*electronics*

*Article*

# Copa-ICN: Improving Copa as a Congestion Control Algorithm in Information-Centric Networking

**Zhiyuan Wang [1,2], Hong Ni [1,2] and Rui Han [1,2,*]**

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; wangzy@dsp.ac.cn (Z.W.); nih@dsp.ac.cn (H.N.)
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China
* Correspondence: hanr@dsp.ac.cn

**Abstract:** To fundamentally improve the efficiency of content distribution in the network, information-centric networking (ICN) has received extensive attention. However, the existence of a large number of IP facilities in the current network makes the smooth evolution of the network architecture a realistic requirement. The ICN architecture that separates the process of name resolution and message routing is widely accepted for its better compatibility with IP networks. In this architecture, the user first obtains the locator of the content replica node from the name resolution system (NRS) and then completes the data transmission through the locator. In data transmission, receiver-driven congestion control algorithms need to be studied. Therefore, we introduce the Copa algorithm into ICN and propose an improved Copa-ICN algorithm. Experiments show that the Copa-ICN algorithm has a high convergence speed and fairness, and when there is a transmission process in the opposite direction, it can still have a high throughput different from the original Copa algorithm.

**Keywords:** information-centric networking; congestion control; receiver-driven



**Citation:** Wang, Z.; Ni, H.; Han, R. Copa-ICN: Improving Copa as a Congestion Control Algorithm in Information-Centric Networking. *Electronics* **2022**, *11*, 1710. https://doi.org/10.3390/electronics11111710

Academic Editor: Jaime Lloret

Received: 16 May 2022
Accepted: 21 May 2022
Published: 27 May 2022



## 1. Introduction

With the development of emerging network applications such as short video, virtual reality (VR) and the Internet of Things (IoT), the distribution of massive amounts of content makes it increasingly difficult for traditional TCP/IP networks to meet quality of service (QoS) requirements. A content delivery network (CDN) [1] makes up for the shortcomings of TCP/IP networks in content distribution to a certain extent by deploying a large number of caching servers on the user side to cache content, but as application layer technology, its performance is still limited.

ICN has received increasing attention from researchers [2]. Different from traditional TCP/IP, ICN names the content and places a cache in the network, only cares about the content and not the location and naturally supports the distribution of large-scale network content. In recent years, the research on information-centric networking (ICN) has been very active. For example, due to the advantages of ICN in content distribution efficiency, some researchers combine ICN with software-defined networking (SDN) to propose the SDN-based ICN approach [3]. With the support of flexible programmability and efficient manageability of SDN, ICN has received more attention [3,4]. Furthermore, most IoT applications inherently follow a content-oriented communication paradigm [5], which is similar to the way ICN operates. Some research on the IoT is even easy to apply to ICN [6], and the application of ICN in the IoT has also attracted the attention of researchers [5].

There are many ICN architectures [7–14]. Based on the routing and forwarding approach, there are two main types: the name-based routing approach and the stand-alone name resolution approach [15]. Due to the existence of a large number of IP facilities in the current network, smooth evolution of the network architecture is a realistic requirement.

The former's name resolution process and message routing process are coupled, so they are less compatible with existing IP architectures, such as CCNs [8] and NDNs [7]. In contrast, the latter has better compatibility with IP architectures such as MobilityFirst [9], NetInf [13] and SEANet [14].

In the ICN architecture adopting the standalone name resolution approach, the information of the named data chunk (NDC) is registered in the name resolution system (NRS). To complete the acquisition of the NDC, the receiver first needs to obtain the locator from the NRS and then uses the locator for routing to complete the data transmission. To prevent over-utilization of the NRS and the overhead of the cache system, the size of the NDC should not be too small [16,17]. Larger NDCs require congestion control of traffic during transmission; otherwise, this will easily cause severe network congestion or even collapse.

In traditional TCP/IP networks, congestion control algorithms have been well-studied [18–20]. The separation of the name resolution process and the message routing process makes it easier to apply congestion control algorithms from TCP/IP networks to ICN networks. Considering that the ICN obtains the NDC based on the pull method, the congestion control algorithm in the traditional network must first adapt to the receiver-driven transmission mode [21]. Secondly, each NDC has a fixed size, which is very different from the communication method based on the byte stream in traditional networks. In a high-speed network, if the size of the NDC is not large enough, the data transmission will end quickly, and it is difficult to achieve a relatively fair throughput between different NDC transmission processes. In this case, the convergence speed and fairness of the algorithm are very important.

In this paper, we study some excellent congestion control algorithms in traditional networks, such as BBR [22], Cubic [23], NewReno [24] and Copa [25]. Due to the excellent performance of the Copa algorithm in terms of convergence speed and fairness, we propose the Copa-ICN algorithm based on the original Copa algorithm. In order to observe the performance of different algorithms more accurately, we propose an ICN transport protocol prototype compatible with the IP architecture. It first obtains the locator from the NRS and then uses the set congestion control algorithm for data transmission. The transport protocol allows switching of the NDC replica nodes to take advantage of ICN's multi-replica feature. We implement the proposed transport protocol prototype in Network Simulator 3 (NS-3) [26] and implement Copa-ICN as well as several other receiver-driven algorithms. The experiment results show that the Copa-ICN algorithm has huge advantages in algorithm convergence speed and delay compared with the BBR, Cubic and NewReno algorithms. Compared with the original Copa algorithm, Copa-ICN not only further improves the convergence speed but also maintains a high throughput when there is an NDC transmission process in the opposite direction.

The remaining parts of the paper are structured as follows. We review the related work about congestion control mechanisms of ICN in Section 2. In Section 3, we introduce a complete ICN transport protocol. In Section 4, we describe the motivation of the Copa algorithm in ICN and propose the Copa-ICN algorithm. Section 5 evaluates the NS-3 experiment results of the Copa-ICN by comparison with the Cubic, BBR, NewReno and Copa algorithms. Conclusions and discussions are carried out in Section 6.

## 2. Related Work

Whether it is the traditional TCP/IP network or the novel ICN network, congestion control mechanisms are essential to ensure network performance and efficient use of network bandwidth. Since ICN was proposed, many ICN architectures have appeared, and researchers have proposed various congestion control algorithms based on various ICN architectures.

A number of new congestion control mechanisms have emerged in ICN. MFTP is the transport protocol of the MobilityFirst architecture, which enables congestion control within the network through a hop-by-hop backpressure scheme, whereby when congestion occurs within the network, the congestion signal is passed back to the traffic source on a hop-by-hop basis, ultimately resulting in a reduction in the amount of traffic injected

into the network [27]. Some researchers have studied the congestion control of wireless links. For example, the authors of [28] tried to reduce the delay of ICN wireless links, while those in [6] proposed a mechanism to reduce wireless network congestion using clustering, which can be easily applied in ICN. Hop-by-hop interest shaping has been proposed as a viable congestion control mechanism in NDN [29,30]. Researchers have proposed many hop-by-hop interest shaping algorithms. In [29], the authors pointed out that both interests and contents can lead to congestion. By shaping interest packets, the loss of data packets can be prevented, and the bandwidth utilization of links can be improved. The authors of [31] proposed a bandwidth fair sharing algorithm between different flows. Interests and NACKs which exceed the fair share will be discarded together, and the additive increase multiplicative decrease (AIMD) mechanism is used to control the receiver's interest rate. PCON [32] uses the CoDel active queue management algorithm to detect network congestion and then sends signals downstream by explicitly marking certain packets. Downstream devices recognize the signal, the router can divert traffic to another path, and the receiver can slow down the interest rate.

Some works try to borrow mature congestion control solutions in TCP/IP networking. ICP [33] is a well-known ICN congestion control solution. It uses a congestion window (CWND) similar to TCP in NDN and uses the AIMD mechanism to control the size of the congestion window. Each round-trip time (RTT) congestion window increases by one. When a congestion event occurs, the CWND is reduced to half of its original value. DCP [34] is a delay-based ICN congestion control protocol that measures the queuing delay on the forwarding path from the producer or caching site to the consumer using a protocol similar to the LEDBAT [35] algorithm, and it addresses issues such as slow start fairness and "latecomer advantage". CCTCP [36] is a transport protocol suitable for the CCN architecture. It redesigns the TCP as a receiver-driven transport protocol, maintains multiple congestion windows, and proposes a new RTT estimation method. The authors of [37] proposed EC-CUBIC, which uses the core idea of the CUBIC [23] algorithm to control the rate at which consumers send interest packets, sensing congestion in the network through the implementation of explicit congestion notifications in routers. NetInf TP [38] is the transport protocol of the NetInf [13] architecture which is compatible with the IP infrastructure. It can share bandwidth fairly with TCP traffic using the NewReno [24] algorithm by implementing the AIMD mechanism for the congestion window. In [21], the authors propose a two-level congestion control mechanism (2LCCM), which first avoids the congestion path through the selection of replica nodes and then uses the receiver-driven BBR [22] congestion control algorithm for data transmission.

Whether it is a hop-by-hop congestion control scheme, a purely receiver-driven scheme, or other work, most of them do not focus on the convergence speed of the congestion control algorithm. In some particular architectures, such as NetInf [13] and SEANet [14], the transmission processes of different NDCs are independent. The size of the NDC is often larger than one maximum transmission unit (MTU), but the size is limited [16]. In this case, the convergence speed of the algorithm is particularly important. If the convergence speed of the algorithm is slow, then NDC transmission will be at an unfair rate most of the time, and the data transmission rate will be unstable, which will adversely affect the user experience. Copa [25] is a new congestion control algorithm which was proposed in 2018. It has better fairness and a faster convergence speed than algorithms such as the Cubic [23] and BBR algorithms [22]. In this paper, we redesign it as a receiver-driven model, improve its performance including queuing delay calculation and propagation delay measurement, and then propose a solution to further accelerate the convergence speed of the Copa algorithm.

