Front.Comput.Sci.
DOI

RESEARCH ARTICLE

Interest-Suppression-based NDN Live Video Broadcasting over Wireless LAN

Menghan LI [1], Dan PEI [1], Xiaoping ZHANG (✉)[1], Beichuan ZHANG [2], Ke XU [1]

1 Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
2 Department of Computer Science, University of Arizona, Tucson, AZ, 85721-0001, USA

Abstract Named Data Networking (NDN) is a new Internet architecture that replaces today's focus on *where* – addresses and hosts with *what* – the content that users and applications care about. One of NDN's prominent advantages is scalable and efficient content distribution due to its native support of caching and multicast in the network. However, at the last hop to wireless users, often the WiFi link, current NDN implementation still treats the communication as multiple unicast sessions, which will cause duplicate packets and waste of bandwidth when multiple users request for the same popular content. WiFi's built-in broadcast mechanism can alleviate this problem, but it suffers from packet loss since there is no MAC-layer acknowledgement as in unicast. In this paper, we develop a new NDN-based cross-layer approach called NLB for efficient and scalable live video streaming over wireless LAN. The core ideas are: using WiFi's broadcast channel to deliver content from the access point to the users, a leader-based mechanism to suppress duplicate requests from users, and receiver-driven rate control and loss recovery. The design is implemented and evaluated in a physical testbed comprised of one software AP and 20 Raspberry Pi-based WiFi clients. While NDN with multiple unicast sessions or plain broadcast can support no more than 10 concurrent viewers of a 1Mbps streaming video, NDN plus NLB supports all 20 viewers, and can likely support many more when present.

Keywords NDN, Video Broadcast, WLAN

Received month dd, yyyy; accepted month dd, yyyy

E-mail: zhxp@tsinghua.edu.cn

1 Introduction

A major trend of Internet traffic in the last few years is the fast increase of video content. For example, in North America, Netflix as a video streaming company has become the largest source of Internet traffic, consuming 29.7% of peak downstream traffic[1)]. Another major trend is the proliferation of mobile devices, such as smartphones, tablets and laptops. Subsequently, more and more contents, especially video contents, are consumed on wireless mobile devices, and because of the cost advantage of WiFi over cellular, most video contents are consumed over WiFi Wireless LAN (*WLAN*) links as the last hop to the users. According to a recent forecast by Cisco[2)], 61% Internet access will be over WiFi and mobile devices by 2018.

The need for efficient and scalable video distribution has not only driven the upgrade and expansion of operational networks, but also the development of new network designs and architectures. Named Data Networking (NDN) [1] is a new Internet architecture that emphasizes the content itself rather than its container (*e.g.*, host) or channel (*e.g.*, connection). By including the content name in each packet, NDN enables native multicast and caching support in the network, which will greatly improve large-scale content distribution, including video distribution.

[1)] Internet Phenomena Report, 2011. https://www.sandvine.com/trends/global-internet-phenomena/

[2)] Cisco visual networking index: forecast and methodology, 2013-2018. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

While video streaming has already received considerable attention in the NDN research community in [2–9], little research work has been done on scalable and efficient video streaming over WLAN. In broadcast media such as WLAN, content delivery will benefit significantly if content is broadcasted to multiple clients when they are requesting for the same popular content. However, in IP networks, multiple clients in the same WLAN are served via multiple unicast sessions, resulting in duplicate data packets and waste of bandwidth. The current NDN implementation, running as overlay on top of IP, suffers from the same problem of not being able to take advantage of the broadcast nature of underlying media. This inefficiency at the last hop may negate all NDN's benefits in wired networks for content distribution.

In this paper, we develop an efficient and scalable solution for NDN live video streaming over WLAN. Live video streaming has more stringent performance requirements than Video-on-Demand (VoD), because it cannot prefetch the content from servers to caches which are much closer to the clients as in VoD. The time interval for buffering live stream cannot be too large and the transmission rate must satisfy the stream bit rate. In fact, live video streaming has to deal with a specific class of problems to ensure timely delivery of an ordered stream of video chunks.

Furthermore, live video streaming over WiFi has important use scenarios. For example, campus-wide live speech with thousands to tens of thousands of viewers in a company or a university [10], country-wide live events such as SuperBowl, NBA finals, World Cup games, Olympic Opening ceremonies, president's speech etc. As the popularity of mobile devices keeps increasing, people will more and more to use mobile devices to watch these live events online over WiFi.

For the above reasons, we believe that supporting live video broadcasting over WiFi has important real-world impacts, and that leveraging and extending NDN protocols is one promising direction to approach this problem.

Our basic idea is to use WLAN's built-in broadcast to deliver the same piece of Data once to multiple clients at the same time, and make it efficient and robust. The proposed approach, called NDN Live video Broadcasting (NLB), is a combination of NDN layer and application layer mechanisms (both above the MAC layer) and does not require any modification to the 802.11 MAC layer. Therefore, NLB is different from and complement previous work that provides 802.11 broadcasting/multicasting mechanism via packet re-transmission, ordering, network coding, or PHY rate adapting, *etc*., such as Medusa [10], Dircast [11], Flexcast [12], and [13–15].

NLB needs to address two major challenges without the help of MAC layer. First, unlike its unicast, 802.11's broadcast doesn't have ACK, which means that more packet loss will be exposed to upper layer and often it will lead to performance degradation. Second, without any coordination, all clients will send the same Interest to the access point, which will waste bandwidth and cause contention with Data transmission.

NLB addresses the above challenges using a couple of intuitive approaches. First, an NDN AP runs as a standard NDN router except that it forwards streaming Data to its WLAN via broadcasting. This way, the Data is only sent once (when there is no re-transmission) to multiple clients. Second, to address the Interest contention problem, the AP chooses the first client whose Interest is received by the AP as the *Leader*. Then the AP will broadcast a periodical Interest ACK (a special Data Packet) to tell all clients about the Leader and the number of active clients upon every P_{ack} received Interest packets from the Leader. All other clients except the Leader are the *Followers*. All clients including the Leader and the Followers use an Interest pipeline mechanism to control Interest sending rate and ensure the reliable transmission. The Leader forwards Interest packets with best-effort manner in NDN layer. On the other hand, the followers use a Delayed Interest Sending mechanism in NDN layer to suppress duplicate Interest/Data packets to save bandwidth and reduce contention.

