Content-Priority-Aware Chunk Scheduling Over Swarm-Based P2P Live Streaming System: From Theoretical Analysis to Practical Design

Chun-Yuan Chang, *Member, IEEE*, Cheng-Fu Chou, and Kwang-Cheng Chen, *Fellow, IEEE*

***Abstract*—In this work, we revisit two chunk, which is the smallest video data unit for scheduling and transmission, scheduling policies that are essentially mutually exclusive but beneficial for swarm-based P2P live streaming systems. The first is *content-diversified* oriented (*cd-oriented*) policy, which regards each chunk of equal importance and schedules chunks to be sent in a near-random fashion. With this approach, peers hold different parts of stream content and contribute their available bandwidth to the system. The second is *importance-first* oriented (*if-oriented*) policy, which gives each chunk a content-dependent priority, usually in a rate-distortion (*RD*) sense, and first schedules the highest-priority chunk to be sent. In doing so, important chunks are more likely to be successfully received before their playback; the reconstructed video quality is thus enhanced under poor network conditions. We successfully identify a simple methodology, which operates on the data availability domain, to leverage both policies. (Data availability here means the *set* of data units that the user can get from its source(s)). This allows us to deploy dynamic strategy switch scheduling in practical systems to further improve the received video quality of each peer. Simulation results show that our data-availability driven dynamic strategy switch not only overcomes the drawbacks of the two individual policies but also retains the benefits of both. Most importantly, it bridges the gap between rate-distortion analysis on compressed video and P2P content delivery research.**

***Index Terms*—Chunk scheduling, P2P streaming, rate-distortion optimized.**

I. Introduction

RECENT years have witnessed the success of swarm-based P2P live streaming systems, including CoolStreaming [1], [15], GridMedia [2], PPTV [3], PPStream [4], UUSee [5], and others. In these systems, peers not only serve as stream content receivers but also play the role of stream content forwarders to assist the streaming server in broadcasting live stream content to a large number of peers on the Internet. Compared with the conventional client-server paradigm, this emerging content diffusion paradigm allows content providers to save on more infrastructure cost and to instead serve more peers. It is worth noting that in these systems there is no rendezvous peer to tell each peer what to do. Instead, each peer arbitrarily builds neighbor relationships with other peers, and exchanges its available content with its neighbor peers in an epidemic fashion. Since each peer may maintain several neighbor peers, when some of the peer's neighbors leave or are congested (that is, busy serving other peers), the peer can immediately shift its request to other neighbors to prevent sudden interruptions in stream content delivery. Hence, these systems are highly robust to peer churn and are able to maintain excellent load-balancing among peers.

Nevertheless, these systems are still in their early stages, and much research, from different points of view, has been conducted to improve system performance. For example, [11] and [12] aim at an adaptive neighbor selection approach such that peers with more available bandwidth can crawl as close to the streaming server to speed up chunk diffusion. Works [12] and [13] add incentives to encourage peers to contribute more to the system; this increases the system's maximal sustained streaming rate. To reduce the impact of peer churn, [14] takes into account peer stability during neighbor selection. To effectively utilize each peer's available bandwidth, [15]–[22], [43] focus on a chunk scheduling mechanism to determine which chunk to diffuse first. Works [40]–[42] take into account the advantages of a content delivery network (CDN) to further address critical issues in swarm-based P2P streaming systems, such as a high buffering requirement, frequent disruption of playback, unfairness in upload contributions, and network-unfriendliness. Our work is orthogonal to these excellent works in that we further take into account the characteristics of compressed video to explore a content-priority-aware chunk scheduling mechanism to improve received video quality for poor network conditions or insufficient aggregate available bandwidth.

In the past, many systems have adopted such techniques to improve video quality. Since these systems were built in a client-server manner (i.e., an always-on server holds complete stream content from which clients obtain their stream content), there is no *data availability* problem in these systems Simply using an *if-oriented* scheduling policy [23]–[26] significantly improves the received video quality of each peer. The main

Manuscript received September 01, 2013; revised December 16, 2013; accepted December 29, 2013. Date of publication January 20, 2014; date of current version March 07, 2014. This work was supported by the National Science Council of China under Contract NSC 101-2221-E-002-059-MY2, Contract 102-2221-E-002-096-MY3, and Contract 102-2622-E-002-013-CC2. This paper was recommended by Guest Editor C. W. Chen.

C.-Y. Chang and C.-F. Chou are with Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 10617, Taiwan (e-mail: left@cmlab.csie.ntu.edu.tw; ccf@cmlab.csie.ntu.edu.tw).

