Survey of model and algorithms of host-to-host congestion control

Pengtao Kang[1], Tao Feng[1,*] and Xianming Gao[1]

[1] *Systems Engineering Institute AMS PLA, Beijing 100039, China*

Abstract

Host-to-host congestion control is an important technology that avoids network congestion and improves network transmission performance by controlling the transmission rate. However, it is difficult for host-to-host congestion control mechanisms to remain effective in dynamic networks due to factors such as imperfect feedback information, error-prone congestion detection, slow convergence control algorithm, etc. In order to provide a basis for the development of new mechanisms, this paper provides a comprehensive review of control models, congestion detection and control algorithms, as well as evaluation indicators and the main challenges associated with such networks. Furthermore, the paper discusses the potential of intelligent algorithms and edge-host cooperation in host-to-host congestion control.

Keywords

host-to-host congestion control, control model, congestion detection algorithm

1. Introduction

Communication networks have seen a revolutionary shift in recent decades, largely due to soaring data consumption and the widespread adoption of connected devices [1]. As the number of devices and data usage surge, network congestion is emerging as a critical concern once again. Moreover, applications like live broadcasting and gaming, which are increasingly popular, demand stringent requirements for low latency, high bandwidth, and stable host-to-host communication [2, 3]. Consequently, refining the host-to-host congestion control mechanism has become an urgent practical matter.

To furnish valuable references for researchers with a keen interest, we conduct a comprehensive review of host-to-host congestion control, focusing on model and algorithms. The primary contributions encompass three key aspects:

- In this paper we efficiently categorize control models based on their workflow, offering an in-depth analysis of each category's unique features.
- The congestion control process is methodically broken down into two phases: detection and control. By examining commonly used algorithms in these stages

ICCIC 2024: International Conference on Computer and Intelligent Control, June 29–30, 2024, Kuala Lumpur, Malaysia

* Corresponding Author

k1067311368@163.com (P. T. Kang); feng09@163.com (T. Feng); nudt_gxm@163.com (X. M. Gao)

0009-0004-6857-9351 (P. T. Kang); 0000-0001-9791-9694 (T. Feng); 0000-0002-4173-3740 (X. M. Gao)

through network feedback signals, the paper effectively compares their strengths and weaknesses.

- We highlight the main challenges in mechanism design, delving into how intelligent algorithm and edge computing technologies could address issues like hosts collaboration, feedback delay, and inaccuracy.

The structure of the paper unfolds as follows: Section 2 classifies and compares control models, while Section 3 provides a summary of congestion detection. Section 4 delves into the analysis of control algorithms, and Section 5 introduces the evaluation indicators. Section 6 elucidates the main challenges faced in mechanism design. Section 7 discusses potential future directions, and the paper concludes in Section 8.

2. Control model

Host-to-host congestion control models can be categorized based on the involvement of switches into implicit feedback model and explicit feedback model. The implicit feedback model operates without direct communication from switches, whereas the explicit feedback model relies on direct signals or data from switches to manage congestion. This distinction is vital for understanding the underlying mechanisms and effectiveness of each approach in different network environments.

2.1. Implicit feedback model

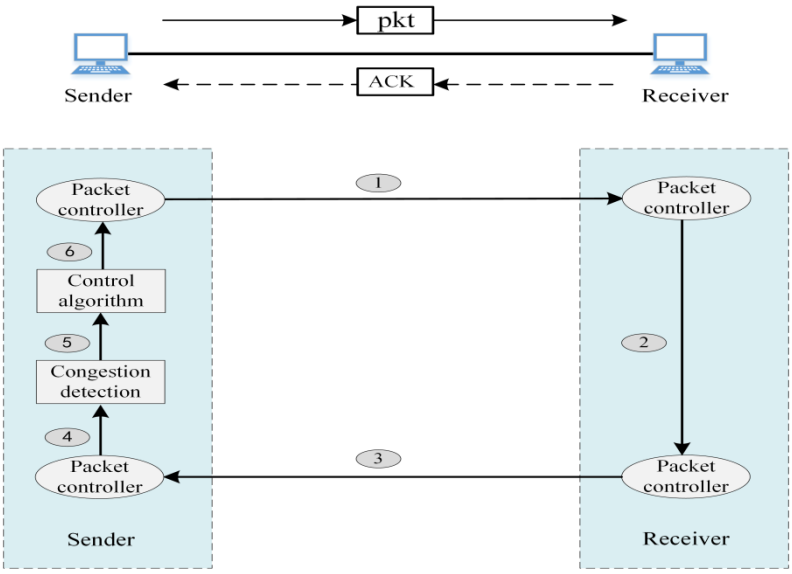

Figure 1: Implicit feedback model

The implicit feedback model is shown as Figure 1. This model is fundamentally composed of the sender and the receiver. The basic operation process can be outlined in the following steps:

1. The sender initiates the process by transmitting packets at a predetermined rate.
2. Upon receiving these packets, the receiver generates ACKs (acknowledge packets).
3. The receiver then sends these ACKs back to the sender.
4. The sender, upon receiving the ACKs, analyzes them to extract the network state.
5. Based on the results of this congestion detection, the sender implements a control algorithm.
6. The sender adjusts its packet sending rate according to the outputs of the control algorithm.

The emergence of the implicit feedback model can be traced back to post-1986 developments, driven by the early switches' limitations in providing network status information to terminals. This led to the development of various host-to-host congestion control mechanisms based on this model, such as Tahoe [4], TCP Reno [5, 6], TCP NewReno [7], TCP BBR [8], and referenced in [9-12].

2.2. Explicit feedback model

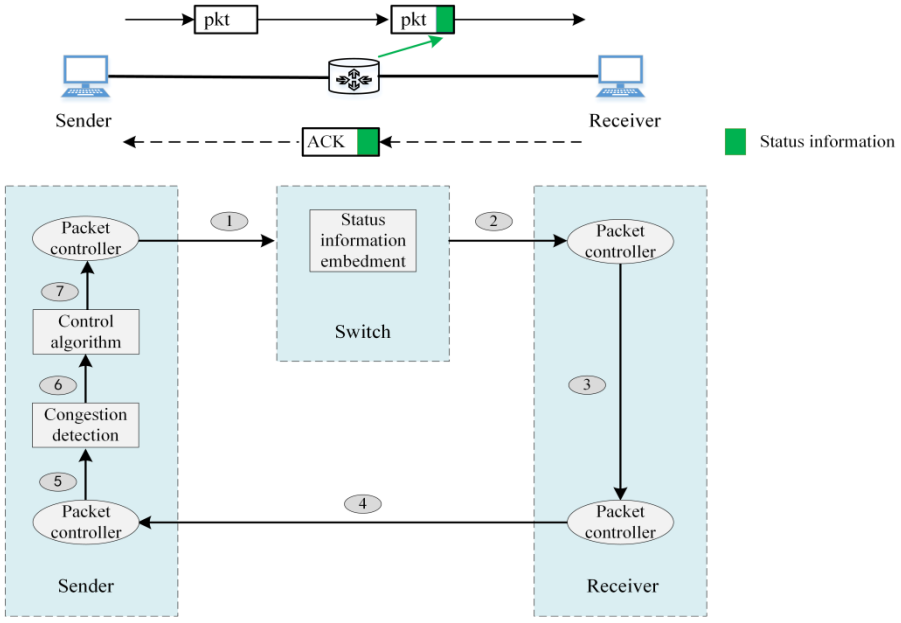

Figure 2: Explicit feedback model

As illustrated in Figure 2, the explicit feedback model is composed of three key components: the sender, the switch, and the receiver. The operation process unfolds as follows:

1. The sender starts by transmitting packets at a predetermined rate.
2. As packets pass through, the switch embeds network status information into them.
3. The receiver, upon receiving enhanced packets, adds network status information into ACKs.
4. The receiver then sends these ACKs back to the sender.
5. The sender analyzes the ACKs to extract network status information.

6. Based on the congestion detection, the sender implements a specific control algorithm.
7. The sender adjusts its transmission rate according to the results of the control algorithm.

