Adaptive Traffic Scheduling via Network Coding in NDN

Zhifan Zhao[1 2], Xiaobin Tan[1 2], Junxiang Su [1]

1. CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System, University of Science and Technology of China

2. Laboratory for Future Networks, University of Science and Technology of China, Hefei, China, 230027

Email: xbtan@ustc.edu.cn, {zzfan@, jxsu@}mail.ustc.edu.cn

Abstract: Named Data Networking (NDN) architecture promises significant advantages over current Internet architecture by replacing its host-centric design with a content-centric one. In NDN, the mode that one Interest packet gets one Data packet can quite easily lead to Interest flooding and a huge number of the related entries. Moreover, the absence of predefined connections is a challenge for NDN to efficiently manage the successive requests from consumers or the in-network concurrent flows. In this paper, we describes an adaptive traffic scheduling mechanism where multiple Data packets per Interest packet are allowed without violating the operational principle of original NDN protocol and in-network caching can be collaborated with network coding to achieve multi-path transmission. Evaluation results show that the proposed scheme can significantly improve transmission efficiency and reduce system load including the number of Interest packets and lookup operations performed on the related tables in routers.

Key Words: Named Data Networking, Multi-path, Network Coding

1 Introduction

Named Data Networking (NDN) proposed by Zhang L, et al. in [1] which promises significant advantages over current Internet architecture by replacing its host-centric design with a content-centric one. NDN relies on the pull model to acquire content, with the user sending explicit requests (referred to as Interest packets) for the named content (referred to as Data packet). Each request message contains the name of the desired content, and it is forwarded by the name to get the corresponding content which may be cached in more than one in-network node.

The mechanism of one Interest packet getting one Data packet in NDN may be more flexible when dealing with the problem of mobility, path failure, and multi-path forwarding etc. However, this may cause overload problems due to the countless Interest packets. Considering a request for a large-size content object (e.g. a 1GB video) in NDNx prototype [2] (the maximum size of the data packet is 8800 Bytes), 1GB video will be split into approximately 120,000 chunks. Thus, the same number of Interest packets will be generated to get the corresponding Data packets just for a 1GB video. Furthermore, each Interest is forwarded using name-based forwarding, with each content router resolving the content name to an outgoing interface by involving multiple accesses to the off-chip high-latency memory to facilitate the Longest Prefix Match (LPM) operation. In the case of millions of such requests, a noticeable increase in the cost of performing lookups on related tables of NDN may be observed.

Although there are many researches that operate on the per-packet basis to reduce the number of lookups on the forwarding tables or to decrease the size of the related tables (e.g. [10] [12] etc.), flow-based method still have great advantages when it comes to improving traffic efficiency and reducing network overload. For example, in [6], the authors present a method that exploits the correlations in user traffic to create active flow states in content routers to bypass the default forwarding for future requests, just like IP network with the features of creating network pipes between hosts identified by addresses, which can address the recurring lookups on FIB overhead problem. And in [5], they introduce a virtual control plane to optimize the flow transmission in actual plane which can maximize the user-demand. However, these approaches may offer limited gains especially when the *multi-path transmission* occurs, and a comprehensive analysis of the joint problem of flow transmission and multi-path transmission in NDN still needs to be concerned.

In this paper, we propose an adaptive traffic scheduling mechanism for NDN. Our mechanism combines the idea of *Interest packets aggregation* (the continuous requests for the successive chunks of the same content object can be merged into one *flow Interest* packet) with the principle of Random Linear Network Coding (RLNC) [4] [14]. The proposed scheme concerns the multi-path flow control in a connectionless and stateful transport model. The main contributions of this paper are as follows:

- We propose a new Interest packet format (flow Interest) and it can dramatically decrease the number of Interest packets. To some degree, we solve the problem of Interest flooding.
- Our proposal can decrease the number of lookups on the related forwarding tables without causing more system load by the method of flow transmission.
- We can achieve multi-path transmission, which significantly improves service reliability and network flexibility.
- We propose a forwarding strategy on a per-flow basis, and it is implemented on the ndnSIM prototype to present the performance of our scheme.

