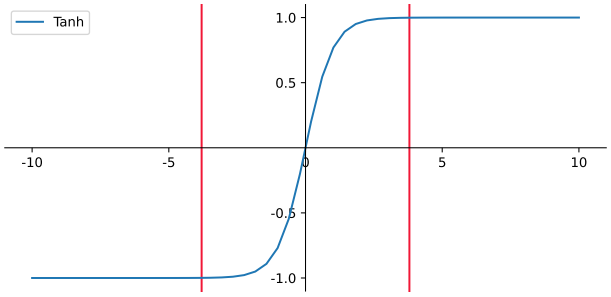


Fig. 4. The change of tanh function.

Finally, we determine the size of the final rwnd in the *Action Enforce* module, and the formula is shown in Eq. 3.

$$w_{new} \leftarrow w_{old} + (\theta \cdot k)/w_{old} \tag{3}$$

Where k is the window tuning aggressiveness parameter, which is used to modify the magnitude of the adjustment to the window. The last window size is the w_{old}. We use w_{old} as a parameter so that the window does not change too much in a single adjustment.

C. Nuwa Algorithm

Algorithm 1 The Nuwa Algorithm

1: $initial:$
2: $T_d \leftarrow 0, S_m \leftarrow 500\ (ms), R_m \leftarrow 0;$
3: $Delay_{min} \leftarrow a\ large\ value;$
4: $\rho \leftarrow a\ suitable\ value$
5: **for** each ACK **do**
6: $S_t \leftarrow Timestamp\ of\ packet\ delivery;$
7: $R_t \leftarrow Timestamp\ of\ packet\ receive;$
8: $D_c = (S_t - S_m) - (R_t - R_m)$
9: **if** $Delay_{min} = 0 || D_c < Delay_{min}$ **then**
10: $S_m \leftarrow S_t$
11: $R_m \leftarrow R_t$
12: $Delay_{min} \leftarrow D_c$
13: **end if**
14: x $\leftarrow \frac{T_d - D_c}{\rho}$
15: $\theta \leftarrow tanh(x)$
16: $w_{new} \leftarrow w_{old} + (\theta \cdot k)/w_{old}$
17: **end for**

The details of the Nuwa algorithm are shown in Algorithm 1. Different systems have different clock frequencies, so we use statistics to calculate the current queuing delay. Timestamps of the send time and receive time of the currently received packets are kept by collecting information on each ack (lines 6-7). Also, we record the send time and receive time of packets with a small time difference (Lines 9-12). By default, the smallest timestamp means that there is no queuing delay or a slight queuing delay exists in the current link. Finally, the delay of the link is estimated using equation (1) (Line 8). When D_c is known, we take $T_d - D_c$ to indicate the current adjustment direction the network should take (Line 14) and use equation (3) to adjust the window (Lines 15-16).

These ideas are derived from delay-based congestion control algorithms but work better for two reasons. First, we designed Nuwa based on the idea of delay. Nuwa can be independent of other algorithms in wireless networks because the base station provides a separate buffer space for each user. In WiFi or other wireless network environments, Nuwa can maintain good bandwidth allocation fairness with other algorithms in natural network environments as long as T_d is set reasonably well. In addition, Nuwa is based on the traditional TCP architecture, and the traditional congestion control algorithm can be implemented with minor modifications; we only improve the congestion control algorithm in the CA phase, which can effectively avoid the typical throughput degradation during data transmission.

V. Evaluation Results

In this section, we compare the Nuwa with other congestion control algorithms. We first implement our algorithm on the CUBIC algorithm and compare the algorithm performance in a simulated environment. In addition, we also compare our scheme with other algorithms in a real environment.

A. Simulation Environment

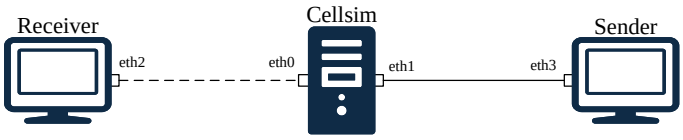


Fig. 5. Simulation topology.

