A proactive transport mechanism with explicit congestion notification for NDN

Jianer Zhou*†, Qinghua Wu*†, Yonggong Wang*†, Zhenyu Li*, Gaogang Xie*
*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
†University of Chinese Academy of Sciences, Beijing, China
{zhoujianer, wuqinghua, wangyonggong, zyli, xie}@ict.ac.cn

***Abstract*—Named Data Networking (NDN) is a new Internet architecture that shifts the communication paradigm from *where* to *what*. The data transport in NDN is completely driven by consumers via the Interest sending rate. An efficient transport control mechanism therefore should provide consumers with accurate network condition variation quickly. We in this paper presents an ECN-based (Explicit Congestion Notification) approach that explicitly piggybacks the network condition information in Interest and Data packets. As Data packet follows the exactly same path of Interest packet, the information carried back by Data packets accurately indicates the network condition. Consumers then proactively adjust Interest sending rates according to such accurate information to maximize the link utilization while avoiding congestion. We further propose a SDN-based smart forwarding mechanism that schedule traffic flows among available paths with global network information. Such a smart forwarding mechanism could further improve the resource utilization of the whole network. We finally evaluated the proposed approaches by packet-level simulation in ndnSIM. The results show that our approaches outperform TCP-style NDN transport mechanisms in link utilization, packet dropping and flow complete time.**

***Index Terms*-NDN, Transport mechanism, ECN, Adaptive forwarding**

I. INTRODUCTION

Internet consumers typically care more about *what* information they want, instead of *where* it is located. However, the Internet architecture is initially designed for point-to-point communication. This mismatch between user demand and network infrastructure makes Internet applications struggling with the gap between where and what. NDN (Named Data Networking) [1], a clean slate network architecture, is proposed to address this problem. It shifts the communication paradigm to data chunks, where each data chunk is uniquely identified by a name for addressing and caching.

The data transmission in NDN is consumer-driven. Consumer sends *Interest* to express the intent on content. The transmission is hop-by-hop and intermediate routers keep states of the transmission. Data packet follows the reverse path of Interest. Intermediate routers could use adaptive forwarding [3] to select the next-hop among the multiple available ones according to the network condition and policies. If there is no satisfying interface for forwarding deal to congestion, the router responds to downstream routers with explicit NACK [3]. This could be viewed by the downstream routers as an indication of congestion. The consumer slows down the Interest sending rate when receive NACK from the upstream router. As one Interest packet retrieves one Data packet, slowing down the Interest sending rate would essentially decrease the data traffic. While simple, such a reactive congestion control mechanism cannot react to the congestion quickly on links several hops away from consumers. Thus, proactive transport control that leverages the information of whole path is highly demanded for NDN, in order to achieve effective and efficient data transmission.

Existing transport control mechanisms in NDN, such as ICP[2], CCTCP[4] and HR-ICP[5], adopts the basic idea of TCP in NDN. They still use reactive congestion control and therefore suffer from low link utilization and high packet dropping rate. In fact, the fact that Data packet follows the exact path of Interest provides us an opportunity for consumer to perceive the congestion in the path and consequently adjust the Interest sending rate. We in this paper explore this design space by using Explicit Congestion Notification (ECN) [7]. ECN provides consumers with the network resource usage condition by explicitly carrying such information in the response packet. In such a way, consumers can proactively adjust the Interest sending rate according to network condition. We implement ECN-based transport control by only piggybacking very limited information which could be updated by intermediate routers.

The transmission efficiency of the network could be further improved by global coordination of traffic flows. To this end, we further propose a SDN-based mechanism, which employs controllers to collect the network condition of the whole network. The availability of global information enables the scheduling of flows among different paths, which in turn could maximize the network resource utilization and the transmission performance as well. In detail, we make three technical contributions as follows.

1. We propose an ECN transport mechanism in NDN, which can ultimately use link bandwidth using ECN information to control consumers' Interest sending rate.
2. Based on the ECN transport mechanism, we further utilize SDN to design smart forwarding mechanism to fully exploit the whole network resource.
3. Packet-level simulation shows that by joining ECN and smart forwarding, the bottleneck link utilization improves 10%-20%, and there is almost no dropping packets.

The rest of the paper is organized as followed: Sec. II discusses related works. Sec. III introduces the ECN-base transport mechanism and the smart forwarding. In Sec. IV, we demonstrate the effectiveness of our mechanism. Finally Sec. V concludes the paper.

