# NDN Live Video Broadcasting over Wireless LAN

Menghan Li*, Dan Pei*, Xiaoping Zhang*, Beichuan Zhang†, Ke Xu*
*Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University
Email: limenghan12@mails.tsinghua.edu.cn, {peidan, zhxp, xuke}@tsinghua.edu.cn
†Department of Computer Science, University of Arizona
Email: bzhang@cs.arizona.edu

***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 idea is to use 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 a commodity residential access point and 20 WiFi clients. While NDN with multiple unicast sessions or plain broadcast can support no more than 7 concurrent viewers of a 1Mbps streaming video, NDN plus NLB supports all 20 viewers, and can likely support many more when present.**

## I. 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 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) [3] 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.

*Xiaoping Zhang is the corresponding author.

While video streaming has already received considerable attention in the NDN research community in [4]–[11], 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 [12], 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 retransmission, ordering, network coding, or PHY rate adapting,

978-1-4799-9964-4/15/$31.00 ©2015 IEEE

*etc*., such as Medusa [12], Dircast [13], Flexcast [14], and [15]–[17].

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 ported the AP's code to OpenWrt [18], an embedded Linux that can run on many commodity WLAN APs. The client's code runs on Ubuntu and MacOS. Thus NLB is a practical and deployable solution. We have deployed NLB on a physical testbed consisted of a commodity 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 7 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, and CPU usage.

The remainder of the paper is organized as follows. Section II introduces NDN background and our problem settings. NLB's design and implementation are described in details in Section III. In Section IV, we evaluate NLB using real implementation and testbed experiments. Section V briefly reviews the related work. Finally, a conclusion and the future work are presented in Section VI.

## II. Background and Problem Settings

### A. NDN

NDN architecture [19] 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 [19]'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 for 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.

### B. 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 [11] 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.

### C. 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. 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 [12], 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.*

## III. NLB Design and Implementation

### A. 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 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.

### B. 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 running OpenWrt, 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.

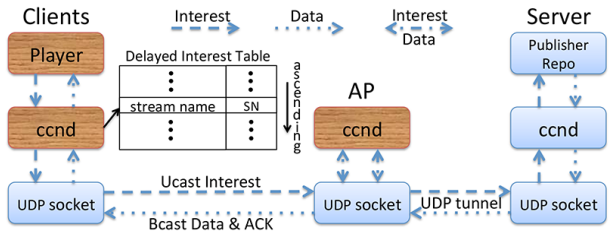



Fig. 1. NLB protocol layers

As illustrated in the Fig. 1, 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 [11] publisher. Multiple clients run a modified NDNVideo player to support live video streaming over 802.11 wireless medium, which typically has high data loss rate and severe delay jitter.

We use ccnx-0.8.1 [20] 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.

### C. Controlling Interest production in application layer

**Live stream bootstrap.** When bootstrapping a live stream in NDN, the player must locate the first newly valid Data segment of the video stream. This mechanism has 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 player needs to buffer the successive Data segments for a time period before actually playing the video. This time period is called *playout delay* and denoted by $T_{pd}$.

**Interest sending rate.** For NDN live streaming, the player 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 player 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 player must get the stream bit rate and then determine the proper Interest sending rate. Although the stream bit rate can be obtained from the metadata of the video stream, this stream bit rate is an average value set by the video codec. 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 player can first obtain the dynamics of stream bit rate from the video server and then adjust the Interest sending rate. We use a straightforward method to recurrently (every $T_s$ seconds) request the latest Data ($S$: 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 Interest pipeline size ($W$) which is the maximum number of pending Interests sent by the player. For example, the average RTT of an Interest to get back a Data is $T_r$. And the two latest Data segments obtained by the player in $T_s$ interval are $S_1$ and

$S_2$ successively. Then we can get

$$W = (S_2 - S_1)T_r/T_s \quad (1)$$

When the player starts, it first requests two latest Data segments and then gets $W$ from the equation 1. The player sets the Interest pipeline size to $W$ and updates it every $T_s$ seconds.

