entropy

MDPI

Article

Network Delay and Cache Overflow: A Parameter Estimation Method for Time Window Based Hopping Network

Zhu Fang [1,*] and Zhengquan Xu [2]

[1] School of Electronic Information, Wuhan University, Wuhan 430064, China
[2] State Key Laboratory of Mapping and Remote Sensing Information Engineering, Wuhan University, Wuhan 430079, China
* Correspondence: fangzhu@whu.edu.cn

Abstract: A basic understanding of delayed packet loss is key to successfully applying it to multi-node hopping networks. Given the problem of delayed data loss due to network delay in a hop network environment, we review early time windowing approaches, for which most contributions focus on end-to-end hopping networks. However, they do not apply to the general hopping network environment, where data transmission from the sending host to the receiving host usually requires forwarding at multiple intermediate nodes due to network latency and network cache overflow, which may result in delayed packet loss. To overcome this challenge, we propose a delay time window and a method for estimating the delay time window. By examining the network delays of different data tasks, we obtain network delay estimates for these data tasks, use them as estimates of the delay time window, and validate the estimated results to verify that the results satisfy the delay distribution law. In addition, simulation tests and a discussion of the results were conducted to demonstrate how to maximize the reception of delay groupings. The analysis shows that the method is more general and applicable to multi-node hopping networks than existing time windowing methods.

Keywords: network security; hopping network; delayed time windows; time window compensation; time window parameter estimation

Citation: Fang, Z.; Xu, Z. Network Delay and Cache Overflow: A Parameter Estimation Method for Time Window Based Hopping Network. *Entropy* **2023**, *25*, 116. https://doi.org/10.3390/e25010116

Academic Editor: José F. F. Mendes

Received: 17 November 2022
Revised: 24 December 2022
Accepted: 3 January 2023
Published: 5 January 2023

1. Introduction

With the advancement of information technology, the wide application of the network promotes the development of the social economy, but it also faces serious security problems. In the face of various malicious network assaults such as Trojans, worms, and viruses, these attacks substantially increase the risks of disclosing user information and resulting in property loss. Traditional network protection approaches such as security vulnerability screening, firewalls, intrusion detection, and so on have been investigated to achieve this goal. Those defensive technologies have raised the degree of network protection and are increasingly becoming the typical network defense configuration.

However, network attack techniques are always improving, and current defensive systems are unable to defend against all new types of network assaults, such as Trojan port hopping, hopping proxy, protocol conversion, distributed denial of service (DDoS), and other unknown attacks. To that end, the idea of moving target defense (MTD) [1–4] was first proposed in the Astronomy Picture of the Day (APOD) project [5], which was presided over by the USA Ministry of Defense (MoD) department. In that project, they provided a brand-new technical route to deal with new attacks. Unlike past cyber security approaches, MTD is devoted to building a dynamic, heterogeneous, and unpredictable network that may avoid, delay, or stop network assaults by increasing the randomness or decreasing the predictability of the system. Furthermore, MTD technologies attempt to create an unpredictable cyber environment by continuously changing the attack surface

and reducing exposure to static targets. Because this technology subverts the traditional defense concept of "fixed and die-hard defense" for specific attacks, it is still effective in defending against some uncertain or unknown attacks and is thus referred to as dynamic or active defense technology, whereas the traditional defense technology is referred to as static or passive defense. According to popular belief, genuinely effective network security must comprise both active and passive defense technology.

Network active defense systems are based on hopping networks, and thus hopping networks have been one of the most active research directions, where some of the primary studies on hopping networks are randomization of network parameters [6–9], etc. Chang et al. [9] randomized the IP addresses of SDNs to improve the randomness of hopping networks; Luo et al. [10] proposed a random port and address hopping (RPAH) mechanism; Lee et al. [11] proposed a UDP/TCP port-hopping method that allows the server parameters to vary with time and a shared secret key function. To prevent attackers from continuously tracking their targets, Fenske et al. [12] proposed a deployment scheme based on MAC address randomization. Since IPv6 has a larger address space, Dunlop et al. [13] proposed an IPv6-based moving target defense (MT6D) technique. These network parameter randomization techniques break the offensive–defensive balance to some extent, making it more difficult for attackers to attack network targets. However, they sacrifice certain network resources when implementing a hopping network, which impacts the network's performance.

The influence on network performance includes two main aspects: one is the problem of resource congestion, i.e., in order to carry out network hopping, it is inevitable to consume network resources, thus indirectly causing a reduction in network transmission performance; the other is the so-called network delay packet loss problem, i.e., when each node in the network synchronizes hopping, due to transmission delay, some data transmitted with pre-hopping parameters do not reach the receiving node at the time of hopping, and it also cannot be correctly received by the node after hopping, thus causing transmission packet loss. This not only degrades performance but also interferes with the normal network function when network resources are tight, transmission paths are long, or hops are frequent.

The solution to network delayed packet loss is mainly to increase the data reception time (time window) for a period of time compared to the data sending time in order to balance the network transmission delay. For example, in a port-hopping network, Lee [14] makes the data reception time longer than the data sending time by a period of time and uses the overlapping time of the time window of the two periods before and after to receive delayed data from different ports; to further adapt to the network environment through repeated interactions with the receiver, Kong et al. [15] proposed a sliding time window based on an IPv6 address hopping (AHSTW) model, and Ma et al. [16] proposed a dynamic address tunneling model for IPv6, which they based on the results of the interaction feedback and made the length of the data-reception time window larger than the length of the data-sending time window. These techniques are mainly used in end-to-end networks, and they are tightly coupled to the service, which to some extent severely disrupts normal network operations and is therefore not suitable for general network scenarios.

To address the above problems, we propose a delayed time window estimation algorithm. Firstly, we introduce the concept of delay time window and the estimated value of the delay time window; secondly, we propose the basic conditions for avoiding non-interaction and IP address conflicts and mathematically prove that the time window compensation mechanism can satisfy these conditions, and we give the calculation of the estimated value of the delay time window; that is, after analyzing the network transmission of different data services, we obtain the expression for the estimated value of the delay time window, and at the same time, mathematically, we prove that the estimated value of the delay time window is theoretically close to the optimal value of the delay time window and finally provide the implementation algorithm. The algorithm not only minimizes the loss of delayed data but also does not require interaction with the service. In addition, the method allows the

network to consume fewer network resources and is able to maintain normal system service under network resource constraints, and it is therefore applicable to networks in general and is no longer limited to end-to-end networks.

2. Preliminaries

2.1. Hopping Network

As shown in Figure 1, assume that the network G is made up of physical nodes N_1, N_2, ... , N_z. The network parameters (IP address, port number, etc.) configured for the corresponding node N_i, $i = 1, 2, \ldots, z$ is denoted as h_{ij}, $j = 1, 2, \ldots, \xi$. Typically, the set $\boldsymbol{H} = (h_{ij})_{z\times\xi}$ remains constant during the work, while nodes can rely on $\boldsymbol{H}$ for mutual access. For each node, it is specified that at every period T, a parameter is selected from $\boldsymbol{H}$ and configured to it by some rule, and its network parameters are changed every period T. For this change, it is called network hopping, and T is the hopping period.

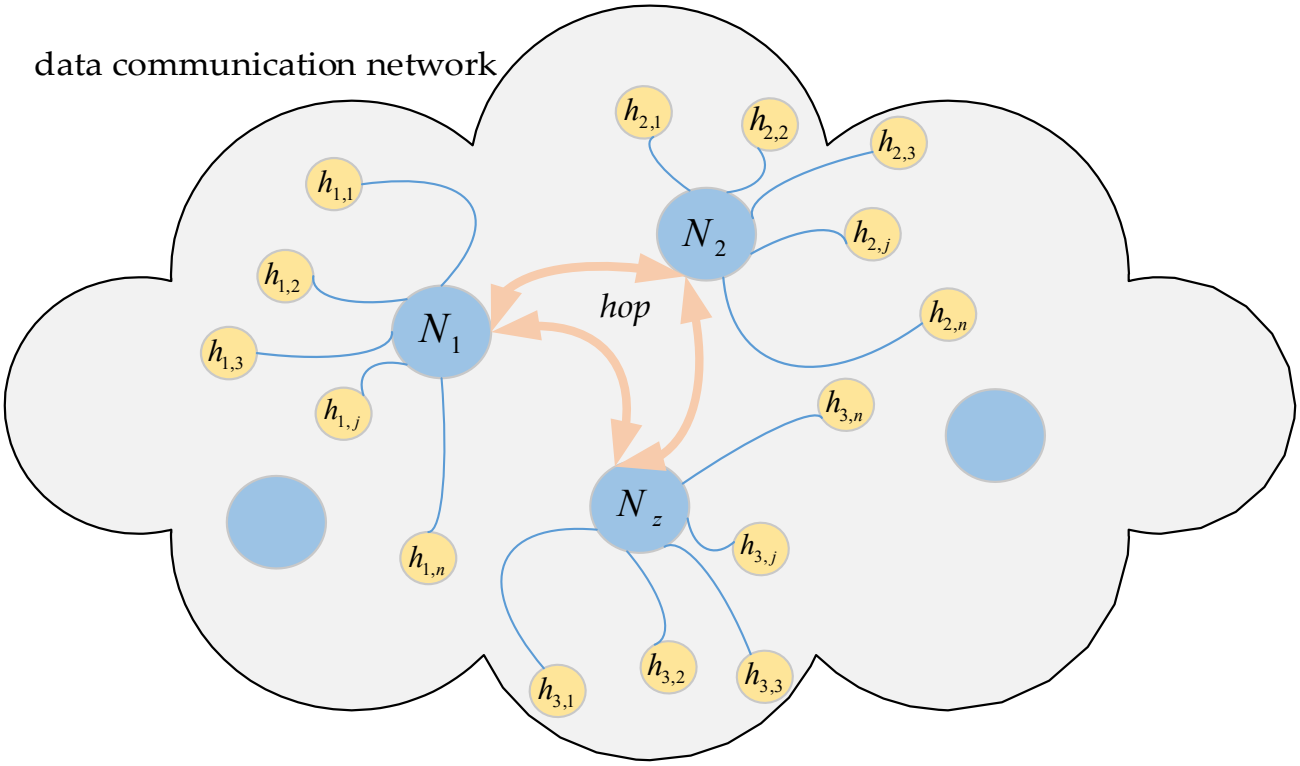

Figure 1. Schematic diagram of the hopping network.