In the simulation environment, we use the topology diagram shown in Fig. 5 for simulation experiments to verify. We use three sets of computers to build the experimental platform, Sender and Receiver use the same computer as the server and client of the experiment, and a desktop computer is used in the middle of the link as the running environment of Cellsim to simulate the network. In Cellsim, the traces collected under the real network are used as the experimental environment. The trace used is shown in Fig. 6. The trajectory contains mainly the link changes of the 5G network at two different moments.

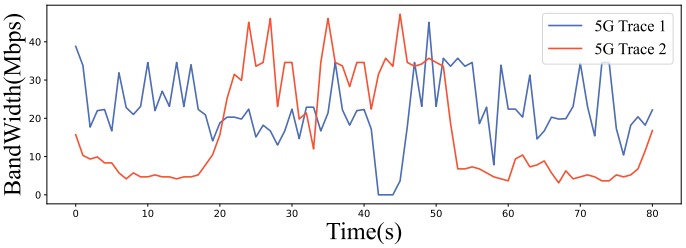
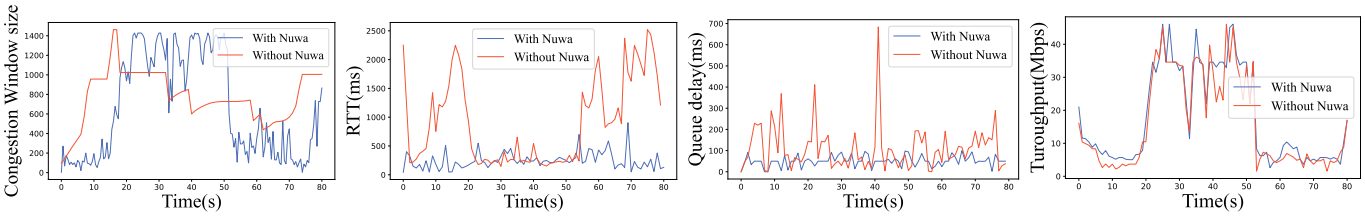


Fig. 6. Simulation Trace.

B. The Adaptive Nature of Nuwa

Firstly, we test the network adaptability of the algorithm. We judge the algorithm's adaptability to network changes from four perspectives: RTT, Throughput, send window variation, and queue delay. We conducted the experiments on a simulated experimental platform. For the experimental trace, we chose 5G Trace2 for the experiment, and the image of the trajectory change is shown in Fig. 6. The trace has a significant amount of network fluctuation environment. There are two relatively significant network bandwidth changes around 20 seconds and 55 seconds, and there are multiple network fluctuations in the trace between 20 and 55 seconds. On this trace, the algorithm can better show the adaptability to network changes. We perform multiple file transfer experiments on this track.

The Nuwa adaptation to network fluctuations is illustrated in Fig. 7. Due to the variability of network fluctuations, the packet loss-based algorithm CUBIC does not perform well in the network. Between 5 and 18 seconds, CUBIC misjudges the change in network bandwidth, and the congestion window continues to increase, leading to congestion in the network due to packet overload. This leads to an increase in RTT in the network and an increase in queueing delay, ultimately leading to a decrease in data throughput. In the network with the Nuwa, the control of the network link and the control of the network condition at the receiver side is better than the



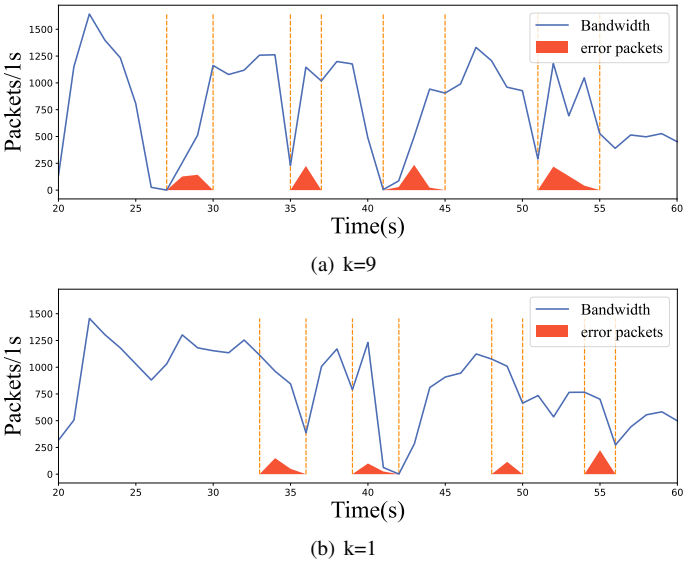
(a) Cwnd Size (b) RTT (c) Queue Delay (d) Throughput