The explicit feedback model gained prominence following the introduction of ECN (Explicit Congestion Notification) in 2001, as per RFC 3168 [13]. This model has been further developed and refined in various mechanisms, including DCQCN [11], HPCC [14] and referenced in [15-21].

2.3. Comparative analysis

Through the above analysis, the implicit feedback model is characterized by its straightforward structure, ease of implementation, and excellent scalability. However, it has a limitation in collecting detailed network feedback, such as queue size and link utilization.

The explicit feedback model enhances the implicit model by providing more detailed link information, such as queue size and link utilization, for a deeper network understanding. However, it has scalability and deployment challenges due to higher network device capability requirements.

Table 1
Control Model Comparison

Model	Features, Advantages and Disadvantages
Implicit Feedback model	No switch involvement; Ease of implementation; Excellent scalability; Limited range of network feedback.
Explicit Feedback model	Switch involvement; More detailed link information; Poor deployment and scalability.

3. Congestion Detection

3.1. Network status collection

Network status collection is crucial for detecting congestion effectively. Here's a summary of the five primary methods for collecting network status information:

1. Using Packet Transmission Status: This basic method uses sent packets and received ACKs to calculate RTT (round-trip time), infer available bandwidth from ACK rates, and detect packet loss from ACKs. It's simple, but limited in the data it can collect.
2. Using SDN Controller [22]: In SDN networks, the controller provides a full view, collecting data like queue size and delay from switches. For instance, TCCS mechanism [23] uses OpenFlow protocol to collect switch queue size for addressing TCP incast issues. However, larger networks and loads can affect the timeliness of data collection.

3. Using In-band Network Telemetry (INT) Technology [24]: The INT framework combines packet forwarding with network measurement for real-time, selective switch info collection. For example, HPCC [14] employs INT to gather bottleneck link utilization for precise control. The downside is increased communication bandwidth usage with link length.
4. Using the ECN Protocol [13]: ECN protocol facilitates active feedback collection from switches, like ECN marks. Mechanisms like DCTCP [25] use ECN-marked packets for active congestion control. Post-deployment dynamic control and flexibility are challenges of this method.
5. Using Cross-Layer Information: This approach involves collecting link status information from various layers during packet transmission. In cellular networks, methods like CQIC [26] and PBE [27] use physical layer data to measure link bandwidth and enhance performance. However, its applicability is limited to specific network types, affecting its versatility.

3.2. Congestion detection algorithms

Congestion detection algorithms categorize the network into congested and uncongested states using network status information. The normalized representation can be described as follows:

$$f_{congestion} = \begin{cases} true & condition1 \\ false & otherwise \end{cases} \tag{1}$$

The *condition1* denotes boundary conditions and its setting depends on the specific network feedback information. Congestion detection algorithms can be categorized into seven types based on the network status information they utilize.

3.2.1. Loss-Based detection algorithm

The Loss-Based detection algorithm uses packet loss as a key indicator of network congestion. Packet loss is typically identified by observing duplicate ACKs. Notable Loss-Based congestion detection mechanisms include TCP Tahoe [28], TCP Reno [29], and TCP NewReno [7]. The formula for Loss-Based congestion detection [28] is:

$$f_{congestion} = \begin{cases} true & N_{ACK_i} \geq 3 \\ false & otherwise \end{cases} \tag{2}$$

NACKi represents the number of times ACK_i.

3.2.2. RTT-Based detection algorithm

The queue size measures network congestion, and RTT mirrors this by reflecting the time from packet send to ACK receipt. RTT fluctuates due to network queuing, with its average (R_{vag}) rising with queue size. The RTT range widens with network load ρ (ratio of packet arrival to departure rate), given as $(-(1-\rho)-1\delta R_{avg}\ to\ (1-\rho)-1\delta R_{avg})$, where δ is a window factor. Thus, RTT serves to evaluate network congestion. The RTT is calculated using the following formula:

$$R(t_c) = t_c - t_s - \sigma \tag{3}$$

Here t_s denotes the sending time of packet i, t_c represents the time when the sender receives ACK_i, and σ stands for an error factor.

There exist two types of congestion detection algorithms based on RTT:

1. Instantaneous RTT-Based Congestion Detection

$$f_{congestion} = \begin{cases} true & R \geq T_{high} \\ false & otherwise \end{cases} \tag{4}$$

R represents the most recent measured value of RTT, and T_{high} is a threshold factor. There are congestion detection mechanisms grounded in instantaneous RTT, examples being TIMELY [10] and Swift [30].

2. Average RTT-Based Congestion Detection

$$R_S = \alpha R_S + (1-\alpha)R, \quad f_{congestion} = \begin{cases} true & R_S \geq T_{high} \\ false & otherwise \end{cases} \tag{5}$$

Similar to the instantaneous RTT approach, R_S denotes the estimated value of RTT [28], the variable α represents a smoothing factor. Schemes based on average RTT include Vegas [31] and FAST TCP [32], among others.

3.2.3. BDP-Based detection algorithm

In the congestion control pipeline model, Figure 3 illustrates the interrelationships among RTT, delivery rate, and the amount of data in flight (i.e., data sent but not yet acknowledged) [8]. The blue line represents the stage where the queue size is zero, the green line represents the stage where the queue size grows, and the red line represents the stage where the packets exceed the buffer capacity.

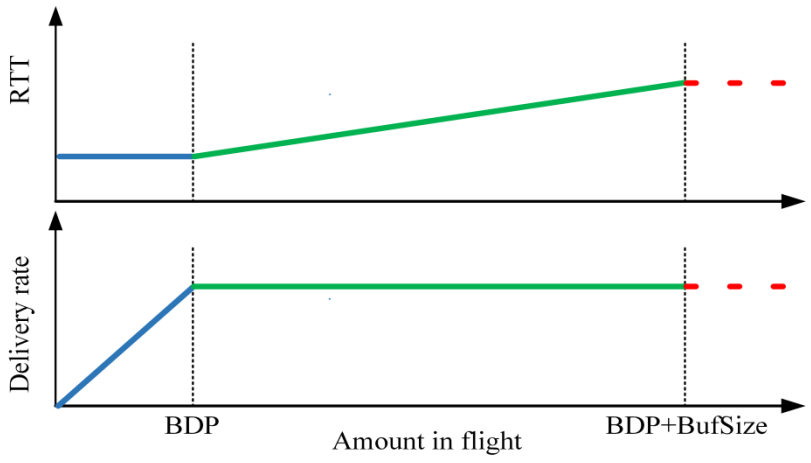

Figure 3: Delivery rate and RTT vs Inflight

The Figure 3 shows that the optimal network transmission performance occurs where the blue and green lines intersect, indicating the lowest RTT and highest delivery rate. In

the BDP-Based Detection Algorithm, the blue line's region signifies no network congestion, while other areas suggest congestion. The Bandwidth-Delay Product (BDP) is calculated as the product of the minimum RTT and the maximum delivery rate, using statistical data for these values. The formula for the minimum RTT [8] is given by:

$$R_{\alpha}(t) = min(R(t)), \quad \forall t \in [t - W_R, t] \tag{6}$$

$R(t)$ denotes the latest measured RTT, and W_R represents a window factor.

The formula for calculating the maximum delivery rate [8] is:

$$B(t) = max(R_d(t)), \quad \forall t \in [t - W_B, t] \tag{7}$$

$R_d(t) = N_d/\Delta t$, where N_d is the number of ACKs received during Δt. W_B is a window factor.

In accordance with the definition of BDP, the BDP-Based congestion detection formula is:

$$f_{congestion} = \begin{cases} true & N_{flight} \geq R_{\alpha} \cdot B \\ false & otherwise \end{cases} \tag{8}$$

N_{flight} represents the amount of data in flight, R_a is the minimum RTT, and B is the maximum delivery rate. BDP-Based congestion detection mechanisms, such as BBR [8] and BBR-ACD [33], leverage these principles for effective congestion control.

