WinCM: A Window based Congestion Control Mechanism for NDN

Minxiao Wang
School of Information Engineering and Automation
Civil Aviation University of China
Tianjin, China
mxwangroot@outlook.com

Meng Yue
School of Information Engineering and Automation
Civil Aviation University of China
Tianjin, China
myue_23@163.com

Zhijun Wu
School of Information Engineering and Automation
Civil Aviation University of China
Tianjin, China
zjwu@cauc.edu.cn

Abstract*—Named Data Networking (NDN) architecture inherits the hourglass shape of IP, whereas the narrow waist part is changed from IP addresses to content names. In NDN architecture, consumers can retrieve data from multiple content chunks and through multiple paths by sending Interest packets carrying a given name rather than data objects' location. These new features lead to the invalidation of end-to-end congestion control mechanisms. In this paper, we propose a novel window based congestion control mechanism (WinCM) to support high-throughput applications in NDN. WinCM contains three modules: Active Queue Management (AQM) module, Consumer Window Adjustment module and Forwarding Strategy module. AQM module detects congestion by monitoring packet-sojourn time and notices consumers and downstream nodes along delivery path. AQM keeps marking each Data packets that are dequeued continuously for the duration of a queue delay, so that consumers can decrease their windows every time they receive a congestion tag. Simultaneously, each downstream node's Forwarding Strategy module can accurately and instantly adjust the per-interface Unsatisfied Interest Window, which is one of parameters recorded in Measurements Table to decide how to forward Interest packets and avoid congestion. Simulations based on ndnSIM show that WinCM can exploit available bandwidth faster and maximize bandwidth utilization while maintaining lower queue delay.

***Keywords*—NDN, congestion control, multipath flows, high-throughput network transport**

I. INTRODUCTION

Named Data Networking [1] is a typical Information-Centric Networking (ICN) architecture [2], which is a hot research topic about future Internet architecture. In NDN architecture, communication is driven by the receiving ends. Data consumers and producers exchange Interest packets and Data packets identified by same name prefix. Because NDN forwards packets based on names and supports caching Data objects in routers, the content objects could be delivered by multiple sources. These changes benefit applications in sharing large distributed data sets and raising throughputs.

However, traditional end-to-end congestion control mechanisms do not fit the multi-source transport form in NDN for three reasons. Firstly, both delay-based and loss-based traditional congestion controls are deployed merely on the senders, which own data objects. While, NDN should deploy congestion control on consumers, which require data. In addition, since the need to control forwarding rate of alternative paths, NDN congestion control should also be deployed on all in-network nodes. Secondly, delay-based congestion control (such as Vegas and BBR [3]) can adjust window size or sending rate by estimating RTT value, because TCP works on a single fixed link between two hosts. But NDN transports packets through multiple paths, which have different delay. Thirdly, loss-based congestion control (such as Reno, BIC [4] and CUBIC [5]) is able to react to duplicate ACKs caused by packets loss, because single-source TCP transports packets and ACKs sequentially. In NDN, however, it is impossible to maintain the Data packets from different producers or routers arriving in strict sequence. Therefore, a congestion control mechanism should be designed for NDN to exploit the high-throughput multi-path transport potential.

Congestion control mechanism for NDN should be "*Interest control*", and "*hop-by-hop*". The principle of maintaining one-on-one flow balance provides potential for avoiding congestion in NDN by controlling Interest packet, which pulls back exactly one Data packet [6]. In [6], Yi et al. also indicate that forwarding strategy should use per prefix-interface interest limit to control congestion hop-by-hop. Many existing researches proposed their congestion control solutions. HoBHIS [7], [8], [9] uses hop-by-hop Interest shaping to anticipate the drop of data packets due to buffer overflow. However, HoBHIS assumes known available bandwidth when computing the shaping rate. ICP [10] [11] prevents congestion by estimating RTT and route label, which has been questioned about its practicality. PCON [12] used forwarding percentage to adaptively adjust Interest rate hop-by-hop. But this percentage is only adjusted when congestion occurs and only represents long-term forwarding performance. There is unnecessary rate decline on aggregate nodes in PCON.

In this paper, we propose WinCM which has the features of "*multi-path distinguish*" and "*Instant forwarding adaptation*". We implement WinCM in ndnSIM [13]. The simulation results show that WinCM can exploit available bandwidth faster and maximize bandwidth utilization while maintaining lower queue delay.