II. RELATED WORK

Congestion control is a hot topic in both TCP/IP network and NDN network. We first survey the mechanism in TCP/IP network:

Congestion Window. Congestion window mechanism is widely deployed nowadays. Each end in the communication maintains a congestion window, which additively increases if the RTT is below the estimated value, otherwise multiplicatively decrease. This mechanism is known as AIMD[9]. Congestion window mechanism is easy to deploy since it only requires modification in end hosts. However, congestion window results in low utilization in high bandwidth-delay networks because of the slow start function. Moreover, it results in unfairness for short flows when mixed with long flows[16].

Explicit Congestion Notification (ECN). ECN mechanism notifies network condition by explicit information, such as in XCP[7], RCP[8]. Router generates ECN information according to network condition and send it to end host. End host adjusts its transmission rate according to the ECN information it has received. Compared with implicit congestion information in TCP, ECN can effectively utilize network resource and achieve fairness. However, RCP uses congestion information to control server's transmission rate, which is contrary to the consumer-driven nature in NDN. Besides, RCP does not consider the impact of Data size on the estimation of flow size, which is also essential in congestion control in NDN.

Several transport mechanisms have been proposed for NDN. Most of them is TCP-style. Some researchers use NDN's features, such as the NACK, to improve the TCP-style mechanism.

Receiver driven TCP-style. Most of proposed transport mechanisms in NDN are receiver-driven. Receiver adjusts its Interest sending rate according to the RTT of the coming back Data, using the AIMD and slow start principle[12]. Contug et al.[15] propose transport mechanism which mixes receiver-driven mechanism and the basic principles in TCP. It changes the congestion-controller from sender to receiver. In[13], Sara et al. separate the traffic into different flows according to name prefix. Each flow uses TCP-like mechanism to control the congestion. Routers fairly allocate the bandwidth among flows, and use optimal buffer algorithm to improve the performance. In NDN, due to in-network cache, content can be retrieved from different providers or routers. Different locations of content may result in multipath in transport layer. In [14], Givonaan et al. deal with the multipath problem in NDN, using similar way as the MTCP.

Interest NACK. An Interest is much smaller than a Data. Intuitively, it is much more "resource-saving" if we drop Interest instead of Data when congestion happens. In HR-ICP[5], routers shape the Interest hop-by-hop to handle or prevent congestion. Each router shapes Interest by itself, according to the bandwidth it can supply to incoming Data. In [17] Wang et al. improve the Interest shaping mechanism. Once an Interest is shaped, router sends NACK back to the receiver to inform that congestion has occurred. Although NACK implies congestion, it does not inform how severe the congestion is. The receiver simply cuts half of the Interest sending window and begin slow-start when it receives NACK. Although cutting the window to half and slow-start can deal with congestion, they reduce the link utilization. Furthermore, the shaping mechanism, no matter Interest or Data, causes retransmission, and that increases the flow complete time.

Adaptive forwarding. Adaptive forwarding is a main feature of NDN data plain. In TCP/IP, forwarding table completely follows the route table without any adaptability.

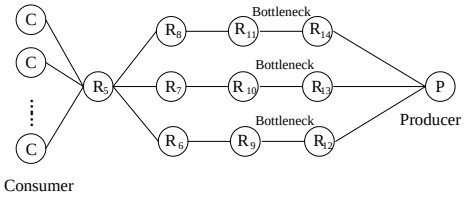

Fig. 1. A mesh NDN network topology

However in NDN, during the forwarding process, router can adaptively choose a forwarding interface from several available paths according the network situation. In [3] Cheng, et al. make use of the adaptive forwarding mechanism to design a hop-by-hop congestion control mechanism. Routers adaptively forward the Interest to another interface when it detects that the next hop has been congested. However such just one hop detection is not enough if the congestion happens on later link, because the one hop detection cannot sense the congestion on later link immediately.

III. Design

In this section we first describe our design rationale. Then we introduce the ECN-based Interest sending rate and smart adaptive forwarding mechanism.

A. Design rationale

When designing transport control mechanism for a new network architecture, the first question is which entity should be responsible for the congestion control. In TCP/IP, as its push principle, it is the sender who deals with network congestion by adjusting the packet sending window. Under the pull nature of NDN, it is obvious that receivers should be responsible for the network congestion. The one-to-one mapping of Interest-Data makes it possible to control the network traffic through the control of Interest sending rate. Such control congestion is called receiver-driven transport mechanism[15]. Our design also follows the receiver-driven principle.

If multiple flows are transmitted through one link, it is ideal that the link bandwidth is fully utilized and at the same time fairly shared by all the flows. ECN information reflects the network resource situation such as the available bandwidth and the number of flows on the path. According to the ECN information, receiver adjusts its Interest sending rate to fully use the bandwidth and achieve fairness. So to fully use link bandwidth we first use the ECN information to control the receiver's Interest sending rate.