This paper is organized as follows. The NDN architecture's challenges and our analysis are summarized in Section 2. The details of our mechanism including architecture description, working mechanism, and forwarding protocols are presented in Section 3. Section 4 is the performance evaluation of our proposed mechanism. Finally, we concludes

This work was supported by the State Key Program of the National Natural Science Foundation of China under Grant 61233003, and National "863" Project of China under Grant 2014AA06A503.

our paper in Section 5.

2 PROBLEM ANALYSIS

NDN transport model brings new challenges that TCP/IP based network, even in its multi-path versions, can not address, and it motivates the definition of novel transport control and forwarding mechanisms. However, in TCP/IP based network, in regard of its connection-based character, the multi-path solutions (e.g. [13]) perform load-balancing mechanisms over static paths which are precomputed by a routing protocol. Instead, in NDN, it's a great challenge to achieve more efficient flow transmission and multi-path transmission due to the absence of predefined connections and the lack of knowledge of available source(s).

Previous efforts have always been devoted to exploiting the transfer capacity of multiple paths on the per-packet basis. Actually, if we want to better exploit the transfer capacity of network, the processing capacity of in-network nodes and the mechanism of multi-path transmission should be taken into account (Cisco Visual Networking Index [11] suggests that the sum of all forms of video traffic is expected to reach 80-90% of the global consumer traffic by 2019). In this paper, our adaptive traffic scheduling mechanism in NDN focus on three aspects.

Multi-path: As for the multi-path transmission in NDN, our goal is to make users can continuously request for the desired content object from multi-sources at the same time. But the Interest forwarding is controlled by in-network nodes which take on-the-fly decisions on forwarding interfaces for each arriving Interest, which means there is no predefined connections in NDN transport.

However, the principle of Network Coding can fully utilize the flexibility brought by the absence of predefined flow transmitting which have been analyzed in [9]. Considering the scenario in Fig. 1, the consumer wants to retrieve $\{X_1, ..., X_{10}\}$. A, B and C all have the copy of $\{X_1, ..., X_{10}\}$, when they received the flow request of $\{X_1, ..., X_{10}\}$, they return the different linear combination of the $\{X_1, ..., X_{10}\}$ ($Y_i (i = 1, 2, ..., 10)$). Obviously, the consumer can get $\{X_1, ..., X_{10}\}$ if he/she receives 10 linear independent encoded Data messages.

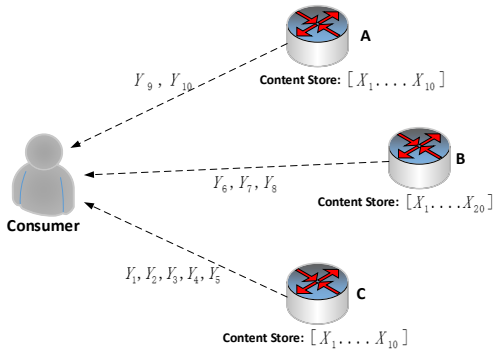

Fig. 1: An example for multi-path transmission via Network Coding

Flow transmission: We use the *flow Interest* to pack the original Interest and it has especially better performance in terms of aggregation to reduce the repetitive transmissions for multiple requests for the same data. For example, different flow Interests which carry the flow request for the different successive chunks of the same content object can be aggregated into one flow Interest. Furthermore, different flows can enjoy the different services according to applications' requirement and network condition. Indeed, the operation on the flow Interest greatly simplifies the problem of flow control. But the flexibility brought by the absence of predefined connections is also a challenge for the flow transmission.

Security: As a novel idea to build secure applications, a fundamental security primitive is embedded in the "this waist" of NDN: the name in each NDN message is bound to content with a signature [1]. This basic feature provides data integrity and origin authentication. Signature information is generated by the name and the content of message. The content of message is encoded in source nodes, and the destination node verifies the content, name and signature information after decoding the received message.