**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 lost in wireless medium. The player adjusts Interest retransmission timer based on previous RTT values [11]. It uses a low pass filter

$$SRTT = (1 - \alpha) * SRTT + \alpha * RTT, (\alpha = 1/8) \quad (2)$$

to smooth RTT values, and uses the TCP/IP formula

$$\begin{aligned} RTTVAR = (1 - \beta) * RTTVAR + \\ \beta * |SRTT - RTT|, (\beta = 1/4) \end{aligned} \quad (3)$$

to smooth variance. The Interest retransmission timer is calculated using

$$T_i = SRTT + K * RTTVAR, (K = 3) \quad (4)$$

If the Data is not returned back after $T_i$ when the Interest is sent out, the player will retransmit the Interest. Here, $\alpha = 1/8$ and $\beta = 1/4$ are the recommended values from RFC2988 [21]. And $K = 3$ is refereced from the NDNVideo technical report [11]. These parameter value work well for the single client requesting the stream. In multiple clients situation, if the player 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 player 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 exclude the very short RTT values (less than 10ms in our experimentation) caused by the local CS in equation 2, so that it can reflect the actual network delay.

The main optimization goal of NLB is to suppress duplicate Interest packets. Therefore, we do not tune the parameter values for calculating the Interest retransmission timer and just use a conservative method to exclude the very short RTT values.

### *D. 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 player 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 player, 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 player. 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 player, 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 ifself 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.

## IV. 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 compare the scalability of the NLB with other approaches as the client number increases.

### A. Performance Metrics

According to [22], 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 player*.
- **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 player*.

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 players is requested by just one Interest packet, with no redundant Interest/Data packets.

### B. Compared Approaches and Experiment Setups

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

- UCast: The clients run NLB's player. The clients and the AP running the original *ccnd* send the Interest and Data over UDP unicast.
- BCast: The clients running NLB's player 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.

We have implemented the AP *ccnd* on OpenWrt [18], a software router system on embedded linux that can run on all major commodity APs. We install OpenWrt and AP *ccnd* on a Mercury MW4530R Router with a 720MHz CPU. To avoid the unpredictable background traffic interference from affecting our measurement results, we use the router's 802.11ac 5GHz channel (which is less crowded than 2.4GHz Channels). Its combined receiving and sending unicast bandwidth is 750Mbps, but the broadcast bandwidth is limited to 6Mbps by the 802.11ac standard.

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 the NDNVideo player and the client *ccnd* on Linux virtual machines running on PCs. Each VM, running a single client, combines an independent wireless card. We have up to 20 clients connected to the Mercury AP via WLAN.

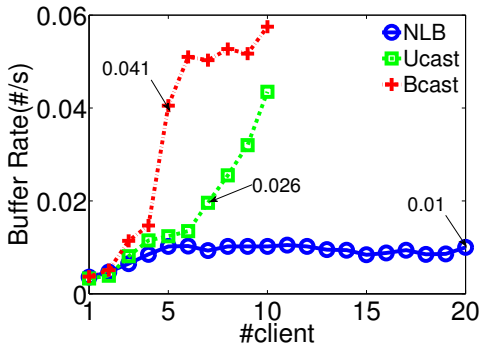
We experiment with a benchmark Mobile calendar video clip [23]. The video is encoded at the rate of 1Mbps using FFmpeg [24] tool with H.264 codec. We have repeated each experiment of 300 seconds duration for 5 runs. In our experiments, for all compared approaches, the playout delay $T_{pd}$ is set to 5 seconds and this duration is included in the time spent on buffering. The time interval $T_s$ of players 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 three compared approaches. 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.

### C. Performance Results

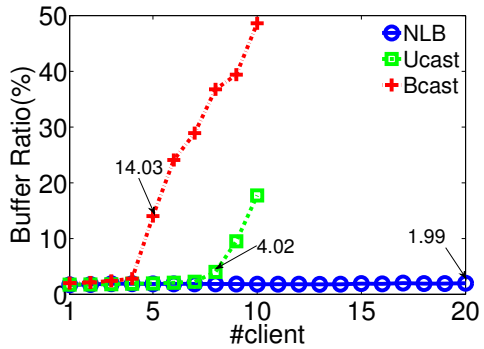
We vary the number of clients and measure the above-defined metrics over multiple clients to evaluate the scalability of 3 different approaches.