By controlling receivers' Interest sending rate, we can perhaps fully use the link bandwidth and avoid congestion. But if all the flows choose a path that goes through the same bottleneck, even we can fully use the link bandwidth and achieve fairness, the flow complete time can be very low, because too many flows share the same bottleneck. So if there are several paths available for the flows, how can we allocate the multiple paths to each flow to minimize the total flow complete time?

In TCP/IP, only a single path is available for each source destination pair, and the forwarding process strictly follows the single path, thus adaptively selecting alternative path is very difficult. However, in NDN, the adaptive forwarding makes it possible. Taking the mesh network topology in Fig. 1 for example, a flow goes through $R5$. The flow has two available

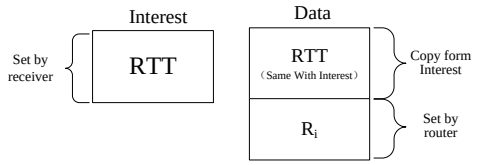

Fig. 2. Explicit congestion notification header for Interest and Data.

paths to get the data from provider. Router $R5$ can easily measure the congestion condition at the link1 (between router $R5$ and $R7$) and link2 (between router $R5$ and $R8$). If the router senses that link1 is much more congested than link2 then the router can adaptively forward the flow to link2. By such adaptive forwarding, the network resource can be effectively used and the flow complete time will be reduced.

However, such one-hop measurement is very limited. Taking the same situation for example, the true bottleneck happens on link3 (between router $R10$ and $R13$). But the router $R5$ can measure just one hop situation, it cannot sense the true bottleneck is on the link3, so it will still forward the Interest to link2. The wrong forwarding decision will put more burdens to link3, and increase the flow complete time of all the flows.

The SDN-style controller can get the whole network information. We will introduce the SDN's network-wide information to overcome the "limited information" problem. By the network-view information, the Interest can be forwarded to the best path. The whole network bandwidth utilization will be improved and total flow complete time can be reduced. So to improve the whole network bandwidth utilization and minimize total flow complete time, we further use SDN's network-wide information to design smart forwarding mechanism.

B. ECN-based Interest sending rate

We suppose every router fairly assigns its bandwidth to every flow. Our motivation is to send Interest at a maximum rate that the path can handle the corresponding coming back Data without dropping. To achieve this, every router on the path should calculates maximum Interest sending rate for the flows that goes through it. We define such rate of every router as R_r.

Fig. 2 shows the Interest and Data ECN header. The ECN header contains the explicit congestion notification information on the path. The Interest carries the RTT(Round Trip Time) of the flow and this flow's Interest sending rate R_i. The RTT is defined as the interval between the time receiver sends an Interest to the time it receives the corresponding Data. In our design, RTT is used to determine the updating interval of R_r. The provider copies the Interest's RTT to Data when it send back the Data, and the routers do not change it along the path. As the first Interest of a flow does not know the RTT, we set the first Interest's RTT as 0. R_i is the minimum R_r of routers along flow i's path. The router compares the R_i that is recorded in the Data with its own R_r. If the router's rate is less than the one in the packet, it replaces it. By this way, the coming back Data carries the minimum R_r of all the routers along the path. As the Data comes back along the same path of the Interest, the rate on the Data also reflects the path of the Interest. Each receiver sends its Interest according the R_i it receives from Data.

Suppose we know how many flows going through a link, then R_r can be determined as follows:

$$R_r = \frac{C}{Flow_{num} * Size_d} \quad . \tag{1}$$

Where $Size_d$ is the size of incoming Data, C is the bandwidth of the link, $Flow_{num}$ is the number of flows on the link. Eq. 1 is the ideal situation. When the network condition changes, such as a new flow is added, R_r should be updated. To make the update process reasonable and let the system enter stable situation, Eq. 1 is improved as:

$$R_r(t) = R_r(t - RTT_{avg}) + \frac{\alpha(C - S(t)) - \beta\frac{Q(t)}{RTT_{avg}}}{Flow_{num} * Size_d} \tag{2}$$

Where $R_r(t)$ is the Interest sending rate that the router assigns to all flows at time t, $S(t)$ is the speed of coming back Data, $Q(t)$ is the packets in the queue at time t, α and β are the parameters that influence the convergence and performance, RTT_{avg} is the average RTT of all the flows that go through this router. We set RTT_{avg} as the updating interval of $R_r(t)$. As Eq. 2 shows, α influences how the bandwidth is used. If α is larger then bandwidth will be occupied quickly. Parameter β influences how quickly that the packets in the queue can be drained. Obviously, larger α and β can help the system to use the resource quickly. But larger α and β will make the system become unstable, as the network is difficult to convergence. In Sec. IV, we will further discuss the α and β's influence to stability using evaluation results.