3 PROPOSED MECHANISM

In this section, we describe the design details of our mechanism.

3.1 Flow NDN Model

The NDN architecture design stipulates hierarchically structured names and bases forwarding on LPM. To obtain data, the requester should send out an Interest packet, which carries the name of the desired data (*e.g. /supplied-name/version/chunk-number*). Each router has a *Content Store (CS)* to make a cache of some content and maintains a *Pending Interest Table (PIT)* to keep track of the currently unsatisfied Interest packets. Arriving Interest packets are forwarded using name-based forwarding by looking up the data name in the locally maintained *Forwarding Information Base (FIB)* when there are no pending Interest packets with the same name in PIT and no corresponding entry in CS.

In order to better support multi-path flow transmission, we define the format of flow Interest by including some additional fields to primary Interest packet head. Moreover, the scheduling of the flow Interest forwarding should be based on the new design of the FIB/PIT/CS.

3.1.1 Flow Name

A summary representation of the hierarchical name structure is depicted in Fig. 2. The chunk name can uniquely clarify the Data packet and the flow name distinguishes the object with the supplied name and version.

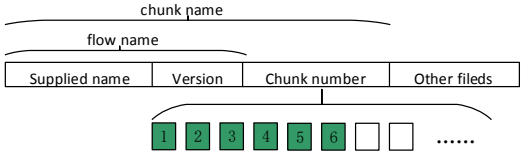

Fig. 2: *Flow name*

NDN allows name aggregation, e.g., */ustc/video* could correspond to an autonomous system originating the video

and each chunk of the data split will be added into that name. In this way, we can regard the "*flow name*" as the name of combination of total chunks. For example, */ustc/video/demo.mpg* is corresponding to the first chunk to the end chunk of the *demo.mpg* (the prefix: */ustc/video*).

3.1.2 Flow Interest and Data Packet

Flow Interest mechanism takes advantage of merging many original Interest packets into a packing Interest to speed up the interest processing[1]. We include four additional fields in the primary Interest packet header: *Flow Type* (*f*-Type) field, *Start Chunk of flow Interest* (SC), *End Chunk of flow Interest* (EC) field and *Request Size of flow Interest* (RS).

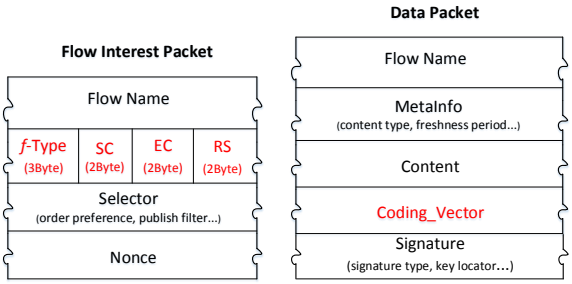

Fig. 3: Flow Interest packet and Data packet

In flow Interest packet (Fig. 3), the 3-Byte field of *f-Type* have strong flexibility and expansibility which can meet the different service requirements. For example, in our implementation, the basic functions are representing the type of this Interest (1-bit), determining whether this flow Interest can be split (1-bit), and the limiting condition of the flow Interest split (e.g. link condition and the maximum of subflows). The *SC* and *EC* fields represent the start and the end chunk of the request field. The *RS* field represent the size of the request filed (i.e. the number of chunks).

As for the Data packet, we add a h Bytes length *Coding_vector* field. Coding_vector indicates *global encoding vector*. When h chunks are considered to encode together, h Byte *global encoding vector* is necessary.

3.1.3 Flow PIT (*f*-PIT)

There are some extra work needs to be done for the original PIT to meet the need of flow Interests. A flow Interest carries more than one request for the content chunks, which means the PIT should meanwhile generate the same number of PIT entries to record the different requests. However, corresponding Data packets do not arrive at the same time and some of the later reached Data packets will be dropped.

