An explicit congestion notification transport control mechanism 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 Network(NDN) is a new Internet architecture. Its change of the network layer also sheds light on the transport layer. The Data which comes back along the same path of the Interest natively acts as a barrier of the Explicit Congestion Notification (ECN) information to receiver. Avoiding a congesting path can be achieved by adaptive forwarding which is a main feature of the NDN data plane. In this paper we implement an ECN transport mechanism in NDN, using the Data to carry ECN information. And we make use of network-wide information of the SDN controller to design smart forwarding mechanism. Our simulations in ndnSim show that the ECN transport mechanism outperform TCP-style NDN transport mechanisms in link utilization, packet dropping and flow complete time. The network-view information of the SDN controller can optimize the adaptive forwarding in NDN. By joining the ECN transport mechanism and smart forwarding, the total flow complete time can be reduced.**

***Index Terms*-NDN, Transport mechanism, ECN, Congestion control, 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. Several clean slate network architectures have been proposed to address this problem. Among those proposals, NDN[1] assigns each data a unique name for addressing and caching, which is compatible with Internet demand and exhilarate information dissemination greatly.

As NDN adopts simple best-effort packet transport between neighboring NDN nodes, congestion might occur and packets could be dropped due to congestion. Therefore transport control is essential for effective and efficient data transmission. Some TCP-style transport control mechanisms in NDN have been proposed, such as ICP[2], CCTCP[4] and HR-ICP[5]. These TCP-style proposals still suffer from the same problems as that in TCP, e.g. low link utilization, high packet dropping rate. Explicit Congestion Notification (ECN) is a promising technique to achieve high link utilization[7]. Adaptive forwarding[3] in NDN is proposed which lets router adaptively select forwarding interface among the multiple available ones according to network condition. Different from TCP/IP in which forwarding interfaces are determined statically, adaptive forwarding could react to congested links and bypass them quickly.

As far as we know, there's still no ECN-style congestion mechanism proposed for NDN and little research about congestion avoidance mechanism by utilizing adaptive forwarding has been investigated. In this paper, by utilizing the feature in NDN that there's one-to-one mapping of Interest-Data and Data is forwarded back along exactly the path Interest is transmitted, in Data packet we introduce an ECN field, which contains ECN information to notify consumer or down-stream routers to adapt their transmission rate. What's more, we also exploit the power of network-wide information by SDN[6] to design a smart forwarding mechanism. By combining ECN and smart forwarding mechanism, we could not only improve bandwidth utilization on single links but also the resource utilization in the whole network. In summary, our main contribution are:

1. We first propose an ECN-based transport mechanism in NDN.
2. We are the first to utilize SDN technique to design smart forwarding mechanism to fully exploit the resource in the whole network.
3. Packet-level simulation shows that joint ECN and smart forwarding in NDN could improve link utilization and total Flow Complete Time (FCT).

The rest of the paper is organized as followed: Sec. II discusses related works. In Sec. III we introduce the NDN's feature which we utilize to design our transport mechanism. In Sec. IV we discuss the design rationale. Sec. V introduces the ECN congestion mechanism and the smart forwarding. In Sec. VI, we demonstrate the effectiveness of our mechanism. Finally Sec. VII concludes the paper.

II. Related work

Congestion control mechanisms in TCP/IP network can be grouped into two categories:

Congestion Window Congestion window mechanism is widely deployed nowadays. Each end in the communication maintains a congestion window, which will additively increase if the RTT is below the estimation value, otherwise multiplicatively decrease, known as AIMD[9]. Congestion window mechanism is easy to deploy since it only requires modification in end hosts. However, congestion window results to 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.

Explicit Congestion Notification (ECN) ECN mechanism notifies network condition by explicit information, such as in XCP[7], RCP[8]. (describe how ECN works.) 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 use 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.