K.-C. Chen is with Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2014.2298278

idea of the scheduling policy is to assign each chunk a content-dependent priority (usually in an RD sense) and schedule to be sent first that chunk with the highest priority. In doing so, important chunks are more likely to be successfully delivered (or received) before their playback, thus enhancing the reconstructed video quality.

However, in a swarm-based P2P streaming system, chunks are diffused by the exchange of available peer chunks. Under this kind of content diffusion framework, peers' buffered stream content must be made sufficiently diverse. Otherwise, due to insufficient data availability, peers are unable to efficiently utilize available bandwidth [16]. To address this, many *cd-oriented* chunk scheduling mechanisms have been proposed (for instance in [15]–[22]), in which chunks are regarded as equally important and are scheduled to be sent in a near random fashion. Consequently, peers can hold different parts of the stream content, maintaining data availability, and thus are enabled to contribute their available bandwidth to the system.

Clearly, it is not feasible to apply an *if-oriented* chunk scheduling mechanism directly into a swarm-based P2P streaming system, because doing so tends to make similar the buffered content for different peers. This effect would not be beneficial for peers' available bandwidth utilization, and could be exacerbated significantly for large populations or when considering finer rate-distortion (RD) characteristics. This only complicates the design of a content-aware chunk scheduling mechanism (in an RD sense). In this work, we seek to leverage *cd-oriented* and *if-oriented* chunk scheduling policies to further improve the received video quality of each peer.

In the real world, swarm-based P2P streaming systems are quite complex due to the many design considerations (neighbor selection policy, chunk scheduling mechanism load balancing) and dynamics (peer churn, heterogeneous peer capability, and video streaming rate) that affect performance. To isolate effects caused only by chunk scheduling, we begin with an abstract stochastic model [18], [43] that assumes neighbor selection is random and that all peers have the same capability. Since the model takes into account only buffer positions and does not consider the inherent content importance, we modify it to render it suitable for the analysis of the *if-oriented* scheduling policy. A comparison of the *cd-oriented* chunk scheduling policy with the *if-oriented* chunk scheduling policy reveals several properties of the *if-oriented* chunk scheduling policy that are unique to swarm-based P2P streaming systems. This suggests a simple way, from the data availability domain, to effectively leverage the two policies with differing population sizes. In short, we achieve the objection by simply maintaining the data availability of each peer above a certain amount. This finding led to the design of a dynamic strategy switch chunk scheduling mechanism for systems in practice to further improve the received video quality of each peer.

To best of our knowledge, this is the first series of work that makes use of the data availability of each peer to leverage both the *cd-oriented* and *if-oriented* chunk scheduling policies. Simulation results show that the proposed data-availability driven dynamic strategy switch not only provides better video quality compared with existing approaches, but also is scalable even to large population sizes. Most importantly, it bridges the gap between rate-distortion analysis on compressed video and P2P content delivery research.

A. Contributions

Our contributions are the following.

- We build a general analytical model to analyze the performance of the *cd-oriented* and if*-oriented* chunk scheduling mechanisms, and identify several properties of the *if-oriented* chunk scheduling policy that are unique to swarm-based P2P streaming systems.
- We successfully identify a simple methodology, which operates on the data-availability domain, to leverage the *cd-oriented* and if*-oriented* chunk scheduling mechanisms. This allows us to deploy dynamic strategy switch scheduling (DSW) in practical systems. Most importantly, it bridges the gap between rate-distortion analysis on compressed video and P2P content delivery research.
- We conduct a performance evaluation of DSW in a practical simulation environment and theoretically explain why our dataavailability driven DSW is better than other approaches.
- We illustrate the limitations of content-priority-aware chunk scheduling in swarm-based P2P streaming systems.

B. Organization

The remainder of this paper is organized as follows. In Section II we further enhance our extended model in [29] so that we can analyze and identify the properties of the *if-oriented* scheduling policy that are unique to swarm-based P2P streaming systems when finer priorities are assigned to each chunk, and share some useful insights. After that, in Section III we successfully devise a chunk scheduling design methodology, which resolves our initial concern in [28], based on the extended abstract stochastic model with advantages from both *cd-oriented* and *if-oriented* chunk scheduling policies, and then describe implementation issues in Section IV. The simulation environment and settings are described in Section V. In Section VI, we evaluate the performance of DSW in comparison with other approaches. We conclude in Section VII.