2.2. Time Window

As network parameters constantly change in the hopping, a particular network parameter is only valid for a specific period, which is the time window for that network parameter, as shown in Figure 2. Then the network parameter h_{ij} corresponding to the i node at the j hopping period is valid for the time range $[tw_a(j), tw_b(j)]$ for the h_{ij} time window, denoted as $TW(h_{ij})$. Since the network nodes are generally synchronous hopping, the time windows of the parameters corresponding to each node are fully overlapping, i.e., $TW(h_{1j}) = TW(h_{2j}) = \ldots = TW(h_{zj}) = [tw_a(j), tw_b(j)]$, so they are denoted uniformly as $TW(j)$. In the normal case, the time window within which the data is sent is the sending time window $STW(j)= [st_a(j), st_b(j)]$. After receiving the data within the time window, the time window for receiving the data can be denoted as receiving time window $RTW(j) = [rt_a(j), rt_b(j)]$. The set of times at which data arrives after it has been sent within the time window is the arrival-data time window $ATW(j) = [at_b(j), at_b(j)]$. In the ideal case of no network delay, $RTW\ (j) = STW(j) = ATW(j)$.

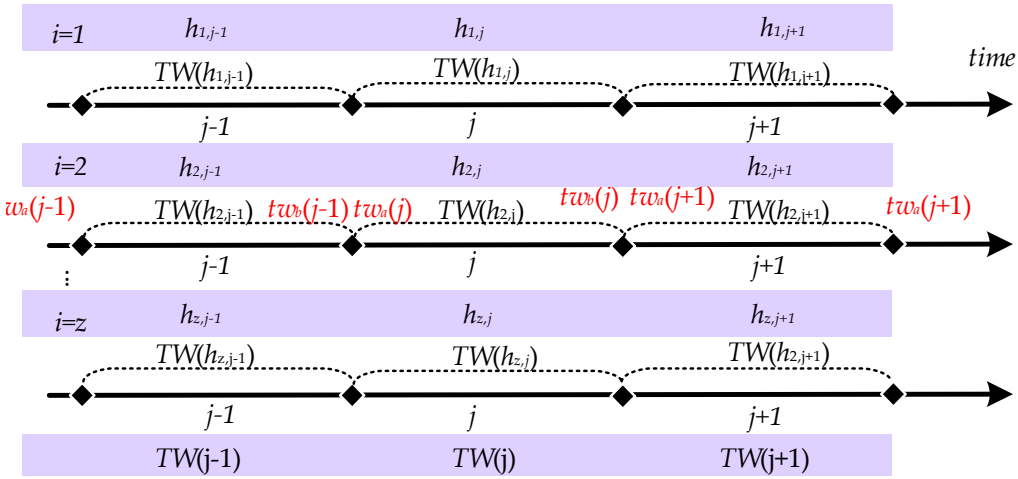

Figure 2. Description of Time Window.

2.3. Delayed Packet Loss

In hopping networks, some mechanisms are usually designed so that all legitimate nodes can sense the change of $\boldsymbol{H}$ in real time, so the regular communication of legitimate nodes is unaffected. However, some transmission packet loss is still generated during a short period of time when $\boldsymbol{H}$ is hopped, which is called delayed packet loss. As shown in Figure 3, the specific cause of delayed packet loss is that $ATW(j)$ is slightly delayed by a period of time compared to $STW(j)$ due to the presence of network delay d, which results in $RTW(j)$ not being able to cover $ATW(j)$ completely. As a result, different methods have been tried to make $RTW(j)$ cover $ATW(j)$as much as possible. We summarize the descriptions in the literature [15–17], where they present two methods for $RTW(j)$ to cover $ATW(j)$. One is to maximize $RTW(j)$ coverage of $ATW(j)$ by extending $RTW(j)$ on the basis that $STW(j)$ does not change [14]; the other is to shorten $STW(j)$ on the basis that $RTW(j)$ does not change, which is equivalent to delaying $RTW(j)$ to give it a chance to cover $ATW(j)$ completely [15,16], but both methods have drawbacks.

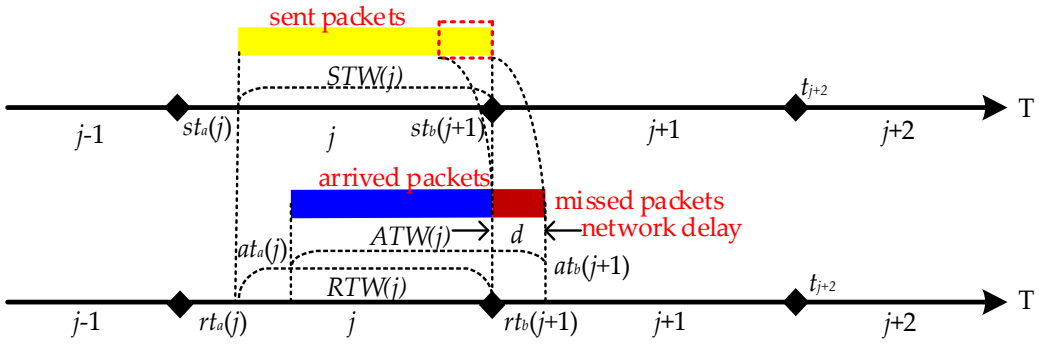

Figure 3. Delayed packet loss may be lost during network hopping.

2.4. Problem Statement

These two methods have two disadvantages:

One is the problem of conflicting IP addresses. When extending $RTW(j)$ for some time, then $RTW(j)$ is a period longer than $STW(j)$, and there will be a period of overlap between this extra time $ODRT = [rt_a(j+1), rt_b(j)]$ and $RTW(j+1)$. Since the overlapping time can receive data from different ports but not from different IP addresses, receiving data during the overlapping time can lead to a problem of conflicting IP addresses, as shown in Figure 4.

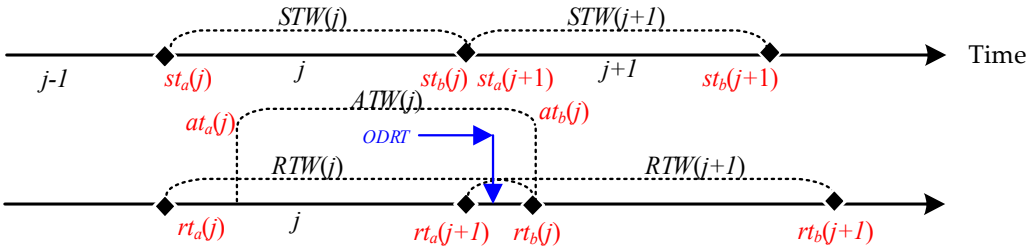

Figure 4. *ODRT* for overlapping data reception time.

The other is the problem of tight business coupling. When $STW(j)$ is shortened by a period of time, then a period of time $FDTT = [st_b(j), st_a(j+1)]$ for which $STW(j)$ is less than $RTW(j)$ can be used without being used to send data and without the need to interact with the tasks to obtain $st_b(j)$ and $st_a(j+1)$ for this period of time through the time window. The aim is to allow delayed data to arrive in total, and such a time window is mainly used in end-to-end networks (see Figure 5).

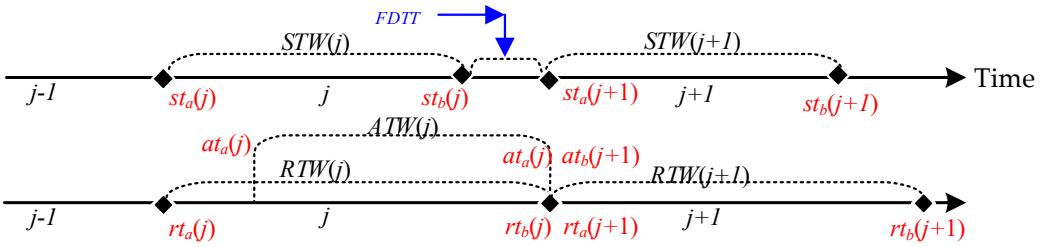

Figure 5. *FDTT* for idle data sending time.

3. Proposed Scheme

In this section, we present a scheme for a delayed time window compensation mechanism and its parameter estimation to solve the problem of IP address conflicts and tightly coupled services. The implementation of such a mechanism then requires two conditions: firstly, a time window compensation mechanism and defined associated parameters that meet the above requirements; secondly, such a mechanism can effectively perform the function as a time window when the influence of external factors (it is assumed that there are no network attacks on the hopping network and that the hopping network is hopping at a uniform rate) on the compensation mechanism is negligible.