The rest of paper is organized as follows. Section 2 discusses the congestion control principle for WinCM. Section 3 describes the design of WinCM. Section 4 evaluates simulation performance and compares with PCON. Section 5 concludes the paper.

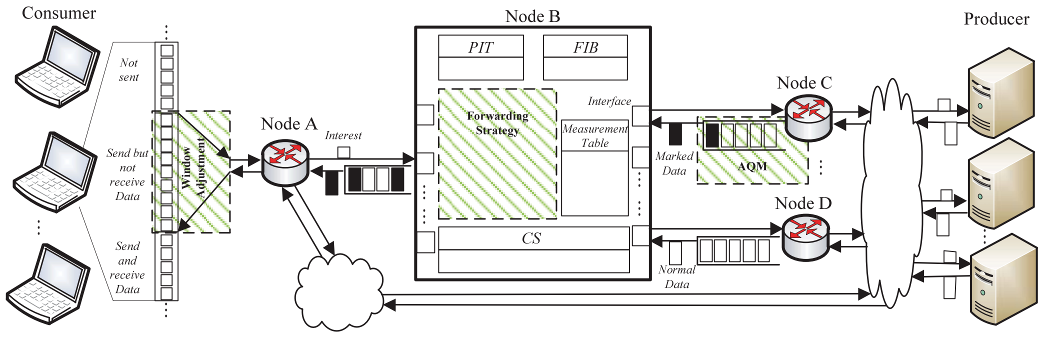

Fig. 1. The model of WinCM.

II. CONGESTION CONTROL PRINCIPLE

A NDN congestion control mechanism should include four functions: "congestion detection", "back-pressure signaling", "consumer rate adjustment" and "multi-path strategies" [14].As a system, the congestion control function should be realized by distributed modules. WinCM contains three modules to realize these functions: 1) AQM module supports congestion detection and provides back-pressure signaling. 2) Consumer Window Adjustment module controls Interest sending rate. 3) Forwarding Strategy module provides per prefix-interface Interest forwarding rate control. In this section, we introduce the design principles of each module separately.

A. AQM module

"Packet networks require buffers to absorb short-term arrival rate fluctuations [15]." Since the meaning of queues, buffers shouldn't be filled up and remained full. In order to achieve this goal, many AQM algorithms were proposed, such as RED, REM et al. In these algorithms, current length of queue doesn't mean congestion, because it may be caused by bursty traffic. They detect congestion by calculating average length of queue that indicates transport rate has exceeded link capacity. Kathleen Nichols and Van Jacobson et al. provided CoDel [15] to solve "*buffeerbloat*" problem. This algorithm has also been deployed in PCON [12] to detected congestion for NDN by monitoring packet-sojourn time, which is a better predictor of congestion. In WinCM, we adopt the same congestion detection method as CoDel, but transmit congestion feedback signals in a different way.

One of the purposes of congestion control is to help network applications to exploit bandwidth. Therefore, congestion will always occur when applications attempt to probe a link which bandwidth has been fully used. In NDN, congestion may occur on all paths when multi-path traffic reaches the maximum transmission rate. Moreover, all bottleneck routers' AQMs on these paths will inform downstream nodes and consumers that congestion has been detected. This will result in too many unclear congestion signals for consumers and downstream nodes so that they can't exploit bandwidth fully. Hence, NDN requires a back-pressure congestion signal containing more information. We propose *"multi-congestion marking"* for AQM module. In WinCM, we mark several packets instead of one when congestion is detected. The marking will persist for a duration. During the time period, all packets which are dequeued will be marked continuously. The value of the duration is equal to current queue delay, so that the number of marked packets will reveal how serious congestion is on this path. Both Consumer Window Adjustment module and Forwarding Strategy module adjust Interest sending rate based on the *multi-congestion marking*. More benefits about how AQM works with other modules will be discussed later.

B. Consumer Window Adjustment module

Traditional Congestion Avoidance algorithms use Congestion Window (cwnd) to control the amount of outstanding data in order to control congestion. However, congestion is not really avoidable, because the sender always try to increase cwnd to exploit more bandwidth. BDP (bandwidth-delay product) is equivalent to the maximum amount of data on the network circuit. If the amount of outstanding data is greater than BDP, the excessive data will be buffered in the queue of bottleneck router and result in increasing of queue delay and congestion ultimately. Most of Congestion Avoidance algorithms are Additive Increase Multiplicative Decrease (AIMD). Too many congestion signals will cause consumer to overreact to congestion. Especially, multiplicative decrease will cause the window to be extremely small in this case. Hence, we believe that AIAD is more suitable for NDN's multi-path traffic than AIMD.