In our scheme, we organize the PIT entries in a "flow" way and the structure is {*Prefix, (Start Chunk_1, End Chunk_1, Face IDs, Request Size_1, Timer_1), ..., (Start Chunk_n, End Chunk_n, Face IDs, Request Size_n, Timer_n)*}.

[1]In this paper, we generally think the request in the flow Interest is the successive chunks of the same content object, and the request size of the flow Interest is under 100.

When a flow Interest arrives (Fig. 4), if there exits a matched f-PIT entry (the request field and size of the same content are exactly the same as the corresponding f-PIT entry, and the incoming face is already in the pending interface set), then this flow request will be merged into the corresponding entry by simply resetting the lifetime of this f-PIT entry. Otherwise the flow Interest should be forwarded to next-hop and a new f-PIT entry will be generated. When a Data packet arrives, update the *Timer* and the value of the corresponding f-PIT entry's *Request Size* should minus one.

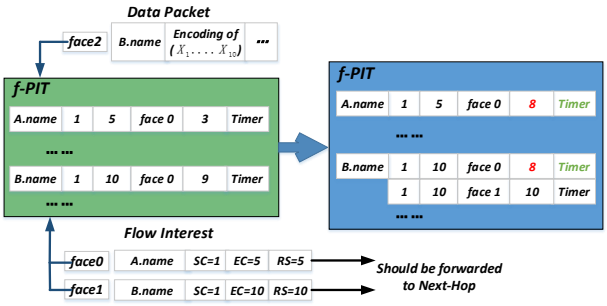

Fig. 4: The structure and working process of f-PIT

3.1.4 Flow FIB (*f*-FIB)

In-network nodes perform dynamic flow-by-flow packet request scheduling by taking on-the-fly decisions on the selection of forwarding interfaces. It is necessary to update the interfaces' weight parameters due to the everchanging network conditions. To make the forwarding of the flow Interest more efficient, we add a new field to the original FIB entry that contains state information of all output interfaces for each flow. The general structure for f-FIB is {*Prefix, (Face_1, Value), ..., (Face_n, Value)*}, which is kept on all output interfaces and is real-time updating. When the matched f-FIB entries have multiple available output interfaces for a flow Interest, the router can acquire the price of each interface by checking the Value field to split the flow Interest or select the minimum cost interface.

3.2 Working Process

Next, we will present the working process of our adaptive traffic scheduling mechanism in flow NDN model, and we also propose a Network Coding based forwarding strategy on a per-flow basis.

3.2.1 State Information Update

The f-PIT contains the incoming interfaces of Interests that have been forwarded but are still waiting for matching Data. Each outgoing interface records the time when the Interest is forwarded via this interface, so that when Data packet returns, the RTT time can be computed.

Let us consider the download of a given content s, and in-network node i has available out interfaces $J = \{1, 2, ..., n\}$. Let c_j be the capacity of link l for each interface and $c = [c_1, c_2, ..., c_j]$. Let $x^j(t)$ be the sum of transmission rates on the interface j at time t.

First, we give the current price of each interface of node i and define $p^i_{s,j}(t)$ as the interface prices of node i at time t. The interface price algorithm can be modified to

$$p^i_{s,j}(t+1) = [p^i_{s,j}(t) + \gamma(x^j(t) - c_j)]^+,$$

then we can deduce the aggregate price on each interface from observed round trip time (RTT) by using a step size β/c_j:

$$p^i_{s,j}(t+1) = [p^i_{s,j}(t) + \frac{\beta}{c_j}(x^j(t) - c_j)]^+, \qquad (1)$$

The number of pending Interests reflects (i) the path length and response time associated to a given content; (ii) the quality of residual path towards repository/closet copy. Actually, the PIT queuing length $b_j(t)$ evolves according to

$$b_j(t+1) = [b_j(t) + (x^j(t) - c_l)]^+,$$

multiplying both sides by β/c_j, we have

$$\frac{\beta}{c_j}b_j(t+1) = [\frac{\beta}{c_j}b_j(t) + \frac{\beta}{c_j}(x^j(t) - c_j)]^+. \qquad (2)$$