To the best of our knowledge, NLB is the first NDN-based cross-layer design to support live video broadcasting over WiFi without any modification to the MAC layer. To evaluate NLB, we have implemented it in NDN and run it on a software AP. The client's code runs on embedded Linux on Raspberry Pi[3] systems. We have deployed NLB on a physical testbed consisted of a software AP and 20 clients. The results have shown that NLB performs significantly better than NDN video streaming over plain broadcasting and over multiple unicast sessions. While the latter can support no more than 10 concurrent clients for 1Mbps video, NLB supports all 20 clients and likely many more when they are present. The analysis has shown that NLB's mechanisms are effective in reducing duplicate packets, contention, buffering ratio.

The remainder of the paper is organized as follows.

[3] Raspberry Pi. https://www.raspberrypi.org/

Section 2 introduces NDN background and our problem settings. NLB's design and implementation are described in details in Section 3. In Section 4, we evaluate NLB using real implementation and testbed experiments. Section 5 briefly reviews the related work. Finally, a conclusion and the future work are presented in Section 6.

2 Background and Problem Settings

2.1 NDN

NDN architecture [16] has two basic packet types: Interest packet and Data packet. Each content chunk is identified by the meaningful and hierarchically structured name. In NDN, all communication is driven by the consumer who sends the Interest packet firstly. NDN routers forward the Interest packet according to its name to the content provider, and maintain a pending interest table (PIT) which tracks the entrance interface (called *face* in the rest of the paper following [16]'s convention) of the Interest. Therefore, a returning Data packet can trace the reverse path back to the consumer and be also cached in the Content Store (CS) at the NDN routers along the path. When an NDN router receives multiple Interest packets destined to the same content, it only creates one PIT entry which records all the entrance faces of the Interest packets, and forwards only one Interest to the upstream.

2.2 NDNVideo

Video streaming services could benefit significantly from NDN. The Interest aggregation mechanism and Data caching in NDN can greatly save the bandwidth of backbone routers and offload the pressure of video servers. In fact, NDN project team has released an NDNVideo software [9] and conducted several live demos already.

The NDNVideo publisher cuts the video stream into equal-length segments and put them into NDN repository with the name prefix like /repo/stream/video/segments/=number. The NDNVideo player uses a pipeline to send successive Interest packets to request video segments. If a specific Interest or Data is lost during transmission, the player will retransmit this Interest according to the Interest retransmission timer for at most R_{max} times.

2.3 Problem Settings: receiver-driven broadcasting over WLAN

We consider a typical WLAN in which an access point (AP) provides N wireless clients (clients) the access to the Internet-based service. The video server and AP are connected through high-speed wired network. In our application scenario, the wired network is not the bottleneck. We assume that N clients are simultaneously accessing the same live video stream service from the server. In order to save bandwidth and eliminate the redundant packets in wireless medium, it is intuitive and desired that the AP broadcasts (as opposed to unicast) the stream data to multiple clients.

In traditional IP broadcasting over WLAN such as in [10], the server first pushes the data to the AP, then the AP broadcasts the data to clients in WLAN. Unlike IP broadcasting, NDN broadcasting over WLAN is a receiver-driven broadcasting. The client must first send the Interest to the AP, then the AP adds the Interest to the PIT and forwards it to the content provider. When the Data tracks the reverse path of the Interest back to the AP, if the PIT entry records multiple entrance faces from WLAN, then the AP can broadcast the Data to WLAN.

However, the efficiency of this receiver-driven broadcasting may be impacted by the Interest transmission. If the Interest is broadcasted over WLAN, WiFi broadcast's lack of MAC layer ACK introduces a lot of Interest loss in upper layer. NDN's receiver-driven Interest retransmission mechanism (with long timeout value which references RTT between the consumer and the provider) cannot recover the lost Interest in time, restricting the streaming performance potentially. If multiple clients send Interest packets to the AP via unicast simultaneously, they will send repeated Interest packets before receiving corresponding Data. Besides, multiple clients sending Interest packets via unicast could result in wireless channel contention with the Data broadcast, and aggravate the Data loss in WLAN. Therefore, in order to mitigate wireless channel contention, we also need a mechanism to suppress redundant Interest packets. Based on the above analysis, the design problem of NLB can be stated as:

For a live video broadcasting with a given bit rate and a given number of clients in a WLAN, minimize the amount of Interest and Data packets transmitted over wireless medium, under the premise of smoothed playback.

3 NLB Design and Implementation

3.1 Leader-based receiver-driven broadcasting over WLAN

Because NDN broadcasting is receiver-driven, there must be at least one Interest to trigger a Data broadcasting. The design goal of NLB is to reduce the overhead of Interest sending as much as possible and avoid the competition between multiple clients. Our design intuition to suppress repeated Interest packets is shown as follows. Ideally, for each Data, *just* one Interest is sent over WLAN by one of the clients, and then all the clients can receive the Data broadcasted by the AP without sending their own Interest to WLAN. Instead of selecting a potentially *different* client to send the Interest for *each* Data, NLB chooses to use a very straightforward approach: we make one client (called the *Leader*) stay ahead of other clients to send the Interest and other clients (called the *Followers*) wait for Data packets broadcasted by the AP. Once a Leader is chosen, we do not risk frequently selecting a new Leader because the time granularity of client joining/leaving the live video broadcast is much larger than the Interest/Data packet rate. In the rest of the section, we will describe the NLB design crossing application layer and NDN layer.

3.2 NLB protocol layers

In order to maximize the deployment opportunities, NLB should be able to compatible with existing IP network and 802.11 protocols. In this way, it can be deployed onto most commodity wireless APs, and can use UDP tunnels to connect to the rest of NDN networks on the Internet to run NDN live video streaming, while the same APs can still run IP applications. As such, we seek solutions that do not require modifying 802.11 MAC protocols.

We build a NDN-based cross-layer solution over UDP tunnels. In particular, NLB can be run directly on top of 802.11 MAC layer, besides UDP is not absolutely necessary. However, because our purpose is to verify the effectiveness of NLB, so in order to reduce the implementation difficulty, we use UDP sockets to wrap NDN packets. Like 802.11 broadcast, UDP broadcast does not support the reliable transmission. Therefore, running NDN over UDP tunnels does not affect the functionality of NLB design and just introduces the overhead of UDP headers.