The definition of $R_r(t)$ in Eq. 2 is explained as follows. The available bandwidth and queue should be fairly shared by all the flows, so the link's available resource is divided by the number of flow. If $(C - S(t)) > 0$, more available bandwidth could be used and the transmission rate of each flow should be increased. Otherwise, the bandwidth has been over used and the sending rate should be reduced. We assume that the packets in the queue should always come to zero. If it is not zero, it means too many Data packets flow into this link, and the Interest sending rate should be reduced. In every RTT_{avg} the router should drain $Q(t)/RTT_{avg}$ Data packets. Because the $R_r(t)$ is the Interest sending rate, and the available resource is supplied to the Data, so we divide it by the size of Data.

If router tends to make the system converge to stable stage more quickly, it can update $R_r(t)$ with shorter interval T ($0 < T \leq RTT_{avg}$). Then Eq. 2 becomes:

$$R_r(t) = R_r(t - T) + \frac{\frac{T}{RTT_{avg}} * (\alpha(C - S(t)) - \beta\frac{Q(t)}{RTT_{avg}})}{Flow_{num} * Size_d} \quad . \tag{3}$$

In NDN, Interests which have same prefix can be treated as in a same flow[13]. And it is possible to calculate the number of flows that goes through a path by the prefix of Interest. However, using the prefix of the Interest name to estimate the number of flows traversing through the router will add complexity to the router. In[8], it has been proved that the processor-fair resource allocation method can estimate the flow number by the each flow's sending rate. Processor-fair means routers fairly allocate link bandwidth and queue resource to all flows through this link. Thus we also use process-fair resource allocation method to calculate the number of flows that go through this link:

$$Flow_{num} = \frac{C}{R_r(t - RTT_{avg}) * Size_d} \quad . \tag{4}$$

As we set every flow share the link bandwidth equally, and every flow's rate is the same, it is reasonable to use Eq. 4 to estimate the number of flows. In Sec. IV, we will evaluate by simulation that the estimation is correctly. Combining Eq. 4 with Eq. 3 we have:

$$R_r(t) = R_r(t-T)[1+\frac{\frac{T}{RTT_{avg}} * (\alpha(C - S(t)) - \beta\frac{Q(t)}{RTT_{avg}})}{C}] \ . \tag{5}$$

Many factors may influence the size of Data in NDN, such as the different MTU of different link. So it will be very difficult to exactly measure the size of Data. When we need to use the size of Data to test the accuracy of the flow number estimation, we have to use the historical information to estimate the size of Data. From Eq. 5 we can find that, $R_r(t)$ does not need to measure the flow number directly and the size of Data. That greatly simplifies the router's calculating process.

C. Smart adaptive forwarding

In this paper we just control the forwarding process, not the route calculating process. In NDN, Data may have multiple replicas which are distributed in different hosts, so there are multiple paths to get a Data. And even the Data from the same provider may have different available paths. We assume the receivers have multiple paths to get a Data, and the routers have known every hop of different paths. The smart adaptive forwarding mechanism we proposed is independent from the route calculation.

We set that there is a controller to gather the network information. By the network information, the controller calculate the best forwarding decision and then inform the routers. Every router sends its own $R_r(t)$, the transmit delay and the bandwidth of the next hop to the controller at an interval of RTT_{avg}. After several RTT_{avg}, the controller can know every router's $R_r(t)$ and the transmit delay of every hop. We denote the information as forwarding-assistant information. The network's route information can also be stored in the controller. The forwarding-assistant and route information stored in the controller is showed in Fig. 3. As Fig. 3 shows, the forwarding-assistant information table contains the $R_r(t)$, the transmit delay and the bandwidth of every router in the network. The route information record that through what paths Interest can get its Data. For example, Interest with prefix1 can get its corresponding Data from two path: R1-R2-R3 and R1-R4-R3. By the forwarding-assistant information and route information, we can calculate the best forwarding strategy.

Using the Interest sending rate we propose above, the FCT of each flow is:

$$FCT = \frac{Size_f}{Size_d * R_b} + RTT \tag{6}$$

where $Size_f$ is the size of the flow, R_b is the bottleneck router's Interest sending rate. Eq. 6 means that FCT is the time that the flow goes through the bottleneck plus the RTT of this flow. The RTT of this flow can be calculated by the forwarding-assistant information.

We suppose the i-th flow has several available paths. If it chooses path j, then this flow's FCT on path j should be:

$$FCT_{i,j} = \frac{Size_f}{Size_d * R_b^{'}} + RTT_j \tag{7}$$

where RTT_j is the RTT of flow i if it chooses path j and $R_b^{'}$ is the bottleneck router's Interest sending rate on path j.

Forwarding assistant information			
R1	$R_r(t)$	Trans delay	Bandwidth
R2	$R_r(t)$	Trans delay	Bandwidth
R3	$R_r(t)$	Trans delay	Bandwidth
.....			