In Consumer Window Adjustment module of WinCM, we modify nothing but window decrease algorithm compared with TCP. In order to handle the problem about too many congestion signals, Consumer Window Adjustment module decreases cwnd by β for each marked packet received from AQM modules. The amount of marked packets equals to the excessive data in bottleneck queues on all paths. In addition, some efficient congestion control algorithms, such as BIC and CUBIC, adjust cwnd increasing rate based on the decrease amount caused by the last congestion. The more cwnd has been decreased, the faster it recovers. The less decrease provided by AIAD can help consumers to adjust their windows more cautiously than AIMD. In consequence, AIAD can work well with *multi-congestion marking* to adjust consumers' sending rate in NDN.

C. Forwarding Strategy module

In NDN's Interest-Data exchange communication, Data packets are more likely to cause congestion because of their larger packet size. Hence, a hop-by-hop Interest rate control should be proposed to control congestion caused by Data packets on multi-path. Congestion control has been considered

This work was supported in part by the National Natural Science Foundation under grant No.U1533107 and the Major Program of Natural Science Foundation of Tianjin under grant No.17JCZDJC30XXX.

as part of the NDN Forwarding Strategy [16]. However, most existing congestion control mechanisms haven't taken "*multi-path distinguish*" into account. Aggregate node and its downstream nodes may receive congestion signals from different upstream paths. There may be a case that some paths are congested but the others are not fully used, because these upstream paths have unknown bandwidth. In this case, these congestion signals inform all the downstream nodes to adjust Interest forwarding rate ,although that is not really needed. Hence, we design a *multi-path distinguish* Forwarding Strategy to weaken the adverse impaction among multi-path.

To control congestion on multi-path, a router must decide which interface should be used for forwarding an arrived Interest packet. Meanwhile, routers also must control how fast each interface forwards Interests. In WinCM, we introduce "*Unsatisfied Interest Number*" (**UIN**) and " *Unsatisfied Interest Window*" (**UIW**) to realize the function of *multi-path distinguish* and *Instant* Forwarding Strategy without assumed knowledge. **UIN** records current number of unsatisfied Interest carrying a same FIB prefix for a given interface. **UIW** is used to exploit the sum of available capacity of all this interface's upstream paths for Interests carrying a given FIB prefix. **UIW** records the maximum **UIN** that has not caused congestion since the last congestion. In general, **UIW** is greater than or equal to **UIN**. If the interface received a marked packet carrying the same prefix, **UIN** will be decreased by 1 as normal and **UIW** will be decreased by *d* which is greater than 1. After receiving several congestion signals, **UIW** will be smaller than **UIN**. In this case, router will not forward Interests through that Interface until **UIN** is less than **UIW** again. Our Forwarding Strategy module will react to both marked Data packets and normal Data packets. The normal Data packets from uncongested upstream paths will reconcile the congestion signals so that weakening the adverse impaction among multi-path and reducing unnecessary rate decline of downstream nodes. In addition, we use ***ForwardingWeight***(**UIN**,**UIW**) to realize functions about multi-path forwarding and Instant Interest rate controlling. More design details will be given in the next section.

III. MECHANISM DESIGN DETAILS

Through analysis, we have proposed the congestion control principles of the three modules in WinCM.

For AQM module:

1) Packet-sojourn time based congestion detection: AQM should monitor the packet-sojourn time of each packet to detect congestion.

2) Multi-congestion marking: Once congestion has been detected, AQM should continue marking every dequeued packets for a duration which equals to current queue delay.

For Consumer Window Adjustment module:

3) AD Window Adjustment: Consumer should decrease window by β for each marked packet.

For Forwarding Strategy module:

4) Instant Forwarding Adaptation: Forwarding Strategy should adaptively adjust the real forwarding ratio instantly.

5) Multi-path distinguish forwarding: Forwarding Strategy should react to both marked Data packets and normal Data packets to adjust per prefix-interface Interest forwarding rate, so that reducing unnecessary rate decline.

Figure 1 shows the model of WinCM. The three modules are indicated by dotted boxes. Next we design each module's algorithm with their principles separately.