In application layer, the video server runs an original NDNVideo [9] publisher. Because of focusing on the transmission efficiency of video streaming and the testbed limitation, on the client side, we modified NDNVideo player code to develop a new NDN Simulative Video Player (NSVP) to replace a real NDNVideo player. Since we have demonstrated that NLB system can be deployed on real NDNVideo player in our last conference paper [17], NSVP is not a deployment constraint of NLB. NSVP has three modules: Streamer, Tracer and Simulative Decoder (SDecoder). The Streamer first gets the metadata of the video stream, and pass it to the SDecoder. The SDecoder can simulate the data acquisition process of a real decoder according to the codec parameters, such as codec type, frame rate, GoP size, and etc.. Then the Streamer requests the Data to satisfy the SDecoder with its best effort. The Streamer must reassemble NDN Data packets into encoded video frames and pass them to the Tracer. The Tracer is a queue buffer between the Streamer and the SDecoder. It can adjust the buffer size to smooth network delay. When the Streamer can't satisfy the SDecoder, the Tracer will suspend the transfer of the encoded frames to the SDecoder and buffer for a while. Besides buffering, the Tracer can also choose to skip over the overdue frame. And the Tracer will record the buffering duration, the buffering frequency and the overdue frames for statistical results. At last, the SDecoder can record all the received encoded frames for offline analysis, such as PSNR calculation.

We use ccnx-0.8.1[4)] as the base implementation of NDN protocol. NLB involves three different *ccnd* (NDN forwarding daemon implemented in ccnx) implementations: 1) a video server running an unchanged *ccnd*, 2) an AP running a modified *ccnd* which can broadcast the live streaming data and the periodic Interest ACK, and 3) multiple clients running a modified *ccnd* which can identify the Interest ACK and then determine to run as the Leader or Follower. The Delayed Interest Table is NLB's major newly added module to implement the Interest suppression mechanism.

3.3 Controlling Interest production in application layer

Live stream bootstrap. When bootstrapping a live stream in NDN, the Streamer must locate the first newly valid Data segment of the video stream. This mechanism has already been supported by NDNVideo prototype. In order to handle the high loss rate of WiFi broadcast and smooth the resulting severe delay jitter of arriving Data, after receiving the first valid Data segment of the live video stream, the Tracer in

[4)] CCNx. http://www.ccnx.org/

NSVP player needs to buffer the successive Data segments for a time period before actually feeding the SDecoder. This time period is called *playout delay* and denoted by T_{pd}.

Interest pipeline. In order to continuously acquire the video stream Data efficiently, the Streamer needs to maintain a receiver-based, timeout-driven Interest pipeline to perform flow control and reliable transmission. Unlike the Interest transport protocols proposed by [18–21], we do not use the Additive Increase Multiplicative Decrease (AIMD) mechanism. For NDN live video streaming, the Streamer must send Interest packets to match the Data producing rate of video stream. However, AIMD's target is to respond to the network congestion and sufficiently use the unoccupied network bandwidth, which does not match the requirements of live video streaming. Besides, for the AIMD mechanism, a timer expiration means that the window size will be cut in half, which will lead to a sharp decrease of the Interest sending rate. Furthermore, in the environment of WiFi's broadcast, the lack of loss recovery mechanism in Layer 2 will aggravate the jitter of Interest sending rate and lead to a poor utilization of wireless bandwidth.

Due to NDN's Interest and Data one-on-one flow control mechanism, we use a maximum number of pending Interests (a pending Interest is the Interest which has not been consumed by the requested Data) to limit the Interest sending rate instead of a continuous sliding window for pending Interests. This mechanism, which in principal is similar to TCP's SACK [22], is not only suitable for NDN, but also works well for WiFi's broadcast with high loss rate. The pseudocode in Algorithm 1 details the Interest pipeline design. Besides, in the following sections we describe the operation of how to determine the Interest sending rate and the Interest retransmission timer.

Interest sending rate. For NDN live video streaming, the Streamer should not fetch the Data too quickly, nor request the Data that is not yet produced by the publisher. Otherwise, if it does not receive a Data by the timeout, the Streamer will retransmit the corresponding Interest. Thus, consistently being ahead of the Data production will cause a lot of repeated Interests.

To solve this problem, the Streamer needs to get the stream bit rate and then determine the proper Interest sending rate. According to the Interest pipeline design described in Algorithm 1, the Interest sending rate is determined by the maximum number of pending Interests (N_{pi}), which represent the average number of in-flight Interests in the network. When we get the average RTT (T_r) of an Interest to get back a Data and the payload size of a

Algorithm 1 Interest pipeline

```
1:  function INTEREST_PIPELINE
2:      N_pi = get_initial_npi()
3:      pos = check_latest_data()
4:      S_1 = pos
5:      requested = pos − 1
6:      counter = 0
7:      request_more_data()
8:      while running = True do
9:          // run callback for 5 seconds;
10:         run_cb(5)
11:         S_2 = check_latest_data()
12:         N_pi = (S_2 − S_1) ∗ T_r/5
13:     end while
14: end function
15:
16: function REQUEST_MORE_DATA
17:     left = 2 if counter == 0 else 1
18:     counter = (counter + 1)%2
19:     while (len(p_ints) < N_pi) and (left > 0) do
20:         requested+ = 1
21:         left− = 1
22:         p_ints.add(requested)
23:         send_interest(requested)
24:     end while
25: end function
26:
27: function RECEIVE_DATA_CB(number, data)
28:     calculate_rtt(number)
29:     push_out_data(number, data)
30:     p_ints.remove(number)
31:     request_more_data()
32: end function
33:
34: function INTEREST_EXPIRED_CB(number)
35:     if retries(number) == R_max then
36:         push_out_data(number, None)
37:         p_ints.remove(number)
38:         request_more_data()
39:     else
40:         retries(number) + +
41:         send_interest(number)
42:     end if
43: end function
```

Data (L), we can derive the Data transmission rate (DR) in the unit time.

$$DR = N_{pi} * L/T_r \tag{1}$$

we can get:

$$N_{pi} = DR * T_r/L \tag{2}$$

The metadata of the video stream provides an average bit rate value VR set by the video codec. However, after the Streamer starts, the *actual* stream bit rate will vary with the dynamics of video frames. In the situation of IP's push-based transmission, the video server can timely adjust the sending rate according to the varied stream bit rate. For NDN's receiver-driven transmission, the Streamer can first obtain the dynamics of stream bit rate from the video server and then adjust the Interest sending rate.