3.2.4. ECN-Based detection algorithm

ECN is a mechanism that uses the last two bits of the DSCP (Differentiated Services Code Point) field in the IP header to signal congestion. When a switch's queue size exceeds a set threshold, it marks subsequent packets with ECN. Upon receiving an ECN-marked packet, the receiver generates a Congestion Notification Packet (CNP) and forwards it to the sender. This process allows the detection of network congestion, and the formula is as follows:

$$f_{congestion} = \begin{cases} true & getCNP = true \\ false & otherwise \end{cases} \tag{9}$$

ECN-Based congestion detection mechanisms, such as DCTCP [25] and D2TCP [34], utilize these principles to effectively detect and respond to network congestion.

3.2.5. Link-Utilization-Based detection algorithm

In the context of network congestion, link utilization serves as a direct indicator of the congestion level. The congestion detection formula based on link utilization is expressed as:

$$f_{congestion} = \begin{cases} true & R_{sum}/C \geq \varphi \\ false & otherwise \end{cases} \tag{10}$$

R_{sum} represents the total traffic of the bottleneck link, C represents the bandwidth of the bottleneck link, and φ is a threshold factor. Link-Utilization-Based congestion detection mechanisms, such as HPCC [14] and EagerCC [35], leverage these principles to effectively identify and respond to network congestion based on link utilization metrics.

3.2.6. Equation-Based detection algorithm

Equation-based congestion detection evaluates network state by constructing a utility function. The formula based on equations is expressed as:

$$f_{congestion} = \begin{cases} true & F(x_1, x_2, ..., x_N) \geq T_{high} \\ false & otherwise \end{cases} \quad (11)$$

F represents the utility function. x_1, x_2 *and* x_N represent network status. T_{high} is a threshold factor. Equation-based congestion detection mechanisms allow for customizable design to fit specific requirements. For instance, the network performance-oriented congestion control framework [36] assesses network status by constructing a utility function based on throughput, loss rate, and delay.

3.2.7. Composite-Approach-Based detection algorithm

Practical congestion control mechanisms often employ a combination of algorithms. The congestion detection formula based on composite approach is expressed as:

$$f_{congestion} = \begin{cases} true & f_1 = true \quad or \quad f_2 = true \\ false & otherwise \end{cases} \quad (12)$$

f_1 and f_2 represent the results of certain detection algorithms. Composite-Approach-Based Detection mechanisms leverage these principles to combining the advantages of multiple detection methods. For example, TCP Africa [37] integrates both loss-based and RTT-based algorithms for effective congestion detection.

3.2.8. Comparative analysis

- Loss-Based Detection Algorithm: The Loss-Based detection algorithm operates on a simple principle and can be implemented in the implicit feedback model. It effectively determines network congestion, serving as the fundamental detection algorithm for TCP. However, it lacks the ability to differentiate between packet loss due to damage and congestion, performing poorly in high-error-rate networks like wireless networks. It exhibits low sensitivity in low congestion levels, leading to potential buffer overflow issues.
- RTT-Based Detection Algorithm: The RTT-Based detection algorithm is implementable in the implicit feedback model, allowing the adaptation of congestion detection thresholds to different environments. Yet, its accuracy is influenced by the randomness of RTT, and it suffers from a feedback time delay problem. As congestion intensifies, the RTT feedback duration increases, resulting in delayed congestion control.
- BDP-Based Detection Algorithm: Implementable in the implicit feedback model, the BDP-Based detection algorithm is insensitive to packet loss, avoiding misjudgments caused by damage-related losses in the Loss-Based detection. It effectively determines worsening congestion based on RTT sensitivity. However, it heavily

relies on RTT and has a long detection period, leading to poor performance in highly dynamic environments, such as wireless networks.

- ECN-Based and Link-Utilization-Based Detection Algorithms: Both ECN-Based and Link-Utilization-Based detection algorithms can only be implemented in the explicit feedback model. They accurately determine network congestion and proactively control congestion through factor settings. However, they require network devices to support corresponding protocols, resulting in poor scalability.
- Equation-Based and Composite-Approach-Based Detection Algorithms: The Equation-Based detection algorithm establishes a mapping between network status information and congestion using complex functions, offering fine granularity. The Composite-Approach-Based detection algorithm combines the advantages of multiple detection methods to improve adaptability. However, these algorithms have high complexity, and their performance is challenging to guarantee in dynamic environments.

Table 2
Comparison of Major Congestion Detection Algorithms

Algorithm	Advantages	Disadvantages
Loss-Based	Simple structure.	Misjudgment during packet loss; Buffer overflow problems.
RTT-Based	Simple structure; Adaptive congestion thresholds.	Limited accuracy due to delay; Feedback delay problem.
BDP-Based	Simple structure; Avoiding misjudgments from Loss-Based.	Heavy reliance on RTT; Long detection period; Poor performance in dynamic environments.
ECN-Based and Link-Utilization-Based	Accurate congestion detection; Active congestion control.	Complex structure; Poor scalability.
Equation-Based	Flexible design approach; Fine granularity.	High complexity; Unstable performance in dynamic environments.
Composite-Approach-Based	Combining the advantages of multiple approaches.	High complexity; Unstable performance in dynamic environments.

4. Control Algorithm

The control algorithm computes the new sending rate based on congestion detection. The basic strategy is to decrease the sending rate when congestion occurs and increase the sending rate when there is no congestion. Its normalized representation is

$$R_{new} = \begin{cases} R_C \cdot A - B & f_{congestion} = true \\ R_C \cdot C + D & f_{congestion} = false \end{cases}, \tag{13}$$

R_{new} represents the new sending rate. R_c represents the current sending rate. A, B, C and D are the control parameters whose values depend on the network feedback information. It should be noted that there is a corresponding linear relationship between the sending rate and the congestion window, and that adjusting the congestion window can be converted equivalently to adjusting the sending rate. According to the network feedback information and calculation rules, six typical control algorithms are selected to be analysed in this section.

4.1. Loss-Based control algorithm: TCP Reno [38]

4.1.1. Overview of the algorithm

The TCP Reno algorithm encompasses several key components, including the slow start, congestion avoidance, timeout retransmit, fast retransmit, and fast recovery algorithms.

1. Slow start algorithm

When the congestion window (*cwnd*) is less than the slow-start threshold (*ssthtesh*), and a new ACK is received, *cwnd* is incremented by 1 (in MSS, Maximum Segment Size units).

2. Congestion avoidance algorithm

When *cwnd* ≥ ssthtesh, upon receiving a new ACK, *cwnd=cwnd+1/cwnd*.

3. Timeout retransmission algorithm

If *RTT>RTO* (the retransmission timeout), *ssthtesh=max(2, cwnd/2)*, and *cwnd=1*, entering the slow-start state.

4. Fast retransmit algorithm

Upon receiving 3 duplicate ACKs, *ssthtesh=max(2, cwnd/2)*, and *cwnd=cwnd/2*. This triggers the fast recovery state.

5. Fast recovery algorithm

In response to duplicate ACKs, *cwnd=cwnd+1*; upon receiving a new ACK, *cwnd=ssthresh*, transitioning into the congestion avoidance state.

4.1.2 Fluid model

$$p(t) = \begin{cases} 0, & q(t) \le K_{min} \\ p_{max} \cdot (q(t) - K_{min}), & K_{min} < q(t) \le K_{max} \\ 1, & q(t) > K_{max} \end{cases} \tag{14}$$

$$\frac{dq}{dt} = \sum_{i=1}^{N} R_i(t) - C \tag{15}$$

$$R_i(t) = \frac{W_i(t)}{\tau_i} \tag{16}$$

$$\tau_i(t) = \frac{q(t)}{C} + D^i_{prop} \tag{17}$$

$$\frac{dW_i(t)}{dt} = \frac{1}{\tau_i(t)} - \frac{W_i(t)}{2} R_i(t) p(t) \tag{18}$$

Table 3
TCP Reno Model Equation Parameters

p	Packet loss probability	C	Bandwidth of bottleneck link
K_{min}	Low threshold	R_i	Sending rate of flow i
K_{max}	High threshold	τ_i	Round-trip time
P_{max}	Probability factor	D^i_{prop}	Propagation time
q	Queue size of bottleneck link	W_i	Congestion window size

4.1.3 Algorithm analysis

1. Stability and Proof

If the algorithm is stable, there must be a point where the sending rate of all flows remains constant. Assuming $\frac{dW_i}{dt} = 0$, the equations at the stable point can be derived as follows:

$$\begin{cases} 2 = W_i^2 \cdot p \\ \sum_{i=1}^{N} \frac{W_i}{\tau_i} = C \end{cases} \tag{19}$$

The equations with two unknowns have a unique solution. Furthermore, the unique solution satisfies$W_1 = W_2 = \cdots = W_N$, indicating the stability of the TCP Reno algorithm.