## 3. Proposed Transport Protocol

In this section, we introduce a receiver-driven transport protocol that can be used in the ICN architecture by adopting the standalone name resolution approach and can be deployed over an IP infrastructure.

### *3.1. Overview of the Transmission Process*

In the ICN architecture of the standalone name resolution approach, the process of name resolution and message routing are separated, and the NRS maintains the mapping between the identifier and the locator [39]. In this paper, Entity-ID (EID) is the identifier, and the network address (NA) is the locator. Specifically, EID represents a receiver device or an NDC, and the NA represents an IP address. When a new NDC is generated in a device, the NDC is assigned a unique EID. The EID of the NDC and the IP of the device will be registered in the NRS. If the receiver wants to obtain the NDC identified by the EID in the network, it first needs to query the NRS for the NA list of nodes that have replicas of the NDC and then select a replica node to obtain the NDC. For security reasons, receiver devices often cannot directly obtain the NAs of nodes in the network, so replica selection often requires the assistance of edge nodes. Since the size of the NDC is often larger than the maximum segment size (MSS), the NDC should be split into segments for transmission. The congestion control algorithm controls the transmission rate, and the retransmission algorithm performs error recovery. In this process, the sender only needs to reply with data according to the request of the receiver, which is stateless. ICN routers may cache forwarded segments. When the ICN router successfully assembles a complete NDC, it will register with the NRS to become a new replica node. In addition, the dynamic binding of identifiers and locators makes it easier to modify the NA during packet forwarding. This means that when severe network congestion occurs between the receiver and the selected replica node, the NDC transmission can be successfully completed by switching the replica node. Figure 1 is a schematic diagram of NDC transmission in ICN.

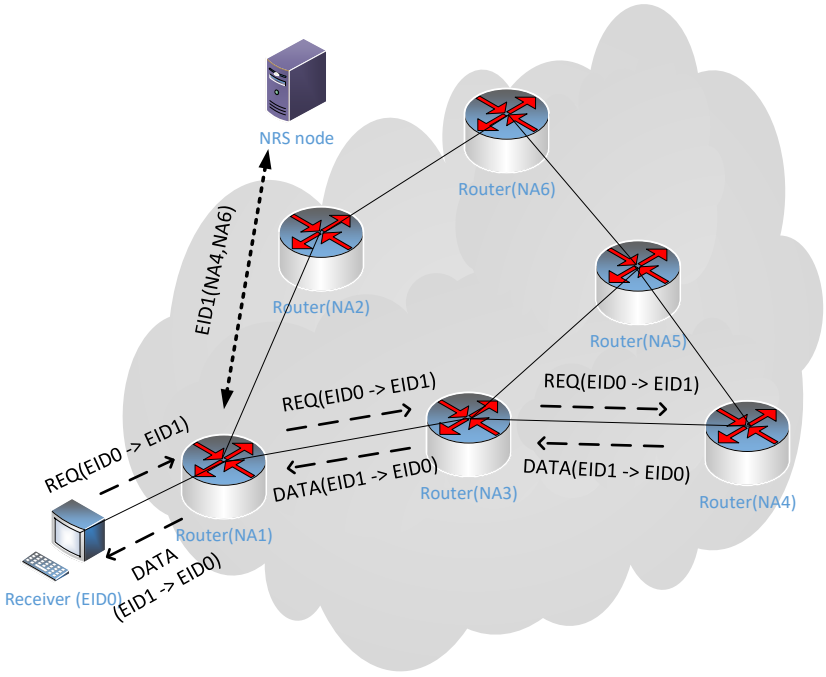


**Figure 1.** NDC transmission in ICN networks.

### *3.2. Message Type*

Transport protocols use IP addresses for routing, so all messages are encapsulated as IP packets. An unassigned value for the "Protocol" [40] field of IPv4 or the "Next Header" [41] field of IPv6 can be set to indicate that the IP header is followed by an EID header. The EID header is followed by the transport layer header. There are mainly two types of packets used in the transmission process: one is the request (REQ) packet, and the other is the data (DATA) packet. The receiver sends the REQ packet to the network, and the replica node replies with the DATA packet after receiving the REQ packet. Figure 2 is the packet format in the ICN transport protocol.

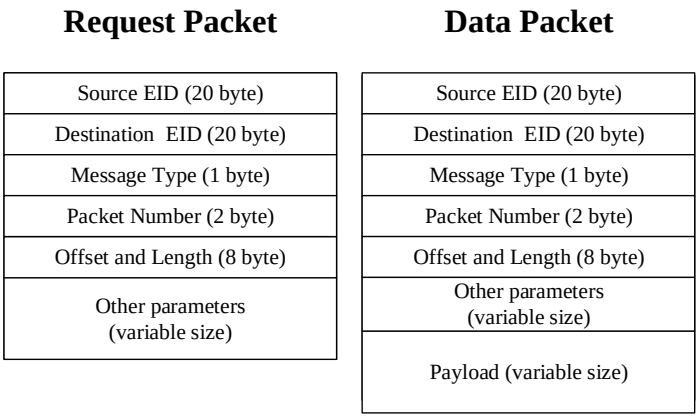


**Figure 2.** Packet format in the ICN transport protocol.

In the REQ packet, "Source EID" indicates the EID of the receiver device, "Destination EID" is the EID of the NDC, "Message Type" indicates that the packet is an REQ packet, "Packet Number" is a monotonically increasing request packet sequence number, "Offset and Length" indicates the offset and length of the requested data in the NDC, and "Other parameters" are other optional parameters, such as specifying the MSS and checksum.

In the DATA packet, "Source EID" is the EID of the NDC, "Destination EID" is the EID of the receiver device, "Message Type" indicates that the packet is a DATA packet, "Packet Number" is the "Packet Number" of the REQ packet, (An REQ packet usually requests multiple DATA packets, so the "Packet Number" of the DATA packet will be repeated.) "Offset and Length" indicates the offset and length of the data in the packet in the NDC, and "Other parameters" are other optional parameters, such as the NDC size and checksum.

### *3.3. Rate and Congestion Control*

We use a sliding window (WND) to control the rate during transmission. The size of the WND is the smaller of the congestion windows (CWNDs) and the available buffer to the receiver. This paper focuses on the congestion window. In fact, for NDCs with a limited size, except for resource-limited embedded devices, normal computers rarely have an available buffer smaller than the congestion window. In the transmission process controlled by the sliding window, the amount of data that the receiver can request is the difference between the WND and inflight data. When the size of the WND and inflight data are equal, the receiver will not be able to send REQ packets, and then when DATA packets are received, inflight data changes to be small, the WND will be larger than inflight data, and the receiver will be able to request the data.

When DATA packet loss occurs, we need to detect it as soon as possible and subtract the amount of lost data from the inflight data so that the receiver can send a new request. For the loss of DATA packets, we have two ways to find this out. One is the number of DATA packets received that should arrive after the packet exceeds a certain threshold, which is similar to the TCP's fast retransmission, with the difference being that the threshold in the TCP is a value that can be dynamically changed. Our design refers to the practice of ICTP [16] and NetInf TP [38] and sets the threshold to a fixed value of three. Another way to detect packet loss is timer timeout, which is consistent with the calculation method of RTO in the TCP [42]. Data found to be lost will be re-recorded in the queue waiting to be transmitted.

Based on the above protocol details, we have implemented a set of congestion control algorithm interfaces, which can transplant some congestion control algorithms of the TCP. It should be noted that in the implementation of the interface, the switching of the replica node is also transmitted to the congestion control algorithm as a congestion

control signal. In addition, according to the needs of some congestion control algorithms, the pacing [43] mechanism is also supported. We have implemented four congestion algorithms—NewReno [24], Cubic [23], BBR [22], and Copa [25]—in the transport protocol.

## 4. Copa Algorithm in ICN

In this section, we first describe the motivation for applying the Copa algorithm to the ICN network, introduce the Copa algorithm, and finally propose our improvements to the Copa algorithm for ICN scenarios.

### *4.1. Motivation*

To prevent network collapse, congestion control is necessary during the transmission of NDCs. Congestion control of a standard transmission process usually has two phases: slow start and congestion avoidance. The size of the NDC has a great impact on the performance of the congestion control algorithm. In addition, an NDC that is too small will also bring a large overhead to the cache and NRS of the ICN [16,17]. Therefore, the NDC discussed in this paper is not smaller than 1 MB.

An NDC transmission process controlled by a qualified congestion control algorithm will try to occupy all the bottleneck bandwidth to avoid wasting bandwidth and reduce the time required for transmission. To improve the user experience, the NDC should be delivered at a higher transmission rate as much as possible. When a bottleneck link has a transmission process joining and ending, all transmission processes through the bottleneck link should be able to quickly converge to a new fair rate. Since the NDC size in ICN is limited, the transmission time is often not very long. During the transmission process, the congestion control algorithm may restart due to the switching of replica nodes. If the congestion control algorithm takes a long time to converge to the fair rate, it will cause the NDC transmission to be at an unfair rate most of the time.

The convergence speed and fairness of the congestion control algorithm in ICN should be two important indicators. Copa is a new congestion control algorithm proposed in recent years which has been applied in the TCP and QUIC [44,45]. Compared with several widely used algorithms, it has good performance in terms of bandwidth utilization, convergence speed, fairness, and delay. Therefore, we implement the Copa algorithm in ICN and improve it so that it has better performance in the ICN transport protocol.

### *4.2. Improving the Copa Algorithm in ICN*

In the ICN transport protocol, we propose the Copa-ICN algorithm, which has three improvements over the original Copa: improving the calculation of the queuing delay, improving the accuracy of the queuing delay, and improving the convergence speed of the algorithm. Algorithms 1 and 2 show the details of Copa-ICN. Table 1 is units and meanings of main notation.