3.1. Delay Time Window Compensation Mechanism

According to the above, there are problems with both methods of $RTW(j)$ covering $ATW(j)$ by extending $RTW(j)$ on the basis that $STW(j)$ does not change length or shortening $STW(j)$ on the basis that $RTW(j)$ remains unchanged, so we propose the idea of keeping $STW(j)$ equal to $RTW(j)$ and delaying $RTW(j)$ to $STW(j)$ for a period of time, with the aim of allowing $RTW(j)$ to override $ATW(j)$. This not only avoids IP address conflicts and tight task coupling, but it also solves the problem of delayed packet loss. The proof is as follows:

Firstly, when $STW(j) = RTW(j)$, then there is not a period of time when $RTW(j)$ is longer than $STW(j)$; this period of time does not create an overlap with $RTW(j+1)$, and therefore this does not lead to IP address conflicts. Furthermore, there is also not a period of time when $STW(j)$ is less than $RTW(j)$, which indicates that the time window does not require interactive data tasks to obtain such a period of time, and therefore this does not lead to tight service coupling.

Secondly, when $RTW(j) = ATW(j)$, then all the data from $STW(j)$ can be received in $RTW(j)$, including the delayed data.

Thus, we let $STW(j) = RTW(j)$ and $RTW(j) = ATW(j)$; they can be achieved by the following two steps:

(1) For $RTW(j) = STW(j)$, let $rt_b(j) - rt_a(j) = st_b(j) - st_a(j)$, then $rt_b(j) = st_b(j)$ and $rt_a(j) = st_a(j)$.

(2) For $RTW(j) = ATW(j)$, assume that d is the network delay and $w^{\sim}$ denotes a period of time (delay time window). Under the condition that step (1) and (2) are satisfied, it follows that $at_a(j) = st_a(j) + d$, $at_b(j) = st_b(j) + d$, $rt_a(j) = st_a(j) + w^{\sim}$ and $rt_b(j) = st_b(j) + w^{\sim}$. Conversely, the values of $rt_a(j)$ and $rt_b(j)$ can be determined as long as the estimated value of $w^{\sim}$ is obtained and $w^{\sim} = d$, again satisfying $STW(j) = RTW(j)$ and $ATW(j) = RTW(j)$.

In summary, the delay time window compensation mechanism has the characteristics of necessity and possibility:

1. Necessity means that a delay time window compensation mechanism must solve the problem of delayed packet loss and related problems.
2. Possibility means that there may be a suitable delay time window length, i.e., a value of $w^{\sim}$ (see Figure 6).

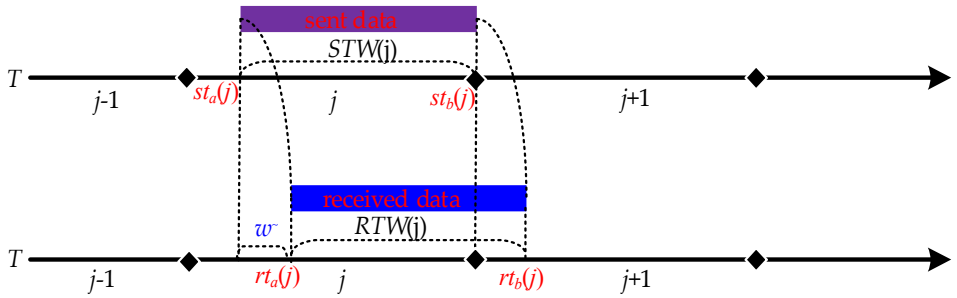

Figure 6. Diagram of the delay time window.

3.2. Estimation for Delay Time Window

In this section, we obtain the value of $w^{\sim}$ by calculation, as getting the value of $w^{\sim}$ helps us to determine the values of $rt_a(j)$ and $rt_b(j)$. Since d is unknown, then we can only find a way to estimate a value for the network delay and use it as an estimated value for $w^{\sim}$. From this, we present an example with multiple (l) data tasks distributed in an observable wholly connected network. By progressively computing the network transmission delay for the typical data tasks in the example, we finally obtain an estimated value of the network delay for $w^{\sim}$. These typical data tasks include an end-to-end data task, a multi-node data task, and multiple multi-node data tasks. This regular data task is more complex, so we plan to start with the simpler data tasks, as shown in Figure 7.

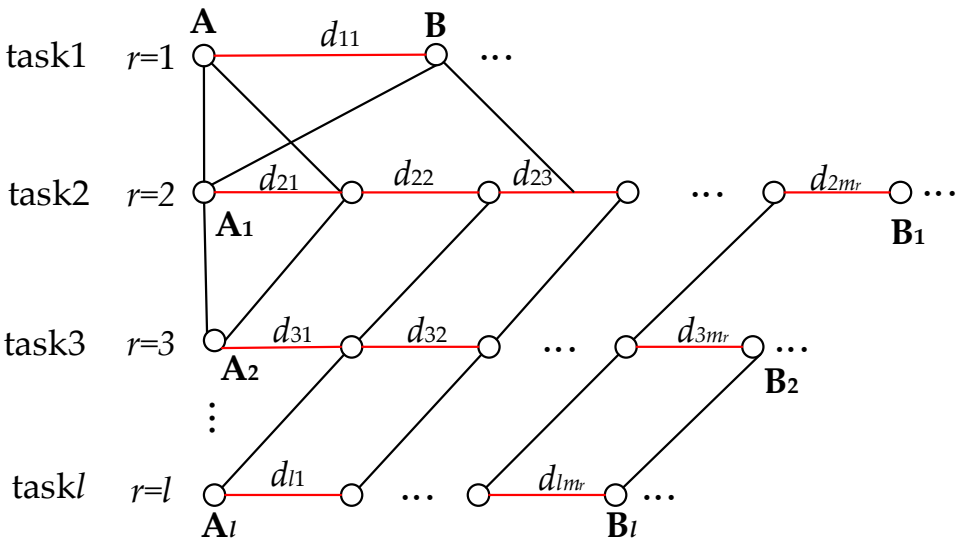

Figure 7. Whole connected network consisting of two or more nodes connected in a serial or parallel network.

(1) A simple data task. Under normal situations, routers and network agents have sufficient network cache [17,18] to store and forward data, so in most cases, the network cache is still significantly effective in reducing delay in packet loss. However, a few instances can still cause packet loss problems for some data. These are the rare cases where the processor does not have enough processing capability due to large amounts of data or unusual data, and therefore the network cache is insufficient. The few cases that cause delayed packet loss are then our target. Consequently, we examine task 1 ($r = 1$), as shown in Figure 8. In task 1, we start with two nodes, with a network delay of d_{10} between node A and node B. When data is sent from A to B, a small amount of data is lost, except for most of the data that is received by B. This is because when B has sufficient network cache and $w^{\sim} > d_{10}$, the network cache can store data arriving before $rt_a(j)$ without data loss. However, when B does not have sufficient network cache due to oversized data, data arriving before $rt_a(j)$ is lost; furthermore, when $w^{\sim} < d_{10}$ and $RTW(j)$ cannot completely cover $ATW(j)$, the data arriving after $rt_b(j)$ are lost (see Figure 9). The lost data can be expressed as:

$$L_{AB} = v \cdot | d_{10} - w^{\sim} | \quad (1)$$

Let $L_{AB} = 0$, $w^{\sim} = d_{10}$.

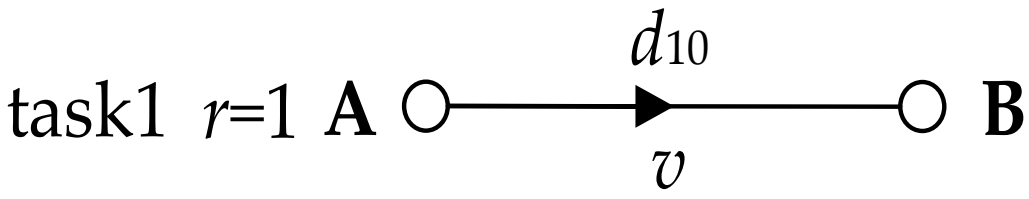

Figure 8. In the absence of intermediate nodes, end-to-end network transmission.

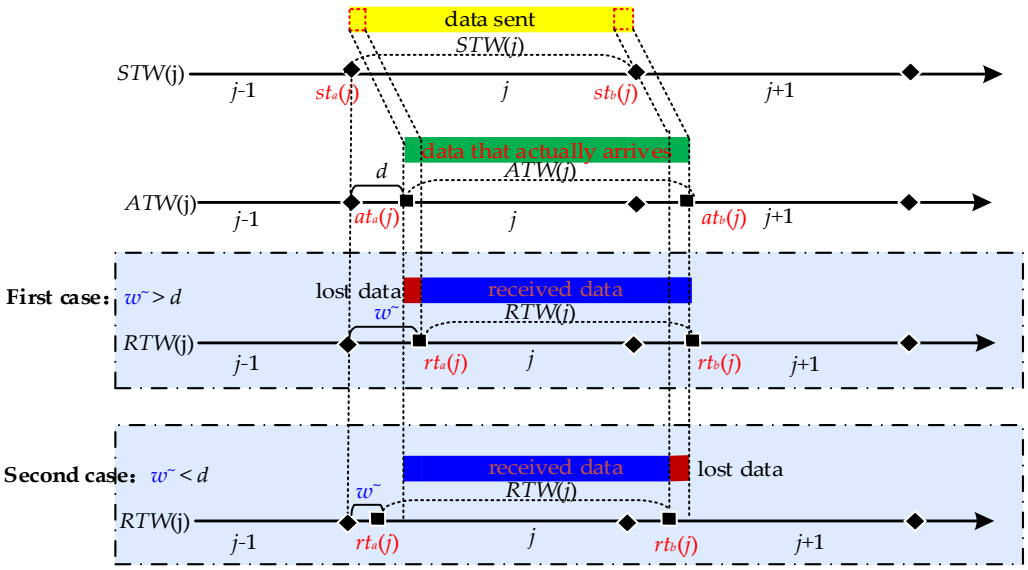

Figure 9. Data packet loss may occur when $w^{\sim} > d$ or $w^{\sim} < d$.