2. Unfairness and Proof

Assuming flow A and B share the same path, with $W_A(t) > W_B(t)$, let $f(t) = W_A(t) - W_B(t)$. Utilizing equation (18), we can derive:

$$f'(t) = \frac{dW_A(t) - dW_B(t)}{dt} = \frac{p(t)}{2\tau(t)}(W_B^2(t) - W_A^2(t)) < 0 \tag{20}$$

The decreasing function $f(t)$ leads to equal congestion windows $W_1 = W_2 = \cdots = W_N$ across flows, showing algorithm convergence. However, because $R_i(t) = W_i(t)/\tau_i$, the algorithm does not ensure fair bandwidth sharing, lacking inherent fairness.

4.2. RTT-Based control algorithm: Patched TIMELY [39]

4.2.1. Overview of the algorithm

Patched TIMELY algorithm consists of two parts: RTT algorithm and rate algorithm.

1. RTT algorithm

$$RTT = t_c - t_s - \frac{seg.size}{NICrate} \tag{21}$$

t_s refers to the time when packet i is sent, and t_c indicates the time when the ACK_i is received. *NICrate* denotes the bandwidth capacity of the Network Interface Card (NIC), while *seg.size* represents the size of an individual data burst.

2. Rate algorithm

$$R = \begin{cases} R + \delta & newRTT < T_{low} \\ R \cdot (1 - \beta \cdot (1 - T_{high} / newRTT)) & newRTT > T_{high} \\ \delta \cdot (1 - weight) + R \cdot (1 - \beta \cdot error \cdot weight) & otherwise \end{cases} \tag{22}$$

$newRTT$ represents the newly measured round-trip time. The δ acts as a step modifier in the algorithm. T_{low} and T_{high} are critical threshold factors that guide the rate adjustment. Additionally, *error* and *weight* function as regulatory factors in this context. The algorithm employs specific formulas, incorporating α and τ^* as further regulatory elements.

$$\begin{aligned} &error = \frac{newRTT - T_{low}}{T_{low}}, \quad g = r(t) / D_{minRTT}, \quad r(t) = (1-\alpha) r(t - \tau^*) + \alpha \Delta RTT, \\ &weight = \begin{cases} 0 & g \le -0.25 \\ 2g + 0.5 & -0.25 < g < 0.25 \\ 1 & g \ge 0.25 \end{cases}. \end{aligned} \tag{23}$$

4.2.2. Fluid model

$$\frac{dq}{dt} = \sum_{i=1}^{N} R_i(t) - C \tag{24}$$

$$\frac{dR_i}{dt} = \begin{cases} \frac{\delta}{\tau^*} & q(t - \tau') < C \cdot T_{low} \\ -\frac{\beta}{\tau^*}(1 - \frac{C \cdot T_{high}}{q(t - \tau')}) R_i(t) & q(t - \tau') > C \cdot T_{high} \\ \frac{(1 - w_i)\delta}{\tau^*} - \frac{\beta w_i R_i(t)}{\tau^*} \frac{q(\tau - \tau') - C \cdot T_{low}}{C \cdot T_{low}} & otherwise \end{cases} \tag{25}$$

$$\tau' = \frac{q}{C} + D_{prop} \tag{26}$$

$$\frac{dg_i}{dt} = \frac{\alpha}{\tau^*}(-g_i(t) + \frac{q(t - \tau') - q(t - \tau' - \tau^*)}{C \cdot D_{minRTT}}) \tag{27}$$

$$w_i = \begin{cases} 0 & g_i \le -0.25 \\ 2g_i + 0.5 & -0.25 < g_i < 0.25 \\ 1 & g_i \ge 0.25 \end{cases} \tag{28}$$

$$\tau^* = max\{\frac{seg}{R_i}, D_{minRTT}\} \tag{29}$$

Table 4

Patched TIMELY Model Equation Parameters

R_i	Sending rate of flow i	C	Bandwidth of 13 ottleneck link
g_i	RTT gradient of flow i	β	Multiplicative decrease factor
q	Queue size of the bottleneck	δ	Additive increase step
t	Time	T_{low}	Low threshold
τ^*	Rate update interval	T_{high}	High threshold
τ'	Round-trip time	D_{minRMT}	Minimum RTT for normalization
N	Number of flows at bottleneck	D_{prop}	Propagation time
α	EWMA smoothing factor	seg	Burst size
w	EWMA smoothing factor	MTU	Maximum Transmission Unit

4.2.3. Algorithm analysis

1. Stability and Proof

Considering the characteristics of a constant transmission rate and a stable queue size at equilibrium, setting equation (24, 25) to zero leads to the following conclusion:

$$\begin{cases} R_i = \dfrac{C}{N} \\ q^* = \dfrac{N\delta C T_{low}}{\beta C} + C T_{low} \end{cases} \tag{30}$$

The unique solution implies that the algorithm demonstrates stability.

2. Fairness and Proof

In the scenario where flows A and B share a bottleneck link, with $R_A(t) > R_B(t)$ and $C^*T_{low} < qt < C^*T_{high}$, The derivative of $f(t) = R_A(t) - R_B(t)$ is:

$$f'(t) = \frac{dR_A - dR_B}{dt} = \frac{\beta w_i}{\tau^*} \frac{q(t) - C \cdot T_{low}}{C \cdot T_{low}} (R_B(t) - R_A(t)) \tag{31}$$

When $R_A(t) > R_B(t)$, the function $f(t)$ decreases until $R_A(t) = R_B(t)$, leading to convergence. This ensures fair bandwidth sharing among flows in the Patched TIMELY algorithm model, reflecting its fairness characteristic.

4.3 BDP-Based control algorithm: TCP BBR [8]

4.3.1 Overview of the algorithm

TCP BBR consists of four main algorithms: Start-up, Drain, ProbeBW, and ProbeRTT. For an in-depth understanding of the round-trip propagation time (D_{prop}) and the methods used to measure the bottleneck bandwidth (BW), one should refer to Section 3.2.3.

1. Start-up algorithm

Start-up phase fills the network pipe and transitions to Drain when the bandwidth estimate stabilizes (no more than 1.25x increase over three assessments). During this phase,

the sending rate *(R)* and congestion window *(cwnd)* are calculated with the following considerations:

$$R(t) = \frac{2}{\ln 2} \cdot BW(t), \qquad cwnd(t) = \frac{2}{\ln 2} \cdot BW(t) \cdot D_{prop}(t) \tag{32}$$

2. Drain algorithm

In Drain, once the inflight data falls beneath the BDP threshold, the algorithm transitions to the ProbeBW state. In this state, the calculations for rate and congestion window are as follows:

$$R(t) = \frac{\ln 2}{2} \cdot BW(t), \qquad cwnd(t) = \frac{2}{\ln 2} \cdot BW(t) \cdot D_{prop}(t) \tag{33}$$

3. ProbeBW algorithm

ProbeBW algorithm regularly evaluates the bottleneck link's bandwidth, typically every 8 round-trip propagation times. If D_{prop} increases within a 10-second window, the algorithm moves to the ProbeRTT state. In ProbeBW, the rate and congestion window are calculated as follows:

$$\begin{aligned} &R(t) = R_{gain} \cdot BW(t) \qquad R_{gain} = [1.25, 0.75, 1, 1, 1, 1, 1, 1], \\ &cwnd(t) = 2 \cdot BW(t) \cdot D_{prop}(t) \end{aligned} \tag{34}$$