**Table 1.** Units and meanings of main notation.

| Parameter | Units | Meanings |
|---|---|---|
| $\lambda$ | Mbps | The average throughput of the flow |
| $\delta$ | 1/(MTU-sized packet) | Determining how much latency is a trade-off to throughput |
| $d_q$ | second | Average queuing time for the round-trip path of the packet |
| $RTT_{standing}$ | second | The minimum RTT within SRTT/2 |
| $RTT_{min}$ | second | The minimum RTT within 10 s |
| $cwnd$ | MTU-sized packet | The size of the congestion window |
| $v$ | MTU-sized packet | The velocity parameter for the change in $cwnd$ |
| $Tdat_{standing}$ | second | Minimum propagation delay of DATA packet within SRTT/2 |
| $Tdat_{min}$ | second | Minimum propagation delay of DATA packet within 10 s |
| $T_q$ | second | Average queuing time of the DATA packet |

**Algorithm 1:** Copa-ICN algorithm (part 1).

**Data:** pTime: Packet Received Time
**Data:** SRTT: Smooth RTT
**Data:** Mss: Maximum Segment Size
**Data:** cwndBytes: Congestion Window size (in bytes)

Initialization:
    $cwndBytes \leftarrow cwndMin, v \leftarrow 1, isSlowStart \leftarrow true$
    $direction \leftarrow UP, numTimesDirectionSame \leftarrow 0$
    $lastCwndRecordTime \leftarrow CurrentTime, lastRecordedCwndByte \leftarrow cwndBytes$
    Request two high-priority DATA packets
**Function** `OnPacketRecv()`**:**
    Update $Tdat_{standing}$, $Tdat_{min}$, $RTT_{standing}$
    $T_q = Tdat_{standing} - Tdat_{min}$
    $currentRate = cwndBytes / RTT_{standing}$
    $\delta = a \cdot (currentRate/1000000 * 8)^2 + b$
    $\delta = min(\delta, 0.5)$
    $increaseCwnd \leftarrow false$
    **if** $T_q == 0$ **then**
        $increaseCwnd \leftarrow true$
    **else**
        $targetRate = \frac{Mss}{\delta \cdot T_q}$
        $increaseCwnd = targetRate \geq currentRate$
    **if** $!(increaseCwnd \&\& isSlowStart)$ **then**
        UpdateDirection()
    **if** $increaseCwnd$ **then**
        **if** $isSlowStart$ **then**
            $cwndBytes = cwndBytes + Mss$
        **else**
            **if** $direction != UP \&\& v > 1$ **then**
                ChangeDirection(UP)
            $cwndBytes = cwndBytes + \frac{Mss^2 \cdot v}{\delta \cdot cwndBytes}$
    **else**
        **if** $direction != DOWN \&\& v > 1$ **then**
            ChangeDirection(DOWN)
        **if** $isSlowStart$ **then**
            $isSlowStart \leftarrow false$
        $cwndBytes = cwndBytes - \frac{Mss^2 \cdot v}{\delta \cdot cwndBytes}$
        $cwndBytes = max(cwndBytes, cwndMin)$
    Update Pacing Rate to $\frac{cwndBytes \cdot 2}{SRTT}$

**Algorithm 2:** Copa-ICN algorithm (part 2).

```
Function UpdateDirection():
    if PTime − lastCwndRecordTime > SRTT then
        newDirection = cwndBytes > lastRecordedCwndBytes?Up : Down
        if newDirection! = direction then
            v = 1
            numTimesDirectionSame = 0
        else
            numTimesDirectionSame + +
            if numTimesDirectionSame >= 4 then
                v = 2 · v
        direction = newDirection
        lastCwndRecordTime = pTime
        lastRecordedCwndBytes = cwndBytes

Function ChangeDirection(newDirection):
    if newDirection! = direction then
        direction = newDirection
        v = 1
        numTimesDirectionSame = 0
        lastCwndRecordTime = pTime
        lastRecordedCwndBytes = cwndBytes

Function OnReplicaChanged():
    All parameters reset, goto Initialization
```

#### 4.2.1. Copa Overview

As a delay-based congestion control algorithm, Copa defines an objective function that includes the average throughput $\lambda$ of the flow and the packet queuing delay $d$:$U = \log \lambda - \delta \log d$. According to the author's proof, when the data transmission rate is $\lambda = 1/(\delta \cdot d_q)$, $U$ achieves the maximum value.

Copa defines two directions: "Up" and "Down". When an ACK is received, CWND increases if the direction is "Up" and decreases if the direction is "Down" so that the transmission rate converges to the target rate. The value of CWND changes dynamically each time to speed up the convergence of the algorithm. Specifically, the algorithm performs the following steps when an ACK is received:

(1) Update packet queuing delay $d_q$:

$$d_q = RTT_{standing} - RTT_{min} \tag{1}$$

where $RTT_{standing}$ is the minimum value within half the standard SRTT and $RTT_{min}$ is the minimum RTT within 10 s.

(2) Calculate the target transmission rate $\lambda$:

$$\lambda_{target} = 1/(\delta \cdot d_q) \tag{2}$$

(3) Calculate the current transmission rate. Then, determine the direction:

$$\lambda_{current} = cwnd/RTT_{standing} \tag{3}$$

where $cwnd$ is the size of the current congestion window. If $\lambda_{current}$ is less than $\lambda_{target}$, then $cwnd = cwnd + v/(\delta \cdot d_q)$, and set the direction to "Up", where $v$ is defined by step (4).

If $\lambda_{current}$ is greater than $\lambda_{target}$, then $cwnd = cwnd - v/(\delta \cdot d_q)$, and change the direction to "Down".

(4) The parameter $v$ is used to speed up the convergence, and its initial value is one. If the current direction is different from the last time, $v$ will be reset to one. If the direction is the same, and the direction remains unchanged for three consecutive RTTs, then $v$ is doubled.

In addition, when data starts to be transmitted, Copa will also have a slow start process, in which the congestion window size increases exponentially until $\lambda_{target}$ is no longer greater than $\lambda_{current}$. The sender of Copa uses pacing and sends packets at twice the $cwnd/RTT_{standing}$ rate. The default value of $\delta$ is 0.5, and when the Copa algorithm competes with buffer-filling schemes, $\delta$ will be reduced.

#### 4.2.2. Improved Calculation of the Queuing Delay

Delay-based congestion control algorithms are mostly based on changes in the RTT to sense the delay and reduce the data transmission rate as the delay increases. During the transmission of an NDC, there are two directions of packet transmission: one is REQ packet transmission from the receiver to the replica node, and the other is DATA packet transmission from the replica node to the receiver. When the queuing delay of DATA packet transmission increases, the congestion control algorithm reduces the transmission rate so that the newly added NDC transmission process can quickly reach a reasonable transmission rate. When the queuing delay of REQ packet transmission increases, the congestion control algorithm will also reduce the transmission rate of the NDC. However, since there are much fewer REQ packets than DATA packets, and their size is much smaller than that of the DATA packets, REQ packets occupy very little bandwidth and cannot achieve the effect of giving up bandwidth to the newly added NDC transmission process. In addition, the increase in the REQ packet queuing delay does not mean an increase in network congestion in the opposite direction. Blindly reducing the transmission rate of the NDC will only cause insufficient utilization of bandwidth in the opposite direction. Therefore, when the queuing delay in one direction increases, reducing the rate of the REQ packets in that direction should be avoided, but one should consider reducing the rate of the DATA packets.

Copa, as a delay-based congestion control algorithm, also has the above problems. It uses the difference between the minimum RTT in $SRTT/2$ and the minimum RTT in 10 s as an input parameter to calculate a target rate, where the difference is inversely proportional to the calculated target rate. Thus, as the queuing delay experienced by REQ packets increases, the target rate decreases.

A reasonable approach is that the calculated target rate should only be related to the queuing delay experienced by DATA packets. This requires the measurement of the queuing delay of the DATA packets. When there is no clock synchronization problem between the replica node and the receiver, we can easily obtain the transmission delay of the DATA packet. However, there is a common clock asynchrony between devices, and the clock synchronization between different devices is a complex question. Therefore, we identify the clock difference between the receiver and the replica node as $clock_{delta}$. The transmission delay of the DATA packet is

$$Tdat = Trecv - Treplica + clock_{delta} \tag{4}$$

where $Treplica$ is the timestamp of the replica node which sends the DATA packet and $Trecv$ is the timestamp of the receiver which receives the DATA packet.

Take the minimum value $Tdat_{min}$ of $Tdat$ within 10 s and the minimum value $Tdat_{standing}$ within $SRTT/2$. Therefore, the queuing delay of the DATA packet is

$$T_q = Tdat_{standing} - Tdat_{min} \tag{5}$$

From Equations (4) and (5), we get the following expression:

$$T_q = (Trecv_{standing} - Trecv_{min}) - (Treplica_{standing} - Treplica_{min}) \quad (6)$$

According to the expression in Equation (6), we know that the clock being asynchronous between the receiver and the replica node will not affect the value of $T_q$.

We use the queuing delay $T_q$ of the DATA packets to replace the queuing delay of the round-trip path in the Copa algorithm so that we can still calculate a stable target rate when the queuing delay of the REQ packets increases. However, the target rate remains unchanged, and the increase in queuing delay experienced by the REQ packets will lead to an increase in the RTT, and the bandwidth delay product (BDP) of the link will increase. At this time, *cwnd* should also increase. Because Copa decides whether to increase *cwnd* according to the size of the current transmission rate and the target rate, the current rate is $cwnd/RTT_{standing}$, and when the REQ packet queuing delay increases, the RTT will become larger, and the current rate will be lower than the target rate, leading to *cwnd* getting larger. Therefore, our improvement to $d_q$ computation is reasonable.