Where v denotes the transmission rate, and L_{AB} denotes the number of delayed packet losses.

(2) Depicts a multi-node data task. We observe this for task 2 ($r = 2$), where the variable m combined with the corresponding subscript number r indicates the number m_r of nodes of the corresponding data task, e.g., the number of nodes for the data task 2 ($r = 2$) is m_2. The variable d combined with the corresponding subscript numbers r and i denotes the network delay d_{ri} of the corresponding task r and node i, e.g., the network delay for task 2 and node 1 is d_{21}, as shown in Figure 10. The data tasks for multiple nodes are more

complex because data passing through multiple nodes will generate network delays, and there is an accumulation of multiple network delays starting with three nodes. The network transmission of three nodes can be seen as two end-to-end network transmissions, i.e., node 1 to node 2 and node 2 to node 3. As the data causes one network delay after passing node 2 and another network delay when the data reaches node 3, then the sum of the two network delays is the total network delay when the data comes to node 3. We can then calculate the number of packet losses based on the total network delay, but it needs to be clear that Equation (1) already provides the delayed packet loss for the first end-to-end network transmission, so the delayed packet loss for the second end-to-end network transmission is cumulative on top of that (see Figure 11). The delayed packet loss for the three nodes can then be expressed as:

$$L_2 = v \cdot |\, d_{21} - w^{\sim} \,| + v \cdot |\, d_{21} + d_{22} - w^{\sim} \,| \tag{2}$$

Based on the delayed packet loss for three nodes, we introduce the delayed packet loss for m_2 nodes, which can be expressed as:

$$L_2 = v \cdot |\, d_{21} - w^{\sim} \,| + v \cdot |\, d_{21} + d_{22} - w^{\sim} \,| + \cdots + v \cdot |\, d_{21} + d_{22} + \cdots + d_{2m2} - w^{\sim} \,| \tag{3}$$

Combining for (3), we get:

$$L_2 = \sum_{i=1}^{m_2} v \cdot \left| \sum_{\kappa=1}^{i} d_{2\kappa} - w^{\sim} \right|$$

Let $d_{2i} = \sum_{\kappa=1}^{i} d_{2\kappa}$, we have:

$$L_2 = \sum_{i=1}^{m_2} v \cdot |d_{2i} - w^{\sim}|$$

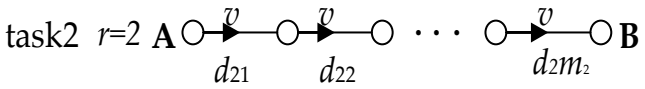

Figure 10. Multi-node network transmission.

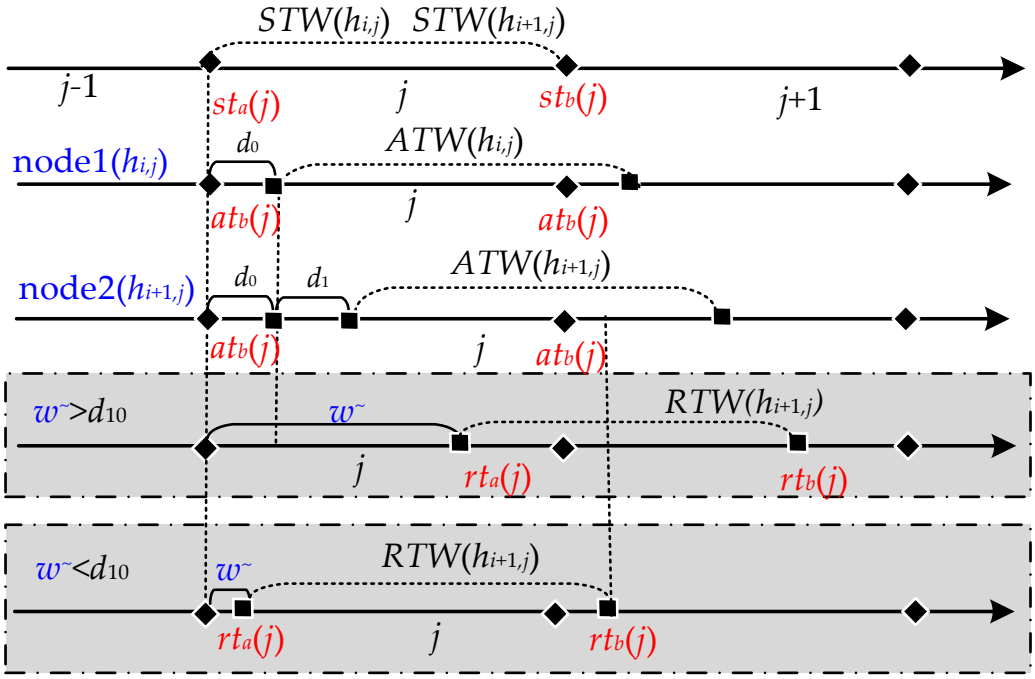

Figure 11. Delayed packet loss for two nodes based hopping network.

Based on Equation (3), we get:

$$f_2 = \frac{L_2}{T \cdot v} \tag{4}$$

where f_{AB} is the function of packet loss rate for hopping period T.

Let $\frac{\partial f_2}{\partial w^{\sim}} = 0$, we have:

$$\begin{gathered}
\partial \frac{\left(\frac{L_2}{v \cdot T}\right)}{\partial w^{\sim}} = 0 \\
\frac{\partial\left(\frac{(d_{21}-w^{\sim})^2}{T}\right)}{\partial w^{\sim}} + \frac{\partial\left(\frac{(d_{22}+d_{22}-w^{\sim})^2}{T}\right)}{\partial w^{\sim}} + \cdots + \frac{\partial\left(\frac{\left(d_{21}+d_{22}+\cdots+d_{2m_2}-w^{\sim}\right)^2}{T}\right)}{\partial w^{\sim}} = 0 \\
w^{\sim} = \frac{1}{m_2}[m_2 d_{21} + (m_2 - 1)d_{22} + \cdots + d_{2m_2}]
\end{gathered}$$

(3) Depicts multiple data tasks for the whole network. In the observable network, any two or multiple nodes in the network are connected as a single path. Thus, there are multiple paths (α) across the network, and in most cases, there are multiple data tasks (l) on a single path. Then, after observing the data tasks on this one path, we can further examine the data tasks on multiple paths. Since data tasks on multiple paths are more complex, we first examine data tasks on one path. By observation, accumulating the network delay for each data task can be used as the total network delay for the data tasks on this path. Then similarly, based on (4), accumulating the delayed packet loss for multiple data tasks on the path one can be used as the total delayed packet loss for this path (see Figure 12). The total delayed packet loss on path one can be expressed as:

$$\sum_{r=1}^{l} L_{\mathrm{AB}}(r) \tag{5}$$

Based on the delayed packet loss for path 1, we introduce the delayed packet loss for the α path expressed as:

$$L_l = \sum_{r=1}^{l} L_{\mathrm{AB}}(r) \tag{6}$$

Based on (6), the data packet loss rate for period (T) can be expressed as:

$$f_l = \frac{L_l}{v \cdot T} \tag{7}$$

Let $\frac{\partial f_l}{\partial w^{\sim}} = 0$, we have:

$$\begin{gathered}
\frac{\partial f_\alpha}{\partial w^{\sim}} = \partial \frac{\left(\frac{L_l}{v \cdot T}\right)}{\partial w^{\sim}} = 0 \\
\frac{\partial\left(\sum\limits_{r=1}^{l}\left(\frac{\left(d_{d_{r1}}-w^{\sim}\right)^2}{T} + \frac{(d_{r1}+d_{r2}-w^{\sim})^2}{T} + \cdots + \frac{(d_{r1}+d_{r2}+\cdots+d_{rm_r}-w^{\sim})^2}{T}\right)\right)}{\partial w^{\sim}} = 0 \\
\sum_{r=1}^{l}\left[\left(d_{d_{r1}} - w^{\sim}\right) + (d_{r1} + d_{r2} - w^{\sim}) + \cdots + (d_{r1} + d_{r2} + \cdots + d_{rm_r} - w^{\sim})\right] = 0 \\
\sum_{r=1}^{l}\left[m_r d_{d_{r1}} + (m_r - 1)d_{r2} + \cdots + d_{rm_r} - m_r w^{\sim}\right] = 0 \\
w^{\sim} = \frac{1}{\sum_{r=1}^{l} m_r} \sum_{r=1}^{l} [m_r d_{r1} + (m_r - 1)d_{r2} + \cdots + d_{rm_r}]
\end{gathered}$$