Route information	
Prefix1	R1-R2-R3/R1-R4-R3
Prefix2	R2-R2-R4/R2-R1-R4
Prefix3	R1-R2-R3-R4
.....	

Fig. 3. Forwarding assistant and route information stored in the controller.

$$R_b^{'} = \frac{C}{Flow_{num} + 1} = \frac{C}{C/(R_b * Size_d) + 1} \tag{8}$$

For simplicity, we set the value of $Size_d$ and $Size_f$ as fixed values, and suppose $Size_{data} = Size_{flow}$. So Eq. 7 becomes:

$$FCT_{i,j} = \frac{C/(R_b * Size_d) + 1}{C} + RTT_j \tag{9}$$

Our design goal is to minimize the Total Flow Complete Time (TFCT) in the network. TFCT is the sum of all the flows' complete time. To achieve our goal we define the objective of the smart forwarding as:

$$\begin{aligned} \min \quad & \sum_{i=0}^{n} FCT_i \\ \text{s.t.} \quad & \text{path } j \text{ is available;} \\ & \forall i, P_i \in \{0,1\}; \\ & \max \min R_i \ . \end{aligned} \tag{10}$$

Where P_i is the number of paths that flow i chooses, R_i is flow i's Interest sending rate. $P_i \in \{0,1\}$ means flow i can choose at most one path. $maxmin\ R_i$ means flow i's Interest sending rate should be maximized. The reason we maximize R_i is to achieve fairness between different flows. If we do not set $maxmin\ R_i$, some flows may choose a min R_i to minimum the TFCT, and that will influence this flow's FCT.

Routers send the updated forwarding-assistance and route information to the controller at the interval of RTT_{avg}. The controller sends back smart forwarding decision for each flow when it receives the router's updating information. The smart forwarding decision is based on the unit of flow, not the unit of packet. So the overhead introduced by the controller's help is limited compared with the whole volume that goes through the router. To timely reflect the change of network information to the controller, routers can change the sending interval, and that will raise the overhead. The balance between the overhead and the accuracy of updating information can be adaptively controlled.

IV. Performance Evaluation

In this section, we study the performance of the proposed mechanism using ndnSim[18], [19].

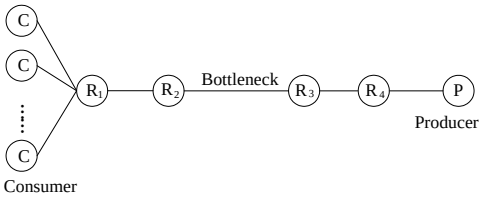

Fig. 4. Bottleneck topology using in simulation.

Our evaluation includes four parts. In the first part we discuss the influence of α and β in Eq. 5 to system stability. In the second part we evaluate the flow number estimating process, which we propose in Eq. 4. In the third part, we evaluate the ECN-based Interest sending(ECN-based) mechanism using the bottleneck network topology. At last we join the Smart forwarding with ECN-based(SECN) and evaluate SECN using the mesh network topology. The evaluation results show the flow number can be estimated accurately. Compared with ICP and ICP-shape, ECN-based mechanism performs better in link utilization, packets dropping and flow complete time. The SECN has better TFCT(Total Flow Complete Time) compared with adaptive forwarding mechanism.

A. Network Setup

Fig. 1 and Fig. 4 show the two network topologies we use in the simulation. One is bottleneck topology and the other one is mesh topology. In both network topologies, there are many consumers who send Interest into network and retrieve the corresponding Data from the producer. The number of consumers varies from 1 to 100. In the mesh network, consumers can get Data from three paths and each path's bottleneck bandwidth is different. Each link's capacity varies from 30 Mbps to 200 Mbps and the propagation delay of each hop is 10 ms. The buffer in each node is equal to the production of bandwidth and delay. In the following, the bandwidth refers to the bottleneck's bandwidth.

B. Stability analysis

To choose suitable α and β that make the system stable, we test under what α and β, the flow number can be estimated accurately. Once the flow number of the network can be accurately estimated, the Interest sending rate $R_r(t)$ can also be estimated accurately, then the system enters stable stage. So we choose the accuracy of estimating flow number as the stability metrics.

At the beginning, there are 10 flows in the network, and we test whether the flow number can accurately be estimated under different value of α and β. Fig. 5 shows that under such values the flow number can be accurately estimated. Fig. 6 shows that under these values the estimated flow number change greatly, which means that the system is not stable. From Fig. 5, we can find (0.2,1.5) is the most suitable value to estimate the flow number. Under this value, we test whether the system is still stable when the system changes. We set the number of flows in the network as 5,10,15 and 20 respectively. Fig. 7 shows that under different situations the flow number can also be accurately estimated when $\alpha = 0.2, \beta = 1.5$. That means a fixed value of α and β can make the system enter stable stage even when the system's situation changes.