Some 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. Different with TCP/IP's push principle, NDN is a receiver-driven network architecture. In NDN, receivers control transmission rate by adapting the rate of sending Interest. Thus, most of proposed transport mechanisms in NDN are receiver-driven. Receiver adjust its Interest sending rate according to the RTT of the coming back Data, using the AIMD and slow start principle. Contug et al.[15] propose transport mechanism in NDN which mixes receiver-driven mechanism and the basic principles in TCP. It just 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 to each flow, 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. In NDN 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 just by itself, according the bandwidth it can supply to incoming Data. In [16] Wang et al. improve the Interest shaping mechanism as follows. 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 when it receives NACK, which will reduce the link utilization significantly as in TCP. Furthermore, the shaping mechanism, no matter Interest or Data will cause retransmission, which will increase the flow complete time.

To fit the pull principle of NDN, all of these above mechanisms use the receiver to deal with congestion. But they still follow the TCP's slow start and AIMD implicit congestion principle. They also have the low utilization and unfairness problems, similar with TCP[12].

III. Background

NDN adopts transport mechanism [1], [3] which is distinct from TCP/IP network or other ICN proposals. In NDN, consumer issues one Interest to retrieve only one piece of Data. Thus consumer could control the transmission rate by adapting the number of issuing Interests. On receiving an Interest, routers maintain an entry for it in its PIT to ensure Data is forwarded back exactly along the path that Interest is transmitted. Such a "reverse path" feature enables that Data could carry Explicit Congestion Notification (ECN) information back to consumers.

Intermediate routers could also facilitate shaping flows by adaptive forwarding [3]. When forwarding an Interest, NDN router adaptively selects an interface among the multiple available ones according to the network condition. If there is no satisfying interface for forwarding, NDN router responds to downstream router with explicit NACK. Thus, downstream routers react to network congestion in a hop-by-hop way. This is much easier than that in TCP/IP because router-assistant congestion mechanism needs the help of route protocol[10]. However only one hop detection is not enough if the congestion happens on links several hop away from consumers, because the one hop detection cannot sense it immediately. Since SDN[6] could be used to gather network-wide congestion information, a SDN-based solution is promising to overcome the limit of the one-hop solution.

IV. Design rationale

When we to design 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 our first design goal is to 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? Our second design goal is to design a smart forwarding mechanism that minimizes 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. 6 for example, a flow goes through $R5$. The flow has two available 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 a 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

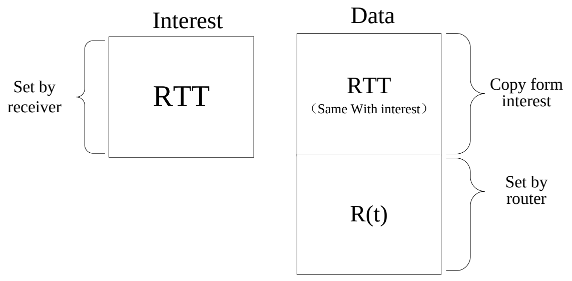

Fig. 1. Explicit congestion notification header for Interest and Data.

$R5$ 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.

V. Design

To achieve the two goals above, our design divided into three parts. First we design the ECN Interest sending rate on the receiver. Second we design a smart forwarding mechanism and at last we analysis the convergence and stability of the system.

A. ECN Interest sending rate

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 each flow's maximum Interest sending rate that it can handle, without causing congestion. We define such maximum rate as $R(t)$.

Fig. 1 shows the Interest and Data ECN header. The ECN header contains the explicit congestion notification information on the path. The Interest carries the RTT of the flow. 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 $R(t)$. The provider copied the Interest's RTT to Data when it send back the Data, and the routers do not change it along the path. The router compares the $R(t)$ that is recorded in the Data with its own $R(t)$. If the router's rate is smaller, it replaces it. By this way, the coming back Data carries the minimum $R(t)$ 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 $R(t)$ it receives last Data.