The estimation of $w^{\sim}$ is to let $w^{\sim} = \frac{1}{\sum_{r=1}^{l} m_r} \sum\limits_{r=1}^{l} [m_r d_{r1} + (m_r - 1)d_{r2} + \cdots + d_{rm_r}]$. When $w^{\sim} = \frac{1}{\sum_{r=1}^{l} m_r} \sum\limits_{r=1}^{l} [m_r d_{r1} + (m_r - 1)d_{r2} + \cdots + d_{rm_r}]$, then $L_a = 0$ satisfies 0 packet loss for data tasks on r. According to [19] as the basis for $w^{\sim}$ estimation, the results obtained by Bolot's [19] tests are consistent with those obtained by [20–22] using simulation and experimental methods. Under the assumption that using bulk traffic for large packets and traffic for small packets in internet traffic estimation is consistent, the structure of the delay time distribution can be described as the relationship between the waiting

time w_n and w_{n+1} for packets n and $n+1$. It influences the network traffic (bits) b, the packets (bits) P, the service rate of the network (bits/ms) μ, and the packet queuing time δ. Their relationship can be expressed as $w_{n+1} - w_n = (b+P)/\mu - \delta$. Taking Figure 13 as an example, it shows the distribution of $w_{n+1} - w_n - \delta$ for n ($n \leq 800$) UDP (32 bts) packets with $w_{n+1} - w_n - \delta$ at δ = 20 ms. $w_{n+1} - w_n - \delta$ is the network load received by the server within $[n\delta, (n+1)\cdot\delta]$ and is measured in ms. From Figure 13, it can be seen that the time is mainly distributed in the area covered by the dashed line. Therefore, our proposed strategy is to select the larger data tasks. This is because they take up more time (delay time = receive time ($at_{r\kappa}$) − send time ($st_{r\kappa}$)) through multi-hop routes [23,24]. For example, in Figure 13, assuming that the largest data task $r = 2$, $i = 1, \ldots, m_2$, then the maximum delay time is $m_2 d_{21} + (m_2 - 1)d_{22} + \cdots + d_{2m_2}$, and the amount of lost data is $L_2 = \sum_{i=1}^{m_2} v \cdot \left| \sum_{\kappa=1}^{i} d_{2\kappa} - w^{\sim} \right|$ (see Equation (3)). Let $d_{2i} = \sum_{\kappa=1}^{i} d_{2\kappa}$, $d_{2\kappa} = at_{2\kappa} - st_{2\kappa}$. When $w^{\sim} = d_{2i}$, then $L_2 = 0$, i.e., $w^{\sim} = \sum_{\kappa=1}^{i} at_{2\kappa} - st_{2\kappa} t$. Our approach is to use $\max\{\sum_{\kappa=1}^{i} at_{2\kappa} - st_{2\kappa}, 0\}$ as an estimate for $w^{\sim}$, i.e., $w^{\sim} = \max\{\sum_{\kappa=1}^{i} at_{2\kappa} - st_{2\kappa}, 0\}$, $w^{\sim} \subset \{0, (b+P)/\mu - \delta\}$.

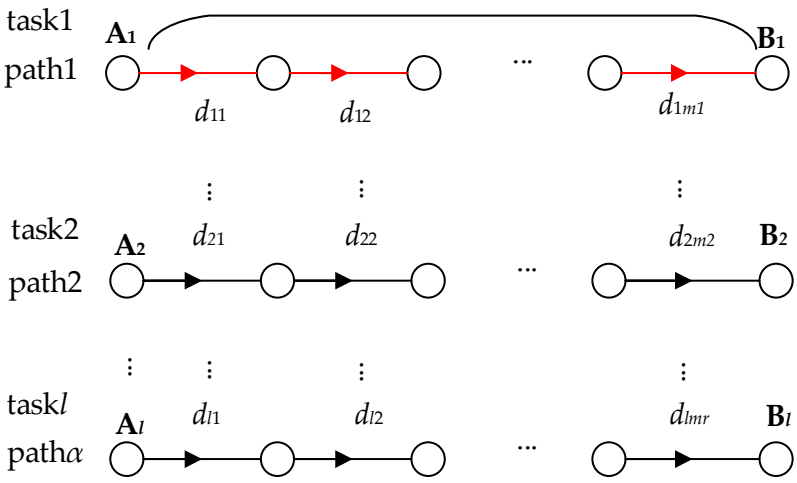

Figure 12. Network transmission for the entire network.

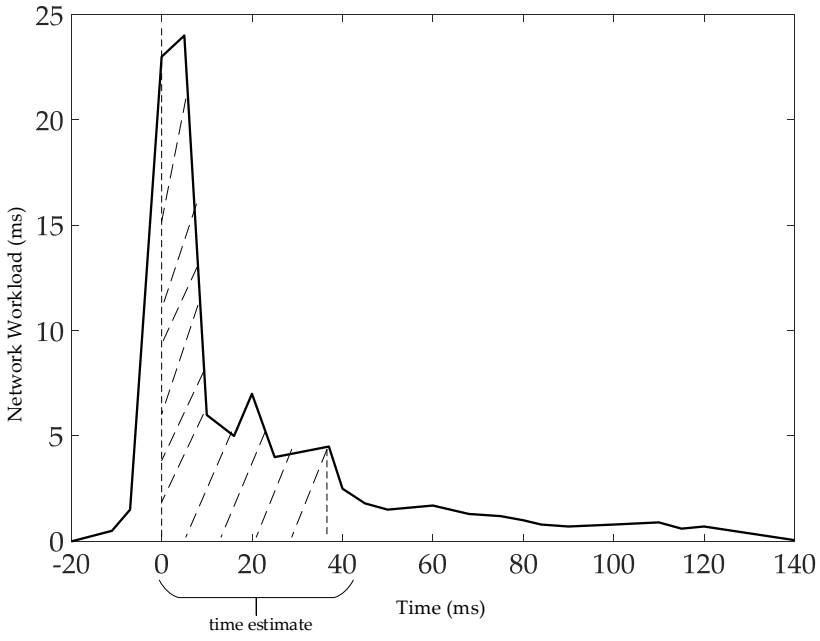

Figure 13. Distribution of $w_{n+1} - w_n - \delta$ at δ = 20 ms.

4. Performance

In this section, we give the error and experimental evaluation of our two proposed delay time window schemes. For error evaluation, we mainly analyze the difference between estimated and optimal values of the delay time window. For experimental evaluation, we test the data loss rate of our schemes.

4.1. Error Evaluation

In this section, to assess the validity of our proposed estimates of the delay time window, we thus present the error assessment of the delay time window. The error is a key part of the error assessment. The error is defined as $\delta = w^{\sim} - \overline{w}$, and the mathematical expectation of the error ($E(\delta)$) reflects the magnitude of the mean of the error between the estimated and optimal values of the delay time window. Therefore, it can be used as an indicator for error assessment. In the error evaluation, let the variables $x_1, x_2, \ldots, x_{2k+1}$ denote the network delay, and $x_1, x_2, \ldots, x_{2k+1} \sim N(0,1)$, and let them be independent of each other, $\overline{w} = (x_1, x_2, \ldots, x_{2k+1})/2k+1$. Let *med* $(x_1, x_2, \ldots, x_{2k+1})$ denote the function that returns an optimal value based on the proposed method. According to [25,26], the probability density distribution for end-to-end delayed packet loss shows gamma distribution. Thus, we consider that its probability density can be viewed as density function $f(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$, and its distribution may be viewed as $F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$. Firstly, the analysis of multiple network delays is more complex, so we start with three network delays, x_1, x_2, and x_3. According to this, we have $\delta = (x_1 + x_2 + x_3)/3 - med(x_1, x_2, x_3)$.

$$
\begin{aligned}
E(\delta) &= \iiint_{x_1<x_2<x_3} (x_1 + x_2 - 2x_2) f(x_1) f(x_2) f(x_3) dx_1 dx_2 dx_3 \\
&= \int_{-\infty}^{\infty} f(x_2) \left[\int_{x_2}^{\infty} f(x_3) dx_3 \int_{-\infty}^{x_2} x_1 f(x_1) dx_1 + \int_{-\infty}^{x_2} f(x_1) dx_1 \int_{x_2}^{\infty} x_3 f(x_3) dx_3 - 2x_3 \int_{x_2}^{\infty} f(x_3) dx_3 \int_{-\infty}^{x_2} f(x_1) dx_1 \right] dx_2 \\
&= \int_{-\infty}^{\infty} f(x_2) [(1 - F(x_2))(-f(x_2)) + F(x_2) f(x_2) - 2x_2 (1 - F(x_2)) F(x_2)] dx_2 \\
&= \int_{-\infty}^{\infty} \left[f^2(x_2)(2F(x_2) - 1) + 2x_2 (F(x_2) - 1) F(x_2) \right] dx_2 \\
&= \int_{-\infty}^{\infty} \left[f^2(x_2)(2F(x_2) - 1) \right] dx + (F(x_2) - 1) F(x_2) f(x_2) |_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f^2(x_2) 2x_2 (F(x_2) - 1) dx_2 \\
&= 0
\end{aligned}
$$

Similarly, δ = *med* $(x_1, x_2, \ldots, x_{2k+1}) - (x_1, x_2, \ldots, x_{2k+1})/2k+1$.

$$
\begin{aligned}
E(\delta) &= E\left(\left(\sum_{i=1}^{2k+1} x_i / 2k + 1 \right) - x^* \right) \\
&= \int_{x^* = med(x_1, x_2, \ldots, x_{2k+1})} \left(\sum_{i=1}^{2k+1} x_i - (2k+1) x^* \right) f(x_1) f(x_2) \cdots f(x_{2m+1}) dx_1 dx_2 \cdots dx_{2m+1} \\
&= \int_{-\infty}^{\infty} \left\{ k[1 - F(x^*)]^k F^{k-1}(x^*)(-f(x^*)) + k[1 - F(x^*)]^{k-1} F^k(x^*) f(x^*) - k x^* F^k(x^*)[1 - F(x^*)] \right\}^k f(x^*) \\
&= \int_{-\infty}^{\infty} k[F(x^*)(1 - F(x^*))]^{k-1} [2F(x^*) - 1] f^2(x^*) dx^* - \int_{-\infty}^{\infty} k x^* [F(x^*)(1 - F(x^*))]^k f(x^*) dx \\
&= 0
\end{aligned}
$$

Therefore, $E(\delta) = 0$, from which it follows that the estimated value of $w^{\sim}$ is an unbiased estimated value.