II. Insights From Abstract Stochastic Model

In this section, we outline the abstract stochastic model developed in [18], [43] and modify it to reveal the difference between the *cd-oriented* and *if-oriented* chunk scheduling mechanisms. This yields several useful insights, based on which we explore the co-existence of the two mechanisms, and derive a unique evaluation metric, *data availability*, that can be used effectively to control the tradeoff between content diversity and content importance under different system parameters (for example, startup latency, population size, and fineness of rate-distortion characteristics). This unique measurable metric enables us to develop a dynamic strategy switch in practical systems.

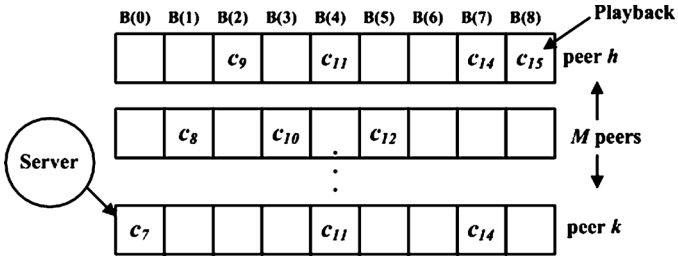

Fig. 1. Abstract P2P streaming model in [18].

A. Abstract Stochastic Model in [18] and [43]

As shown in Fig. 1, a server and M peers are in this model. The server generates a stream of chunks $c_1, \ldots, c_t, \ldots$, and then disseminates them, in playback order, into the system. In this model, the server is assumed to generate u chunks per time slot. Consequently, only u peers can obtain the newly generated chunks from the server. Each peer is equipped with a streaming buffer B that can accommodate up to n chunks received from other peers (including the server). The buffer positions are indexed by $i \in \{0, \ldots, n-1\}$. $B(0)$ is used to store the newest chunk that the server distributes in the current time slot; $B(n-1)$ is reserved for playback. In this model, the buffer acts as a sliding window: in each time slot, each peer fetches the chunk stored in $B(n-1)$ for playback and then shifts the buffer contents by one position toward the playback deadline for the next playback. If $B(n-1)$ is empty, a peer experiences a chunk loss in the time slot. Since the model is for a live streaming system, all peers are to playback the same chunk at the same time.

Apart from obtaining it from the server, a peer could also obtain any chunk from other peers up until the playback time of that chunk. In this model, a peer is assumed to be able to download at most one chunk per time slot. That is, if a peer has been selected by the server, it no longer contacts other peers for download. As such, each peer first checks if it has been selected by the server; if not, the peer may contact other peers to retrieve a chunk not in its buffer. In the real world, a peer may connect to multiple peers. For simplicity, however, in this model, a peer is limited to contacting a single peer at each time slot. If the contacted peer has more than one chunk that the peer desires, the peer selects one of them, according to a chunk scheduling mechanism ϕ, for download. If the contacted peer cannot provide any desired content to the peer, the peer loses the chance to download anything in this time slot, and a *content bottleneck* occurs. Since each peer may randomly contact other peers, a peer may be contacted by multiple other peers in a given time slot. However, when M is large, the likelihood of such an event occurring is quite low [18], [43]; hence it is reasonable to ignore the impact of such an event by assuming that each peer is able to serve all requests. In addition to the above settings and assumptions, [18], [43] assumes that buffer occupancy probabilities across all peers are i.i.d., We thus have the following simple relation between the steadystate buffer occupancy probabilities for chunk scheduling mechanism ϕ as follows:

$$p_\phi(i+1) = p_\phi(i) + (1 - p_\phi(i))p_\phi(i)s_{\phi,\mathcal{A}}(i) \quad i = 1, \ldots, n-2 \text{ and } p(0) = u/M. \quad (1)$$

In above equation, since the buffer position$B(i+1)$ is filled by a shift from $B(i)$, the buffer occupancy probability at the beginning of the time slot $i+1$ is the probability that the probability $B(i)$ was filled at the beginning of the time slot i plus the probability that $B(i)$ was filled by the P2P streaming protocol during the time slot i; this is expressed as $(1-p_\phi(i))p_\phi(i)s_{\phi,\mathcal{A}}(i)$. The terms $(1-p_\phi(i))p_\phi(i)$ and $s_{\phi,A}(i)$ represent the probability that peer k does not have $B(i)$ but its selected peer h does, and the probability that peer k chooses to download $B(i)$ from its selected peer h, given that k does not possess $B(i)$ while h does, respectively. Subscript $\mathcal{A}$ represents the associated buffer positions that$B(i)$ is competing with. Unless otherwise stated, $\mathcal{A}$ covers all of the buffer positions that are before $B(n-1)$, excluding$B(i)$.