More specifically, when the Streamer starts, it first requests 20 Data segments to calculate an average RTT and use the video bit rate VR contained in the metadata to derive the following initial N_{pi} from the equation 2

$$N_{pi} = VR * T_r/L \tag{3}$$

After the Streamer starts, we use a straightforward method to recurrently (every T_s seconds) request the latest Data (we use S to denote the segment number of the latest Data) to determine the upper bound of Interest to be sent. Besides, we can calculate the stream bit rate from the number of segments produced every T_s seconds, then determine the maximum number of pending Interests. For example, assume that the two consecutive S values obtained by the Streamer in a T_s-second interval are S_1 and S_2. Then we can derive N_{pi}, the number of in-flight Interests, using the equation 2, and plug in $DR = (S_2 - S_1) * L/T_s$, and we get:

$$N_{pi} = ((S_2 - S_1) * L/T_s) * T_r/L = (S_2 - S_1) * T_r/T_s \tag{4}$$

Interest retransmission timer. Because WiFi broadcast cannot guarantee the reliable Data delivery, clients have to deal with the problem of how to re-request the Data that are lost in wireless medium. The Streamer adjusts Interest retransmission timer based on previous RTT values [9]. It uses a low pass filter

$$SRTT = (1 - \alpha) * SRTT + \alpha * RTT \tag{5}$$

to smooth RTT values, and uses the TCP/IP formula [23]:

$$\begin{aligned} RTTVAR = (1 - \beta) * RTTVAR + \\ \beta * |SRTT - RTT| \end{aligned} \tag{6}$$

to smooth variance. The Interest retransmission timer is calculated using

$$T_i = SRTT + K * RTTVAR \tag{7}$$

If the Data has not been returned back within T_i seconds after the Interest is sent out, the Streamer will retransmit the Interest. Here, $\alpha = 1/8$ and $\beta = 1/4$ are the recommended values from RFC2988 [23]. And $K = 3$ is referenced from the NDNVideo technical report [9]. These parameter values work well for the single client requesting the stream. In multiple clients situation, if the Streamer continues fetching Data packets from the local CS, T_i will fall to a very small value which is much lower than the RTT between the consumer and the producer. And the Streamer will keep retransmitting Interests according to this very small T_i. This will result in an explosive growth of repeated Interest packets in a short period. Therefore, we use a *min_timer* to limit the minimum Interest retransmission timer value in equation 8.

$$T_i = max(min\_timer, SRTT + K * RTTVAR) \tag{8}$$

In Section 4.3, we will tune the parameter values for calculating the Interest retransmission timer and evaluate the impact of these values on the streaming performance.

3.4 Leader-based Interest suppression in NDN layer

Choosing the Leader. When a client requests the live stream for the first time, it first sends an Interest containing the name of the stream. The AP establishes a new face for this client and records the stream name. If any other clients request the same stream, the AP will update the number of clients N which request this specific stream. The AP chooses the first client requesting the live stream as the Leader and replies an Interest ACK for every P_{ack} received Interest packets from the Leader. The Interest ACK is a special *Data* packet which contains its corresponding Interest and the current number of clients N. When a client receives the Interest ACK, it will check its nonce value. If the nonce value is matched with any Interest in its PIT, a client will know that itself was chosen as the Leader by the AP. Otherwise, a client will run as a Follower. The Leader *ccnd* forwards every new Interest packet received from the Streamer with the best-effort manner in NDN layer. The Follower *ccnd* delays all the Interest packets, which are not matched in the CS and will wait for the matched Data to "consume" the Interest. Besides determining the running mode from the Interest ACK, the Followers can also learn

the Interest sending progress of the Leader (SN_{max}: the maximum segment number of the Interest sent by the Leader) from the Interest ACK.

Clients leaving. For requesting the stream Data segments, a client sends an Interest to get the stream production progress periodically, and this special Interest will not be delayed by the client. The AP determines whether a client is still alive according to this Interest. If an AP has not received this Interest from a client in a period, it will declare this client has left. If the leaving client is a Follower, the AP will just update the client number N in the next Interest ACK. If the leaving client is the Leader, the AP will select the sender of the Interest received right after the ex-Leader leaves as the new Leader. The new Leader will keep running in the Follower mode until the SN of received Interest is larger than SN_{max}, which is the maximum segment number of Interest already sent by ex-Leader (to avoid resending these Interests), before switching to the Leader mode.

Interest suppression. For the Leader, besides forwarding all the newly received Interest packets to the AP, *ccnd* maintains a Delayed Interest Table (DIT) to record them. When the Leader *ccnd* receives a Data, if the Data consumes a matched Interest in the DIT, the *ccnd* records the segment number of this Interest (SN_i) and removes any Interest of which SN is smaller than SN_i from the DIT. The Leader *ccnd* delays the retransmitted Interest which is still in the DIT.

When the Follower *ccnd* receives an Interest from the Streamer, if there is no matched Data in the CS, it adds the Interest into the DIT. There are three events which may trigger the Follower *ccnd* to update the DIT: When **1)** an Interest hits a matched Data in the CS or **2)** a received Data consumes a matched Interest in the DIT, the *ccnd* records the segment number of Interest (SN_i). Then it updates the DIT and sends the Interest of which SN is less than SN_i randomly with the probability $1/(N-R)$. N is the number of clients which are requesting the live stream. R is the times for which the *ccnd* received the same Interest retransmitted from the Streamer. If the *ccnd* successfully sends the Interest out, it will remove the Interest entry from the DIT. **3)** When the *ccnd* receives a retransmitted Interest from the Streamer, if $R > N/2$, the delayed Interest will be sent out unconditionally.