4.2. Experimental Evaluation

The experimental assessment consists of two parts, one for the simulation experiment and the other for the actual examination.

For the simulation experiments, we developed the simulated hopping network program SHN (similar to NS-2) using Dev-C++5.11 and C++, which consisted of seven hopping network nodes, including one transmitter node, one receiver node, and five intermediate nodes, as well as network delays (X) and 14 IP addresses (192.168.1.10 to 192.168.1.13). The host with the SHN is DESKTOP-B4VQAPP, which was configured with CPU Intel (R) Xeon (R) e-2124 3.31ghz, 16GB of RAM, and Windows 10 OS. The experiments were implemented on DESKTOP-B4VQAPP. In our experiments, we first set up the system's initial

values, including the network hopping period $T = 2000$, the delay time window $w = 50$, the number of intermediate nodes $\sigma = 5$, the mean value $mu = 50$ of the random variable X, and the variance $sigma = 5, 10, 15, 20$, bandwidth (100 Mbit/s). Our first experiment was an end-to-end hopping network packet loss rate experiment. The second experiment was a packet loss rate experiment for a multi-node hopping network. In both experiments, we sent 10^5 data (32 bytes) from the sending node to the receiving node and counted the data loss rate at period T based on the data received by the receiving node after the data reached the receiving node. All experiments were repeated 5000 times. For end-to-end networks, as shown in Figure 14 and Table 1, because in this method, the size of the time window was set by sending interaction information to the receiving node and feedback from it. The methods of [15–17] are better than the method presented in this paper. In addition, as shown in Figure 15, we experimented with the proposed delayed time window approach, and the experimental results show that the packet loss rate is better for time windows of 10 ms, 20 ms, and 30 ms compared to no time window.

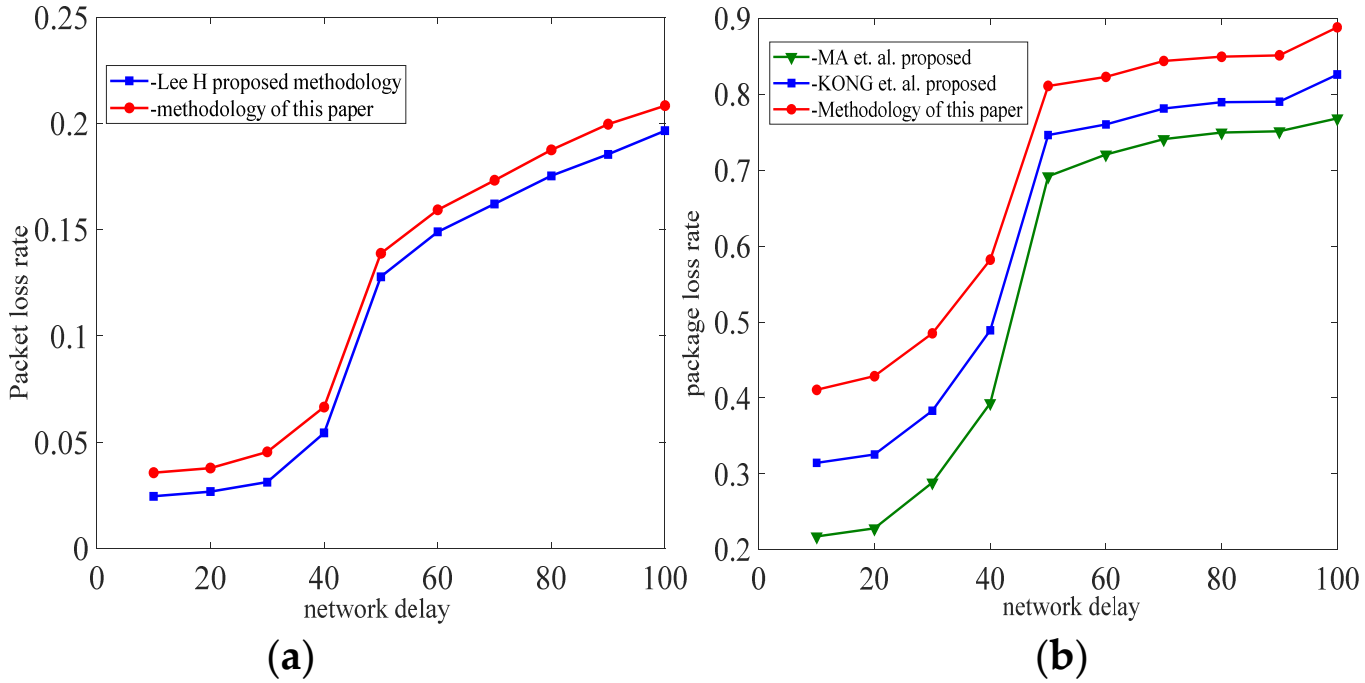

Figure 14. (**a**) Port hopping, (**b**) IP address hopping, the packet loss rates for end-to-end network. Refs [16,17].

Table 1. Comparison of packet loss rates for different time window methods.

Method	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
Ours	0.3	0.35	0.5	0.6	0.8
ODRT [14]	0.6	0.7	0.8	0.9	1.0
FDTT [15]	0.8	0.9	1.1	1.2	1.4
FDTT [16]	0.9	1.2	1.4	1.6	1.7

For the actual examination environment, the physical connection topology of the network is shown in Figure 16. The system is composed of two hopping subnets; hopping subnet 1 and hopping subnet 2 are connected through the IP bearer network. After the start of the network hopping, the source address and source port of the IP packet of the data platform in hopping subnet 1 after the hopping process are transmitted through the two WAN ports of the S5700 router through the IP bearer network to the hopping equipment in hopping subnet 2 after restoration, and then they are transmitted by the visitors in the data plane, thus completing an ordinary network transmission process.

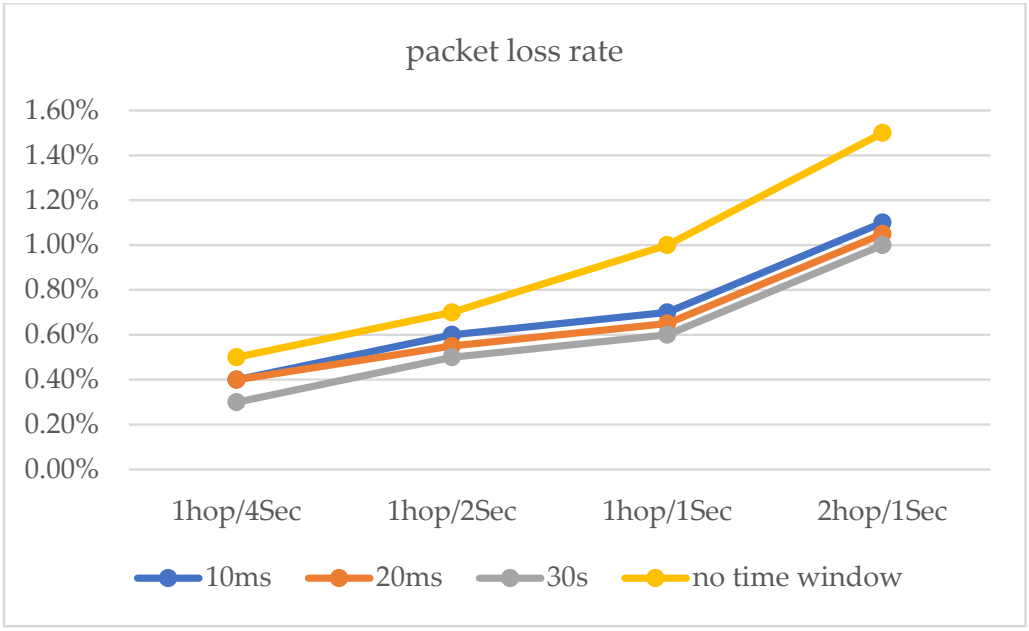

Figure 15. The packet loss rates with delay time window and no delay time window.