In [18], Zhou *et al.* evaluated the system performance of different scheduling policies by specifying $s_{\phi,\mathcal{A}}(i)$. However, they do not consider the inherent content importance. We refer these policies as *content-independent* chunk scheduling mechanisms (random or rarest-first scheduling mechanisms, etc.). Next, we modify the above model and further extend $s_{\phi,\mathcal{A}}(i)$ to build a stochastic probability model like (1) to evaluate the performance of an *if-oriented* chunk scheduling policy.

B. Extended Abstract Stochastic Model

By definition, a peer using *if-oriented* scheduling first schedules the most important chunk to be sent or downloaded. Hence we must first assign an importance to each chunk. This, however, is not trivial, as it must reflect not only the RD impact of each chunk but also its distribution over time, which depends on the video content characteristics (high or low motion videos), encoding setting (dependency configuration, rate control mechanism), as well as the packetization approach. Since the purpose of this extended model is to reveal the difference between the *cd-oriented* and *if-oriented* chunk scheduling policies, it is sufficient to simply assign a relative priority to each chunk. Without loss of generality, we assume there are L kinds of chunks, with priorities $\rho_0 \succcurlyeq \ldots \rho_l \succcurlyeq \ldots \succcurlyeq \rho_{L-1}$, and assume that the injection of each kind of chunk into the system in each time slot is in a round-robin fashion. Consequently, we can construct a stochastic model for specifying buffer occupancy probability of *if-oriented* scheduling as follows. Note that there are two implications with *if-oriented* scheduling.

1) The diffusion of the highest-priority chunks is not affected by lower-priority chunks.
2) Lower-priority chunks may be diffused only when the selected peer cannot provide for the retrieval of any higher-priority chunk.

The first implication indicates that the derivation of the highest-priority buffer occupancy probabilities does not take into account other buffer occupancy probabilities; the second indicates that the derivation of lower-priority buffer occupancy probabilities must take into account the higher-priority buffer occupancy probabilities. These two implications suggest that we should specify the buffer occupancy probabilities of each ρ_l individually and in a recursive fashion. Note that since the injection of each kind of chunk into the system in each time slot

is assumed to be in a round-robin fashion, when $L < n-1$, the selected peer could provide more than one chunk with the same priority. Hence, we must still employ a *content-independent* chunk scheduling mechanismϕ to break ties when the selected peer has more than one chunk with same priority.

Let us denote the resultant buffer occupancy probabilities of each ρ_l that are associated with ϕ as $p_\phi^{\rho_l}(i)$. We thus derive $p_\phi^{\rho_l}(i)$ with different ρ_l as follows.

Initial

$$p_\phi^{\rho_0}(i+1) = p_\phi^{\rho_0}(i) + \left(1 - p_\phi^{\rho_0}(i)\right) p_\phi^{\rho_0}(i) s_{\phi,\mathcal{A}=\mathcal{B}(\rho_0)}(i) \quad i = 1, \ldots, n-2 \text{ and } p(0) = u/M. \quad (2.1)$$

In contrast to $s_{\phi,\mathcal{A}}(i)$ in (1), $\mathcal{A}$ here is here set to $\mathcal{B}(\rho_0)$, where $\mathcal{B}(\rho_0)$ is the buffer position that currently should be filled with the priority ρ_0 chunk. This indicates that the highest-priority chunk competes only with chunks with the same priority.

Recursion

$$p_\phi^{\rho_l}(i+1) = p_\phi^{\rho_l}(i) + \left(1 - p_\phi^{\rho_l}(i)\right) p_\phi^{\rho_l}(i) e_{0,l-1}(i) s_{\phi,\mathcal{A}=\mathcal{B}(\rho_l)}(i) \quad i = 1, \ldots, n-2 \text{ and } p(0) = u/M \quad (2.2)$$

where $e_{0,l-1}(i)$ is the probability that the selected peer h cannot provide any desired chunk with a priority higher than ρ_l. In greater detail

$$e_{0,l-1}(i) = \prod_{i=0}^{l-1} \prod_{j \in \mathcal{B}(\rho_i)} \left(p_\phi^{\rho_i}(j) + \left(1 - p_\phi^{\rho_i}(j)\right)^2 \right) \quad (2.3)$$

where $(p_\phi^{\rho_i}(j) + (1 - p_\phi^{\rho_i}(j))^2)$ is the probability that peer k has $B(j)$ plus the probability that peer k does not have the chunk for $B(j)$, nor does the selected peer h.