A. AQM algorithm design

Because NDN's communication mode is "receiver driven", "multi-source", and "multi-path", there are more bursty traffics in NDN than in IP. Therefore, the lengths of outgoing queues fluctuate more dramatically and congestion is more likely to occur. Hence, the buffer size should be larger than usual to absorb these bursty traffics. Meanwhile, the AQM algorithm should control the amount of buffered data so that keeping available space to absorb bursts and reducing queue delay. In WinCM, we adopt the same congestion detection method of CoDel, which detects congestion by monitoring packet-sojourn time.

In CoDel, each packet will be given a time tag when it is enqueued. The packets' sojourn time can be computed when they are dequeued. After obtaining packet-sojourn time, CoDel compares the minimum packet-sojourn time with **Target** (an acceptable value of standing queue delay, default 5 ms). The first time packet-sojourn time exceeds **Target**, AQM records the current time as **FirstAboveTime** and records the packet-sojourn time as **FirstSojourn**. If the minimum packet-sojourn time exceeds **Target** for at least an **Interval** (default 110 ms), CoDel considers that the outgoing link which the queue belongs to is congested.

Algorithm 1 AQM algorithm

```
function CHECKSOJOURNTIME($Packet, Now$)
    $sojournTime \leftarrow Now - Tag.GetTime$;
    if $sojournTime > Target$ then
        if $FirstAboveTime == 0$ then
            $OverInterval \leftarrow False$ ; $FirstAboveTime \leftarrow Now$;
            $FirstSojourn \leftarrow sojournTime$ ;
        else
            if $Now > (FirstAboveTime + Interval)$ then
                $OverInterval \leftarrow True$;
                $StopMarkingTime()$;
            end if
        end if
    end if
    return $OverInterval$;
end function

function DODEQUEUE($Packet, Now$)
    $okToMark \leftarrow$ CHECKSOJOURNTIME($Packet, Now$);
    if $okToMark$ then
        if $(Now >= NextMarkingTime)\&(Now <= StopMarkingTime)$
then
            $MarkNext \leftarrow True$; $NextMarkingTime \leftarrow Now$;
        end if
    end if
end function
```

After congestion has been detected, AQM should send congestion signals to the downstream nodes and consumers. In WinCM, we design "*multi-congestion marking*" to signal congestion so that the amount of marked Data equals to the current amount of Data which exceed link capacity. Our purpose is to inform the locations where to retrieve these marked Data packets about how many excessive Data they have required and to help them to adjust Interest rate accurately. AQM module marks the first packet when congestion is detected. Then AQM module persists in marking packets until **StopMarkingTime**. The duration from the first marking moment to **StopMarkingTime** should equal to the current queue delay, as in:

$$\begin{aligned}\mathbf{StopMarkingTime} = \mathbf{FirstAboveTime} + \mathbf{Interval} \\ + sojournTime + k\end{aligned} \tag{1}$$

Where *sojournTime* is the current dequeued packet's packet-sojourn time. However, *sojournTime* is equal to the queue delay before a *sojournTime* rather than the current delay. So we use *k* to estimate the queue delay's change after a *sojournTime*, as follow:

$$k = \frac{sojournTime - \mathbf{FirstSojourn}}{now - \mathbf{FirstAboveTime}} \times sojournTime \tag{2}$$

The AQM algorithm is given in Algorithm 1.

B. Consumer Window Adjustment algorithm design

In WinCM, we deploy a TCP-like congestion control algorithm for Consumer Window Adjustment where the relation between Interest and Data is similar to TCP packet and ACK. Consumer Window Adjustment module uses Congestion Window to control the amount of outstanding Interests and Interest sending rate. Congestion Window is increased when consumer receives a normal Data packet, and decreased when marked Data packets or a NACKs [6] are received or Timeouts. NACKs and Timeouts are always caused by packets loss for heavy congestion that AQM module tries to avoid. In addition, NACKs and Timeouts mechanisms have been effectively implemented in PCON. Hence, we adopts PCON's NACKs and Timeouts mechanisms and focus on improving Window Adjustment.