NLB delays the Interest sending in NDN layer because of the following reasons. Due to the interference in wireless channel, wireless transmission may suffer from burst delay jitter. The Interest retransmission timer estimation in application layer cannot timely respond to the delay variation. Therefore, the Interest retransmission mechanism in application layer cannot accurately determine if the packet is indeed lost and may result in repeated Interest and Data packets. When the Interest SN_i is consumed by the Data SN_i, if any Interest's SN is smaller than SN_i has not consumed yet, the Interest itself or its corresponding Data could be determined to be lost. For the Followers, in order to refrain multiple followers from requesting the same lost Data and to ensure the lost Data can be recovered in time, the *ccnd* sends the Interest randomly with a probability which is determined by the client number and the Interest retransmission times.

4 Performance Evaluation

In this section, we first define the performance metrics for the live video streaming, and describe the compared approaches and experiment setups. Using these metrics, we then evaluate the impact of different retransmission timer calculation on NLB's performance. At last, we compare the scalability of the NLB with other approaches as the client number increases.

4.1 Performance Metrics

According to [24], the quality of Internet live video is better measured by buffering rate, buffering ratio, etc., because they are more related to the viewing experience than traditional PSNR metric. Therefore, in this paper we primarily focus on these two metrics, defined for each individual client:

- **Buffering Rate** is defined as *the number of buffering events occur during the stream session* divided by *the duration of the session*.
- **Buffering Ratio** is defined as *the time spent on buffering* divided by *the sum of buffering and playing time*.

As discussed in the previous section, NLB's scalability is primarily achieved by effectively reducing repeated Interest and Data packets in wireless medium. As such, in order to better explain the scalability results, we further define two metrics to evaluate the packet redundancy in a *WLAN* running a live video streaming:

- **Interest Redundancy** is defined as the ratio of *the total number of Interest packets sent by the ccnd of all clients* to *the average number of Data packets delivered to a Streamer*.

- **Data Redundancy** is defined as the ratio of *the total number of Data packets sent by the AP* to *the average number of Data packets delivered to a Streamer*.

In the ideal situation where there are no packet loss or repeated packets, the Interest/Data redundancy is 100%. A 100% redundancy means that each Data packet delivered to all Streamers is requested by just one Interest packet, with no redundant Interest/Data packets.

4.2 Compared Approaches and Experiment Setups

We primarily compare the performance of NLB with two other NDN approaches:

- **UCast**: The clients run NSVP. The clients and the AP running the original *ccnd* send the Interest and Data over UDP unicast.
- **BCast**: The clients running NSVP and the original *ccnd* send the Interest over UDP unicast. The AP running a slightly modified *ccnd* just broadcasts each stream Data segment to the clients.

In our last conference paper [17], we have implemented the AP *ccnd* on OpenWrt[5], a software router system on embedded linux that can run on all major commodity APs. However, the scalability of UCast is limited by OpenWrt router's low-end CPU. Here, we use an Ubuntu PC which runs hostapd to serve as a software AP. The AP which has an 802.11n wireless adaptor runs at 2.4GHz channel with 6Mbps broadcast bandwidth and 100Mbps unicast bandwidth. Here, we do not quantify the background traffic interference. But in order to obtain fair experiment results between different approaches and settings, all the experiments are performed in our lab during the time period from 0 am to 6 am, when there are little background traffic and interference.

The NDNVideo publisher is installed on an Ubuntu server, which keeps generating the live video segments and stores them into NDN data repository. We install NSVP and the client *ccnd* on 20 Raspberry Pies. Each Raspberry Pi, which has 4 cores 900MHz CPU and 1G RAM. Each Pi is equipped with an independent USB 2.4GHz TP-LINK wireless adaptor.

We experiment with a benchmark Mobile calendar video clip[6]. The video is encoded at the rate of 1Mbps using FFmpeg[7] tool with H.264 codec. We have repeated each experiment of 120 seconds duration for 5 runs. In our experiments, for all compared approaches, the playout delay T_{pd} is set to 5 seconds and this duration is included in the time spent on buffering. The time interval T_s of requesting the latest Data is also 5 seconds. Here, we do not evaluate the impacts of changing these settings. And as shown in the following results, 5 seconds is an adequate value and fair for different approaches and settings. For NLB approach, the period P_{ack} of the Interest ACK replied by the AP is every 30 received Interest packets from the Leader. This value is sufficient for the Followers to track Leader's process and the overhead introduced by the Interest ACK does not significantly affect the performance of NLB. We intend to evaluate the benefits to NLB of adaptively modifying these parameters as our future work.

4.3 Parameter Settings

In this subsection we explore how to set a few key parameters in NLB. Table 1 summarizes the parameter spaces explored and the chosen parameters.

Because WiFi broadcast has a high loss rate and meanwhile it is the last hop of whole transmission process, in order to perform timely loss recovery, the Interest retransmission timer should be as close as possible to the latest RTT value. On the other hand, if the retransmission timer is too aggressively short, it will lead to unnecessary retransmissions. Especially with the increasing number of clients, excessive retransmissions can exacerbate the sending contention between clients and eventually reduce the effective bandwidth. Therefore, the parameter setting of retransmission timer calculation should be a trade off between retransmission timeliness and contention avoidance.

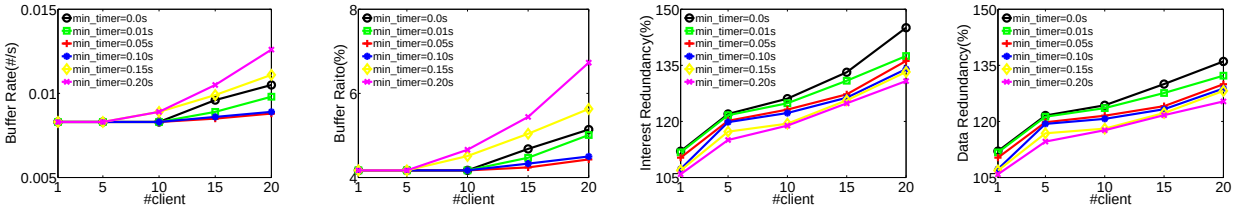
Our goal is to compare the scalability in the client number of different parameter settings in the premise for ensuring reliable transmission. Therefore, we set an infinite R_{max} to guarantee no Data loss in application layer.