Suppose we know how many flows going through a link, and we want to share its bandwidth with all the flows, then $R(t)$ can be determined as follows:

$$R(t) = \frac{C}{Flow_{num} * Size_d} \ . \tag{1}$$

In the equation, $Size_d$ is the size of incoming Data, $Flow_{num}$ 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, $R(t)$ should be updated. To make the update process reasonable and let the system enter stable situation, Eq. 1 should be improved as:

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

In the Eq. 2, $R(t)$ is the Interest sending rate that the router assigns to all flows at time t, C is the bandwidth of the link, $S(t)$ is the speed of coming back Data, $Q(t)$ is the packets that occupied 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 $R(t)$.

The definition of $R(t)$ 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 $(C - S(t)) > 0$, there are more available bandwidth to 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 occupied 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 $R(t)$ 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 $R(t)$ with shorter interval T $(0 < T \leq RTT_{avg})$. Then Eq. 2 becomes:

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

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 allocated way 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. So we also use process-fair way to calculate the number of flows that go through this link:

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

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. VI, we will evaluate by simulation that the estimation is correctly. Combining with Eq. 4, Eq. 3 becomes:

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

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, $R(t)$ does not need to measure the flow number directly and the size of Data. That will greatly simplify the router's calculating process.

Forwarding assistant information			
R1	R(t)	Trans delay	Bandwidth
R2	R(t)	Trans delay	Bandwidth
R3	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. 2. Forwarding assistant and route information stored in the controller.

B. *Smart adaptive forwarding*

In this paper we just control the forwarding process, not the route calculating process. The route algorithm in NDN is on active research. But from the algorithms proposed by now, we can see that the NDN route algorithm is different from TCP/IP[11]. Traditional route protocol such as OSPF and RIP just have one single path for each destination. But 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. In this paper, 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 just has relationship with the choose of forwarding interfaces from different paths.

Every router sends its own $R(t)$, 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 $R(t)$ and the transmit delay of every hop. We call the information as forwarding-assistant information. The network's route information can also be stored in the controller. By the forwarding-assistant information and route information, we can calculate the best forwarding strategy. The forwarding-assistant and route information stored in the controller is showed in Fig. 2.

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's Interest sending rate, R_b can be easily calculated by the forwarding-assistance and route information. 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 $R^{'}_{bottleneck}$ is the bottleneck's Interest sending rate on path j.

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

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:

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

In the equation, P_i is the number of path that flow i chooses. $P_i \in \{0,1\}$ means $Flow_i$ can choose at most one path. $max\ min\ R_i$ means $Flow_i$'s Interest sending rate should be maximized. The reason we set $max\ min\ R_i$ is to achieve fairness between different flows. If we do not set $max\ min\ R_i$, some flows may choose a min R 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 routers send 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 each packet. So the overhead introduced by the controller's help is very 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. But the balance between the overhead and the accuracy of updating information can be adaptively controlled.

C. *Stability analysis*

The parameters α and β influence the stability and convergence of the system. As Eq. 2 shows, α influences how the bandwidth is used. If α is large then bandwidth will be occupied quickly. Parameter β influences how quickly that the packets in the queue can be drained. Obviously that large α and β can help the system to use the resource quickly. But large α and β will make the system become unstable, as the network is difficult to convergence.

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 $R(t)$ can also be estimated accurately, then the system will enter 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. 3 shows that under such values the flow number can be accurately estimated. Fig. 4 shows that under these values the estimated flow number change greatly, which means that the system is not stable. From Fig. 3, we

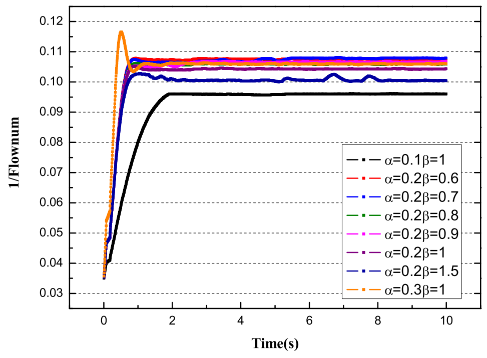

Fig. 3. Under such α and β the flow number can be accurately estimated.