From Fig. 6, we find that when α is close to 0.5, the system becomes unstable, and when it gets larger, the system becomes

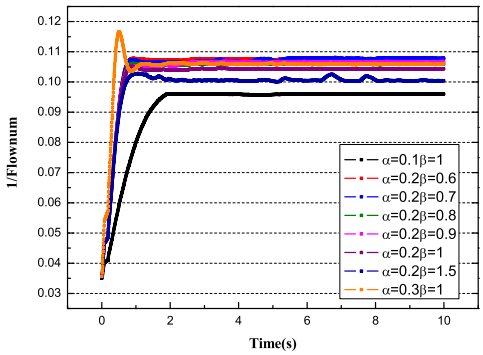

Fig. 5. Under such α and β the flow number can be accurately estimated.

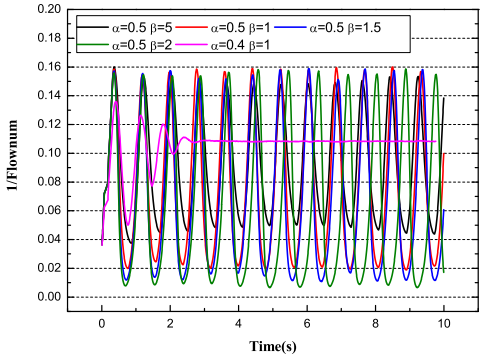

Fig. 6. Under such α and β the flow number can not be accurately estimated.

even more unstable. This can be explained as follows. Large α makes the system react too radically to increase $R_r(t)$ when $(C - S(t)) > 0$ or decrease $R_r(t)$ when $(C - S(t)) < 0$. Radical reaction will make the system difficult to converge. From Fig. 5, we find that when $\alpha < 0.2$, the system will take longer time to accurately estimate the flow number. It is because small α makes the system increase or decrease $R_r(t)$ conservatively, and that will result in longer time to convergence. When β is smaller than 1.5, although the system can be stable, the estimated flow number is not accurate compared with $\beta = 1.5$. It is because small β can not drain the packets in the queue quickly enough. And that will result in inaccurate $R_r(t)$ and flow number.

The key point of the stability analysis is that, for α and β we can choose a fixed value to make the system stable. Even when the system changes, such as the flow number, RTT and bandwidth, the fit value can also make the system keep stable. Although now we can not prove the best value by theory, it is possible to choose a stable value. In our simulation, we set $\alpha = 0.2$ and $\beta = 1.5$, according the analysis above.

C. The estimated flow number

Following Eq. 4, the flow number of the link can be estimated by the rate of this link. The size of Data can be estimated as the average size of the Data that goes through. Under the bottleneck network topology, at t=0, 10 flows start. At t=10 another 10 flows start. And at t=20, 10 flows finish, there are only 10 flows left. From Fig. 8 we can see that the flow number can be accurately estimated. Even when the flow number changes, the estimation can also converge.

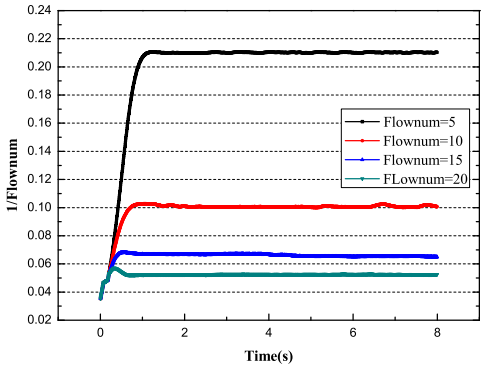

Fig. 7. Stable α and β can make the network stable even when network situation changes.

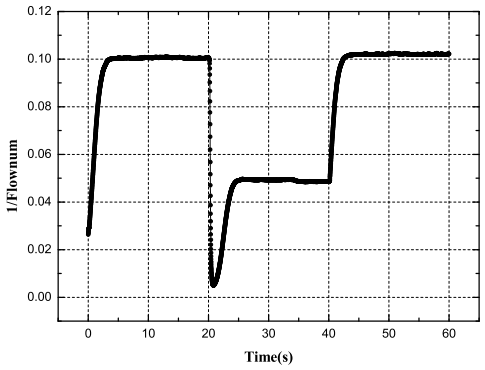

Fig. 8. The accuracy of estimation of flow number.