After obtaining each $p_\phi^{\rho_i}(i)$, we have the buffer occupancy probability of an *if-oriented* scheduling policy that takes into account ϕ and L

$$p_{if(\phi),L}(i) = \left(\frac{1}{L}\right) \sum_{i=0}^{L-1} p_\phi^{\rho_i}(i). \quad (3)$$

Remark 1: When $L \geq n-1$, $p_{if(\phi),L}(i)$ is independent of ϕ. This is because when $L \geq n-1$, $s_{\phi,\mathcal{A}=\mathcal{B}(\rho_l)}(i)$ becomes 1.

Remark 2: When $L = 1$, $p_{if(\phi),L}(i)$ becomes $p_\phi(i)$.

This is because when $L = 1$, all chunks are regarded as equally important. So $e_{0,l-1}(i)$ becomes 1 and $p_{if(\phi),L}(i)$ is reduced to $p_\phi(i)$.

Below, we take rf[1] (the *rarest-first* scheduling), and its corresponding $if(rf)$ as example to show the difference between the *cd-oriented* and *if-oriented* chunk scheduling policies.

C. Numerical Result Analysis

In this subsection, we seek to compare rf and *if(rf)* in the context of a pure P2P environment (that is, $u = 1$) based on the extended stochastic model, and investigate parameters like n, M, and L on the performance of the chunk scheduling policy, for instance buffer occupancy probabilities. Since the simplest case for an *if-oriented* chunk scheduling policy is $L = 2$, below we start with the analysis of case$L = 2$. For $L = 2$, we have

Result 1: $p_{rf}^{\rho_0}(i) \leq p_{rf}^{\rho_1}(i)$, where $0 \leq i \leq n-1$ (see Fig. 2).

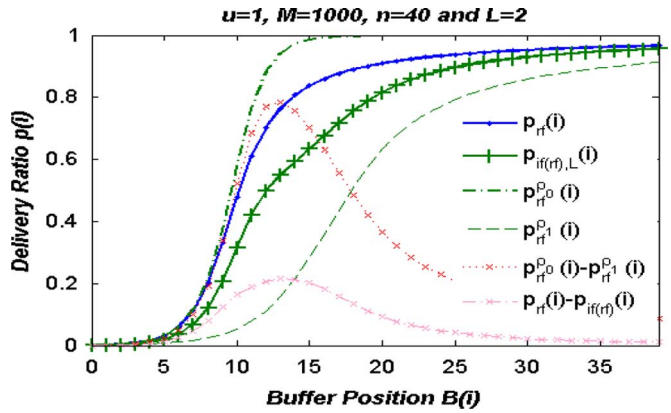

Fig. 2. Delivery ratios for rf and $if(rf)$ with $u = 1$, $M = 1000$, $L = 2$, and $n = 40$.

This result reveals the inherent property of the *if-oriented* chunk scheduling policy. To further understand this policy, we investigate the variation of $p_{rf}^{\rho_0}(i)$ and $p_{rf}^{\rho_1}(i)$ for i in Fig. 2. It is interesting that $\delta = p_{rf}^{\rho_0}(i) - p_{rf}^{\rho_1}(i)$ increases rapidly before position $i = 13$ and then decreases significantly. In practice, after each chunk c is produced by the server and pushed to a peer, chunk c attempts to traverse all peers in the system. Over time, a diffusion tree forms for chunk c is formed. Clearly, for a chunk with a higher diffusion priority than other chunks, its diffusion tree grows larger than that of other chunks. So $p_{rf}^{\rho_0}(i)$ is essentially larger than $p_{rf}^{\rho_1}(i)$. Note that unlike client-server models, chunks must be relayed by participating peers in a swarm-based P2P streaming system. Over time, the size difference between the diffusion trees formed by the chunk with priorities ρ_0 and ρ_1 grows significantly. Hence, $\delta = p_{rf}^{\rho_0}(i) - p_{rf}^{\rho_1}(i)$ initially increases rapidly. However, as shown, after $i = 13$, this difference shrinks, because the peer population is not unlimited: the growing of the diffusion tree is bounded by the peer population M. Therefore, the diffusion speed of priority-ρ_0 chunks lowers. In contrast, after the system is full of priority-ρ_0 chunks, priority-ρ_1 chunks gradually catch up with the occupancy of the higher-priority chunks.