Algorithm 2 Consumer Window Adjustment algorithm

```
function OnData($dataPkt$)
    if $dataPkt.isMarked()$ then
        $++numMarked$;
        WindowDecrease($numMarked$);
    else
        if $numMarked > 0$ then
            $++normData$;
            NormalIncrease();
            if $normData > n$ then
                $normData \leftarrow 0$; $numMarked \leftarrow 0$;
            end if
        else
            BicIncrease();
        end if
    end if
end function

function WindowDecrease($numMarked$)
    if $numMarked == 1$ then
        $prevmax \leftarrow bicmax$; $bicmax \leftarrow cwnd$; $cwnd \leftarrow cwnd - beta$;
    else$numMarked > 1$
        $cwnd \leftarrow cwnd - beta$; $bicmin \leftarrow cwnd$;
    end if
    if $numMarked = 0$ then
        BicDecrease();
    end if
end function
```

Consumer Window Adjustment can adapt to different ***WindowIncrease***() functions for different kinds of applications. However, it needs the same ***WindowDecrease***() function to react to congestion signals. ***WindowDecrease***() uses an *AD* algorithm to decrease cwnd by β (greater than 1, default 4) for each marked Data packet. The amount of marked Data packets according to "*multi-congestion marking*" equals to the mount of excessive Data that required by this consumer. Meanwhile, based on the one-on-one flow balance principle it also equals to the amount of excessive Interest this consumer has sent. Thus, Consumer Window Adjustment can adjust cwnd and Interest rate accurately. In addition, consumer starts counting the number of marked Data since the first marked Data until *n* (default 8) continuous normal Data packets are received. During congestion state, each normal Data packet increase cwnd by 1/cwnd. In this paper, we take BIC as an example to describe algorithm detail for the reason of comparing WinCM's performance with PCON conveniently. The algorithm is given in Algorithm 2.

C. Multi-path distinguish and Instant forwarding strategy design

In WinCM, we introduce "*Unsatisfied Interest Number*" (**UIN**) and " *Unsatisfied Interest Window*" (**UIW**) for each FIB prefix and each interface to realize the function of *multi-path distinguish* and *Instant* Forwarding Strategy without assumed knowledge.

UIN records current number of unsatisfied Interests carrying a same FIB prefix for a given interface. **UIN** is increased by 1 for an Interest packet and is decreased by 1 for a Data packet whether is marked. **UIW** is used to exploit the sum of available capacity of all this interface's upstream paths for Interests carrying a given FIB prefix. **UIW** records the maximum **UIN** that has not caused congestion since the last congestion. **UIW** is decreased by *d* (greater than 1, default 1.5) for a marked Data packet. Different from the window of end device, **UIW** works on the router's interface and it is impossible to known how many Interest packets will arrive. Although Forwarding Strategy should contain hop-by-hop congestion control, however there is no reason to reject any legal Interest packet. Thus, **UIW** should be increased by 1 rather than strictly limit **UIN**, when **UIN** equals to **UIW** and an Interest packet is decided to be forwarded through this interface.

After defining **UIN** and **UIW**, we use them to calculate the each alternative interface's forwarding weight **ForWeight** for Forwarding Strategy module, as follow:

$$\begin{aligned}\mathbf{ForWeight} = (\mathbf{UIW} - \mathbf{UIN})(1 - \frac{\mathbf{UIN}}{\mathbf{UIW}}) \\ + \frac{ex\mathbf{UIW}}{(\mathbf{UIW} - ex\mathbf{UIW} + 1)(\mathbf{UIW} - \mathbf{UIN} + 1)}\end{aligned} \tag{3}$$

Where *ex***UIW** is the old **UIW**, it is updated to **UIW** every 100ms or when receiving a marked Data.

Forwarding Strategy module uses the percentage of all interfaces' **ForWeight** to decide which interface should be used for forwarding an incoming Interest packet and control how fast each interface forwards Interests. Hence, calculating **ForWeight** should take account of the state of interface for the given FIB prefix. **UIW** and **UIN** can indicate the state of interface for this FIB prefix: **UIW**>**UIN** means the window is not yet saturated, this interface should be preferred; **UIW**=**UIN** means the window is saturated, nevertheless this interface still can be used; **UIW**<**UIN** means the window is too big and this interface shouldn't be used until **UIN** reduces to less than **UIW** (when **UIW**<**UIN,** we regard the interface as ineligible and it's **ForWeight** is 0).