**Scalability in the client number.** Our goal is to compare the scalability in the client number of different approaches in the premise for ensuring reliable transmission. Therefore, we set an infinite $R_{max}$ to guarantee no Data loss in application layer. As illustrated in the Fig. 2, for NLB approach within the period of 300 seconds, the average number of buffering events occured for each client is no more than 2 and the average duration of buffering time is at most nearly 1 second (the initial buffering time is excluded). These two values show that NLB can support 20 clients to play back 1Mbps video smoothly. The curve of NLB does not show any clear turning point yet up to 20 clients, although the results were limited to 20 clients due to logistic difficulty in obtaining more client machines at the time of our experiments. The performance of BCast degrades sharply when the client number is larger than 4. And UCast can only support up to 7 clients to play back the video smoothly. For the scalability in the client number, NLB outperforms at least 3 times than UCast and 5 times than BCast.

**Packet redundancy.** The results in the Fig. 2 can be further explained by the packet redundancy results in the Fig. 3(a) and Fig. 3(b), which illustrate the ratio of redundant Interest and Data packets transmitted over wireless medium

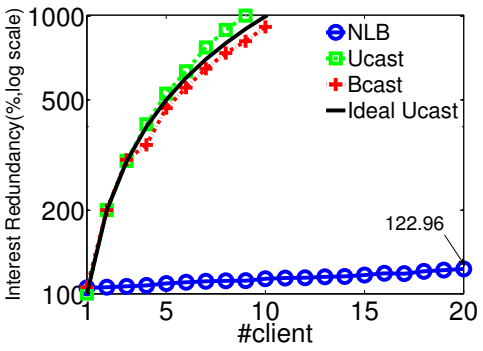



(a) Average Buffering Rate.

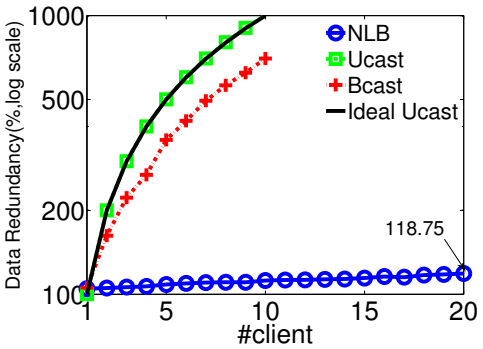



(b) Average Buffering Ratio.

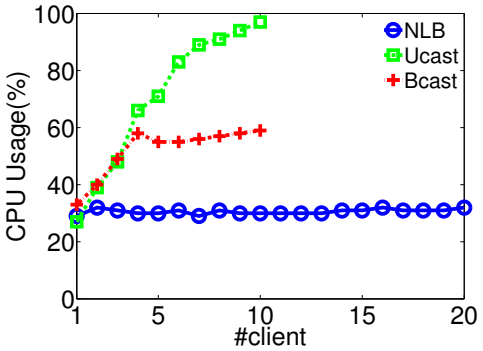
Fig. 2. Scalability comparison




(a) Interest Redundancy.




(b) Data Redundancy.




(c) CPU Utilization.

Fig. 3. The utilization of wireless channel and computing resource

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 only about 120%, which means that the additional packet redundancy introduced by one client is about 1%. 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 II-C 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.

**CPU usage.** The Fig. 3(c) shows the CPU usage by the *ccnd* running on the AP. The CPU usage can reflect the amount of packets processed by the *ccnd* in the unit time. When the client number is larger than 7 in UCast, the CPU usage by the *ccnd* is more than 90%. Such high CPU utilization on a low-end CPU (720MHz) is apparently a major contributor to the sharp performance degradation in UCast. However, for BCast approach, when the client number is larger than 4, the CPU usage by the *ccnd* stays at around 60%. We can infer that the performance of BCast is limited by the AP's 6Mpbs broadcast bandwidth when the client number is larger than 4. For NLB approach, the CPU usage by the *ccnd* always stays at around 30%, which means that the total amount of packets transmitted over wireless medium grows very slowly with 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*.

## V. 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 [4] propose an architecture for mapping HTTP-based streaming applications in CCN. A

cooperative caching strategy is introduced to enable time-shifted TV services in [5]. In order to reduce the transmission overhead of Interest packets, a time-based Interest protocol is proposed in [6], 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 [7]. In [8] 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 [9], 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 [10] 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 [11] 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 [12], Dircast [13], Flexcast [14], and [15]–[17]. 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.

## VI. 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 scalable content distribution end to end.

As advocated in [3], building and experimenting with applications is the way to push forward network architecture research. We believe that NLB not only provides 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 NDN over broadcast.