4. ProbeRTT algorithm

The ProbeRTT algorithm runs for a maximum of 200 milliseconds before transitioning. It switches back to Start-up or ProbeBW based on network load. During this state, the algorithm calculates rate and congestion window as follows:

$$R(t) = BW(t), \qquad cwnd(t) = 4(MSS) \tag{35}$$

4.3.2 Fluid model

$$\frac{dq}{dt} = \sum_{i=1}^{N} R_i(t) - C \tag{36}$$

$$T_i(t) = \frac{q(t)}{C} + D_{prop}^i \tag{37}$$

$$\frac{dR_i(t)}{dt} = \begin{cases} \dfrac{0.03}{D_{prop}^i} R_i(t - D_{prop}^i) & \sum_{i=1}^{N} R_i(t - D_{prop}^i) < C \\ \dfrac{q(t - 2D_{prop}^i) - q(t - D_{prop}^i)}{CD_{prop}^i T(t - D_{prop}^i)} R_i(t - D_{prop}^i) & q(t - D_{prop}^i) > 0 \\ 0 & otherwise \end{cases} \tag{38}$$

Table 5
TCP BBR Model Equation Parameters

q	Queue size of bottleneck link	T_i	Round-trip time of flow i

C	Bandwidth of bottleneck link	D^i_{prop}	Propagation time of flow i
R_i	Sending rate of flow i		

4.3.3 Algorithm analysis

1. Stability and Proof

Setting equation (38) to zero yields a specific result:

$$\begin{cases} \sum_{i=1}^{N} R_i(t) = C \\ q(t) = 0 \end{cases} \tag{39}$$

The result shows that the equations have multiple solutions, indicating that TCP BBR has numerous stable points and confirming its inherent stability.

2. Unfairness and Proof

Given $R_A(t) > R_B(t)$ for flows A and B on the same path, the following observation can be made:

$$\frac{d(R_A(t) - R_B(t))}{dt} = \begin{cases} \frac{0.03}{D^i_{prop}}(R_A(t) - R_B(t)) > 0 \\ \frac{q(t - D^i_{prop}) - q(t)}{CD^i_{prop}T(t)}(R_A(t) - R_B(t)) > 0 \end{cases} \tag{40}$$

This leads to the conclusion that the rate difference between flows A and B is growing, suggesting that flow A continues to dominate bandwidth at the bottleneck. This behavior indicates a lack of fairness in the algorithm.

4.4. ECN-Based control algorithm: DCQCN [11]

4.4.1. Overview of the algorithm

DCQCN consists of switch, receiver and sender algorithms.

1. Switch algorithm

$$p(t) = \begin{cases} 0 & q(t) \le K_{\min} \\ \frac{q(t) - K_{\min}}{K_{\max} - K_{\min}} P_{\max} & K_{\min} < q(t) \le K_{\max} \\ 1 & q(t) > K_{\max} \end{cases} \tag{41}$$

Here p represents the marking probability. K_{min} and K_{max} define the minimum and maximum thresholds, respectively, and P_{max} denotes the maximum probability limit.

2. Receiver algorithm

After receiving an ECN-marked packet, the receiver sends a CNP. Maximum one CNP can be sent per *N* microseconds.

3. Sender algorithm

The sender algorithm primarily governs the packet transmission rate (R_C) utilizing factors such as α (initial value is 1), τ', R_T, F, Timer N_T, and Byte counterN_{BC}. When the sender receives a CNP,

$$R_T = R_C, \quad R_C = R_C(1-\alpha/2), \quad \alpha = (1-g)\alpha + g \tag{42}$$

If sender gets no CNP forτ',

$$\alpha = (1-g)\alpha \tag{43}$$

If no CNP is received and *max(N_T, N_{BC}) ≤ F* occurs,

$$R_C = (R_T + R_C)/2 \tag{44}$$

If no CNP is received and *min(N_T, N_{BC}) > F* occurs,

$$R_T = R_T + iR_{AI} \quad i = N_T \text{ or } N_{\mathrm{BC}}, \quad R_C = (R_C + R_T)/2 \tag{45}$$

In other cases,

$$R_T = R_T + R_{AI}, \quad R_C = (R_C + R_T)/2 \tag{46}$$

4.4.2. Fluid model

$$p(t) = \begin{cases} 0 & q(t) \le K_{min} \\ \dfrac{q(t)-K_{min}}{K_{max}-K_{min}} p_{max} & K_{min} < q(t) \le K_{max} \\ 1 & q(t) > K_{max} \end{cases} \tag{47}$$

$$\frac{dq}{dt} = \sum_{i=1}^{N} R_i(t) - C \tag{48}$$

$$\frac{d\alpha^{(i)}}{dt} = \frac{g}{\tau'}((1-(1-p(t-\tau^*))^{\tau' R_C^{(i)}(t-\tau^*)}) - \alpha^{(i)}(t)) \tag{49}$$

$$\begin{aligned} \frac{dR_C^{(i)}}{dt} = & -\frac{R_C^{(i)}(t)\alpha^{(i)}(t)}{2\tau}(1-(1-p(t-\tau^*))^{\tau R_C^{(i)}(t-\tau^*)}) \\ & + \frac{R_T^{(i)}(t)-R_C^{(i)}(t)}{2}\frac{R_C^{(i)}(t-\tau^*)p(t-\tau^*)}{(1-p(t-\tau^*))^{-B}-1} \\ & + \frac{R_T^{(i)}(t)-R_C^{(i)}(t)}{2}\frac{R_C^{(i)}(t-\tau^*)p(t-\tau^*)}{(1-p(t-\tau^*))^{-TR_C^{(i)}(t-\tau^*)}-1} \end{aligned} \tag{50}$$

$$\begin{aligned} \frac{dR_T^{(i)}}{dt} = & -\frac{R_T^{(i)}(t)-R_C^{(i)}(t)}{\tau}(1-(1-p(t-\tau^*))^{\tau R_C^{(i)}(t-\tau^*)}) \\ & + R_{AI}R_C^{(i)}(t-\tau^*)\frac{(1-p(t-\tau^*))^{FB}p(t-\tau^*)}{(1-p(t-\tau^*))^{-B}-1} \\ & + R_{AI}R_C^{(i)}(t-\tau^*)\frac{(1-p(t-\tau^*))^{FTR_C^{(i)}(t-\tau^*)}p(t-\tau^*)}{(1-p(t-\tau^*))^{-TR_C^{(i)}(t-\tau^*)}-1} \end{aligned} \tag{51}$$

Table 6

DCQCN Model Equation Parameters

R_C	Current rate	N	Number of flows at bottleneck
R_T	Target rate	C	Bandwidth of bottlenck link
α	Decrease factor	F	Fast recovery steps
q	Queue size of bottleneck link	B	Byte counter for rate increase
t	Time	T	Timer for rate increase
K_{min}	Low threshold	R_{AI}	Rate increase step
K_{max}	High threshold	τ	CNP generation timer
P_{max}	Probability factor	τ^*	Control loop delay
g	Decrease factor	τ'	Update Interval of α

4.4.3. Algorithm analysis

1. Stability and Proof

Initially, if the bandwidth is underutilized, the sending rate increases. If it surpasses the bottleneck capacity, congestion occurs, and the sending rate decreases. Therefore, at the stable point, the bottleneck link bandwidth must be fully occupied and the queue size will remain constant. Assuming the queue size at stable time is q^*, the following equations can be obtained.

$$\begin{cases} \sum_{i=1}^{N} R_C^{(i)}(t) = C \\ p^* = \dfrac{q^* - K_{min}}{K_{max} - K_{min}} P_{max} \\ \dfrac{d\alpha}{dt} = 0, \quad \dfrac{dR_T}{dt} = 0, \quad \dfrac{dR_C}{dt} = 0 \end{cases} \tag{52}$$

Combining equation (49-51), we see that

$$R_C^{(i)} = \frac{a^2 \alpha^{(i)*}}{\tau^2 R_{AI}(b+c)(d+e)} \tag{53}$$

Here $\alpha^{(i)*}$, a, b, c, d, e is denoted as follows：

$$\begin{gathered} \alpha^{(i)*} = 1-(1-p^*)^{\tau' R_C^{(i)}}, \quad a = 1-(1-p^*)^{\tau R_C^{(i)}}, \quad b = ((1-p^*)^{-B}-1)^{-1} p^*, \\ c = ((1-p^*)^{-TR_C^{(i)}}-1)^{-1} p^*, \quad d = ((1-p^*)^{-B}-1)^{-1}(1-p^*)^{FB} p^*, \\ e = ((1-p^*)^{-TR_C^{(i)}}-1)^{-1}(1-p^*)^{FTR_C^{(i)}} p^*. \end{gathered} \tag{54}$$

Let $f(p^*) = \dfrac{a^2 \alpha^{(i)*}}{\tau^2 R_{AI}(b+c)(d+e)}$. When $R_{C(i)} > 0$, $f(0) = 0$ and $f(1) \to +\infty$. According to the continuity principle, when $R_{C(i)} > 0$, there exists p^* such that $f(p^*) = R_C^{(i)*}$. Since all flows share p^*, it follows that $R_C^{(1)*} = R_C^{(2)*} = \cdots = R_C^{(N)*} = C/N$.