Fig. 7. As Trace changes, Nuwa tracks the changes network so that RTT and one-way delay are in a smooth state while avoiding throughput loss.

traditional algorithm because it is implemented at the receiver side. In Fig. 7, we can see that the control of the window in the Nuwa is closer to the actual change of the network link. Moreover, Nuwa can send data using a better window so that the number of packets in the network can fit the network changes and reduce packet loss. Therefore, the throughput of Nuwa in Fig. 7(d) is higher than the throughput under the CUBIC algorithm.

C. The Impact of k

k is a key parameter in Nuwa to control the adjustment magnitude of the window. Packet loss occurs in the current experiment in two cases. First, packet loss occurs when the instantaneous bandwidth of the network suddenly becomes smaller, the link sending window is unable to sense the bandwidth reduction and sends more packets than the link can tolerate. There is also a case when the link bandwidth becomes wider, the congestion control algorithm causes the congestion window to be too large due to incorrect link calculation, resulting in too much data sent to generate packet loss. And the accuracy of the k-value directly leads to the data transmission efficiency. The experimental results for different k are shown in Fig. 8.



(a) k=9

(b) k=1

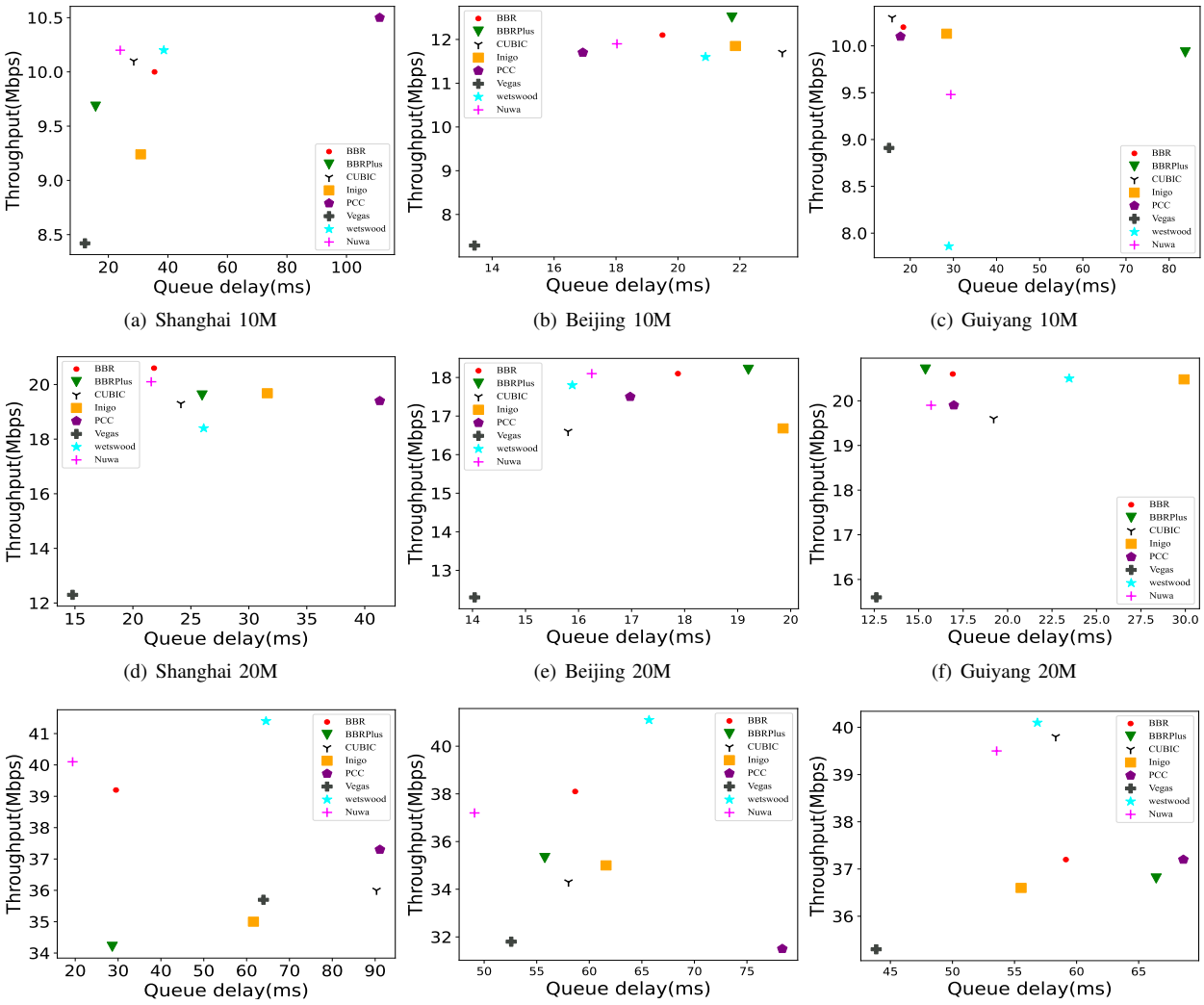
Fig. 8. The impact of k on the performance of TCP.