## Acknowledgement

This work has been supported by the National Key Basic Research Program of China (973 program) under Grant No.2013CB329105, and the National Natural Science Foundation of China (NSFC) under Grant No.61233007, No.61472210, No.61472214.

We would like to thank the reviewers for their valuable comments. And we sincerely appreciate Lixia Zhang, Jeff Burke, Ilya Moiseenko, and Lijing Wang for their useful experience on NDN project. Also, we express thanks to Jianxun Cao, Yantao Lai and Changhua Pei for their assistance on experimenting. At last, we thank Juexing Liao for her proofreading of this paper.

## References

[1] "Internet Phenomena report," 2011. [Online]. Available: https://www.sandvine.com/trends/global-internet-phenomena/

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

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc cllaffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 44, no. 3, July 2014.

[4] H. Xu, Z. Chen, R. Chen, and J. Cao, "Live streaming with content centric networking," in *Proc. 3rd Int. Conf. on Networking and Distributed Computing*, Hangzhou, China, 2012.

[5] Z. Li and G. Simon, "Time-shifted TV in content centric networks: the case for cooperative in-network caching," in *Proc. of IEEE ICC*, 2011.

[6] J. Park, J. Kim, M. Jang, and B. Lee, "Time-based interest protocol for real-time content streaming in content-centric networking (CCN)," in *Proc. of IEEE Int. Conf. on Consumer Electronics (ICCE)*, 2013.

[7] V. Ciancaglini, G. Piro, R. Loti, L. A. Grieco, and L. Liquori, "CCN-TV: a data-centric approach to real-time video services," in *Proc. of IEEE AINA*, Barcelona, Spain, Mar 2012.

[8] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, "Offloading cellular networks with Information-Centric Networking: the case of video streaming," in *Proceedings of the IEEE WoWMoM*, 2012.

[9] B. Han, N. Choi, T. Kwon, and Y. Choi, "AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking," in *Proc. of IEEE NOMEN*, 2013.

[10] A. Detti, M. Pomposini, and N. Blefari-Melazzi, "Peer-to-peer live adaptive video streaming for information centric cellular networks," in *Proc. of IEEE 24th PIMRC*, 2013.

[11] D. Kulinski and J. Burke, "NDNVideo: Random-access Live and Pre-recorded Streaming using NDN," NDN, Tech. Rep. TR-0007, 2012.

[12] S. Sen, N. Madabhushi, and S. Banerjee, "Scalable WiFi Media Delivery through Adaptive Broadcasts." in *NSDI*, 2010, pp. 191–204.

[13] R. Chandra, S. Karanth, T. Moscibroda, V. Navda, J. Padhye, R. Ramjee, and L. Ravindranath, "Dircast: A practical and efficient Wi-Fi multicast system," in *Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on*. IEEE, 2009, pp. 161–170.

[14] S. Aditya and S. Katti, "Flexcast: graceful wireless video streaming," in *Proceedings of the 17th annual international conference on Mobile computing and networking*. ACM, 2011, pp. 277–288.

[15] S. Jakubczak and D. Katabi, "A cross-layer design for scalable mobile video," in *Proceedings of the 17th annual international conference on Mobile computing and networking*. ACM, 2011, pp. 289–300.

[16] Y. Park, C. Jo, S. Yun, and H. Kim, "Multi-room IPTV delivery through Pseudo-Broadcast over IEEE 802.11 links," in *Vehicular Technology Conference, 2010 IEEE 71st*. IEEE, 2010, pp. 1–5.

[17] Y. Bejerano, J. Ferragut, K. Guo, V. Gupta, C. Gutterman, T. Nandagopal, and G. Zussman, "Scalable WiFi Multicast Services for Very Large Groups," in *the 21th IEEE ICNP*, 2013, pp. 7–11.

[18] OpenWrt, "https://openwrt.org/."

[19] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[20] CCNx, "http://www.ccnx.org/."

[21] V. Paxson and M. Allman, "Computing TCP's retransmission timer," RFC 2988, Nov. 2000.

[22] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 339–350.

[23] "MPEG-2 hd test patterns." [Online]. Available: http://www.w6rz.net/

[24] "FFmpeg - digital audio converter." [Online]. Available: http://www.ffmpeg.org/