Based on the above analysis, the DCQCN algorithm has a unique stable point, where all flows have equal sending rates. Therefore, the DCQCN algorithm is stable.

2. Fairness and Proof

The DCQCN algorithm's fairness isn't directly discernible from equation (51) due to the interplay between RC and RT. To prove fairness, the paper uses a discrete method, examining RC changes between CNPs. Here, we first prove two theorems.

Theorem 1. In DCQCN algorithm, the smaller the sending rate of flow i is, the more additive growth occurs between adjacent CNPs.

Proof of Theorem 1. The switch algorithm estimates a flow collects CNPs at a rate of p*RC*Δt. When a CNP is collected, the average packet sending rate is 1/p packets. This means that the sending rates of adjacent CNPs for each flow are the same. DCQCN controls this rate via a timer and byte counter. For equal data sent, the increment of the byte counter is the same, but the timer increments more with smaller sending rates. Therefore, for flows with smaller sending rates, the additive growth between adjacent CNPs occurs more frequently.

Theorem 2. In DCQCN algorithm, $\alpha^i(t)$ is a converging function.

Proof of Theorem 2. If the time interval between adjacent CNPs of flow i is Δt, then

$$\alpha^{(i)}(t+\Delta t) = (1-g)^{\Delta t/\tau'}((1-g)\alpha^{(i)}(t)+g) < (1-g)^{1+\Delta t/\tau'}\alpha^{(i)}(t) < \alpha^{(i)}(t) \tag{55}$$

The function $\alpha^i(t)$ is monotonically decreasing. Given that $\alpha^i(t) > 0$ and 0<g<1, it follows that $\alpha^{(i)}(t+\Delta t)$>0. By induction, $\alpha^i(t)$ >0 for all t>0. As $\alpha^i(t)$ is monotonically decreasing with an upper bound, it must converge.

Below, we study the changes between adjacent CNPs. Let the receiving times of adjacent CNPs be t and t+Δt, besides, the rate decreases at $t+\delta(\delta \to 0)$ and then experiences additive growth for $N(i)$ times.

Let $R_{C^{(i)}}(t+\delta) = (1-2^{-1}\alpha^{(i)}(t)R_{C^{(i)}}(t)$ and $\alpha^i(t) = \alpha^*(\alpha^* > 0)$, according to the DCQCN algorithm and Theorem 2, we get that when the rate increase occurs 0 times, $R_{T^{(i)}}[0] = R_{C^{(i)}}(t), R_{C^{(i)}}[0] = R_{C^{(i)}}(t+\delta), \Delta R = R_{C^{(i)}}[0] - R_{C^{(i)}}(t) = -2^{-1}R_{C^{(i)}}(t)\alpha^*$ .By induction, we can obtain that at t+Δt, after N(i) growth times, $\Delta R = \left(1-2^{-N(i)}\right)R_{AI} - 2^{-N(i)-1}R_{C^{(i)}}(t)\alpha^*$.

Assuming the number of rate increases between adjacent CNPs at the stable point $R_C(t) = \frac{C}{N}, \Delta R = 0$ is $N(k)$. According to Theorem 1, if $R_{C^{(i)}}(t) > C/N$ for flow i, $N(i) < N(k)$. We can get that $\Delta R(i) = \left(1-2^{-N(i)}\right)R_{AI} - 2^{-N(i)-1}R_{C^{(i)}}(t)\alpha^* < \Delta R(k) = 0$. So $R_{C^{(i)}}(t)$ will converge to the stable point where$R_{C^{(i)}}(t) = C/N$. When $R_{C^{(i)}}(t) < C/N$, the proof method is the same.

In summary, in the DCQCN algorithm, all flows will share the bottleneck bandwidth. So the algorithm is fair.

4.5. Link-Utilization-Based control algorithm: HPCC [14]

4.5.1. Overview of the algorithm

HPCC consists of bottleneck link utilization and congestion window algorithms.

1. Bottlenck link utilization algorithm

$$U = \max(U_j)$$
$$U_j = \frac{q_j + txRate_j \times T}{C \times T} = \frac{q_j}{C \times T} + \frac{txRate_j}{C} \tag{56}$$
$$txRate_j = \frac{ack_1.txBytes_j - ack_0.txBytes_j}{ack_1.ts_j - ack_0.ts_j}$$

U represents the bottleneck link utilization, ack_1 and ack_0 are two ACKs, T denotes the minimun RTT. For link $j, U_J, q_j, txRate_j, ack.ts_j$ and C represent the bandwidth utilization, queue size, sending rate, sending data volume, timestamp and bandwidth, respectively.

2. Congestion window algorithm

$$W = \frac{W}{U/\eta} + W_{AI} \tag{57}$$

W_{AI} is a step factor, and η is generally set to 0.95.

4.5.2. Fluid model

$$R_a(t) = \frac{W_a(t)}{\tau} \tag{58}$$
$$U^j(t) = \frac{1}{C}\sum_i R_i(t) \tag{59}$$
$$\frac{dW_a(t)}{dt} = \left(\frac{\eta}{U^j(t-\tau)} - 1\right)\frac{W_a(t-\tau)}{\tau} + \frac{W_{AI}}{\tau} \tag{60}$$

Table 7
HPCC Model Equation Parameters

$\boldsymbol{R_a}$	Sending rate of flow *a*	$\boldsymbol{i}$	flow at bottleneck link *j*
$\boldsymbol{W_a}$	Congestion window size of flow *a*	$\boldsymbol{R_i}$	Sending rate of flow *i*
$\boldsymbol{\tau}$	Control loop delay	$\boldsymbol{\eta}$	Threshold
$\boldsymbol{j}$	Bottlenck link of flow *a*	$\boldsymbol{C}$	Bandwidth of bottlenck link
$\boldsymbol{U_j}$	the utilization of link j	$\boldsymbol{W_{AI}}$	Additive increase step

4.5.3. Algorithm analysis

1. Stability and Proof

Let the left side of equation (58) be zero, then

$$W_a(t) = \frac{U^j(t-\tau)W_{AI}}{U^j(t-\tau) - \eta} \tag{61}$$

For any $U^j{}_{(t-\tau)}$, $W_a(t)$ has a unique solution, so the algorithm is stable.

2. Unfairness and Proof

Assuming the algorithm is fair, for flows with different initial rates, the rate difference between them will decrease over time. If flows A and B have the same path and $W_{\boldsymbol{a}}(t) > W_B(t)$, $U^j{}_t < \boldsymbol{\eta}$. Let $f_{(t)} = W_{\boldsymbol{a}}(t) > W_B(t)$, then

$$f'(t+\tau) = \frac{dW_A(t+\tau) - dW_B(t+\tau)}{dt} = (\frac{\eta}{U^j(t)} - 1)(\frac{W_A(t) - W_B(t)}{\tau}) > 0 \tag{62}$$

The increasing function $f(t)$ contradicts the fairness assumption. Thus, HPCC algorithm is unfair.