Comparing (1) with (2), we see that link price at time t is proportional to the current delay $q_j(t) := b_j(t)/c_j$ at interface j:

$$p^i_{s,j}(t+1) = \beta\frac{b_j(t)}{c_j} = \beta q_j(t).$$

Given end-to-end propagation delay $\delta^i_{s,j}(t)$, hence the aggregate price can be deduced from $RTT^i_{s,j}(t)$:

$$p^i_{s,j}(t+1) = \beta(RTT^i_{s,j}(t) - \delta^i_{s,j}(t)).$$

And with $P_j \equiv RTT^i_{s,j}(t) - \delta^i_{s,j}(t)$, we define $p^i_{s,j}(t+1) = \beta P_j$.

The measure of $\delta^i_{s,j}(t)$ may be very difficult, but it can be estimated by the minimum $RTT^i_{s,j}(t)$ observed from the history samples.

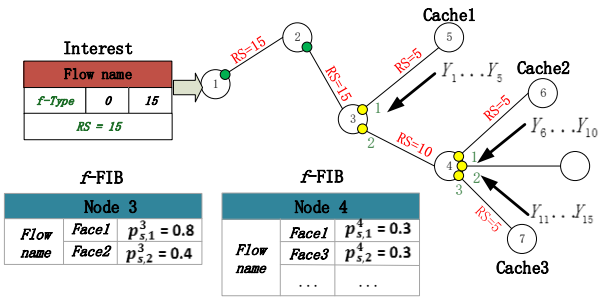

Fig. 5: There are three caches in Node5, Node6 and Node7. When they receive corresponding flow Interest, they return RS encoded Data packets, $Y_i (i = 1, 2, ..., 15)$ is different linearly independent encoding of $X_i (i = 1, 2, ..., 15)$.

3.2.2 Network Coding Based Flow Interest Splitting

In our mechanism, one flow Interest can get RS encoded data packets from more than one available sources through splitting the flow Interest into some sub-flow Interests with different RS_i values ($RS_1 + ... + RS_i = RS$).

Considering a splitting scheme proposed in [8], all the Interfaces in a FIB entry are ranked in order to help choose which interface to use. We can define the interface ranking by setting two boundary values (p^1_i, p^2_i) for the price of each face:

- Green ($0 < p^i_{s,j} < p^1_i$): the interface is in good condition.
- Yellow ($p^1_i < p^i_{s,j} < p^2_i$): the interface is overloaded.
- Red ($p^2_i < p^i_{s,j}$): the interface does not work.

Actually, a flow Interest wishes to meet the f-FIB with the Green interfaces, then the in-network node only selects the interface with the minimum $p^i_{s,j}$ to forward the flow Interest. However, the Green ranking interface does not always exist in in-network nodes and this method is difficult to reach load balancing. Overload occurs when the demand exceeds link capacity leading therefore to unacceptable performance for some flows. In this case, single interface has difficulties to perfectly satisfy the need of the flow Interests. Even if we still choose the interface with the minimum $p^i_{s,j}$ to forward the flow Interests as the method proposed in [7] on the per-packet basis, network may not reach the expected performance.

To achieve certain traffic engineering objectives, the network management system needs to balance traffic between multiple paths. For example, sending 40% of traffic on one path and 60% on another could lead to less congestion in the network and it can be formulated as a simple convex optimization problem (assuming that the flow Interest carries the request for N chunks of content s and there are λ available interfaces in node i):

$$\begin{cases} \min\limits_{\{n_1,n_2,...,n_\lambda\}} \max\{n_1 p^i_{s,1}, n_2 p^i_{s,2}, ..., n_\lambda p^i_{s,\lambda}\} \\ n_1 + n_2 + ... + n_\lambda = N \end{cases} \qquad (3)$$

Obviously, when $n_1 p^i_{s,1} = n_2 p^i_{s,2} = ... = n_\lambda p^i_{s,\lambda}$, we can easily get the optimal solution values $\{n^*_1, n^*_2, ..., n^*_\lambda\}$. In this way, we can split the flow Interest into several sub-flow Interests with different RS values (Fig. 5).