#### 4.2.3. Improved Accuracy of the Queuing Delay

The original Copa algorithm uses the minimum RTT within 10 s as the round-trip propagation time (RTprop) of the link, which means that the measured RTprop will be larger than the actual value until the link queue is completely emptied. Different NDC transmission processes on the same link may have different measured RTprop values. From the expression in Equation (1), we know that the queuing delay is related to $RTT_{standing}$ and $RTT_{min}$. A more accurate measured RTprop means a smaller $RTT_{min}$ and a larger queuing delay. This will adversely affect the fairness of the Copa algorithm. We improved the measurement of the queuing delay $T_q$, but the problem still exists, as the NDC transmission process with a smaller $Tdat_{min}$ value will obtain a smaller data transmission rate. On the other hand, the size of NDC is limited, the transmission time is often less than 10 s, and the practice of using the smallest measurement value within 10 s is difficult to apply to ICN scenarios.

In order to measure the link propagation delay quickly, the scheme of setting higher priority in the IP header of the packet can be used. Whether it is the IPv4 or IPv6 protocol, there is a differentiated services (DS) field. When an NDC starts to transmit, the receiver sets an identification bit in the transport layer header of the first two REQ packets, and the replica node sets a higher priority in the IP header of the reply DATA packet after identifying the identification bit. Finally, the receiver will be able to measure a minimum value of $Tdat_{min}$. In addition, during the NDC transmission process, link changes may occur due to replica node switching and other reasons. At this time, the congestion control algorithm needs to be restarted, and the propagation delay should also be remeasured.

#### 4.2.4. Improved Convergence Speed of Copa

In ICN, the transmission time of an NDC is not very long most of the time. If the convergence speed of the congestion control algorithm is slow, the transmission process of the shared bottleneck link is at an unfair transmission rate most of the time. The Copa algorithm performs well in terms of convergence speed, but it still cannot meet the needs of ICN scenarios. Therefore, we improved the convergence speed of the Copa algorithm.

The average number of packets queued in the bottleneck buffer for an NDC transmission process that reaches a steady rate is $1.25/\delta$. The value of $\delta$ determines how much to weigh the delay compared to the throughput. When the $\delta$ value is small, the number of queued packets in the bottleneck buffer will increase, which will increase the delay, but it is beneficial to improve the throughput of the transmission process. In the same bottleneck buffer, the transmission process with a lower rate should have a stronger aggressiveness than the transmission process with a higher rate so as to obtain a higher transmission rate. $\delta$ is a parameter that can control the aggressiveness. Therefore, we can dynamically adjust

the value of $\delta$ according to the transmission rate to accelerate the convergence process of the algorithm.

We consider constructing a function between $\delta$ and the transmission rate to achieve the effect of $\delta$ dynamically changing with the transmission rate. Since our goal is to increase the rate of those transmission processes at lower rates, $\delta$ should be as small as possible in this case. Considering that the function is called every time a DATA packet is received, the required function should have low complexity in order to reduce computational overhead. For these reasons, we tried to find functions that meet the requirements. We tested some functions, and we found that the expression in Equation (7) satisfied our needs. The first derivative of the expression in Equation (7) is $2 \cdot a \cdot rate$. When the *rate* is small, $\delta$ increases slowly so that it can keep a small value. In addition, the expression in Equation (7) has low complexity:

$$\delta = a \cdot rate^2 + b(rate > 0, a > 0, b > 0) \tag{7}$$

We set $b$ to 0.05 and $a$ to $4.5 \times 10^{-5}$, and *rate* is in units of Mbps. Therefore, the minimum value of $\delta$ is 0.05. When the transmission rate is 100 Mbps, $\delta$ is the default value of 0.5. Considering that $\delta$ should not exceed the default value, when the transmission rate is higher than 100 Mbps, $\delta$ will be the default value of 0.5. When the transmission rate is lower than 100 Mbps, there will be a strong aggressiveness, and because the minimum value of $\delta$ is limited, the aggressiveness will not be too strong. It should be noted that $b$ and $a$ can also be set to other values, and no specific value is optimal. Increasing the speed at which a slower transmission process rate converges to a fair rate is always accompanied by occupancy of the bottleneck buffer.

In this way, we obtain a scheme to improve the convergence speed of the algorithm. When the transmission rate of the transmission process through the bottleneck link is different, the transmission process with a low transmission rate has a smaller delta value and is more aggressive. When several transmissions have the same rate, they have the same $\delta$, so they can still converge to a stable state. Figure 3 is the relationship between $\delta$ and the transmission rate.

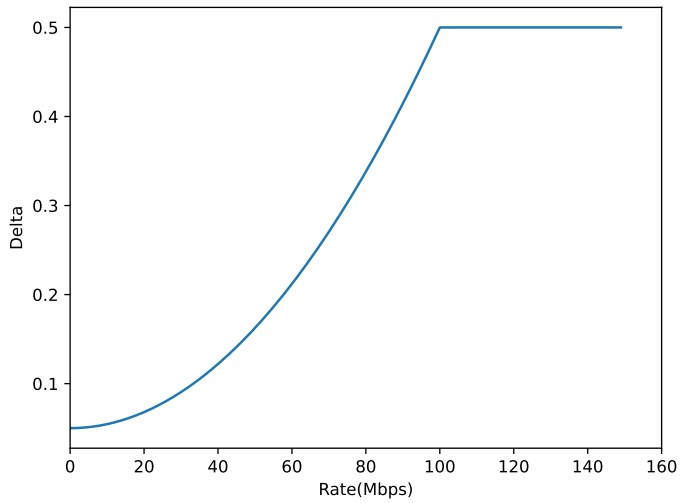


**Figure 3.** The relationship between $\delta$ and the transmission rate.

## 5. Experimental Evaluation and Analysis

NS-3 is an easy-to-use discrete event network simulator. The protocol stack proposed in Section 3 has been implemented in NS-3, which has a set of easy-to-use congestion control interfaces to facilitate the implementation of new congestion control algorithms. Mvfst [46] is Facebook's implementation of the QUIC protocol, which includes a Copa code. We referred to the Copa code of mvfst and implemented the Copa algorithm in our protocol stack. Meanwhile, Copa-ICN was implemented by modifying the Copa code. In this section, we experimentally evaluate Copa-ICN using the NS-3 platform.

### *5.1. Experiment Set-Up*

The simulation experiment mainly included three kinds of nodes: the receiver that initiated the NDC request, the replica node that cached the NDC, and the ICN/IP router for forwarding REQ packets and DATA packets. The nomenclature of the device was determined by its role in the simulation experiment. For example, the replica node may have actually been a server, a router with an ICN cache, or even a user device. We placed NDCs 1 MB, 5 MB, 10 MB, 20 MB, 30 MB, 40 MB, and 50 MB in size on the replica nodes. Since not all algorithms have a high convergence speed, in order to fully observe the performance of different congestion control algorithms in transmitting NDCs, some scenarios chose to transmit the NDC with a size of 50 MB. In practical scenarios, the NDC is often much smaller, and the algorithm with a faster convergence speed will have a greater advantage at this time. Additionally, the header of the DATA packet is 108 bytes, and the valid data contained are a maximum of 1250 bytes. The REQ packet size is generally 106 bytes. Our experiments used two topologies, shown in Figures 4 and 5.

Figure 4 shows simulation experiment network topology 1 (topo-1). It is a linear topology with $n$ receivers (RE1-REn), one replica node (RN), and two routers (IR1 and IR2).

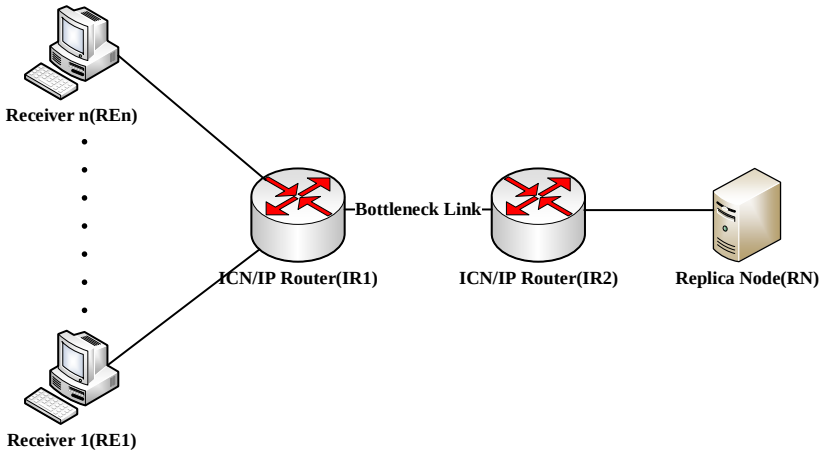


**Figure 4.** Experimental network topology 1.

Figure 5 shows simulation experiment network topology 2 (topo-2). It contains $n$ receivers (RE1-REn), two replica nodes (RN1 and RN2), and two routers (IR1 and IR2).