From Fig. 8, we find that as the value of k increases, the change in throughput varies with the network's fluctuation and becomes more sensitive. When network fluctuations occur, a significant drop in throughput occurs. At the same time, a smaller value of k will cause the change of the window to lag behind the change of the network, resulting in packet loss at the falling edge. The algorithm's k values range is [1-9]. After experiments, it is proved that a k value of 7 is most suitable for the 5G network. At $k = 7$ is our comparison of the Nuwa and other algorithms for data transmission experiments. We use our experiments' data transmission size, packet loss rate, and average queue length in 80 seconds under the same trace as metrics. The experimental results are shown in Table 1.

TABLE I
DATA TRANSMISSION UNDER 5G TRACE2.

Algorithm	***Throughput***	***Bit Error Rate***	***Average Queue Length***
BBR	168.5M	0.54%	144.93ms
BBRPlus	172.3M	0.61%	137.48ms
CUBIC	187.1M	0.87%	165.79ms
Inigo	177.5M	0.71%	146.3ms
PCC	185.3M	1.02%	177.5ms
Vegas	179.1M	0.82%	169.47ms
Westwood	177.8M	**0.51**%	134.74ms
Nuwa	**194.5M**	0.56%	**125.4ms**

We present the data transmission performance of different algorithms under 5G Trace1 in Table 1, where $k = 7$ and $T_d = 750$. We conducted several experiments and averaged the results. The table shows that Nuwa can maintain a high throughput under fluctuating wireless network traces. This is because Nuwa uses queueing delay as the indicator to determine the network state. The table shows that Nuwa has significantly higher control over the link than other algorithms. Compared to the BBR algorithm, Nuwa throughput improves by 15.4% and 4% over CUBIC. And it reduces up to 29% in the pair average queueing delay. We also found that the Westwood algorithm based on the wireless network design can obtain a better throughput in the wireless environment and effectively reduce the packet sending errors. Compared with other algorithms, CUBIC's packet loss-based concept makes it possible to obtain better data transmission capability even under wireless networks. Still, its network packet loss is severe, and there is a high network packet loss. Therefore, CUBIC's overall throughput is limited.



(a) Shanghai 10M (b) Beijing 10M (c) Guiyang 10M

(d) Shanghai 20M (e) Beijing 20M (f) Guiyang 20M

(g) Shanghai 50M (h) Beijing 50M (i) Guiyang 50M

Fig. 9. Nuwa's relationship between throughput and one-way queue latency under 5G network.