4.6. Machine-Learning-Based control algorithm: TCP Drinc [40]

TCP Drinc leverages deep reinforcement learning to regulate transmission rate. It defines event S as a series of states, represented as $S = [s_0, s_1, \ldots, s_N]$. The state space is defined as $s_t = [\Delta w(t), \tau_{RTT}(t), v(t), \delta(t), \tau_{ACK}(t)]$. Here, $\Delta w(t)$ indicates the variation in the congestion window size, $\tau_{RTT}(t)$ signifies the round-trip time, $v(t)$ is the ratio of propagation time to RTT, $\delta(t)$ highlights the disparity between RTT and propagation time, and $\tau_{ACK}(t)$ refers to the interval between successive ACKs.

The action space is defined as $A = [w = w + 1;\ w = w - 1;\ w = w + w^{-1};\ w = w - w^{-1};\ w = w]$. w is the congestion widow size.

The reward is defined as $r(t) = U(t + \tau_{RTT}(t + \tau_{RTT}(t))) - U(t)$. Related functions are listed below:

$$\begin{aligned} U(t) &= U_\alpha(z(t)) - \beta U_\alpha(\tau_{RTT}(t)), \\ z(t) &= \sum_{l=t}^{t+L-1} x(l) \cdot \frac{(1-\eta)}{1-\eta^{L+1}} \cdot \eta^{l-t}, \\ U_\alpha(l) &= \begin{cases} \log(l) & \alpha = 1 \\ \dfrac{l^{1-\alpha}}{1-\alpha} & otherwise \end{cases} \end{aligned} \tag{63}$$

In the aforementioned equations, $x(t)$ symbolizes the network throughput at time t, which is derived from the congestion window and RTT. The variables α, β, η and L are introduced as adjustment factors to fine-tune the algorithm. Importantly, this algorithm is characterized by a dynamic relationship between its inputs and outputs, precluding the formulation of a fluid model equation for analytical purposes.

4.7. Comparative analysis

From the standpoint of control models, algorithms based on packet loss, RTT, and Bandwidth-Delay Product (BDP) are straightforward in their approach, allowing implementation within an implicit feedback model. In contrast, algorithms reliant on ECN and link utilization necessitate an explicit feedback model, entailing more complex protocols.

Regarding algorithm characteristics, the input-output relationship in the first five types of algorithms is typically fixed. Algorithms based on packet loss, RTT, BDP, and ECN struggle to calculate the ideal regulation rate, often resorting to heuristic strategies for incremental adjustments. This approach introduces a degree of latency. Conversely, algorithms focusing

on link utilization avoid this drawback but demand significant network feedback data, thus incurring higher bandwidth and computational resource costs.

Control algorithms integrating machine learning exhibit a dynamic control model contingent on specific network feedback signals. These algorithms excel in adapting to network environment changes, automatically adjusting relevant factors to optimize regulation strategies and enhance performance. However, the complexity of designing appropriate reward functions, ensuring swift convergence, and managing unpredictable outcomes makes them inherently complex.

5. Evaluation Indicators

The criteria for assessing host-to-host congestion control are derived from network performance metrics as outlined in [41]. Despite variations in emphasis among different schemes, four key indicators are universally recognized: Throughput, Delay, Packet Loss Rate, and Fairness Index, as referenced in [42].

1. Throughput

Throughput is defined as the quantity of data successfully transmitted over a network per unit of time. This metric serves as a direct indicator of a control mechanism's efficacy. Generally, higher throughput equates to superior performance. The formula for calculating throughput is as follows:

$$x = N / \Delta t \tag{64}$$

In this formula, Δt represents the duration of the operational period, while N denotes the total amount of data successfully received by the receiver.

2. Delay

Delay is the total time for a packet's transmission and reception, including sending, transmission, processing, and queuing delays. Optimal network performance is characterized by low delay and its stability, which is often associated with higher throughput and stable transmission quality. For practical measurement, the smoothed RTT is commonly used. The formula is as follows:

$$T(t) = \alpha T(t) + (1 - \alpha)(t - t_s) \tag{65}$$

t_s is the send time for packet i, t is the ACK_i receipt time, and α is the smoothing factor.

3. Packet Loss Rate

Packet Loss Rate is the ratio of lost packets to the total packets sent over a period, key for analyzing data flows. Higher loss rates may signal network congestion and are critical for assessing congestion control. The lower the rate, the better the performance, ideally. The formula for Packet Loss Rate is as follows:

$$\eta = (1 - \frac{N}{N_{send}}) \times 100\% \tag{66}$$

N is the amount of data successfully received by the receiver in time interval Δt, and Nsend is the amount of data sent by the sender in the same period.

4. Fairness Index

The Fairness Index serves as a measure of how equitably congestion control mechanisms distribute network resources under congested conditions, quantitatively assessing fairness in resource allocation. A common formula for calculating the Fairness Index [43] is as follows:

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} {x_i}^2} \tag{67}$$

X_i represents the throughput for the flow i. The Fairness Index ranges from 0 to 1, with higher values indicating better fairness. Achieving a high Fairness Index is important for protocol coexistence on the Internet, making it a key metric for assessing congestion control protocols.

6. Main Challenges

A systematic analysis reveals three critical aspects of host-to-host congestion control: real-time collection of network status, precise congestion detection, and efficient packet sending rate control. To address these aspects, host-to-host congestion control mechanisms encounter several challenges:

1. Hosts Coordination

Host-to-host congestion control operates as a distributed system without a central control authority. Each host independently executes control algorithms. Inconsistencies or conflicts in the behavior of different hosts can lead to uneven bandwidth allocation, creating fairness issues. This disparity can also trigger network congestion, adversely impacting network transmission performance. Thus, achieving effective coordination among hosts represents a significant challenge.

2. Feedback Delay

Host-to-host congestion control relies on a feedback mechanism, requiring hosts to gather network feedback and adjust their sending rates accordingly. This process inherently introduces a delay, especially during periods of network congestion, potentially hindering the timely acquisition of current network status. Such delays can result in suboptimal decision-making, affecting congestion control efficacy. Addressing feedback delay is, therefore, a pivotal challenge.

3. Inaccurate Feedback

Beyond the issues caused by feedback delay, the accuracy of feedback in host-to-host congestion control is also compromised by factors like network noise, packet loss, and inherent randomness (e.g., RTT variation). These elements can lead to erroneous congestion control decisions. Consequently, resolving the issue of inaccurate feedback is another crucial challenge.

7. Future Directions

Through the examination of existing research and considering the latest technological trends, we identify two key areas that hold significant potential for future exploration.

1. Edge-host Collaboration Control Model

Using the idea of edge computing, the Edge-host collaboration control model could offload some of the congestion detection and control functions to the edge switches. In terms of distribution, edge switches exhibit a more centralized nature compared to hosts, offering several advantages for congestion detection and control. Firstly, edge switches can aggregate information from multiple hosts, thereby providing more comprehensive data for congestion detection. Secondly, the feedback path can be shortened by leveraging edge switches. Thirdly, the utilization of edge switches can minimize the redundant collection of network state information, consequently reducing transmission overhead. Lastly, the integration of edge switches can effectively coordinate host behavior locally and address local fairness issues. Hence, the Edge-Host Collaboration Control Model signifies a promising research direction.

2. Intelligent Congestion Detection and Control Algorithm

Intelligent congestion detection and control algorithm has emerged as a prominent area of research in recent years, yielding substantial advancements. Innovative mechanisms like Orca [44] and Sage [45] employ deep reinforcement learning to craft intelligent algorithms, facilitating efficient data transmission in intricate network environments. This algorithm, rooted in mathematical and statistical principles, discerns the patterns of network congestion changes. Through logical reasoning, it calculates the optimal sending rate based on network feedback information, thereby enhancing congestion control performance.

Despite current challenges in applying intelligent algorithms to congestion control—such as intricate model training, complex reward function design, and slow convergence—their successes in fields like computer vision, speech recognition, and AIGC underscore their capability to address complex issues. With ongoing technological progress, intelligent algorithms are poised to access more extensive network information and adeptly learn network characteristics and congestion control strategies. Consequently, the Intelligent Congestion Detection and Control Algorithm is set to significantly enhance its efficiency and adaptability, establishing itself as a pivotal area for future research.