We first evaluate the change of *min_timer*. When *min_timer* is zero, it means that the retransmission timer is entirely determined by the smooth RTT and its variation. As a result, when an Interest is served instantly at the local CS, its latest RTT is close to zero, which can confuse the estimation of actual RTT between clients and video server/AP. When *min_timer* is not zero, a smaller *min_timer* means more active retransmission. In contrast, a bigger *min_timer* means more conservative retransmission. The results are shown in the Fig. 1, where the 6 lines in each subfigure respectively shows the results for *min_timer* value

[5] OpenWrt. https://openwrt.org/
[6] MPEG-2 hd test patterns. http://www.w6rz.net/
[7] FFmpeg - digital audio converter. http://www.ffmpeg.org/

Table 1 Parameter settings for retransmission timer calculation

Parameter	Meaning	Parameter spaces	chosen values
R_{max}	The maximum retransmission times of an Interest	25	25
min_timer	The minimum value of retransmission timer	$0s$,$0.01s$,$0.05s$,$0.1s$,$0.15s$,$0.2s$	$0.05s$
(α,β)	The coefficients to calculate SRTT and RTTVAR	(1/8, 1/4),(1/16, 1/8),(1/32, 1/16)	(1/16, 1/8)
K	The coefficient of RTTVAR to calculate retransmission timer	3,4	3



(a) Average Buffering Rate. (b) Average Buffering Ratio. (c) Interest Redundancy. (d) Data Redundancy.

Fig. 1 Scalability comparison with different *min_timer*

from 0 to 0.20 seconds. We can observe that the packet redundancy decrease with the increase of *min_timer*. However, the lowest packet redundancy does not means the best streaming performance, which includes scalability, buffering ratio and buffering rate. Just like our above analysis, a compromised *min_timer* can lead to better performance. We can observe that when $min\_timer = 0.05s, 0.10s$, NLB has better scalability and steadier performance.

According to the results represented by the Fig. 1, we choose $min\_timer = 0.05s$ to further evaluate the change of (α,β,K). In the Fig. 2, each of the 6 lines in each subfigure respectively represents the results with different (α,β,K) combination from $(1/8,1/4,3)$ to $(1/32,1/16,4)$. Theoretically, smaller (α,β,K) combination means smoother retransmission timer variation, and bigger (α,β,K) combination means more timely response to RTT variation. The choice of this combination is also a trade off. However, as can be observed in Fig. 2, the performance differences between different (α,β,K) combinations are not significant. The reason can be found in the Fig 3. Because the maximum average RTT is nearly 0.03s, it means that most RTT variations caused by (α,β,K) are covered by $min\_timer = 0.05s$.

In conclusion, among these parameter settings, we choose the combination of $min\_timer = 0.05s$ and $(\alpha,\beta,K) = (1/16,1/8,3)$ to further evaluate the scalability of 3 different approaches. The chosen parameters are summarized in Table 1.

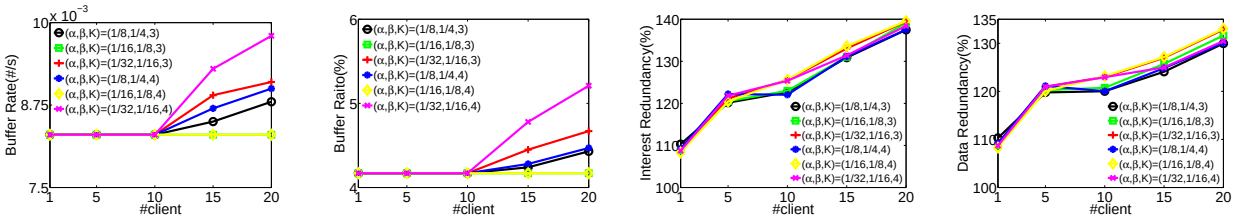
4.4 Performance Comparison

We vary the number of clients and measure the above-defined metrics over multiple clients to evaluate the scalability of 3 different approaches.

Scalability in the client number. As illustrated in the Fig. 4, for NLB approach within the period of 120 seconds, the streaming performance is very steady. And there is no buffering event occurred except the initial buffering time. These two values show that NLB can support 20 clients to play back 1Mbps video smoothly. It is truly achieved that 20 clients have the same user experience as just 1 client does. The curve of NLB remains at the same level from 1 client to 20 clients. This result not only means that NLB does not completely occupy wireless bandwidth, but also shows very high stability of NLB. The performance of BCast degrades sharply when the client number is larger than 5. And UCast can only support up to 10 clients to play back the video smoothly. BCast performs worse than UCast because of its high loss rate and the low bandwidth of WiFi broadcast. For the scalability in the client number, NLB outperforms at least 2 times than UCast and 4 times than BCast.

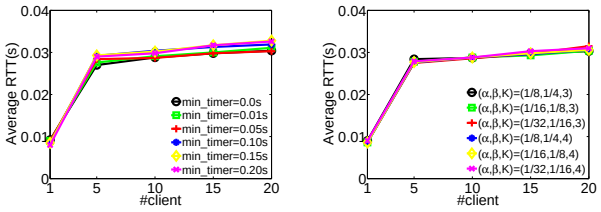
When the client number is bigger than 10, the tendency of BCast's buffering rate curve in the Fig. 4(a) represents an abnormal phenomenon. Because when the buffering event occurs, the number of buffered Data segments before switching to the playing mode is constant. With the increase of the client number, the buffering time for single buffering event of BCast increases sharply. The total buffering ratio even exceeds 50%. In the specified running time of 120 seconds, the total number of the buffering events instead decreases.

Packet redundancy. The results in the Fig. 4 can be



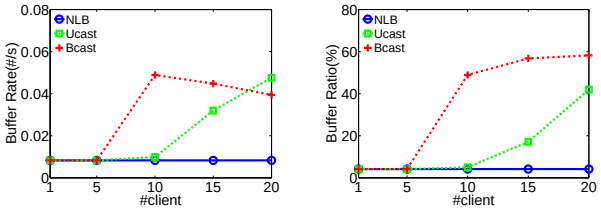
(a) Average Buffering Rate. (b) Average Buffering Ratio. (c) Interest Redundancy. (d) Data Redundancy.