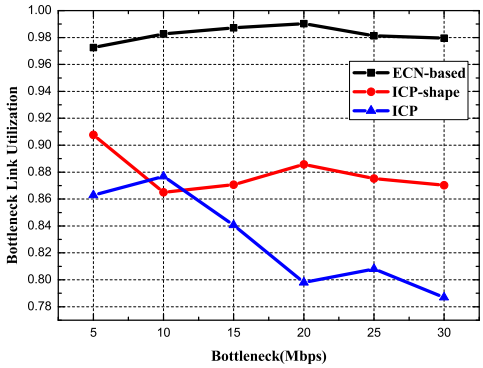

Fig. 9. The bottleneck's link utilization compared with ICP and ICP-shape when change the bottleneck bandwidth.

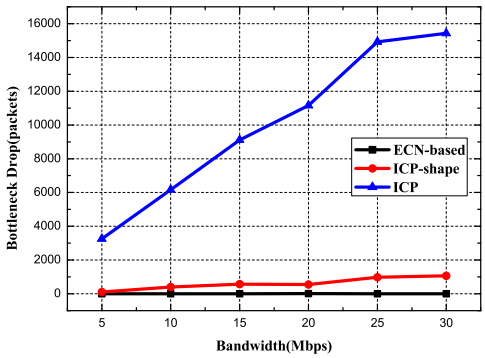

Fig. 10. The number of dropping packets in bottleneck compared with ICP and ICP-shape when change the bottleneck bandwidth.

D. The performance of ECN-based Interest sending rate mechanism

We use bottleneck topology to assess the ECN-based mechanism. ICP is a TCP-style Interest control protocol in NDN. The Interest sending window is passively changed according the RTT and loss of Data, following the AIMD principle[2]. ICP-shape also follows the AIMD principle but it discards the Interest instead of Data when the routers sense congestion[17]. ICP-shape also sends NACK back to receiver if an Interest is shaped. NACK is a feedback information used to inform that the Interest has been discarded or there's no Data corresponding to the Interest. The consumer should retransmit the same Interest when it receives a NACK. As Fig. 9 shows, ICP and ICP-shape waste bandwidth because of the slow start and AIMD principle. The link utilization of ICP and ICP-shape is between 80%-90%. In contrast, even when the bottleneck link's bandwidth changes, the bandwidth utilization of ECN-based always closes to 100%.

Because the ECN-based makes sure that the rate can't exceed the bandwidth, almost no packet (no matter Interest or Data) drops in ECN-based mechanism, as the Fig. 10 shows. ICP uses timeout as the signal to inform congestion, and timeout is caused by dropping Data. In ICP, once the link become congested, the only solution is to drop Data, so the number of dropped Data is very large. The ICP-shape shapes Interest before congestion happens, so it can reduce the number of Data needed to be dropped because of congestion. But as the delay of sending back and the difficulty of estimating the congestion, there are still some dropped Data.

As Eq. 5 shows, the packets in the queue should be drained. This principle makes sure that in ECN-based, the number of packets in the queue are always close to 0, as shown in Fig. 11.In ICP, to use the bandwidth effectively, the receivers always try to enlarge the Interest window until the Data fills up the queue and the Data is dropped, so the number of packets in the queue is largest compared with other two mechanisms. In ICP-shape, the router can shape Interest when it senses that congestion happens. So in ICP-shape, the number of queue size is less than ICP, as shown in Fig. 11.

Fig. 12 shows the FCT(Flow Complete Time) of different flows. FCT in NDN is defined as the time from when receiver sends the first Interest until the receiver receives the last Data of the flow. The ECN-based mechanism's link utilization is much higher than ICP. ECN-based mechanism fairly distributes bandwidth resource to all the flows, so even the short flow can get fair bandwidth resource. No matter flows are short or long, ECN-based'FCT is much better than ICP.

E. Flow complete time compared with ICP

We demonstrate the effectiveness of SECN(Smart forwarding with ECN-based) using mesh network topology ,as shown in Fig. 1. As Fig. 13 shows, under different total bandwidth, the SECN's TFCT is much better than non-adaptive mechanism. As smart adaptive forwarding mechanism can choose the best path on the network-wide view, it's total flow complete time is even better than adaptive mechanism. The total bottleneck bandwidth refers to the sum of bottleneck bandwidth on all the available paths. Using the forwarding-assistance and route

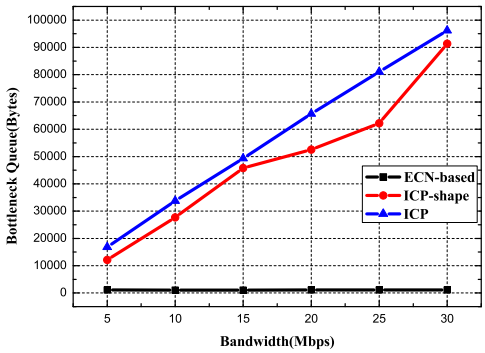

Fig. 11. The queuing packets in bottleneck compared with ICP and ICP-shape when change the bottleneck bandwidth.