Result 2: $p_{if(rf),L=2}(n-1) \leq p_{rf}(n-1)$[2] (see Fig. 2).

We observe that the performance of *if(rf)* is worse than that of rf. This is because in swarm-based P2P streaming, content broadcasting relies mainly on the exchange of available chunks among peers. That is, at any time, each peer must experience sufficient content diversity, or else the peers will be unable to use their bandwidth efficiently. Clearly the *if-oriented* chunk scheduling policy violates this principle. Below, we explain this unique side effect in more detail.

[1]The selection rf here is because it has been recognized as one of the most powerful chunk scheduling policies for bandwidth utilization in swarm-based P2P streaming systems. A derivation of $p_{\phi=rf}(i)$ and $p_{if(\phi=rf),L}(i)$ can be found in the Appendix.

[2]Because we assume a steady state model, buffer position i and buffer size are equivalent.

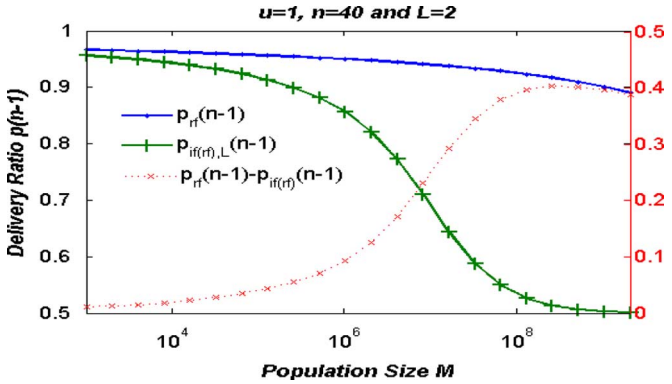

Fig. 3. Delivery ratios for rf and $if(rf)$ with $u = 1$, $n = 40$, $L = 2$, but with different M.

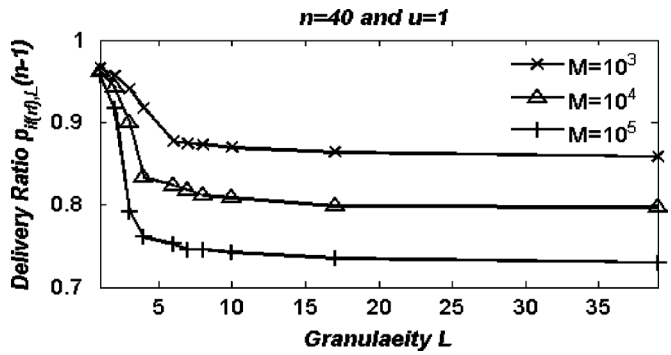

Fig. 4. The delivery ratio of $if(rf)$ with $u = 1$, $n = 40$, but with different L and M.

In the *if-oriented* chunk scheduling policy, peers first obtain high-priority chunks; then when a peer has the whole set of high-priority chunks, it begins requesting and diffusing low-priority chunks. Given a small buffer size, all peers are engaging in requesting or disseminating high-priority chunks and have no spare time to exchange low-priority chunks yet. In such a case, there must be only a small percentage of peers that own the low-priority chunks. Since there are only a small set of peers with low-priority chunks, most peers cannot see those low-priority chunks; thus most peers cannot request those low-priority chunks. In other words, a serious content bottleneck for low-priority chunks occurs. Hence, compared with rf, we cannot leverage the available bandwidth of each peer under the *if-oriented* chunk scheduling policy. The performance of *if(rf)* is thus worse than that of rf.

Scalability is one of the major concerns of a P2P system. Below, we study the impact of M on the performance of rf and *if(rf)*.

Result 3: Let $\delta = p_{rf}(n-1) - p_{if(rf),L=2}(n-1)$. δ increases with M before $p_{rf}(n-1)$ and $p_{if(rf),L=2}(n-1)$ converge (see Fig. 3).

This is due to the gains from multiplexing. That is, for the rf chunk scheduling policy, each peer dedicates its resources to diffusing all chunks as quickly as possible, while under the *if(rf)* chunk scheduling policy each peer allocates its resources first to high-priority chunks; that is, it is only when the selected peer cannot provide any higher-priority chunks for retrieval that *if(rf)* uses the remaining bandwidth to exchange low-priority chunks. Hence, for large populations there are only a small percentage of peers that possess low-priority chunks, similar to the scenario discussed in Result 2. Thus, in this case, even peers with sufficient upload bandwidth experience serious content bottleneck for low-priority chunks. This could lead to poor utilization of upload bandwidth.