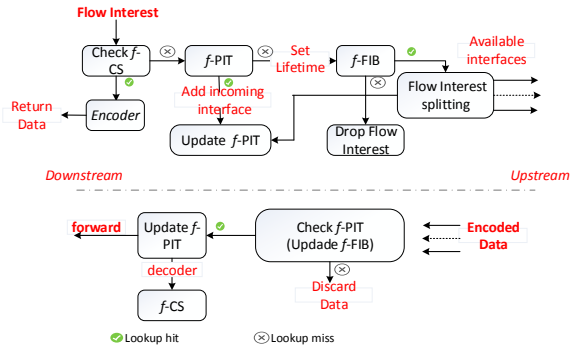

Fig. 6: Forwarding Process at a Node

According to the flow Interest splitting principle, the flow Interest forwarding details are shown in Fig. 6. When a node have the cache of the request of reached flow Interest, this node will return RS encoded corresponding Data packets.

Therefore, each node should have a encoder which can independently and randomly selects a set of coefficients referred to as the *global encoding vector* and uses them to form linear combinations of the request chunks of the corresponding flow Interest.

If the flow Interest should be forwarded to next-hop(s), we can choose one of the available interface with the best condition or choose more. The price p of output interface is real-time updated on each node. Upon the arrival of each Data packet, the price of a particular output interface is updated. And at the f-PIT timeout, the particular interface will be punished by multiplying the maximum RTT_{max} (from history samples) with 2. At each flow Interest's reception, we check the f-FIB to get the available interfaces, and split the flow request according to the price of each interface (In our implementation, we only split the flow Interest into two available interfaces with the minimum costs).

4 PERFORMANCE EVALUATION

In this section, we focus on the evaluating performance of transmission efficiency improvement brought by our adaptive traffic scheduling mechanism including throughput, overload decreasing and the better reactions towards the congestion situation in NDN.

Table 1: Simulation Parameters

Simulation parameters	Value
Producer Capacity (per packet)	0.0001s
Interest Packet Size	35B
Data Packet Size	8000B
Delay Time	10ms
Queue(Maxpackets on the link)	20
CPU	2.60GHz
Memory	4G

4.1 Experimental setting

We implement the protocols described above in NS-3 based Named Data Networking (NDN) simulator (ndnSIM) [3] by developing additional modules and changing some structures of the ndnSIM prototype. The detailed settings are showed in TABLE 1.

4.2 Throughput Improvement

We employ a simple network scenario with a good network condition (see Fig. 7), in which users are connected to two repositories via two paths and the capacity of Data packet generation can be adjusted in repo1 and repo2. We observe the throughput in different uplink bandwidth compared with the original NDN [8] (the unsatisfied Interests or flow Interests will be resent soon). Evaluation results are showed in Fig. 10. We can see that the inefficient use of uplink bandwidth negatively affects the downlink throughput in original NDN. However, in our mechanism, the downlink throughput starts getting affected under 100Kbps of uplink bandwidth.

4.3 Overload Considerations

In this scenario, we focus on the overloaded traffic situation including packet processing, tables maintaining, and the