As equation (3), **ForWeight** has two parts and they play different roles. The first (upside) part is used to make sure the unsaturated interface is preferred. The lower utilization ratio (**UIN/UIW**) of the interface, the greater the first part. The second (downside) part controls that all interfaces' UIW are fairly increased. When UIW should be increased, UIN equals to UIW and the first part equals 0. The more UIW has been increased recently the smaller the second part will be. Hence, the forwarding percentage **ForWeight**/**sumForW** of each interface is sensitive to interface state and is used to choose outface. The **ForWeight** can adaptively adjust the real forwarding ratio instantly. That will be more effective to avoid congestion than a fixed forwarding percentage that only represents a long-term effect. In addition, the **ForWeight** reacts to both marked Data packets and normal Data packets. For example, figure 1 shows an interface of Node A receives both marked Data packets and normal Data packets. However, congestion only occurred on the path where Node C is at, while another path where Node D is at hasn't been fully used. Therefore, this interface of Node A shouldn't simply reduce the forwarding rate for a marked Data. In WinCM, the normal Data packets from uncongested upstream paths will decrease UIN more than UIW, so that weakening the adverse impaction among multi-path and reducing unnecessary rate decline of downstream nodes. The algorithm is given in Algorithm 3.

Algorithm 3 Multi-path distinguish Forwarding Strategy
```
function OnInterest($interestPkt, inface, fibEntry, pitEntry, Now$)
    UpdateExUIW($Now$);
    for auto $m : fibEntry \rightarrow getNextHops$ do
        if $m.UIW >= m.UIN$ then
            $eligibleFace.pushback(m)$;
        end if
    end for
    for auto $n : eligibleFace$ do
        $ForWeight \leftarrow$ ForwardingWeight($n.UIN, n.UIW, n.exUIW$);
        $sumForW \leftarrow sumForW + ForWeight$;
    end for
    ChooseOutface($sumForW, UIN, UIW, exUIW$);
    if $outface.UIN == outface.UIW$ then
        IncreaceUIW($outface$);
    end if
    IncreaceUIN($outface$);
end function

function OnData($dataPkt, inface, fibEntry, pitEntry, Now$)
    DecreaceUIN($inface$);
    if $dataPkt.isMarked()$ then
        DecreaceUIW($inface$);
        SetExUIW($inface$);
    end if
end function

function ChooseOutface($sumForW, UIN, UIW, exUIW$)
    $r \leftarrow$ random(1); $S \leftarrow 0$;
    for auto $m : eligibleFace$ do
        $S \leftarrow S + m.ForWeight$;
        if $r <= S/sumForW$ then
            $outface \leftarrow m$;
        end if
    end for
end function
```

IV. Performance Evaluation

We implement WinCM in ndnSIM [13] and evaluate its performance against PCON. We restored the experimental topology and parameter settings the same as in [12] Topo 4.4. The details of the experimental topology are given in figure 2. PCON's forwarding strategy has two options: "shortest first" (default option) and "equal splitting". We restored both of them and compared WinCM with PCON from three aspects: queue delay, throughputs, and forwarding percentage.

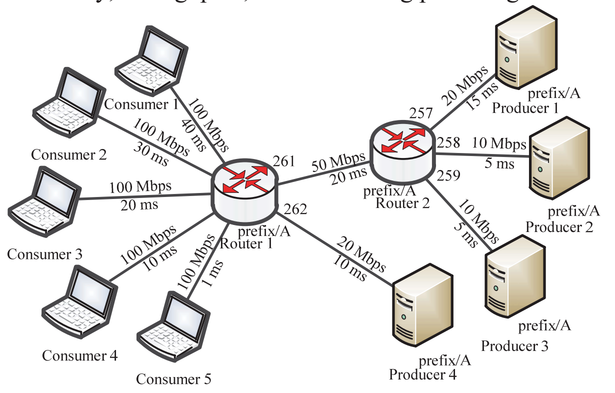

Fig. 2. The experimental topology.

First of all, we compare the queue delay of each path between WinCM and PCON. We define the different paths as the links from router1 to all producers. The results are presented in figure 3, where PCON 1 means true for "shortest first" option and PCON 2 means true for "equal splitting" option. In figure 3, we see that WinCM can control the queue delay of each path to be lower and more stable than PCON 1 and PCON 2. This is because the **ForWeight** can balance the Interest flows instantly to avoid most Interest packets entering the same path in a short time.

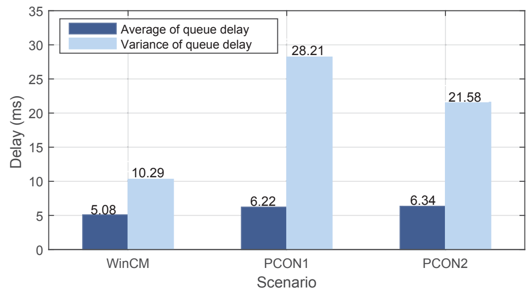

Fig. 3. The difference in queue delay of each scenario between WinCM and PCON.