D. Nuwa under 5G network

We find that the correct estimation of one-way delay can effectively improve the throughput of data transmission. In this experiment, we compare Nuwa with other algorithms in 5G network to show the advantages of Nuwa in wireless networks. The topology of the experiment is shown in Figure 2. We used Nginx [17] to set up a web service on a cloud server and used it to provide file downloads. We deployed servers in different regions to verify the effect of experimental distance on the algorithm's throughput and to show Nuwa's control of queueing latency. Requests are sent from Ningbo, and server cities include Shanghai, Beijing, and Guiyang. We set the server bandwidth to 10M, 20M, and 50M in different regions to verify the impact of different bandwidths on the algorithm. The experimental results are shown in Figure 9.

The throughput does not differ significantly between algorithms when the bandwidth is small. As the bandwidth increases, the throughput gap between different algorithms starts to appear. The wireless design-based congestion control algorithms, such as Westwood, begin to gain more bandwidth as the bandwidth becomes larger. In contrast, packet loss-based algorithms are performing poorly in the experiments, such as Vegas and CUBIC. Due to the large number of buffers in the mobile network, these types of protocols constantly fill the buffer queue, and generating packet loss signals. This signal indicates that the network is congested and forces the sender to reduce its transmission rate. This type of processing creates considerable queuing delays and cycles, resulting in under-utilization of bandwidth. Detection-based algorithms, such as BBR, BBRPlus, and PCC, perform similarly under mobile networks. However, PCC does not perform as well as BBR and BBRplus at large bandwidths. The main reason is due to the fact that the PCC algorithm uses a different utility function to adjust the window. The lack of control over queuing delays by the function leads to many packet drops and retransmissions, allowing PCC to have much lower throughput than other congestion control algorithms. For Nuwa, queuing delay is

used as a criterion for link congestion. The main algorithm is implemented at the receiver side, making the changes in the Nuwa consistent with the user side. The performance of the effect allows Nuwa to maintain a low latency while with the high throughput. For the Inigo algorithm, which is also implemented on the receiver side, the control of throughput and queuing delay is worse than the Nuwa.

Performance of algorithm is relevant to transmission distance. The greater the distance, the more likely it will be that queues will form during transmission. Provided that the bandwidth is 50M, the data transmission efficiency of algorithms begin to vary. BBR algorithms can maintain data transmission in the east region, while their transmission capacities decline when the server is located in the southwest region. PCC, Vegas, and Inigo algorithms share this character. Among them, Westwood algorithm can still maintain high network throughput over long distance transmission. The CUBIC algorithm distinguishes itself from other algorithms based on packet loss for its throughput increase when the distance becomes longer. However, its queuing delay is more serious than other similar algorithms. Additionally, Nuwa keeps its overall throughput at a high level as a result of its excellent control on queuing delay.

E. Fairness

Fairness is also an important evaluation metric for congestion control algorithms. When the delay-based approach suffers significant throughput degradation when competing with loss-based algorithms. As a delay-based algorithm, to prevent the above situation, Nuwa sets different link optimization objectives T_d for different network scenarios. As shown in Eq. 2, the plus or minus of X determines the adding or reduction of windows. And the increase of T_d can boost the competitiveness of Nuwa in the network.

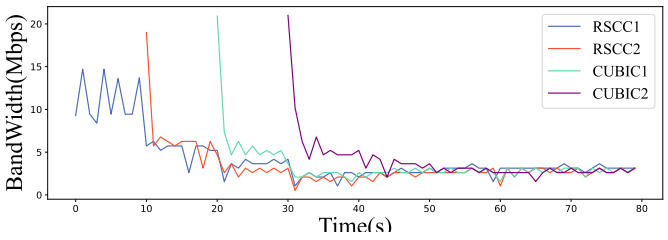


Fig. 10. Fairness experiment results.

We conducted the Nuwa fairness experiments in the experimental topology in Fig. 5. We chose a stable bandwidth of about 10M in CellSim. We used the iperf tool to analyze the bandwidth consumption of the algorithm. Since the CUBIC algorithm is still the default algorithm in Linux and it is widely used nowadays, we chose the CUBIC as the loss-based algorithm used in our experiments. We deployed the CUBIC and Nuwa on the Sender side. For the current network scenario, we adjust the Nuwa algorithm by setting T_d to 512 to resist the CUBIC algorithm. And we add one stream every 10 seconds: two streams of Nuwa and two streams of CUBIC are included.