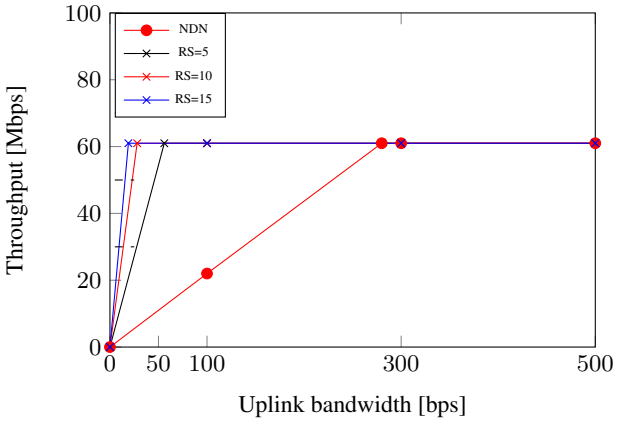

Fig. 10: Our mechanism vs. original NDN adptive traffic scheduling control [8] in download speed.

number of Interests. Notice that the total time (τ_{total}) includes: the forwarding time of Interest packet (τ_{ip}) and Data Packet (τ_{dp}), the lookups time on the CS/PIT/FIB (τ_l), the preset response time of producers ($\tau_r = 10ms$). That is,

$$\tau_{total} = \tau_{ip} + \tau_{dp} + \tau_l + \tau_r$$

The setting rate field in flow Interest is equal to the generation rate of Interest packet in NDN, which means the τ_{dp} is equal to the τ_{f-dp}. We have set both response time of producers $\tau_r = 10ms$. Actually the time of dealing with the flow Interest is more than original Interest packet, but the main impact factor is the number of Interest packets. To completely consider all factors, we can approximate the overhead on packet processing by measuring the total processing time.

Scenario 2 topology is shown in Fig. 8. there are four consumers to request the same content object from repo1 and repo2. The four consumers start to request the content respectively at 0s, 5s, 10s, 15s. Especially, the network nodes in our mechanism is fixed to select the best two interfaces for the flow Interest, and the forwarding strategy in NDN is also the same for the Interest flows. Each node(#3,#4,#5) records the time of dealing with the Interest packet. TABLE 2 demonstrates the total number of Interests sent to acquire the whole content in network and the overall time for the nodes (#3,#4,#5) to deal with the passing Interests. Obviously the improvement of reducing the overhead on packets processing is remarkable.

Table 2: Interest Processing Time

Node ID	Proposed Mechanism / Original NDN		
	Number of Interests	Per-packet processing time [s]	Total time [s]
3	239 / 1265	0.0124 / 0.092	3.038 / 11.638
4	318 / 1465	0.0124 / 0.092	3.943 / 13.478
5	237 / 1270	0.0124 / 0.092	2.939 / 11.684

4.4 Congestion Considerations

In a forwarding architecture, the most critical performance measure is the *forwarding capacity*, and whether or not the proposed solution can support the targeted rates. In this scenario, we consider a typical network scenario to evaluate the