8. Conclusions

Host-to-host congestion control, which is key for high-quality network transmission by managing data rates, must now focus on efficiency and adaptability. This paper reviews congestion control mechanisms across four areas: control models, detection methods, algorithms, and evaluation metrics. It analyzes the three main challenges in this domain and discusses the potential of machine learning and edge collaboration. It concludes that a systematic and comprehensive optimization of host-to-host congestion control is necessary to meet the demands of emerging networks.

References

[1] Cisco U, "Cisco annual internet report (2018–2023) white paper," Cisco: San Jose, CA, USA, volume10(1), pp. 1-35, 2020.

[2] D. Milovanovic, Z. Bojkovic, M. Indoonundon, "5G Low-latency Communication in Virtual Reality Services: Performance Requirements and Promising Solutions," World Scientific and Engineering Academy and Society (WSEAS), 2021.

[3] N. M. QUY, L. A. NGOC, N. T. BAN, "Edge Computing for Real-Time Internet of Things Applications: Future Internet Revolution, Wireless Personal Communications," pp. 1423-52, 2023.

[4] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM Computer Communication Review, pp. 314-29, 1988.

[5] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, pp. 397-413, 1993.

[6] W. Stevens, "Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithm," RFC 2001, 1997.

[7] A. Gurtov, T. Henderson, S. Floyd, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, 2012.

[8] N. Cardwell, Y. Cheng, C. S. Gunn, "BBR: Congestion-Based Congestion Control," Communications of the ACM, 2017.

[9] S. Ha, I. Rhee, L. Xu, "CUBIC: A New TCP-friendly High-Speed TCP Variant," ACM SIGOPS Operating Systems Review, pp. 64-74, 2008.

[10] R. Mittal, V. T. LAM, N. DUKKIPATI, "TIMELY: RTT-based Congestion Control for the Datacenter," Proceedings of the ACM Conference on Special Interest Group on Data Communication, 2015.

[11] Y. Zhu, H. Eran, D. "Firestone, Congestion Control for Large-Scale RDMA Deployments," ACM, 2015.

[12] R. Xie, X. Jia, K. Wu, "Adaptive Online Decision Method for Initial Congestion Window in 5G Mobile Edge Computing Using Deep Reinforcement Learning," IEEE Journal on Selected Areas in Communications, pp. 389-403, 2020.

[13] K. K. Ramakrishnan, S. Floyd, D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, 2001.

[14] Y. li, R. Miao, H. Liu, "HPCC: High Precision Congestion Control," Proceedings of the ACM Special Interest Group on Data Communication, 2019.

[15] D. Giannopoulos, N. Chrysos, E. Mageiropoulos, “Accurate Congestion Control for RDMA Transfers,” Proceedings of the 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), IEEE, 2018.
[16] R. Mittal, A. Shpiner, A. Panda, “Revisiting Network Support for RDMA,” The Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, 2018.
[17] J. Geng, J. Yan, Y. Zhang, “P4QCN: Congestion Control Using P4-capable Device in Data Center Networks,” Electronics, pp. 280, 2019.
[18] W. Cheng, K. Qian, W. Jiang, “Re-architecting Congestion Management in Lossless Ethernet,” Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 2020.
[19] P. Taheri, D. Menikkumbura, E. Vanini, “RoCC: Robust Congestion Control for RDMA,” the Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies, 2020.
[20] Y. Hu, Z. Shi, Y. Nie, “Dcqcn Advanced (dcqcn-a): Combining ECN and RTT for RDMA Congestion Control,” proceedings of the 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2021.
[21] J. Zhang, J. Shi, X. Zhong, “Receiver-Driven RDMA Congestion Control By Differentiating Congestion Types in Datacenter Networks,” Proceedings of the 2021 IEEE 29th International Conference on Network Protocols (ICNP), IEEE, 2021.
[22] Y. Binghao, L. Qinrang, S. Jianliang, “A Survey of Low-Latency Transmission Strategies in Software Defined Networking,” Computer Science Review, pp. 40, 2021.
[23] Y. Lu, Z. Ling, S. Zhu, “SDTCP: Towards Datacenter TCP Congestion Control with SDN for IoT Applications,” Sensors, pp. 109, 2017.
[24] P4.ORG, “In-band Network Telemetry (INT) Dataplane Specificatio,” 2021.
[25] Mohammad, Alizadeh, Albert, “Computer Communication Review: A Quarterly Publication of the Special Interest Group on Data Communication,” Data Center TCP (DCTCP), 2010.
[26] F. Lu, H. Du, A. Jain, “CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks,” Proceedings of the Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, 2015.
[27] Y. XIE, F. YI, K. Jamieson, “PBE-CC: Congestion Control Via Endpoint-Centric, Physical-Layer Bandwidth Measurements,” Proceedings of the Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on Applications, Technologies, Architectures, and Protocols for Computer Communication, USA, 2020.
[28] J. Postel, “Transmission Control Protocol,” RFC 793, 1981.
[29] J. M. Mathis, S. Floyd, A. Romanow, “TCP Selective Acknowledgement Options,” RFC 2018, 2003.
[30] G. Kumar, N. Dukkipati, K. Jang, “Swift: Delay is Simple and Effective for Congestion Control in the Datacenter,” Proceedings of the SIGCOMM’20: Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020.
[31] L. S. Brakmo, L. L. Peterson, “TCP Vegas: End to End Congestion Avoidance on A Global Internet,” IEEE, 1995.

[32] C. Jin, D. X. Wei, “LOW S H. FAST TCP: Motivation, Architecture, Algorithms, Performance,” Proceedings of the INFOCOM 2004 Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 2004.
[33] I. Mahmud, G-H. Kim, T. Lubna, “BBR-ACD: BBR with Advanced Congestion Detection,” Electronics, pp. 136, 2020.
[34] B. Vamanan, J. Hasan, T. N. Vijaykumar, “Deadline-Aware Datacenter TCP (D2TCP),” ACM SIGCOMM Computer Communication Review, pp. 115-26, 2012.
[35] Y. Lu, G. Yuan, Y. Bai, “EagerCC: An Ultra-Low Latency Congestion Control Mechanism in Datacenter Networks,” Computer Networks, pp. 236, 2023.
[36] M. Dong, Q. Li, D. Zarchy, “PCC: Re-Architecting Congestion Control for Consistent High Performance,” USENIX Association, 2014.
[37] R. King, R. Baraniuk, R. Riedi, “TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP,” Proceedings of the INFOCOM 2005 24th Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings IEEE, 2005.
[38] J. Padhye, V. Firoiu, D. F. Towsley, “Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation,” IEEE/ACM Transactions on Networking, pp. 133-45, 2000.
[39] Y. Zhu, M. Ghobadi, V. Misra, “ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY,” ACM, 2016.
[40] K. Xiao, S. Mao, J. K. Tugnait, “TCP-Drinc: Smart Congestion Control Based on Deep Reinforcement Learning,” IEEE Access, pp. 11892-904, 2019.
[41] T. ITU, “Terms and Definitions Related to Quality of Service and Network Performance Including Dependability,” Recommendation E, pp. 800, 1994.
[42] V. Kushwaha, R. Gupta, “Congestion Control for High-Speed Wired Network: A Systematic Literature Review,” Journal of Network and Computer Applications, pp. 62-78, 2014.
[43] R. Jain, K. Ramakrishnan, “Congestion Avoidance in Computer Networks with A Connectionless Network Layer: Concepts, Goals and Methodology,” Proceedings of the [1988] Proceedings Computer Networking Symposium, IEEE, 1988.
[44] S. Abbasloo, C. Y. Yen, H. J. Chao, “Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet,” Proceedings of the SIGCOMM’20: Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020.
[45] C. Y. Yen, S. Abbasloo, H. J. Chao, “Computers Can Learn from the Heuristic Designs and Master Internet Congestion Control,” Proceedings of the ACM SIGCOMM 2023 Conference, 2023.