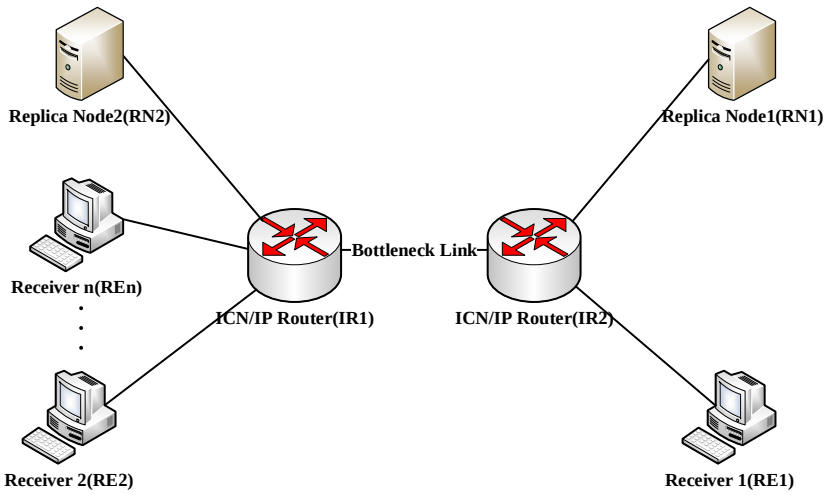


**Figure 5.** Experimental network topology 2.

### *5.2. Comparison of Copa-ICN with BBR, Cubic, and NewReno*

In topo-1, we compared the convergence speed of the Copa-ICN algorithm with the BBR, Cubic and NewReno algorithms. We set up two experimental scenarios. In the first scenario, the bottleneck bandwidth of the link was 20 Mbps, RTprop was 30 ms, the size of NDC-1 was 10 MB, and the size of NDC-2 was 1 MB. In the second scenario, the bottleneck bandwidth of the link was 60 Mbps, RTprop was 50 ms, the size of NDC-1 was 50 MB, and the size of NDC-2 was 10 MB. At 0.1 s, RE1 began to request the transmission of NDC-1, and at 2.1 s, RE2 began to request the transmission of NDC-2.

Figure 6 shows the variation of inflight data over time in two transmission processes in different congestion control algorithms. Before NDC-2 was added, Copa-ICN and BBR had the smallest inflight data, of which Copa-ICN had the smallest extreme value, while NewReno and Cubic had larger inflight data because NewReno and Cubic were buffer-filling algorithms, while Copa-ICN is based on the queuing delay and the BBR algorithm is based on the BDP. Therefore, the inflight of the latter two was small. After the addition of NDC-2, the two transmission processes of the Copa-ICN algorithm converged to fairness quickly, while the Cubic and BBR algorithms failed to converge to fairness. In Figure 6(7), NewReno converges to fairness because the congestion window of NDC-2 accidentally increased to a value similar to that of NDC-1 at the end of the slow start. For the Cubic and NewReno algorithms, only when the packet loss reduced the size of the congestion window multiplicatively did the transmission processes with different rates have a chance to converge; otherwise, the congestion windows of the two transmission processes increased together. A transmission with a smaller congestion window will never catch up with a transmission with a larger congestion window. BBR relies on periodically changing the transmission rate gain and periodically entering the detection minimum RTT phase to empty the bottleneck buffer and make the transmission process converge. The former has less impact on the transmission rate, and the latter has a very low trigger frequency. The default was about 10 s, so the overall convergence speed of BBR was slower.

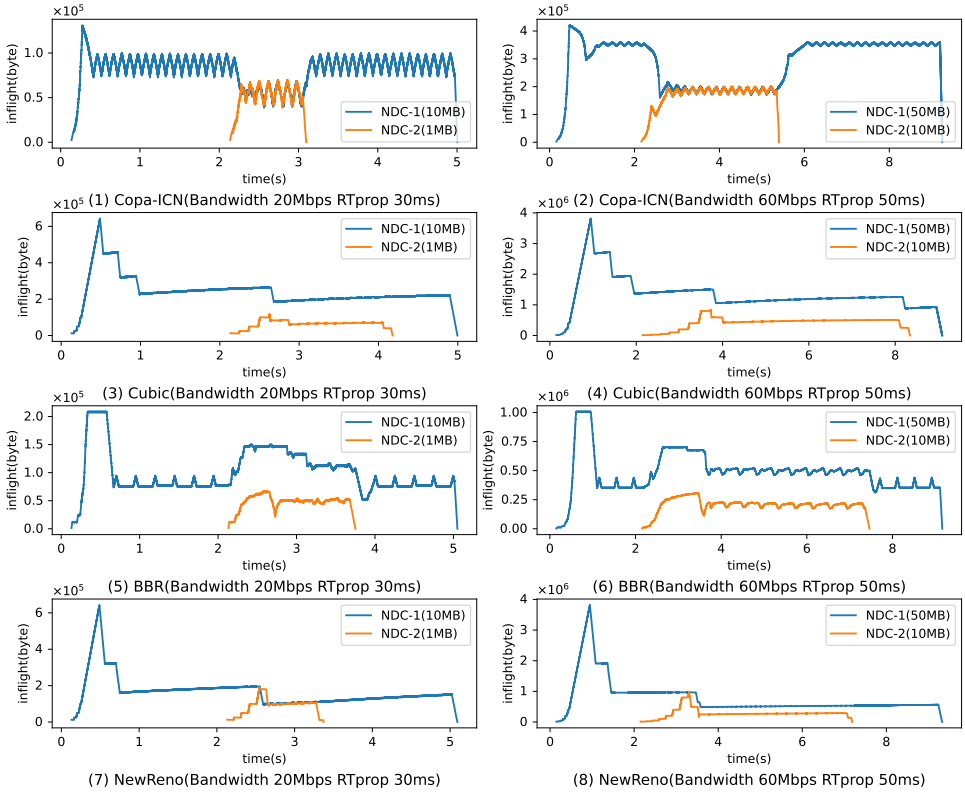


**Figure 6.** Variation of inflight over time for different factors (algorithm, bottleneck bandwidth, RTprop, and NDC size).

When using different congestion control algorithms, the throughputs of the two transmission processes joined at different times are shown in Figure 7. The maximum throughputs in the figure are not 20 Mbps and 60 Mbps due to the existence of the packet header. The relationship between the maximum throughput (*Throughput*) and the bottleneck bandwidth (*BW*) is

$$Throughput = BW/(PH + PP) * PP \tag{8}$$

*PH* is the header size of the DATA packet, and its value is 108 bytes. *PP* is the maximum value of the DATA packet payload, and its value is 1250 bytes. Therefore, for the bottleneck links of 20 Mbps and 60 Mbps, the maximum throughputs that could be achieved by the NDC transmission process were 18.41 Mbps and 55.23 Mbps, respectively. It can be seen from Figure 7 that the addition of NDC-2 reduced the throughput of NDC-1, and the throughputs of the two NDCs in Copa-ICN would soon tend to be the same. Although the throughput of NDC-1 in the Cubic and BBR algorithms was reduced, it was greater than that of NDC-2. The performance of the NewReno algorithm was unstable, although there was better performance in the experiment in Figure 7(7), while the performance of the experiment in Figure 7(8) was poor.

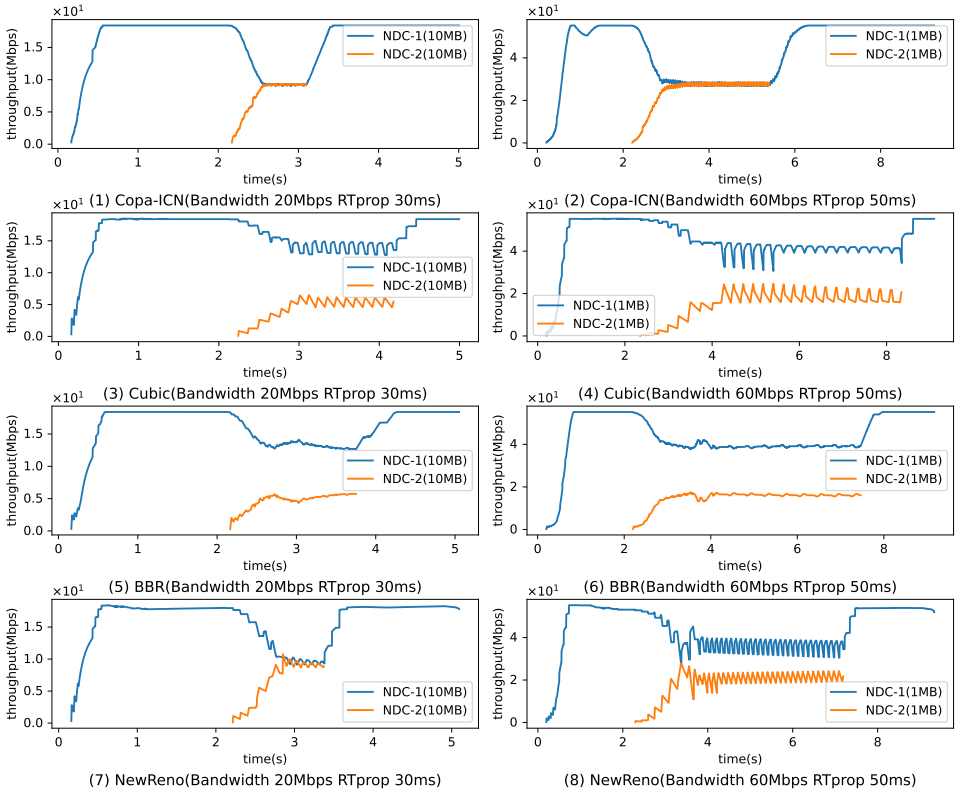


**Figure 7.** Variation of throughtput with time for different factors (algorithm, bottleneck bandwidth, RTprop, and NDC size).

Since the fluctuation of the RTT value was relatively large, in order to clearly observe the change in the RTT, we used the smooth RTT (SRTT), where $SRTT = 0.8 * SRTT + 0.2 * RTT$. Figure 8 shows the variation of the SRTT over time in the transmission process using different congestion control algorithms. It can be seen that the Copa-ICN algorithm always had the smallest delay, while the BBR and NewReno algorithms had relatively large latency, and Cubic was almost always algorithm with the longest delay.