Secondly, we present the throughputs of all consumers and their sum in WinCM and compare the total throughputs with PCON. In figure 4, the upside part shows that WinCM can exploit available bandwidth faster and maximize bandwidth utilization. Usually, the small buffer size will result in low bandwidth utilization of window based congestion control algorithms. The cwnd will be reduced more than the exceeded amount. Hence, the AD algorithm in Consumer Window Adjustment do help in maintain throughput at a high level stably.

The total throughputs contrast in bottom part shows WinCM can exploit bandwidth faster than PCON 1 and also provide a higher bandwidth utilization (96.59%) than both PCON 1 (93.24%) and PCON 2 (89.31%). That is due to that AD algorithm can adjust cwnd based on *multi-congestion marking* more accurately and more cautiously. The BIC algorithm deployed in consumers adjusts cwnd increasing rate based on the decrease amount, so that an accurately and cautiously adjustment will help in transporting stably. That is also due to *multi-path distinguish* and *Instant forwarding adaptation*. They can avoid more congestion and adjust Interest forwarding rate instantly before congestion becomes more serious.

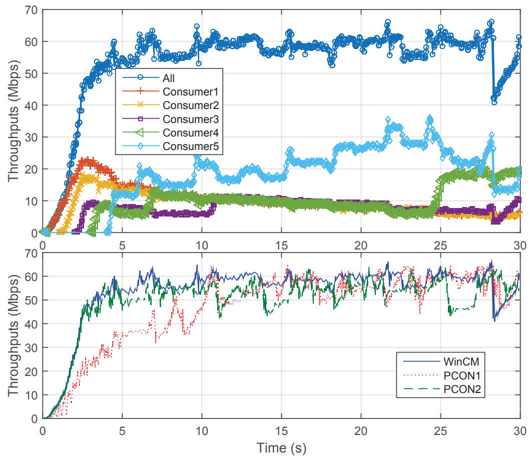

Fig. 4. The throughputs of WinCM compared with PCON.

Finally, the upside of figure 5 shows the variation of **UIW**s of each interface. During the first 5 seconds, all interfaces' **UIW**s increase quickly. Then **UIW**s maintain steady state and the value of them rely on each links' packets capacity. The bottom figure shows "*multi-path distinguish"*, at about 2 seconds, the **UIW**s of Face 258 and 257 are decreased (means congestion occurs) but it will not stop the **UIW** of Face 257 keep increasing. We record both **UIW** and **UIN (UIN** equals to the number of pending Interests that also be used as a forwarding parameter in ICP [11]**)** in the measurement table [17].

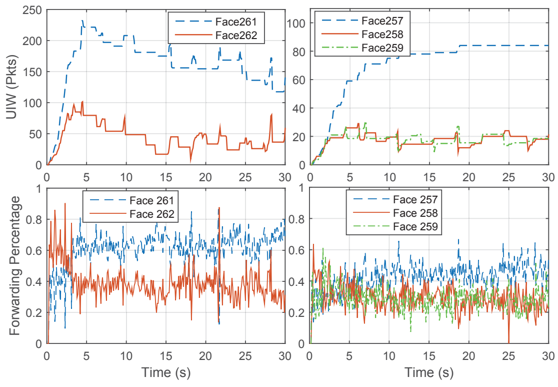

Fig. 5. The **UIW**s and Forwarding Percentages of interfaces in WinCM.

The bottom of figure 5 shows the forwarding percentage of WinCM is very sensitive. In WinCM, Forwarding Strategy module calculates forwarding percentage for each Interest to figure out the most suitable interface. A splitting ratio only represents long-term forwarding performance. The results show that WinCM can react to both marked Data packets and normal Data packets and can instantly balance the current Interest rate of all available interfaces to control congestion. However, PCON only can adjust the forwarding percentage every time when receiving a congestion signal. That means it will spend more time for PCON to adjust forwarding percentage to the optimal value.

Figure 6 shows the difference in the throughputs of each interface between WinCM and PCON. We can see that WinCM can find the optimal value faster. And there are less rate burst in WinCM than PCON. Although the forwarding percentage is more sensitive, the throughput of each interface in WinCM is more stable than PCON. Hence, this result provides that WinCM adaptively adjust the real forwarding ratio instantly to balance multi-path flow rate, so that WinCM can better deal with congestion than PCON and support high-throughput applications in NDN.