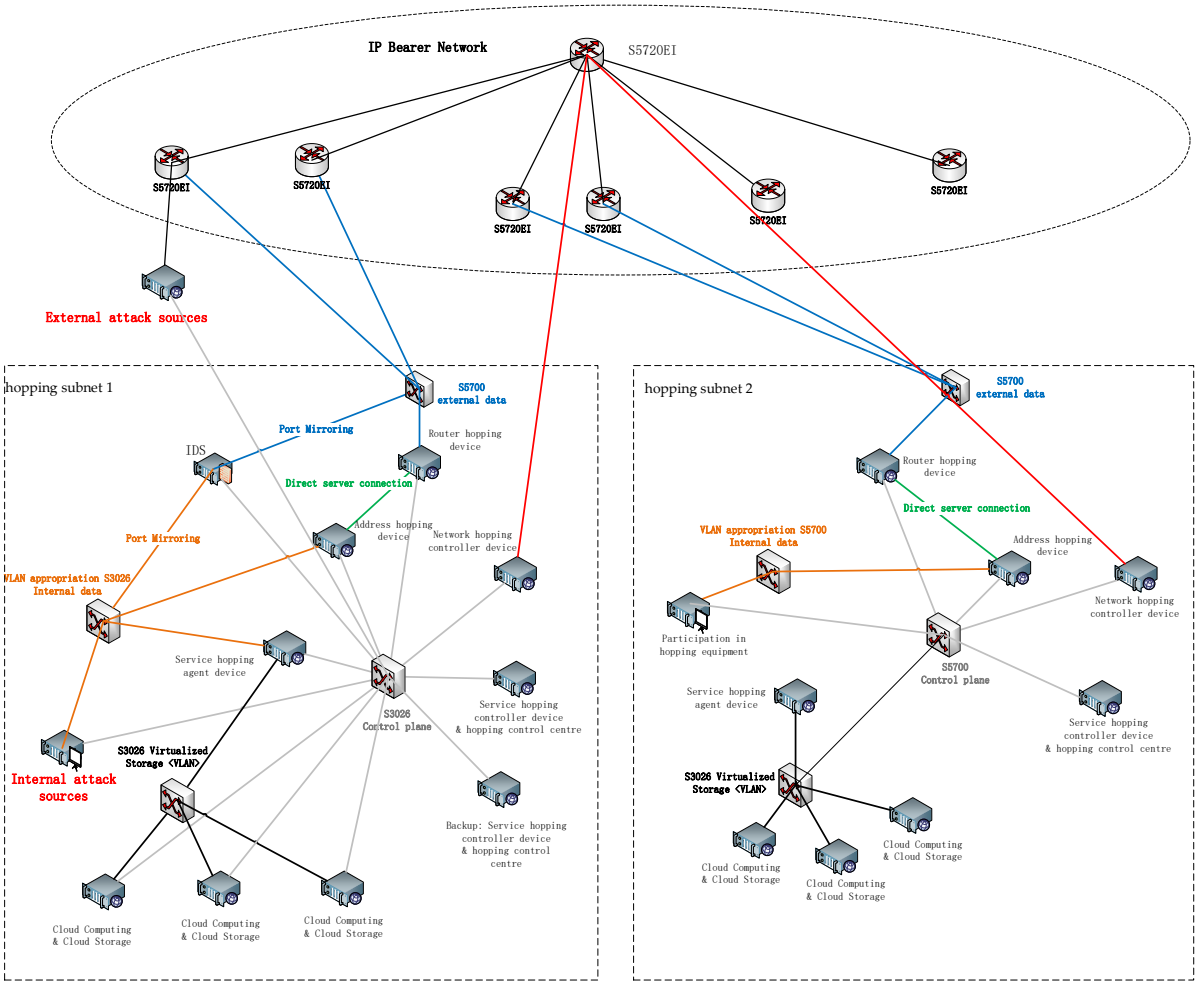

Figure 16. Physical connection topology of the hopping network.

The test environment is built with two hopping subnets, each of which is connected through an IP bearer network. Each hopping subnet is simulated by the relevant equipment in a physical cabinet. Each cabinet includes eight physical servers and three switches; the eight servers, respectively, achieve service hopping, network hopping, posture display, and other functions. The IP bearer network is simulated by an independent cabinet. The IP bearer network consists of four switches with three-layer routing function. The hardware equipment of the test environment mainly includes servers, switches, routers, etc., whose functions and performance indicators are shown in Table 2.

Table 2. List of hardware devices and their configuration parameters for Cabinet 1, Cabinet 2 and Cabinet 3.

	Device Name	Device Type	Hardware Parameter	Network Interface
1	Cloud computing and cloud storage	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*5	2* Gigabit Ethernet port
2	Cloud computing and cloud storage	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*5	2* Gigabit Ethernet port
3	Cloud computing and cloud storage	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*5	2* Gigabit Ethernet port
4	Management platform	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*3	2* Gigabit Ethernet port
5	Network hopping controller	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*3	2* Gigabit Ethernet port
6	Service hopping controller	server (Think-station)	CPU: E5-2609; memory: 16G; harddisk: 500G*3	2* Gigabit Ethernet port
7	Service hopping agent	server (FitServer)	CPU: E5-2609; memory: 16G; harddisk: 500G*2	2* Gigabit Ethernet port
8	Address hopping devices	server (FitServer)	CPU: E5-2609; memory: 16G; harddisk: 500G*2	2* Gigabit Ethernet port
11	Control plane connection	Switche S3026	24-port 2-Layer switch	2* Gigabit Ethernet port
12	Control plane connection	Switche S5700 (Li)	24-port 2-Layer switch	2* Gigabit Ethernet port
13	Cloud computing and cloud storage connection	Switche S5700 (Li)	24-port 2-Layer switch	2* Gigabit Ethernet port

The maximum network transmission rate is 100 Mbit/s under the national standard for category 5 network cables. Network hopping is initiated at one hop/10 s, one hop/5 s, and one hop/2 s, respectively. In addition, a network transmission stress test was conducted (see Figure 17. Moreover, the performance test of the router hopping was carried out at a network transmission pressure of 500 Mb/s (Figures 18 and 19). The experimental results show that the performance of the router still has a relatively large improvement based on various indicators with the use of delay time windows.

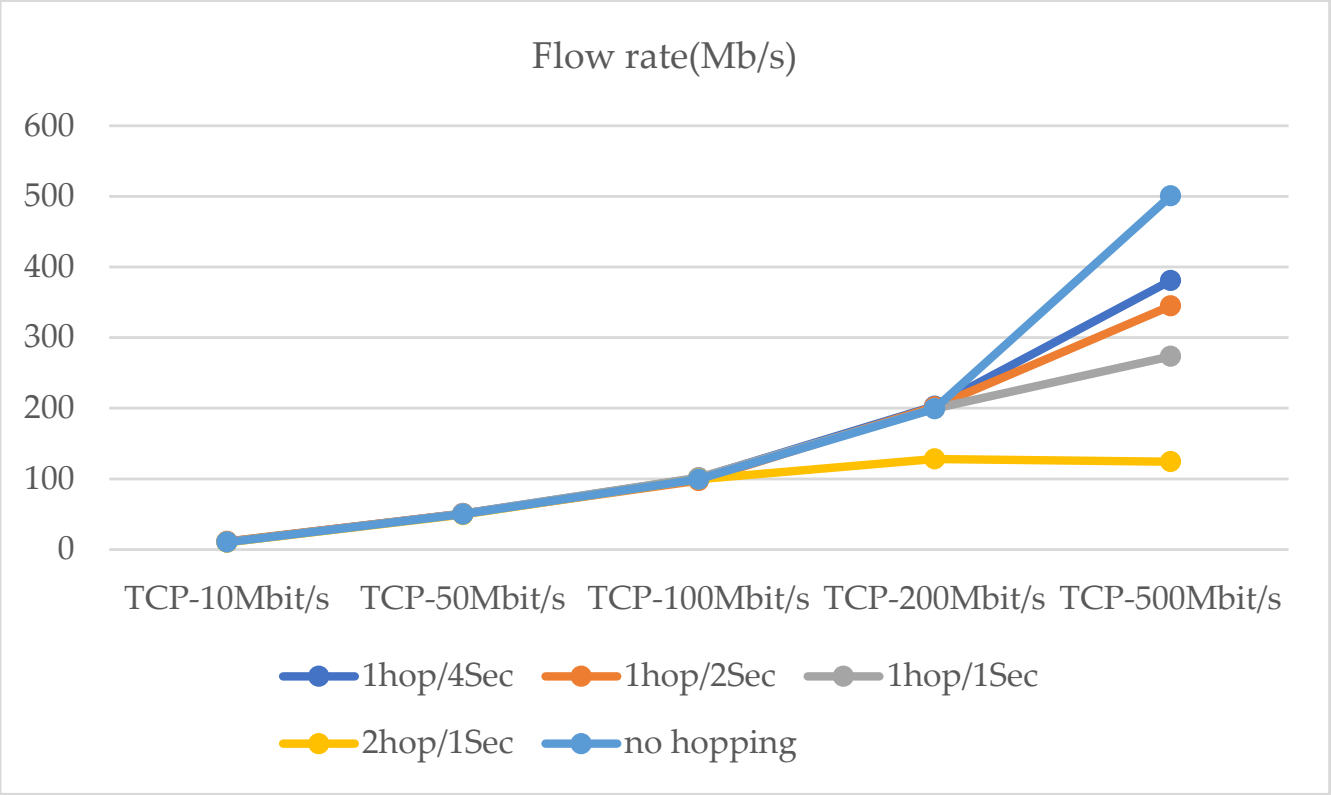

Figure 17. At 5 different hopping speeds, network transmission stress test.