Compared with the receiver-driven implementation of the other three common algorithms, which failed to converge to a fair rate, Copa-ICN not only successfully converged but also had a fast convergence speed. In addition, Copa-ICN also had the lowest la-

tency. After the start of the second transmission process, the average latency of Copa-ICN was more than 37% lower than those of the other three algorithms. This shows the correctness of using Copa-ICN instead of several other algorithms in the ICN scenario.

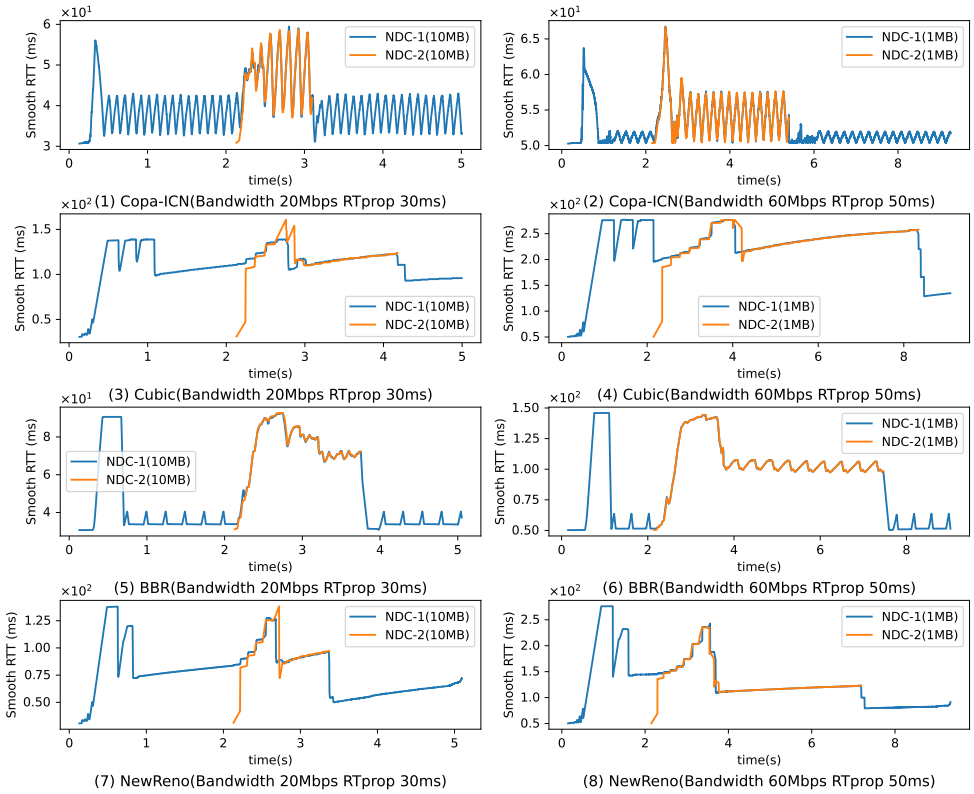


**Figure 8.** Variation of smooth RTT over time for different factors (algorithm, bottleneck bandwidth, RTprop, and NDC size).

### *5.3. Comparison of Copa-ICN with Original Copa*

We changed the Copa algorithm to a receiver-driven model and improved it. To verify the effectiveness of the improvement, we set up four experiments to compare the Copa-ICN and Copa algorithms.

#### 5.3.1. Convergence Experiment of NDC Transmission Process

In topo-1, we set up two experimental scenarios to compare the performance of the Copa-ICN and Copa algorithms in terms of convergence speed. The experimental parameters of the two experimental settings are shown in Table 2. In Experiment-1, RE1 started requesting the NDC at 0.1 s and RE2 at 2.1 s. In Experiment-2, RE1–RE4 started requesting the NDC at 0.1, 2.1, 4.1, and 6.1 s, respectively.

**Table 2.** NDC transmission process convergence experiment parameter settings.

| Parameter | Experiment-1 | Experiment-2 |
|---|---|---|
| Receiver | RE1, RE2 | RE1–RE4 |
| Bandwidth | 20 Mbps, 50 Mbps | 50 Mbps |
| RTprop | 50 ms, 80 ms | 50 ms |
| ICN/IP Router | IR1, IR2 | IR1,IR2 |
| Replica Node | RN | RN |
| Topology | topo-1 | topo-1 |

It should be noted that, according to the expression in Equation (8), when the bottleneck bandwidth was 20 Mbps and 50 Mbps, the maximum throughputs were 18.41 Mbps

and 46.02 Mbps, respectively. In addition, for the convenience of observing the experimental phenomenon, the size of the NDC was set to 50 MB.

Figure 9 shows the change in throughput over time during the first 10 s of Experiment-1. When the second transmission process was added, the throughput of the first transmission process would decrease, and the throughput of the second transmission process would continue to increase so that the throughputs of the two transmission processes would continue to tend to be the same. Observing the time required for the throughput of the transmission process under different bandwidths and RTprops to become the same, it can also be seen that with the increase in the bandwidth and RTprop, the convergence speed of the Copa-ICN and Copa algorithms would become slower and slower. However, the variation in Copa-ICN's convergence speed was much smaller than that of Copa. The time it took for Copa-ICN to converge to a fair rate was only about 20% of that of Copa.

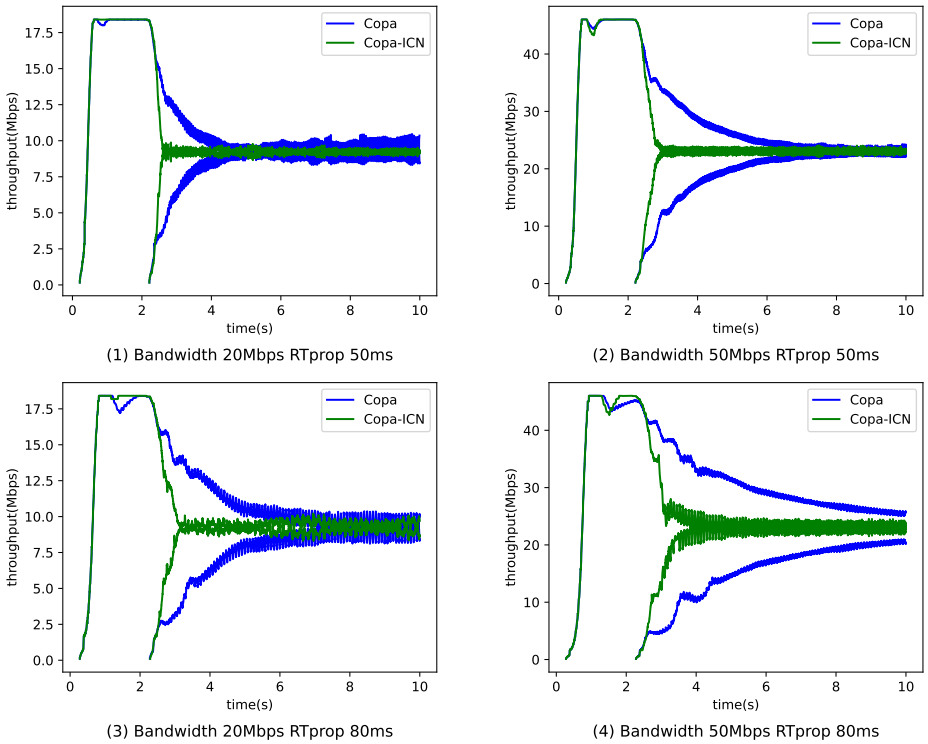


**Figure 9.** Convergence of the two transmission processes for different bandwidths and delays.

The variations of multiple transmission processes over time in the first 10 s of Experiment-2 are shown in Figure 10. It can be seen that with the Copa-ICN algorithm, the throughputs of different transmission processes converged to the same value quickly when a transmission process was added. With the Copa algorithm, however, it took much longer for the throughputs of the different processes to become the same, and a better convergence rate was achieved only when there were many transmission processes in the link. The reason for this is that when there are many transmission processes, the fair share of the throughput of each process becomes smaller, and so the convergence time required is reduced.

A dynamic $\delta$ value was used in Copa-ICN with a maximum value of 0.5. The Copa algorithm used a fixed default value of 0.5. According to the calculation in [25], the number of packets in the bottleneck buffer of a transmission process is $1.25/\delta$. Therefore, using the Copa-ICN algorithm will have more data packets queued in the bottleneck buffer. It can be seen from Figure 11 that the queue length of the bottleneck buffer area of the Copa-ICN algorithm was significantly larger than that of the Copa algorithm. When the fourth transmission process started, the average queue length of Copa-ICN was eight times that of Copa. Moreover, after the third NDC transmission process was added, the packets

in the buffer area had not even been emptied, meaning that it was difficult to measure the RTprop of the link during the transmission process, and thus the measured queuing delay would be too small. If the queuing delay measured by the transmission process added later is too small, it will occupy more bandwidth in the competition with other transmission processes, resulting in unfairness between transmission processes. However, as can be seen from Figure 10, Copa-ICN still maintained good fairness at this time, which shows that our improvement to the queuing delay accuracy was effective.