The experimental results are shown in Fig. 10. A single Nuwa stream can occupy bandwidth quickly and share bandwidth when new Nuwa streams are added. Meanwhile, when CUBIC streams join, the overall occupied bandwidth is less than the Nuwa due to the CUBIC based on packet loss. Still, with the window adjustment, after 50 seconds, the Nuwa and CUBIC can share the bandwidth resources in the link.

VI. Conclution

We propose a new TCP framework named Nuwa, which is a receiver-driven congestion control framework. We implemented the Nuwa in Linux kernel, which can be replaced by a dynamic module loading mechanism. We use network link queuing delay as a network congestion metric and use the $tanh$ function to determine the network window adjustment range. We describe the algorithm and implementation of Nuwa in detail. Through extensive testing, we confirm the feasibility of the Nuwa reception scheme.

References

[1] Narayanan A, Zhang X, Zhu R, et al. A variegated look at 5G in the wild: performance, power, and QoE implications[C]//Proceedings of the 2021 ACM SIGCOMM 2021 Conference. 2021: 610-625.

[2] Jacobson V. Congestion avoidance and control[J]. ACM SIGCOMM computer communication review, 1988, 18(4): 314-329.

[3] Lee S, Joo C. The CDN Pricing Strategies in the Internet Traffic Delivery Chain[C]//2021 International Conference on Information Networking (ICOIN). IEEE, 2021: 680-682.

[4] Allman M, Paxson V, Blanton E. TCP congestion control[R]. 2009.

[5] Parvez N, Mahanti A, Williamson C. TCP NewReno: Slow-but-steady or impatient?[C]//2006 IEEE International Conference on Communications. IEEE, 2006, 2: 716-722.

[6] Zhu L, Ansari N, Liu J. Throughput of high-speed TCP in optical burst switching networks[J]. IEE Proceedings: Communications, 2005, 152(3): 349-352.

[7] Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendly high-speed TCP variant[J]. ACM SIGOPS operating systems review, 2008, 42(5): 64-74.

[8] Brakmo L S, O'Malley S W, Peterson L L. TCP Vegas: New techniques for congestion detection and avoidance[C]//Proceedings of the conference on Communications architectures, protocols and applications. 1994: 24-35.

[9] Kim M J, Cloud J, ParandehGheibi A, et al. Network coded tcp (ctcp)[J]. arXiv preprint arXiv:1212.2291, 2012.

[10] Cardwell N, Cheng Y, Gunn C S, et al. BBR: congestion-based congestion control[J]. Communications of the ACM, 2017, 60(2): 58-66.

[11] Shalunov S, Hazel G, Iyengar J, et al. Low extra delay background transport (LEDBAT)[R]. 2012.

[12] Zahaib Akhtar. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. ACM, 44–58.

[13] Jiang, H.; Wang, Y.; Lee, K.; Rhee, I. Tackling bufferbloat in 3G/4G networks. In Proceedings of the 2012 Internet Measurement Conference, Boston, MA, USA, 14–16 November 2012; pp. 329–342.

[14] Im, H.; Joo, C.; Lee, T.; Bahk, S.Receiver-side TCP countermeasure to bufferbloat in wireless access networks. IEEE Trans. Mob. Comput. 2016, 15, 2080–2093. [CrossRef]

[15] Dong P, Gao K, Xie J, et al. Receiver-side TCP countermeasure in cellular networks[J]. Sensors, 2019, 19(12): 2791.

[16] Feng W, Fisk M, Gardner M, et al. Dynamic right-sizing: An automated, lightweight, and scalable technique for enhancing grid performance[C]//International Workshop on Protocols for High Speed Networks. Springer, Berlin, Heidelberg, 2002: 69-83.

[17] Reese W. Nginx: the high-performance web server and reverse proxy[J]. Linux Journal, 2008, 2008(173): 2.