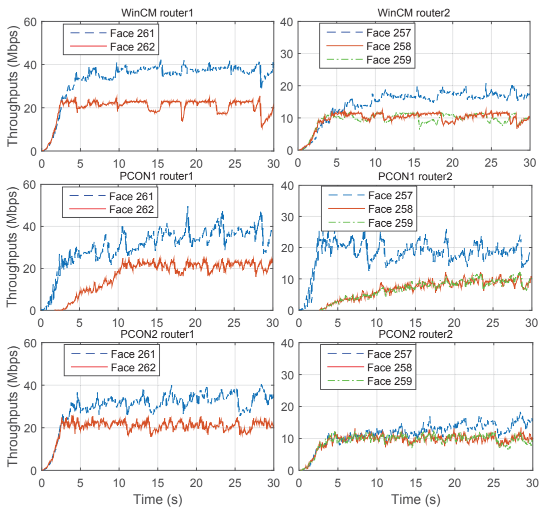

Fig. 6. The difference in throughputs of each interfaces between WinCM and PCON.

V. Conclusions

In this paper we design WinCM, a window based Congestion Control Mechanism for NDN. WinCM has three components: AQM module, which has the features of packet-sojourn time based congestion detection and multi-congestion marking; Consumer Window Adjustment module, which adjusts cwnd based on congestion signals; Forwarding Strategy module, which adaptively adjust the real forwarding ratio instantly and react to both marked Data packets and normal Data packets to adjust per prefix-interface Interest forwarding rate, so that reducing unnecessary rate decline. The performance of WinCM is evaluated by simulations in ndnSIM. The simulation results shows that WinCM can exploit available bandwidth faster and maximize bandwidth utilization while maintaining lower queue delay.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3,pp. 66–73, 2014.

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, and et al, "A survey of information-centric networking research," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson, "BBR: Congestion-Based Congestion Control", *ACM Queue*, vol. 14, no. 5, pp. 50:20-50:53, Oct. 2016.

[4] L. Xu, K. Harfoush, I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", *Proc. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, March 2004.

[5] Rhee Injong, Xu Lisong, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *Proc. Workshop on Protocols for Fast Long Distance Networks*, 2005.

[6] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, L. Zhang, "A case for stateful forwarding plane", *Computer Communications: Information-Centric Networking Special Issue*, 2013.

[7] N. Rozhnova, S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications", *Proceedings of IEEE INFOCOMM NOMEN Workshop*, 2012.

[8] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking", *Proc. of SIGCOMM ICN Workshop*, 2013.

[9] M. Mahdian, S. Arianfar, J. Gibson, D. Oran, "Mircc: Multipath-aware icn rate-based congestion control", *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking*, pp. 1-10, 2016.

[10] G. Carofiglio, M. Gallo, L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking", *Proceedings of IEEE INFOCOMM NOMEN Workshop*, 2012.

[11] G Carofiglio, M Gallo, L Muscariello et al., "Optimal multipath congestion control and request forwarding in information-centric networks", *Network Protocols (ICNP) 2013 21st IEEE International Conference on*, pp. 1-10, 2013.

[12] K. Schneider, C. Yi, B. Zhang, L. Zhang, "A practical congestion control scheme for named data networking", *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking. ACM*, pp. 21-30, 2016.

[13] S. Mastorakis, A. Afanasyev, I. Moiseenko, L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3", pp. 2015.

[14] V.Jacobson, J. Burke, L. Zhang, and et. al, "Named Data Networking Next Phase (NDN-NP) Project May 2015 - April 2016 Annual Report", *Named Data Networking (NDN)*, 2016.

[15] K. Nichols, V. Jacobson, "Controlling queue delay", *Commun. ACM*, vol. 55, no. 7, pp. 42-50, May 2012.

[16] V.Jacobson, J. Burke, D Estrin, L. Zhang, and et. al, "Named Data Networking Next Phase (NDN-NP) Project 2012-2013 Annual Report", *Named Data Networking (NDN)*, 2013

[17] A. Afanasyev, J. Shi, B. Zhang, L.Zhang, and et al., "NFD developer's guide", 2014. http://named-data.net/publications/techreports/ndn-0021-4-nfd-developer-guide/.