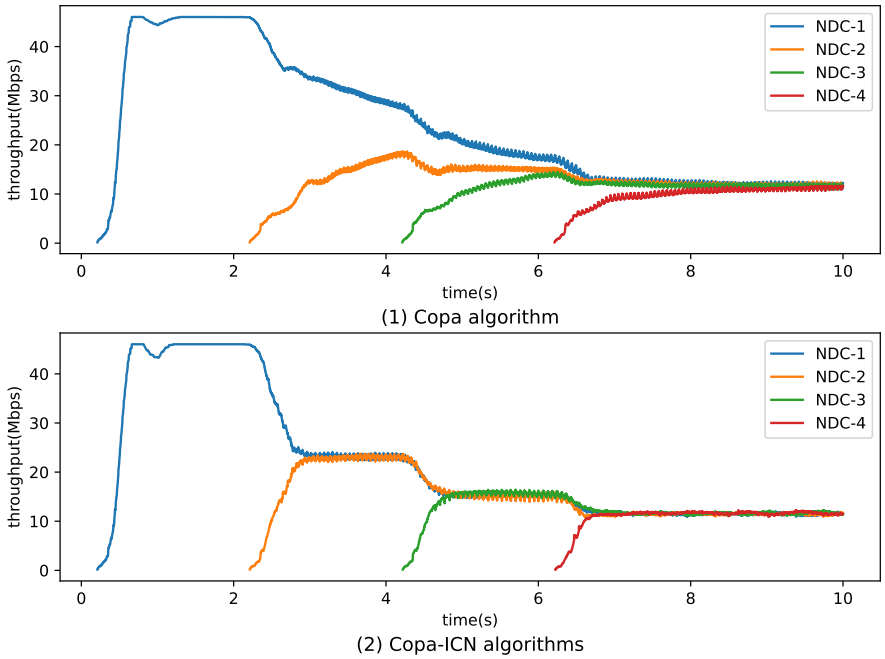


**Figure 10.** The throughput of the transmission process started at 4 different times over time.

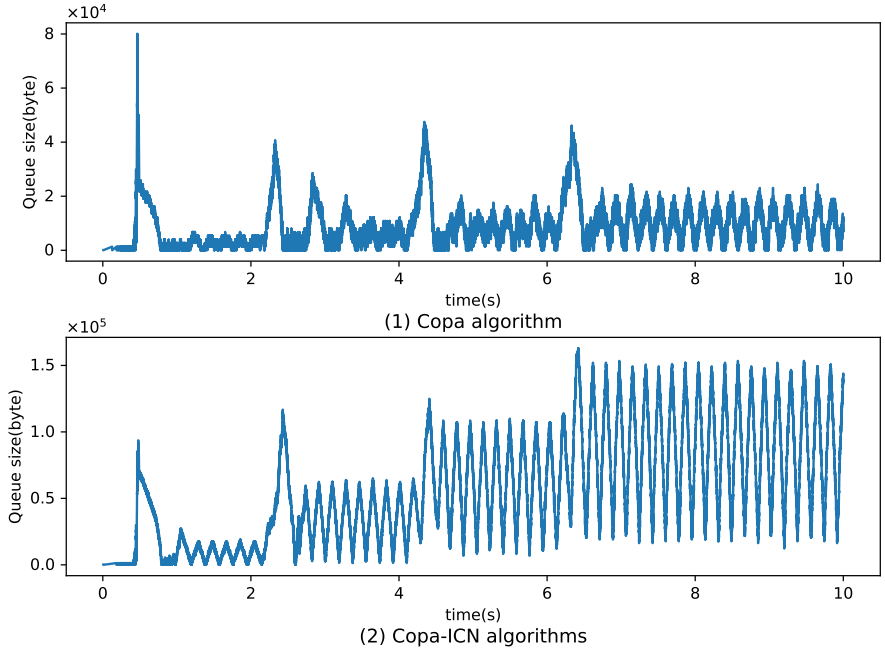


**Figure 11.** The bottleneck buffer queue of the transmission process started at 4 different times over time.

5.3.2. Coexistence with the NDC Transmission Process in the Opposite Direction

In topo-2, we conducted throughput experiments with the Copa-ICN and Copa algorithms in the presence of NDC transmission processes in opposite directions. NDCs were placed at RN1 and RN2. The bottleneck bandwidth and RTprop settings of the link were 20 Mbps and 40 ms and 50 Mbps and 80 ms, respectively. At 0.1 s, $n-1$ receivers (RE2-REn) would start requesting the NDC from RN1 as a reverse NDC transmission process, where n had a value of 1, 2, and 3. At 2.1 s, RE1 started requesting the NDC from RN2.

Figure 12 shows the throughput changes in RE1 upon obtaining the NDC from RN2 under different parameters. It can be seen that in any case, when there was no NDC transmission process in the opposite direction, RE1 obtained the maximum throughput, and when there was an NDC transmission process in the opposite direction, the throughput of RE1 would decrease. In the case of using the Copa algorithm, the throughput would continue to decrease as the number of transmissions in opposite directions increased. However, in the case of using the Copa-ICN algorithm, the throughput would not change much as the number of transmissions in the opposite direction increased. In the network with a bandwidth of 50 Mbps and RTprop of 80 ms, when there were two transmission processes in opposite directions, the throughput of Copa was reduced to 39%, while that of Copa-ICN was 87%. Copa-ICN could still maintain good performance.

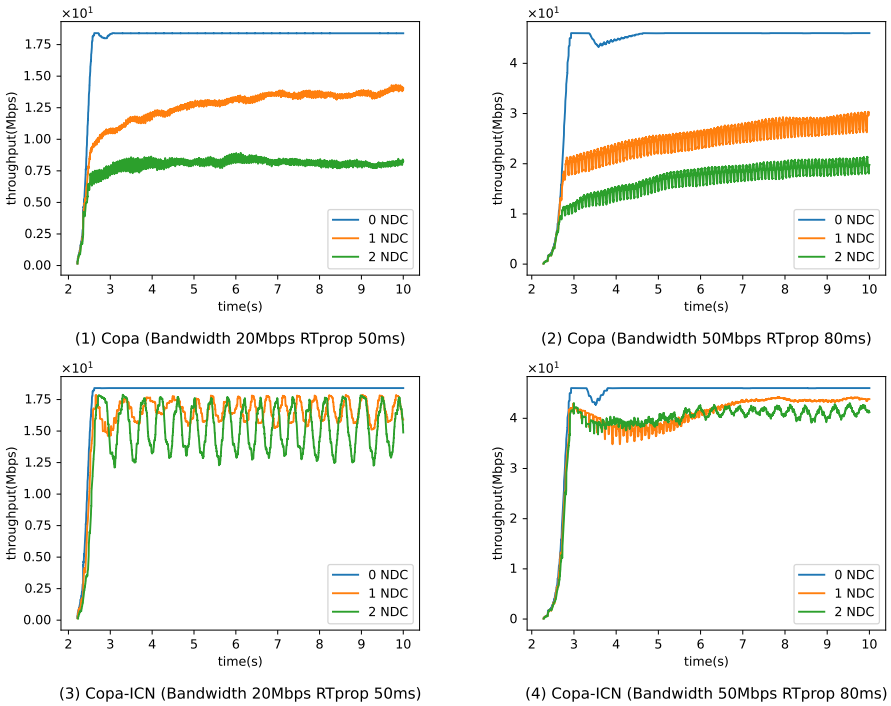


**Figure 12.** Variation of throughput over time for different factors (algorithm, bottleneck bandwidth, RTprop, and number of NDC transmission processes in opposite directions).

5.3.3. Experiment with Different-Sized NDCs in Transmission

We measured the flow completion time (FCT) for NDCs of different sizes in topology 1. The NDC was placed in the RN. The bottleneck bandwidth of the link was 100 Mbps, and the RTprop was 60 ms. We used RE1–RE5 to request the NDC from the RN as 5 background traffic at 0 s, and RE6 started to request the RN to transmit the NDCs of 1 MB, 5 MB, 10 MB, 20 MB, 30 MB, 40 MB and 50 MB respectively at the 8th second.

According to the expression in Equation (8), the maximum throughput was 92.05 Mbps when there were 6 NDC transmission processes. The average throughput of each transmission process was 15.34 Mbps, so we could calculate the optimal FCTs of NDC transmission of different sizes. Figure 13 shows the relationship between the ratio of the FCT measurement value and the optimal value of the NDC and the size of the NDC. As can be seen

from the figure, as the size of the NDC increased, the gap between the FCT measurement and the optimal value became smaller and smaller. When the NDC size was 1 MB, the FCT of Copa-ICN was 85% of that of Copa, and when the NDC size was 50 MB, the FCT of Copa-ICN was 97% of that of Copa. The reason for this is that with the increase in the NDC size, the transmission time will be longer, and when the network conditions remain unchanged, the convergence time of different transmission processes is certain, which will make the transmission process in the optimal throughput time longer. Therefore, it is closer to the optimal FCT value. In the figure, the FCT of Copa-ICN consistently outperformed Copa. The reason for this is that when the Copa-ICN algorithm is used, the transmission process converges faster, which also increases the time during which the transmission process is at the best throughput, resulting in a better FCT.

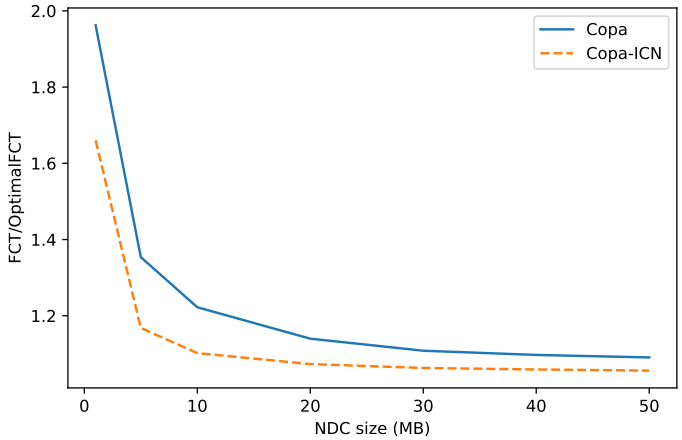


**Figure 13.** The ratio of the FCT to the optimal value for NDCs of different sizes.

## 6. Discussion

The convergence speed is an important indicator to measure the congestion control algorithm. In ICN, especially in some architectures of the stand-alone name resolution approach, the convergence speed of the algorithm is even more important. In this paper, we proposed a congestion control mechanism based on the Copa algorithm, called Copa-ICN, which is suitable for the IP network-compatible ICN architecture. Copa-ICN not only has huge advantages over the BBR, Cubic, and NewReno algorithms in terms of convergence speed, fairness, and delay, but it also has two important improvements over the original Copa algorithm. One is that the Copa-ICN algorithm converges faster without reducing fairness, and the other is that the throughput of the algorithm is significantly improved when there is an NDC transmission process in the opposite direction.