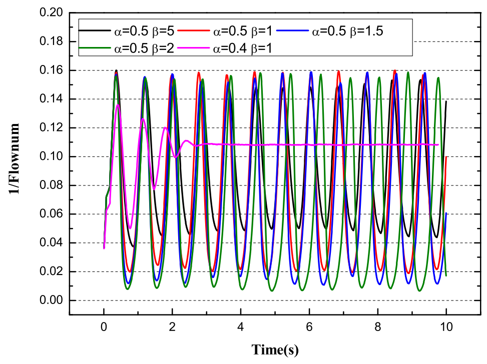

Fig. 4. Under such α and β the flow number can not be accurately estimated.

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. 5 shows that under different situation 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. 4, we find that when α is close to 0.5, the system becomes unstable, and when it gets larger, the system becomes even more unstable. This can be explained as follows. Large α makes the system react too radically to increase $R(t)$ when $(C-S(t)) > 0$ or decrease $R(t)$ when $(C-S(t)) < 0$. Radical reaction will make the system difficult to converge. From Fig. 3, 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 $R(t)$ 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 $R(t)$ 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 suitable value by experimental test. In our simulation, we set $\alpha = 0.2$ and $\beta = 1.5$, according the analysis above.

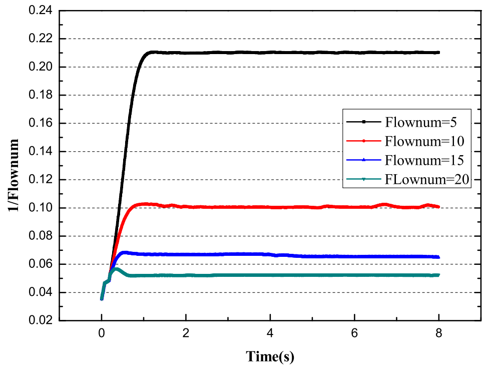

Fig. 5. Stable α and β can make the network stable even when network situation changes.

VI. Simulation

A. Simulation setup

In this section, we study the performance of the smart explicitness congestion notification mechanism we proposed using ndnSim[17], [18].

Fig. 6 shows the network topology we use in the simulation. There are two topologies we use. 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, and each consumer requests Data with a same prefix (means they are a same flow). 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 figures, the bandwidth means the bottleneck's bandwidth.

We set the value of α and β as 0.2 and 1.5 respectively. We compare our smart-ECN with ICP and ICP-shape. 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 Data when the routers sense congestion[16]. 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 are not Data to response the Interest. The consumer should retransmit the same Interest when it receives a NACK.

Our simulation includes three parts. In the first part we evaluate the flow number estimating process, which we propose in Eq. 4. In the second part, we will evaluate the ECN Interest sending rate mechanism using the bottleneck network topology. At last we will estimate the smart forwarding mechanism using the mesh network topology. The simulation result shows the number flows can be estimated accurately. Compared with ICP and ICP-shape, ECN Interest sending rate mechanism performance better in link utilization, packets dropping and flow complete time. Our smart forwarding mechanism has better TFCT compared with adaptive forwarding mechanism.

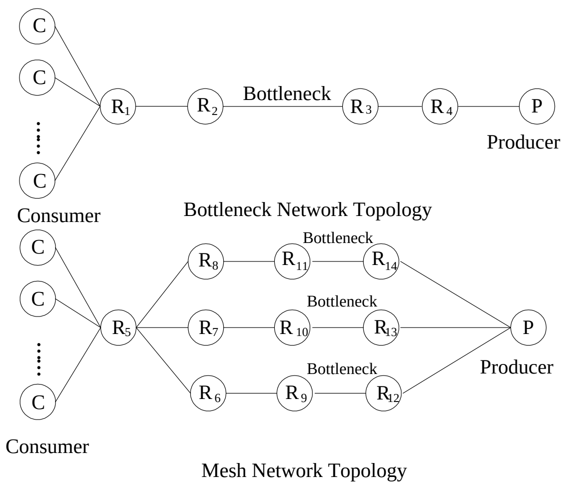

Fig. 6. The two network topologies using in simulation.