However, this performance gap does not always increase with the population size. Instead, when the population size exceeds a valueα, the gap shrinks. This is because excessively large populations lengthen the desired buffering delay for a chunk to traverse the whole system. If each peer's buffering time, that is, buffer size n, cannot tolerate this desired delay, most peers will be unable to get the chunk before the playback deadline. That is, no matter which scheduling policy we choose, the delivery ratio degrades with larger populations because of the playback deadline and the required processing time (including queueing) at each node.

Note that in real-world systems, chunk importance is quite dynamic; that is, each chunk could have a different importance in an RD sense. Hence it is worth investigating cases of larger L are worthy of being.

Result 4: The larger L is, the worse $p_{if(rf),L}(n-1)$ is (see Fig. 4).

From Fig. 4, we see that when finer RD characteristics are taken into account, the performance of *if(rf)* degrades significantly. Below, we build on Remark 2 and Result 2 to provide a justification for this observation.

We first consider the case of $L = 1$ and $L = 2$ for *if(rf)*. As indicated in Remark 2, when L is set to 1, $p_{if(rf),1}(n-1) = p_{rf}(n-1)$. We thus have $p_{if(rf),1}(n-1) \geq p_{if(rf),2}(n-1)$. Suppose $p_{if(rf),L/2}(n-1) \geq p_{if(rf),L}(n-1)$ holds, where $L = 2^k$ and $k \geq 2$.

We check if $p_{if(rf),L}(n-1) \geq p_{if(rf),2L}(n-1)$ as follows. We take the lowest-priority chunk cluster ρ_{L-1} and branch it out into two equal-sized clusters $\rho_{L-1,0}$ and $\rho_{L-1,1}$. This yields the new chunk partition $\{\rho_0, \ldots, \rho_{L-2}, \{\rho_{L-1,0}, \rho_{L-1,1}\}\}$. Suppose $\rho_0 \succcurlyeq \ldots \succcurlyeq \rho_{L-2} \succcurlyeq \rho_{L-1,0} \succcurlyeq \rho_{L-1,1}$, and apply *if(rf)* on this new partition. We obtain the average delivery ratio of the new partition and denote it as follows:

$$p_{if(rf),\{\rho_0,\ldots,\rho_{L-2},\{\rho_{L-1,0},\rho_{L-1,1}\}\}}(n-1) = \frac{\sum_{i=0}^{L-2} p_{rf}^{\rho_i}(n-1) + \frac{p_{rf}^{\rho_{L-1,0}}(n-1) + p_{rf}^{\rho_{L-1,1}}(n-1)}{2}}{L}. \quad (4)$$

By Result 3, we infer that the equality

$$(p_{rf}^{\rho_{L-1,0}}(n-1) + p_{rf}^{\rho_{L-1,1}}(n-1))/(2) \leq p_{rf}^{\rho_{L-1}}(n-1)$$

could hold, and derive the following equality:

$$p_{if(rf),L}(n-1) = p_{if(rf),\{\rho_0,\ldots,\rho_{L-2},\rho_{L-1}\}}(n-1) \geq p_{if(rf),\{\rho_0,\ldots,\rho_{L-2},\{\rho_{L-1,0},\rho_{L-1,1}\}\}}(n-1). \quad (5)$$

Similarly, we continue to branch out ρ_{L-2} untilρ_0, and derive the following equality chain:

$$\begin{aligned} p_{if(rf),L}(n-1) &\geq p_{if(rf),\{\rho_0,\ldots,\rho_{L-2},\{\rho_{L-1,0},\rho_{L-1,1}\}\}}(n-1) \\ &\geq p_{if(rf),\{\rho_0,\ldots,\{\rho_{L-2,0},\rho_{L-1,1}\},\{\rho_{L-2,0},\rho_{L-1,1}\}\}}(n-1) \\ &\ldots \\ &\geq p_{if(rf),\{\{\rho_{0,0},\rho_{0,1}\},\ldots,\{\rho_{L-2,0},\rho_{L-1,1}\}\}}(n-1) \\ &= p_{if(\phi),2L}(n-1). \end{aligned} \tag{6}$$

By mathematical induction, we infer that $p_{if(rf),2^0}(n-1) \geq p_{if(rf),2^1}(n-1) \geq \ldots \geq p_{if(rf),2^k}(n-1)$.