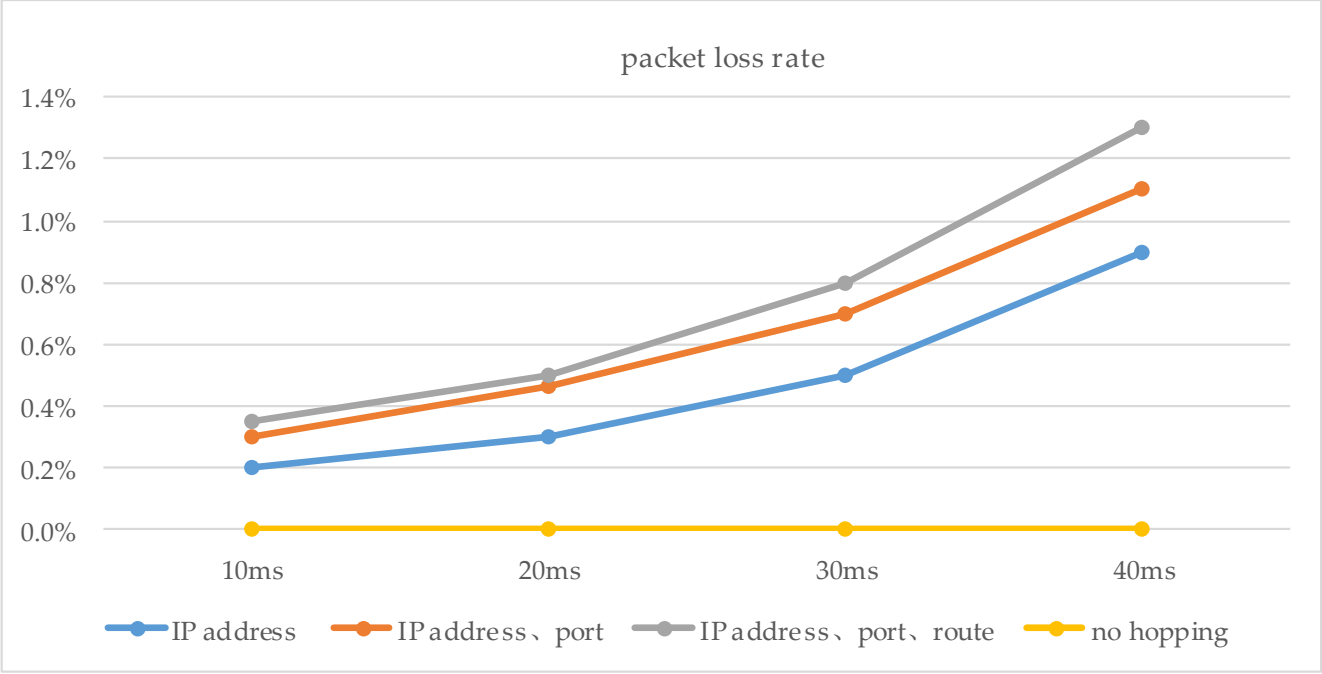

Figure 18. The packet loss rates with delay time window and no delay time window.

Figure 19. Transmission rate for 500 Mb/s, TCP data transfer.

5. Summary and Future Research

In this paper, we construct a multi-node delay time window estimation method. The delay time window and the delay time window compensation mechanism are proposed in the method, as well as the estimation of the delay time window length. A series of experiments were performed to test the effectiveness of the delay time window estimation method. In the SHN simulation experiments, the network transmission packet loss rate was less than 0.6% at 50 ms network delay. At 100 ms network delay, the network transmission packet loss rate was less than 1.1%. In the actual test, the test environment network cable was lower than the super category five standard, and the maximum network transmission rate was 100 Mbit/s. Taking the network transmission speed pressure of 500 Mbit/s as an example, the packet loss rate of network transmission using the delay time window method is less than 0.8% under 30 ms network delay. This is an improvement compared to the packet loss rate of network transmission without delay time windows. This proves the effectiveness of the method in this paper.

The first suggestion for future research relates to the later ones concerning the plausibility of the period of change of the network parameters. In the delay time window estimation method, an estimate of the delay time window is proposed based on the calculated value of the network delay for data services. This reduces the network transmission packet loss rate to a certain extent. However, many factors affect the network transmission packet loss rate, including in hopping networks; one of the important factors is the effect of the hopping period on the network transmission packet loss rate. Therefore, the second recommendation for future research is that we will consider setting a reasonable hopping period. There are two main aspects of a reasonable period. The first is that setting the hopping period length should not consume too many system resources or cause too much packet loss. The second is that setting the hopping period length should not have too great an impact on the resistance to external attacks. It is necessary to analyze the effect of the hopping period on these factors and then propose a reasonable hopping period scheme.

In future research, we will continue to evaluate the impact of the hopping period on the transmission performance of the system.

6. Patents

Zhengquan Xu, Zhu Fang. A method for calculating delay time windows in multi-node networks of hopping networks. National Invention Patent, Patent number 2021107766.

Author Contributions: Z.F. and Z.X. reviewed the literature, drafted the manuscript, and critically revised and approved the final version before submission. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (41971407).

Data Availability Statement: Not applicable.

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

Abbreviations

TW	time window
STW	sending time window
RTW	receiving time window
ATW	arrival-data time window
ODRT	overlapping data reception time
FDTT	*i*dle data sending time

References

1. Jajodia, S.; Ghosh, A.K.; Swarup, V.; Wang, C.; Wang, X.S. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*; Springer Science & Business Media: New York, NY, USA, 2011; Volume 54.
2. Carvalho, M.; Ford, R. Moving-target defenses for computer networks. *IEEE Secur. Priv.* **2014**, *12*, 73–76. [CrossRef]
3. Xu, J.; Guo, P.; Zhao, M.; Erbacher, R.F.; Zhu, M.; Liu, P. Comparing different moving target defense techniques. In *Proceedings of the First ACM Workshop on Moving Target Defense, Scottsdale, AZ, USA, 7 November 2014*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 97–107.
4. Cai, G.; Wang, B.; Wang, T.; Luo, Y.; Wang, X.; Cui, X. Research and development of moving target defense technology. *J. Comput. Res. Dev.* **2016**, *53*, 968.
5. Webber, F.; Pal, P.P.; Atighetchi, M.; Jones, C.; Rubel, P. *Applications That Participate in Their Own Defense (APOD)*; BBN Technologies: Cambridge, MA, USA, 2003.
6. Chang, S.-Y.; Park, Y.; Babu, B.; Babu, A. Fast IP Hopping Randomization to Secure Hop-By-Hop Access in SDN. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 308–320. [CrossRef]
7. Luo, Y.-B.; Wang, B.-S.; Wang, X.-F.; Hu, X.-F.; Cai, G.-L.; Sun, H. RPAH: Random Port and Address Hopping for Thwarting Internal and External Adversaries. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015.
8. Fenske, E.; Brown, D.; Martin, J.; Mayberry, T.; Ryan, P.; Rye, E. Three Years Later: A Study of MAC Address Randomization in Mobile Devices and When It Succeeds. *Proc. Priv. Enhancing Technol.* **2021**, *3*, 164–181. [CrossRef]
9. Chang, S.Y.; Park, Y.; Muralidharan, A. Fast address hopping at the switches: Securing access for packet forwarding in SDN. In Proceedings of the NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 454–460.
10. Hong, S.; Xu, L.; Wang, H.; Gu, G. Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures: New Attacks and Countermeasures. In Proceedings of the Network and Distributed System Security (NDSS), San Diego, CA, USA, 8–11 February 2015.
11. Chang, S.Y.; Hu, Y.C. SecureMAC: Securing wireless medium access control against insider denial-of-service attacks. *IEEE Trans. Mob. Comput.* **2017**, *16*, 3527–3540. [CrossRef]
12. Al-Shaer, E.; Duan, Q.; Jafarian, J.H. Security and Privacy in CommunicationNetworks—SecureComm (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering). In *Random Host Mutation for Moving Target Defense*; Keromytis, A.D., Pietro, R.D., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 106, pp. 310–327.
13. Dunlop, M.; Groat, S.; Urbanski, W.; Marchany, R.; Tront, J. MT6D: A moving target IPv6 defense. In Proceedings of the 2011-MILCOM 2011 Military Communications Conference, Baltimore, MD, USA, 7–10 November 2011; pp. 1321–1326.
14. Lee, H.C.J.; Thing, V.L.L. Port hopping for resilient networks. In Proceedings of the IEEE 60th Vehicular Technology Conference, Los Angeles, CA, USA, 26–29 September 2004; Volume 5, pp. 3291–3295.

15. Kong, Y.; Zhang, L.; Wang, Z. Address hopping proactive defense model in IPv6 based on sliding time window. *J. Comput. Appl.* **2018**, *38*, 1936.
16. Ma, Z.; Huang, X.; Yan, S.; Zhang, P. IPv6 dynamic address tunnel model based on the sliding address window. *Telecommun. Sci.* **2015**, *31*, 74.
17. Chang, Z.; Lei, L.; Zhou, Z.; Mao, S.; Ristaniemi, T. Learn to Cache: Machine Learning for Network Edge Caching in the Big Data Era. *IEEE Wirel. Commun.* **2018**, *25*, 28–35. [CrossRef]
18. Psaras, I.; Chai, W.K.; Pavlou, G. In-Network Cache Management and Resource Allocation for Information-Centric Networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2920–2931. [CrossRef]
19. Blot, J.C. End-to-end packet delay and loss behavior in the Internet. In Proceedings of the SIGCOMM '93: Conference proceedings on Communications architectures, Protocols and Applications, San Francisco, CA, USA, 13–17 September 1993.
20. Mogul, J. Observing TCP dynamicsin real networks. *ACM SIGCOMM Comput. Commun. Rev.* **1992**, *22*, 305–317. [CrossRef]
21. Zhang, L.; Shenker, S.; Clark, D.D. Observationsonthe dynamicsof a congestioncontrol algorithm: The effects of 2-way traffic. *ACM SIGCOMM Comput. Commun. Rev.* **1991**, *21*, 133–147. [CrossRef]
22. Sanghi, D.; Agrawala, A.K.; Jain, B. Experimental assessment of end-to-end behavior on Internet. In Proceedings of the IEEE INFOCOM '93 The Conference on Computer Communications, San Fransisco, CA, USA, 28 March–1 April 1993; pp. 867–874.
23. Paxson, V. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. Netw.* **1997**, *5*, 601–615. [CrossRef]
24. Katz-Bassett, E.; Madhyastha, H.V.; Adhikari, V.K.; Scott, C. Reverse traceroute. In Proceedings of the NSDI, San Jose, CA, USA, 28–30 April 2010.
25. Tsang, Y.; Coates, M.; Nowak, R.D. Network delay tomography. *IEEE Trans. Signal Process.* **2003**, *51*, 2125–2136. [CrossRef]
26. Nistor, M.; Lucani, D.E.; Vinhoza, T.T.V.; Costa, R.A.; Barros, J. On the Delay Distribution of Random Linear Network Coding. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1084–1093. [CrossRef]