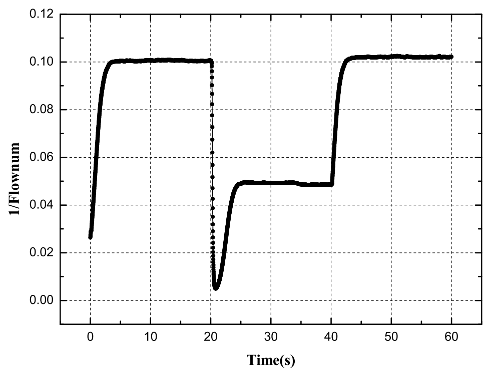

Fig. 7. The accuracy of estimation of flow number.

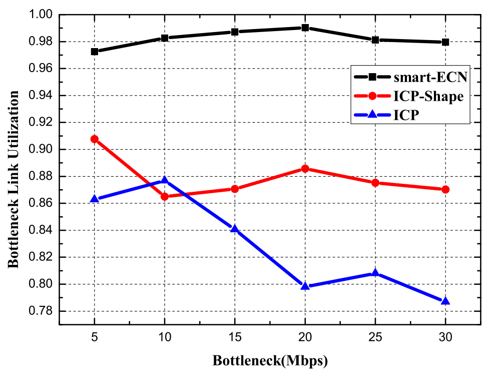

Fig. 8. The bottleneck's link utilization compared with ICP and ICP-shape when change the bottleneck bandwidth.

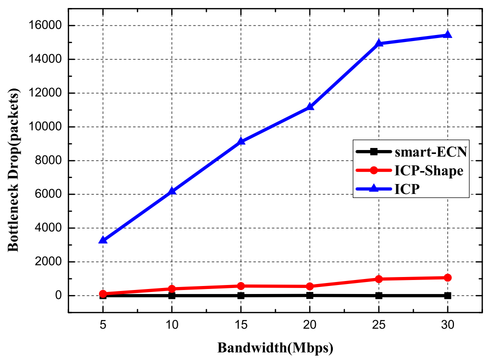

Fig. 9. The number of dropping packets in bottleneck compared with ICP and ICP-shape when change the bottleneck bandwidth.

B. The estimated flow number

By 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, 10flows start. At t=10 more 10 flows start. And at t=20, 10 flows finish, remaining just 10 flows. From Fig. 7 we can see that the flow number can be accurately estimated. Even when the flow number changes, the estimation can also converge.

C. The performance of ECN Interest sending rate mechanism

We use bottleneck topology to estimate the ECN Interest sending rate mechanism. The ECN Interest sending rate is calculated based on the principle that the bandwidth should be ultimately used. The ICP and ICP-shape use the TCP-style window control way. TCP-style window control way uses the timeout-principle to sense the congestion, and once timeout, it cut the interest window to half. TCP-style window control also use the slow-start to send the Interest. These will waste a lot of bandwidth. As Fig. 8 shows, ICP and ICP-shape waste bandwidth because of the slow start. The link utilization of ICP and ICP-shape is between 80%-90%. But even when the bottleneck link's bandwidth changes, the smart-ECN's bandwidth utilization always closes to 100%.

Because the ECN Interest sending rate makes sure that the rate can not overflow the bandwidth, almost no packets(no matter Interest or Data) drop in ECN interest sending rate mechanism, as the Fig. 9 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 dropping Data is very large, as Fig. 9 shows. 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 dropping Data.

As Eq. 5 shows, the packets in the queue should do the utmost to be drained. This principle makes sure that in smart-ECN, the packets in the queue are always close to 0, as Fig. 10 shows. In ICP, to use the bandwidth effectively, the receivers always try to make the Interest window large until the Data fills up the queue and the Data is dropped, so the packets in the queue is large. In ICP-shape , the router can shape Interest when it sense that congestion will happen. So in ICP-shape, the packets in queue is smaller than ICP,as Fig. 10 shows.