The Copa-ICN algorithm is easy to deploy. Although we only implemented it in NS-3, there are no technical difficulties when Copa-ICN is implemented in practical systems. First, the protocol stack supports the mechanism where the receiver sets the flag bit in the REQ packet to notify the replica node to stamp the device time in the DATA packet. Second, the protocol stack needs to modify the congestion control algorithm. Because the protocol stack often has a set of easy-to-use congestion control algorithm interfaces, just like the congestion control interface of the Linux kernel protocol stack, by using the congestion control algorithm interface, we can easily implement various congestion control algorithms, including Copa-ICN.

In the ICN architecture compatible with IP networks, how to coexist with TCP traffic is an important issue. TCP traffic may use a variety of different congestion control algorithms, which will challenge the convergence speed of the Copa-ICN algorithm. This will be the content of our further research in the future.

**Author Contributions:** Methodology, Z.W., H.N. and R.H.; software, Z.W.; writing—original draft preparation, Z.W.; writing—review and editing, Z.W., H.N. and R.H.; supervision, R.H.; project administration, R.H.; funding acquisition, H.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

**Acknowledgments:** We would like to express our gratitude to Jinlin Wang, Yingjie Duan, and Yaqin Song for their meaningful support in this work.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| CWND | Congestion window |
| EID | Entity-ID |
| FCT | Flow completed time |
| ICN | Information-centric networking |
| MSS | Maximum segment size |
| NA | Network address |
| NDC | Named data chunk |
| NRS | Name resolution system |
| RTT | Round-trip time |
| SRTT | Smoothed round trip time |

## References

1. Peng, G. CDN: Content distribution network. *arXiv* **2004**, arXiv:cs/0411069.
2. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutorials* **2013**, *16*, 1024–1049. [CrossRef]
3. Fazea, Y.; Mohammed, F. Software Defined Networking based Information Centric Networking: An Overview of Approaches and Challenges. In Proceedings of the 2021 International Congress of Advanced Technology and Engineering (ICOTEN), Virtual, 4–5 July 2021; pp. 1–8.
4. Badshah, J.; Kamran, M.; Shah, N.; Abid, S.A. An improved method to deploy cache servers in software defined network-based information centric networking for big data. *J. Grid Comput.* **2019**, *17*, 255–277. [CrossRef]
5. Nour, B.; Sharif, K.; Li, F.; Biswas, S.; Moungla, H.; Guizani, M.; Wang, Y. A survey of Internet of Things communication using ICN: A use case perspective. *Comput. Commun.* **2019**, *142*, 95–123. [CrossRef]
6. Tsiropoulou, E.E.; Paruchuri, S.T.; Baras, J.S. Interest, energy and physical-aware coalition formation and resource allocation in smart IoT applications. In Proceedings of the 2017 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 22–24 March 2017; pp. 1–6.
7. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
8. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Orlando, FL, USA, 9–12 December 2009; pp. 1–12.
9. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. Mobilityfirst: A mobility-centric and trustworthy internet architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
10. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.
11. Detti, A.; Blefari Melazzi, N.; Salsano, S.; Pomposini, M. CONET: A content centric inter-networking architecture. In Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, ON, Canada, 15–19 August 2011; pp. 50–55.
12. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In *International Conference on Broadband Communications, Networks and Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–13.
13. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of information (netinf)–an information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
14. Wang, J.; Cheng, G.; You, J.; Sun, P. SEANet: Architecture and Technologies of an On-site, Elastic, Autonomous Network. *J. Netw. New Media* **2020**, *6*, 1–8.

15. Dong, L.; Wang, G. A hybrid approach for name resolution and producer selection in information centric network. In Proceedings of the 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA, 5–8 March 2018; pp. 574–580.
16. Salsano, S.; Detti, A.; Cancellieri, M.; Pomposini, M.; Blefari-Melazzi, N. Transport-layer issues in information centric networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Macao, China, 24–26 September 2012; pp. 19–24.
17. Song, Y.; Ni, H.; Zhu, X. Analytical Modeling of Optimal Chunk Size for Efficient Transmission in Information-Centric Networking. *Int. J. Innov. Comput. Inf. Control* **2020**, *16*, 1511–1525.
18. Jacobson, V. Congestion avoidance and control. *ACM Sigcomm Comput. Commun. Rev.* **1988**, *18*, 314–329. [CrossRef]
19. Liu, Z.; Sun, J.; Hu, S.; Hu, X. An adaptive AQM algorithm based on a novel information compression model. *IEEE Access* **2018**, *6*, 31180–31190. [CrossRef]
20. Polese, M.; Chiariotti, F.; Bonetto, E.; Rigotto, F.; Zanella, A.; Zorzi, M. A survey on recent advances in transport layer protocols. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3584–3608. [CrossRef]
21. Song, Y.; Ni, H.; Zhu, X. Two-Level Congestion Control Mechanism (2LCCM) for Information-Centric Networking. *Future Internet* **2021**, *13*, 149. [CrossRef]
22. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* **2016**, *14*, 20–53. [CrossRef]
23. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
24. Floyd, S.; Henderson, T.; Gurtov, A.; Nishida, Y. *The NewReno Modification to TCP's Fast Recovery Algorithm*; IETF RFC: Fremont, CA, USA, 1999.
25. Arun, V.; Balakrishnan, H. Copa: Practical {Delay-Based} Congestion Control for the Internet. In Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), Renton, DC, USA, 9–11 April 2018; pp. 329–342.
26. NS-3 Project. Available online: https://www.nsnam.org (accessed on 23 March 2022).
27. Su, K.; Bronzino, F.; Ramakrishnan, K.; Raychaudhuri, D. Mftp: A clean-slate transport protocol for the information centric mobilityfirst network. In Proceedings of the 2nd ACM Conference on Information-Centric Networking, San Francisco, CA, USA, 30 September–2 October 2015; pp. 127–136.
28. Ahlgren, B.; Hurtig, P.; Abrahamsson, H.; Grinnemo, K.J.; Brunstrom, A. ICN congestion control for wireless links. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
29. Wang, Y.; Rozhnova, N.; Narayanan, A.; Oran, D.; Rhee, I. An improved hop-by-hop interest shaper for congestion control in named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 55–60. [CrossRef]
30. Siddiqui, S.; Waqas, A.; Khan, A.; Zareen, F.; Iqbal, M.N. Congestion Controlling Mechanisms in Content Centric Networking and Named Data Networking—A Survey. In Proceedings of the 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 30–31 January 2019; pp. 1–7.
31. Oueslati, S.; Roberts, J.; Sbihi, N. Flow-aware traffic control for a content-centric network. In Proceedings of the 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–27 June 2012; pp. 2417–2425.
32. Schneider, K.; Yi, C.; Zhang, B.; Zhang, L. A practical congestion control scheme for named data networking. In Proceedings of the 3rd ACM Conference on Information-Centric Networking, Kyoto, Japan, 26–28 September 2016; pp. 21–30.
33. Carofiglio, G.; Gallo, M.; Muscariello, L. ICP: Design and evaluation of an interest control protocol for content-centric networking. In Proceedings of the 2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; pp. 304–309.
34. Albalawi, A.A.; Garcia-Luna-Aceves, J. A delay-based congestion-control protocol for information-centric networks. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; pp. 809–815.
35. Rossi, D.; Testa, C.; Valenti, S.; Muscariello, L. LEDBAT: The new BitTorrent congestion control protocol. In Proceedings of the 2010 Proceedings of 19th International Conference on Computer Communications and Networks, Zurich, Switzerland, 2–5 August 2010; pp. 1–6.
36. Saino, L.; Cocora, C.; Pavlou, G. CCTCP: A scalable receiver-driven congestion control protocol for content centric networking. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 3775–3780.
37. Liu, Y.; Piao, X.; Hou, C.; Lei, K. A cubic-based explicit congestion control mechanism in named data networking. In Proceedings of the 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Chengdu, China, 13–15 October 2016; pp. 360–363.
38. Potys, R.A.; Ali, N.M.; Marsh, I.; Osmani, F. NetInf TP: A receiver-driven protocol for ICN data transport. In Proceedings of the 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), Portland, OR, USA, 15–16 June 2015; pp. 267–272.
39. Li, J.; You, J.; Deng, H. Adjacency-Information-Entropy-Based Cooperative Name Resolution Approach in ICN. *Future Internet* **2022**, *14*, 68. [CrossRef]
40. Postel, J. *Rfc0791: Internet Protocol*; IETF RFC: Fremont, CA, USA, 1981.
41. Deering, S.; Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification*; IETF RFC: Fremont, CA, USA, 1998.

42. Paxson, V.; Allman, M.; Chu, J.; Sargent, M. *Computing TCP's Retransmission Timer*; Technical Report, rfc 2988; IETF RFC: Fremont, CA, USA, 2000.
43. Aggarwal, A.; Savage, S.; Anderson, T. Understanding the performance of TCP pacing. In Proceedings of the IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064), Tel Aviv, Israel, 26–30 March 2000; Volume 3, pp. 1157–1165.
44. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The quic transport protocol: Design and internet-scale deployment. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; pp. 183–196.
45. Nitin, G. Evaluating COPA Congestion Control for Improved Video Performance. Available online: https://engineering.fb.com/2019/11/17/video-engineering/copa/ (accessed on 14 March 2022).
46. Pronounced Move Fast. Available online: https://github.com/facebookincubator/mvfst (accessed on 11 May 2022).