Fig. 2 Scalability comparison with different (α, β, K)



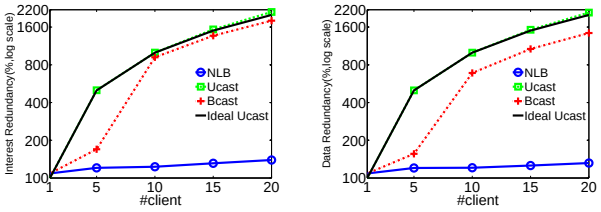
(a) Average RTT of different *min_timer*. (b) Average RTT of different (α, β, K).

Fig. 3 Average RTT of different parameter settings



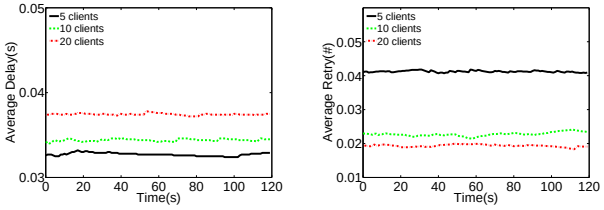
(a) Average Buffering Rate. (b) Average Buffering Ratio.

Fig. 4 Scalability comparison of different approaches



(a) Interest Redundancy. (b) Data Redundancy.

Fig. 5 Packet redundancy over wireless channel



(a) Average Delay. (b) Average Retry.

Fig. 6 NLB stability in time domain

further explained by the packet redundancy results in the Fig. 5(a) and Fig. 5(b), which illustrate the ratio of redundant Interest and Data packets transmitted over wireless medium as the client number varies. With the increase of the client number, the Interest and Data redundancy in NLB increase very slowly. Even for 20 clients in NLB, the ratio of the packet redundancy is still less than 140%, which means that the additional packet redundancy introduced by per client is less than 2%. This result suggests that NLB has the potential to support more clients simultaneously playing 1Mbps video. The black solid curve represents an ideal UCast situation where there is no Interest or Data loss over wireless medium. Because the packet loss rate in wireless medium increases with the number of clients, the two curves of UCast gradually deviate upward from the black solid curves. As our analysis in Section 2.3 shows, if multiple clients send the Interest to AP via unicasting simultaneously, they are likely to send repeated Interest packets before receiving the corresponding Data. Therefore, the Interest redundancy curve of BCast is very close to the black solid curve. Due to Data broadcasting, the Data redundancy of BCast is significantly better than UCast, although it is still much worse than NLB.

Time-domain stability. To further show the stability of NLB in time domain, we present the average delay and retry of one client to fetch Data packets within one-second duration in the Fig. 6. Since the average retry almost maintains stable, the average delay also basically remains steady. These two figures mean that NLB has a very stable performance in packet delivering. In conformity with the results in the Fig 5, the average delay gradually increases with the client number due to the increase of the amount of transmitted packets. Instead, the average retry decreases with the increase of the client number. This is because, as described in Section 3.4, each single Follower sends the retransmitted Interest randomly with a probability which decreases with the increase of the client number.

In summary, above results clearly show that *NLB's*

performance is highly scalable as the client number increases, by effectively suppressing repeated and useless Interest packets and saving bandwidth to transmit useful Data packets.

5 Related Work

Video streaming has already received considerable attention in the CCN/NDN research community. But none of the following works specially consider live video broadcasting over WLAN. The authors in [2] propose an architecture for mapping HTTP-based streaming applications in CCN. A cooperative caching strategy is introduced to enable time-shifted TV services in [3]. In order to reduce the transmission overhead of Interest packets, a time-based Interest protocol is proposed in [4], in which a consumer sends a specific Interest packet requesting a group of content chunks during a specific time interval. A real-time streaming TV service is provided and evaluated in [5]. In [6] the authors present a CCN P2P video streaming application running on mobile devices to offload the cellular radio interface. A CCN adaptive video streaming application named AMVS-NDN is proposed in [7], which enables a mobile device either to use its own 3G/4G connection or to connect via WiFi to another mobile device for exploiting its possibly better 3G/4G link. In [8] the authors present a P2P CCN application for live streaming of videos encoded at multiple bitrates, which allow peers to play a video at a coding rate close to the sum of their cellular capacities to improve video quality. In [9] the authors set up a test-bed for video streaming over CCN, named NDNVideo.

There have been considerable amount of related works on providing WiFi broadcasting/multicasting mechanism via packet re-transmission, ordering, network coding, or PHY rate adapting, *etc*., such as Medusa [10], Dircast [11], Flexcast [12], and [13–15]. NLB is different from these approaches, because NLB is a cross-layer design above the MAC layer and does not try to modify MAC layer behavior. In fact, it can work with and complement these MAC layer-based approaches.

6 Conclusions

NLB achieves scalable and efficient delivery of live streaming video by taking advantage of NDN's data-centric and receiver-driven communication model as well as the broadcast nature of the underlying WLAN medium. It is a practical solution in that it can run on commodity access points and mobile devices without any changes to existing 802.11 MAC layer. Combined with NDN's advantages of caching and multicast in the wired networks, NLB completes the picture of one end-to-end solution for scalable video content distribution over future Internet.

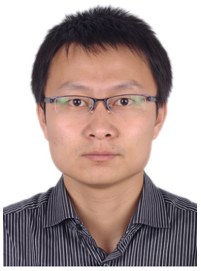
As advocated in [1], building and experimenting with applications is the way to push forward future Internet architecture research. We believe that NLB provides not only a practical solution to live video broadcasting, but also experience and insights into NDN's protocol design. Its cross-layer approach and Interest suppression mechanism may be applicable to NDN over broadcast medium in general. Thus our future work includes experimenting with NLB in larger, more realistic settings, and generalizing the solution to both NDN over broadcast medium and to video distribution over other future Internet architectures.