Fig. 11 shows the FCT of different flows. Flow complete time in NDN is defined as the time from when receiver send the first Interest until the receiver receives the last Data of the flow. The ECN Interest sending rate mechanism's link utilization is much higher than ICP. ECN Interest sending rate mechanism fairly applies bandwidth resource to all the flows, so even if the short flow can get fair bandwidth resource. No matter short flows or long flows, their FCT are much better than ICP, as Fig. 11 shows.

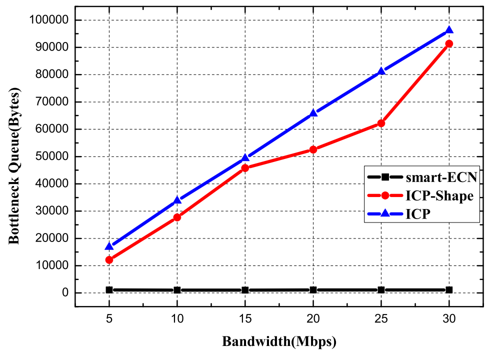

Fig. 10. The volume of queuing packets in bottleneck compared with ICP and ICP-shape when change the bottleneck bandwidth.

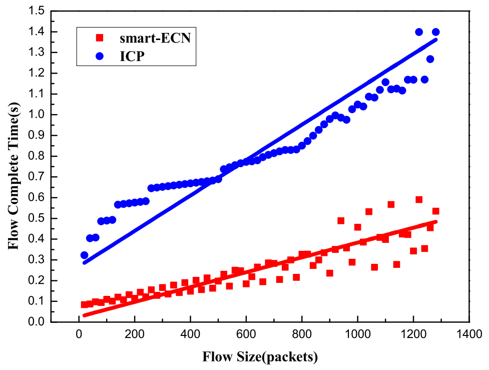

Fig. 11. Smart-ECN's flow complete time compared with ICP.

D. *Flow complete time compared with ICP*

We demonstrate the effectiveness of smart adaptive forwarding mechanism using mesh network topology. The total bottleneck bandwidth means the sum of bottleneck bandwidth on all the available paths. Using the forwarding-assistance and route information, the smart adaptive forwarding mechanism can choose a network-wide best path. 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 will sometimes choose a wrong path whose later link is shared by far more flows. The without-adaptive mechanism chooses the path just by the route information, similar with TCP/IP. The without-adaptive mechanism can not change the forwarding interface according the network condition. As Fig. 12 shows, under different total bandwidth, the smart adaptive forwarding's TFCT is much better than without-adaptive mechanism. As smart adaptive forwarding mechanism can choose a best path on the network-wide view, it's total flow complete time is even better than adaptive mechanism.

VII. Conclusion

In conclusion, we propose a 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 a 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

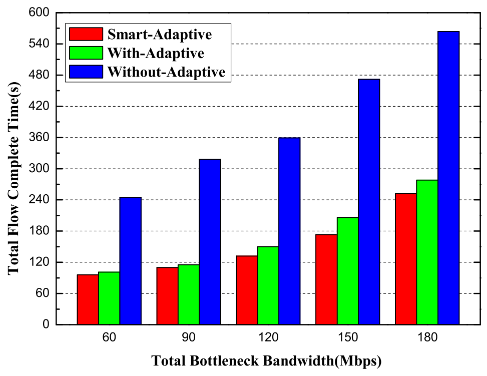

Fig. 12. Total flow complete time compared with other two forwarding mechanisms.

and ECN transport mechanism, the whole network resource can be effective used and total flow complete time can be reduced.

The future work will be focused on the theoretical analysis of system stability. In NDN, as the in-network cache, a Data will have several providers. That will influence the ECN information carried by Data and the RTT. We want to analysis what would happen on such situation and how to make effective resolution for this.

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] 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.

[17] http://ndnsim.net/

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