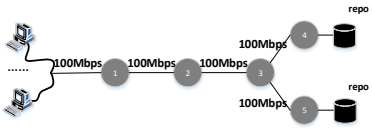

Fig. 7: Scenario 1 topology

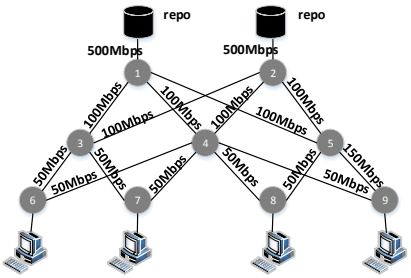

Fig. 8: Scenario 2 topology

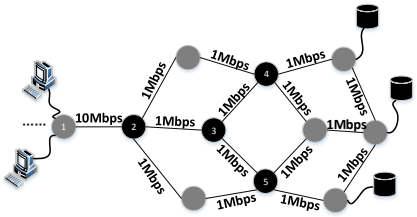

Fig. 9: Scenario 3 topology

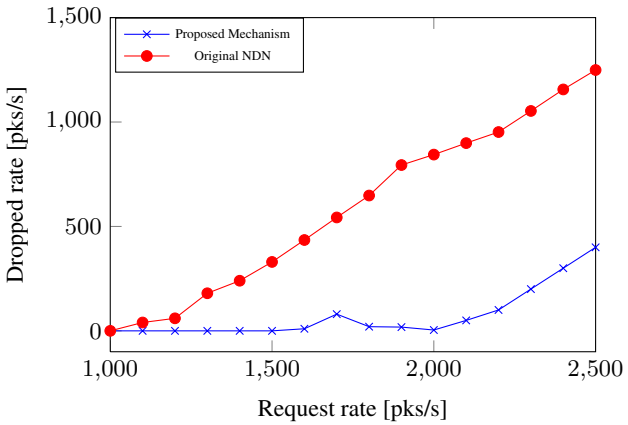

Fig. 11: The number of dropped Data packet per second (pks/s) in different request rate.

performance of Multi-path transmitting when the network is congested, which has also been discussed in Sec. 3-C. The topology is shown in Fig. 9, the bandwidth of access link is 10Mbps, other links 1Mbps. In this case, the network can easily become congested when increasing the user's request rate. The associated FIB entries are set to make all the output interfaces available at nodes (#2,#3,#4,#5). To represent the performance of congestion control and forwarding protocols adapted to different network conditions, we record the dropped Data packets with increasing request rate (Fig. 11). The rise of Dropped Data packet in NDN occurs immediately when each of the link can not satisfy the arriving Interests. In our mechanism, the node can select more than one available interface to meet the sudden growth of demand due to the advantage of the flow Interest.

5 CONCLUSION

The design of our mechanism reflects our understanding of the strengths and limitations of the current NDN architecture. In this paper, we proposed an adaptive traffic scheduling mechanism to improve traffic efficiency and present a forwarding strategy on a per-flow basis. We show the significant improvements of performance achieved in forwarding capacity and energy utilization when in the mode of flow transmission and multi-path transmission.

However, there still exits many works to do in this architecture. For instance, the computation complexity bring by Network Coding and the rate control may be a challenge in our mechanism. The next step of our work is to introduce the AIMD (Additive Increase Multiplicative Decrease) controller into our Architecture.

References

[1] Zhang L, Estrin D, Burke J, et al. Named data networking (ndn) project. *Relatrio Tcnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[2] NDNx. http://named-data.net.

[3] NS-3-based NDN simulator. http://ndnsim.net.

[4] Ho T, Mdard M, Koetter R, et al. A random linear network coding approach to multicast[J]. *Information Theory, IEEE Transactions on*, 2006, 52(10): 4413-4430.
alg1

[5] Edmund Yeh, Tracey Ho, Ying Cui, and Michael Burd. VIP: Joint traffic engineering and caching in Named Data Networks. In *Computing, Networking and Communications (ICN)*, 2015.

[6] Azgin A, Ravindran R, Wang G. Flash-forward CCN: flow-driven forwarding architecture for content centric networks. Proceedings of the 1st international conference on Information-centric networking. ACM, 2014: 189-190.

[7] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and Sen Wang. Optimal Multi-path Congestion Control and Request Forwarding in Information-Centric Networks. In *proc. of IEEE ICNP*, 2013.

[8] Yi C, Afanasyev A, Wang L, et al. Adaptive forwarding in named data networking. *ACM SIGCOMM computer communication review*, 42(3): 62-67, 2012.

[9] Liu W X, Yu S Z, Tan G, et al. Information-centric networking with built-in network coding to achieve multisource transmission at network-layer[J]. *Computer Networks,* 2015.

[10] Haowei Yuan and P. Crowley. Scalable Pending Interest Table Design: From Principles to Practice. IEEE INFOCOM, 2014.

[11] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2014-2019. tech. rep., 2014.

[12] Fukushima, Makoto, Atsushi Tagami, and Toru Hasegawa. Efficiently looking up non-aggregatable name prefixes by reducing prefix seeking. *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013.

[13] Wischik, Damon, et al. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. NSDI. Vol. 11. 2011.

[14] Chou P A, Wu Y, Jain K. Practical network coding. Proceedings of the 41st Annual Allerton Conference on Communication, control, and Computing, 2003.