Acknowledgements This work was partly supported by the State Key Program of National Science of China under grant 61233007, the National Natural Science Foundation of China (NSFC) under grant 61472214 & 61472210, the National High Technology Development Program of China (863 program) under grant 2013AA013302, the National Key Basic Research Program of China (973 program) under grant 2013CB329105, the Tsinghua National Laboratory for Information Science and Technology key projects, the Global Talent Recruitment (Youth) Program, and the Cross-disciplinary Collaborative Teams Program for Science & Technology & Innovation of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

References

1. Zhang L, Afanasyev A, Burke J, Jacobson V, cllaffy k , Crowley P, Papadopoulos C, Wang L, Zhang B. Named Data Networking. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66–73
2. Xu H, Chen Z, Chen R, Cao J. Live streaming with content centric networking. In: Proceedings of IEEE International Conference on Networking and Distributed Computing. 2012, 1–5
3. Li Z, Simon G. Time-shifted TV in content centric networks: the case for cooperative in-network caching. In: Proceedings of IEEE International Conference on Communications. 2011, 1–6
4. Park J, Kim J, Jang M, Lee B. Time-based interest protocol for real-time content streaming in content-centric networking (CCN). In: Proceedings of IEEE International Conference on Consumer Electronics. 2013, 512–513
5. Ciancaglini V, Piro G, Loti R, Grieco L A, Liquori L. CCN-TV: a data-centric approach to real-time video services. In: Proceedings of International Conference on Advanced Information Networking and Applications Workshops. 2012, 982–989

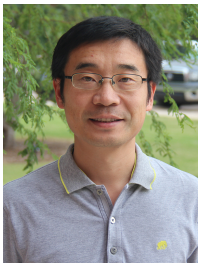
6. Detti A, Pomposini M, Blefari-Melazzi N, Salsano S, Bragagnini A. Offloading cellular networks with Information-Centric Networking: the case of video streaming. In: Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. 2012, 1–3
7. Han B, Choi N, Kwon T, Choi Y. AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking. In: Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2013, 375–380
8. Detti A, Pomposini M, Blefari-Melazzi N. Peer-to-peer live adaptive video streaming for information centric cellular networks. In: Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. 2013, 3583–3588
9. Kulinski D, Burke J. NDNVideo: Random-access Live and Pre-recorded Streaming using NDN. Technical Report TR-0007, NDN, 2012
10. Sen S, Madabhushi N, Banerjee S. Scalable WiFi Media Delivery through Adaptive Broadcasts. In: Proceedings of USENIX Symposium on Networked Systems Design and Implementation. 2010, 191–204
11. Chandra R, Karanth S, Moscibroda T, Navda V, Padhye J, Ramjee R, Ravindranath L. Dircast: A practical and efficient Wi-Fi multicast system. In: Proceedings of IEEE International Conference on Network Protocols. 2009, 161–170
12. Aditya S, Katti S. Flexcast: graceful wireless video streaming. In: Proceedings of ACM International Conference on Mobile Computing and Networking. 2011, 277–288
13. Jakubczak S, Katabi D. A cross-layer design for scalable mobile video. In: Proceedings of ACM International Conference on Mobile Computing and Networking. 2011, 289–300
14. Park Y, Jo C, Yun S, Kim H. Multi-room IPTV delivery through Pseudo-Broadcast over IEEE 802.11 links. In: Proceedings of IEEE Vehicular Technology Conference. 2010, 1–5
15. Bejerano Y, Ferragut J, Guo K, Gupta V, Gutterman C, Nandagopal T, Zussman G. Scalable WiFi Multicast Services for Very Large Groups. In: Proceedings of IEEE International Conference on Network Protocols. 2013, 1–12
16. Jacobson V, Smetters D K, Thornton J D, Plass M F, Briggs N H, Braynard R L. Networking named content. In: Proceedings of ACM International Conference on Emerging Networking Experiments and Technologies. 2009, 1–12
17. Li M, Pei D, Zhang X, Zhang B, Xu K. NDN Live Video Broadcasting over Wireless LAN. In: Proceedings of IEEE International Conference on Computer Communication and Networks. 2015, 1–7
18. Carofiglio G, Gallo M, Muscariello L. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In: Proceeding of IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2012, 304–309
19. Oueslati S, Roberts J, Sbihi N. Flow-aware Traffic Control for a Content-Centric Network. In: Proceeding of IEEE International Conference on Computer Communications. 2012, 2417–2425
20. Saino L, Cocora C, Pavlou G. CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking. In: Proceedings of IEEE International Conference on Communications. 2013, 3775–3780
21. Braun S, Monti M, Sifalakis M, Tschudin C. An Empirical Study of Receiver-based AIMD Flow-Control Strategies for CCN. In: Proceedings of IEEE International Conference on Computer Communications and Networks. 2013, 1–8
22. Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgment options. Technical report, RFC 2018, 1996
23. Paxson V, Allman M. Computing TCP's retransmission timer. Technical report, RFC 2988, 2000
24. Balachandran A, Sekar V, Akella A, Seshan S, Stoica I, Zhang H. Developing a predictive model of quality of experience for internet video. In: Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM). 2013, 339–350



Menghan Li is currently a Ph.D candidate in Department of Computer Science and Technology at Tsinghua University. He received his Bachelor's degree from Tsinghua University in 2007. His current research interests are information-centric network, video streaming and network security.

Dan Pei is an Associate Professor in Computer Science Department at Tsinghua University. Before that he was a researcher at AT&T Research. He received his PhD degree from UCLA in 2005, and his Bachelor's and Master's degrees from Tsinghua University in 1997 and 2000. His current research interests are network measurement, network management, and network security. He is an IEEE Senior Member and an ACM Senior Member. He was selected as an "Expert of China Goverment's Global Talent Recruitment (Youth Program)" in 2012.

Xiaoping Zhang is an Associate Professor in Computer Science Department at Tsinghua University. He received his Ph.D degree from Tsinghua University in 2008. His current research interests are computer networks, high performance router architecture and scalable switch fabric.



Beichuan Zhang is an Associate Professor in the Department of Computer Science at the The University of Arizona. His research interest is in Internet routing architectures and protocols. He has been working on Named Data Networking, green networking, inter-domain routing, and overlay multicast. He received the first Applied Networking Research Prize in 2011 by ISOC and IRTF, and the best paper awards at IEEE ICDCS in 2005 and IWQoS in 2014. Dr. Zhang received Ph.D. from UCLA (2003) and B.S. from Peking University (1995).

Ke Xu received his Ph.D. from the Department of Computer Science of Tsinghua University. He is currently a full professor in the Department of Computer Science of Tsinghua University. He has published more than 100 technical papers and holds 20 patents in the research areas of next generation Internet, P2P systems, Internet of Things(IoT), network virtualization and optimization. He is a member of ACM. He has guest edited several special issues in IEEE and Springer Journals. He is serving as associate editor of Springer journal Networking Science.