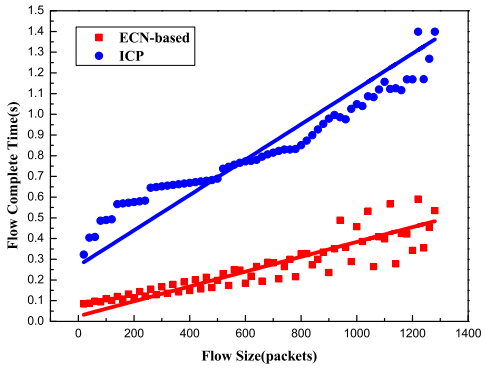

Fig. 12. ECN-based's flow complete time compared with ICP.

information, as shown in Fig. 3, the smart adaptive forwarding mechanism can choose the network-wide best path. The non-adaptive mechanism chooses the path by the route information, similar with TCP/IP. The non-adaptive mechanism can not change the forwarding interface according to the network condition. The adaptive mechanism chooses the best forwarding interface just by the next hop's link information. As the next hop link information can not reflect the whole path's information, the adaptive mechanism sometimes chooses a path whose later link is shared by far more flows.

V. Conclusion

We propose an ECN congestion control and smart forwarding mechanism in NDN. Data carries the ECN information to receivers. Receivers use ECN information to adjust its Interest sending rate. Under smart forwarding mechanism, routers choose the best forwarding interface using the SDN controller's network-view information. The ECN transport mechanism can ultimately use the link utilization and reduce the dropping packets. By joining the smart forwarding and ECN transport mechanism, the whole network resource can be effective used and total flow complete time can be reduced.

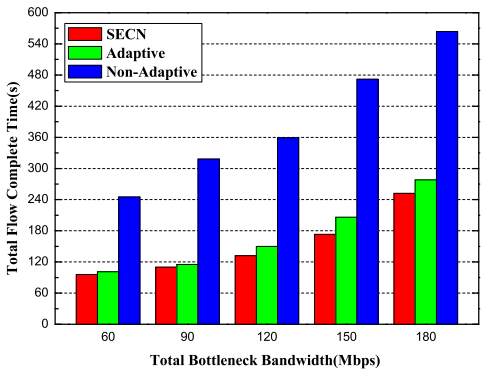

Fig. 13. Total flow complete time compared with other two forwarding mechanisms.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in Proceedings of ACM CoNEXT, 2009.

[2] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in Proc. 1st IEEE Intl Workshop on Emerging Design Choices in Name-Oriented Networking, 2012.

[3] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptivd forwarding in named data networking," in SIGCOMM Comput.Commun.Rev, 2012.

[4] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A scalable receiver-driven congestion control protocol for content centric networking," in Proc.IEEE ICC Intl Conference, 2013.

[5] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in Proc.1st IEEE Intl Workshop on Emerging Design Choices in Name-Oriented Networking, 2012.

[6] A. Voellmy and J. Wang, "Scalable software defined network controllers," in ACM SIGCOMM Computer Communication Review, 2012

[7] D. Katabi, M. Handley and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in Proc.of ACM SIGCOMM,2002.

[8] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the internet," in IWQOS, 2006.

[9] V. Jacobson, "Congestion avoidance and control," in SIGCOMM Comput.Commun.Rev, 1988.

[10] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in internet-like environments," in Proc. of the ACM SIGCOMM,2003.

[11] R. Ahmed, and M. F. Bari, " αRoute: A Name Based Routing Scheme for Information Centric Networks," in Proc. of IEEE INFOCOM 2013.

[12] S. Braun, M. Monti, M. Sifalakis and C. Tschudin, "An empirical study of receiver-based AIMD flow control strategies for CCN," in IEEE ICCCN,2013.

[13] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware Traffic Control for a Content -Centric Network," in IEEE INFOCOM, 2012.

[14] G. Carofiglio, M. Gallo, and L. Muscariello, "Multipath congestion control in content-centric networks," in IEEE INFOCOM, 2013.

[15] S. Arianfar, P. Nikander, L. Eggert, and J. Ott, "Contug: A receiver driven transport protocol for content-centric networks," in IEEE ICNP 2010 (Poster session),2010.

[16] C. Wilson, H. Ballani, T. Karagiannis , and A. Rowtron, "Better never than late: meeting deadlines in datacenter networks," in Proc. of the ACM SIGCOMM 2011.

[17] Y. G. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," in Proc. Of ACM SIGCOMM ICN, 2013.

[18] http://ndnsim.net/

[19] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," in NDN, Technical Report NDN-0005, 2012.