D. Summary and Insights

From the above discussion, we understand the properties of the *if-oriented* scheduling mechanism that are unique to a swarm-based P2P streaming system; that is, n, M, and L significantly influence its performance. We learn that if we want to reap the benefit of the *if-oriented* scheduling mechanism, we must solve the content bottleneck problem. The result of Fig. 2 suggests that increasing the buffer size (startup latency) could be one way to do this. However, in practice, the system designer keeps startup latencies as short as possible so that peers will be willing to enter the system (in general, peers are not tolerant of long startup latencies). Thus, increasing the buffer size is not a good choice. Recall that for small populations (as shown in Figs. 3 and 4), *if(rf)* performs well: it not only yields a high delivery ratio of high-priority chunks but also yields comparable bandwidth utilization to rf. This is because with a small population, high-priority chunks are disseminated to the whole system relatively quickly, so low-priority chunks need not wait long to be diffused. This ensures system-wide content diversity and a more muted content bottleneck effect. This leads to two useful insights: 1) sufficient content diversity allows us to directly apply the *if-oriented* scheduling mechanism and 2) we ought to strike a good balance between content diversity and content importance.

III. Design Methodology

In this section, based on the simple model outlined above, we propose a chunk scheduling design methodology that shares advantages from both rf and *if(rf)* chunk scheduling policies and that is able to adapt to different system parameters such as M and L.

A. Buffer Space Concatenation Approach

One of the simplest ways to achieve the above objective is to combine the two policies in a concatenative fashion. That is, 1) first, we divide the entire buffer space into two parts, a prefix part and postfix part, at a cutoff buffer position k; the prefix part and the postfix part cover $B(0)$ to $B(k-1)$ and $B(k)$ to $B(n-1)$, respectively; and 2) then, we perform rf and *if(rf)* on the prefix and postfix parts, respectively.

To determine the feasibility of this approach, we perform a brute-force search on k to determine whether there exists a k such that the delivery ratio of this concatenated approach, denoted as $p_{c,k}(n-1)$, is comparable to that of rf. We say that there exists a chunk scheduling method that shares advantages from both rf and *if(rf)* chunk scheduling policies if there exists such a k less than the buffer size.

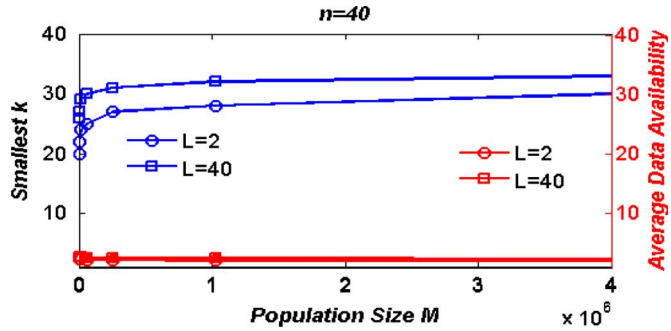

Fig. 5. Blue line stands for buffer size used for rf part. Red line stands for the corresponding average data availability rf part can supply.

For this, we find the smallest $k^* = MIN(\{k : |p_{c,k}(n-1) - p_{rf}(n-1)| < \delta = 0.001\})$ for various population sizes (M) and chunk importance granularities (L), and plot the results in Fig. 5. We can see that 1) k^* is always less than the buffer size; 2) the larger M or L is, the larger k^* is.

The first finding confirms the existence of such a scheduling policy; the second shows that large values of M or L call for increased content availability, which can be achieved by increasing k^*, to support the necessary content diversity. Actually, this indicates the inherent limitation of content-priority-aware chunk scheduling in swarm-based P2P streaming systems because weight of if becomes small. We will discuss this limitation in real simulation results in VI.

Note that the above buffer space concatenation approach leads to a way to leverage rf and *if(rf)*, but it is performed only when peers are aware of M and L. However, in practice, a peer's joining or leaving the system and its video characteristics are hard to predict. Hence, such an ideal approach is hard to realize in real-world systems. Below, based on the ideal approach, we explore a simpler way to adapt to these system dynamics.

B. Data-Availability Driven Approach

From above analysis, we see that the adjustment of the coverage of rf enables it to provide sufficient data availability for later *if(rf)*. Hence, it is instructive to further investigate what data availability the rf part supports. For this, we transform the rf part to its corresponding average data availability as

$$\mathbb{E}[L] = \sum_{\ell=0}^{k^*-1} \ell \cdot \sum_{0 \leq j1 < \ldots < j\ell \leq k^*-1} \left(\prod_{i \in \{j\ell,\ldots,j\ell\}} ((1 - p_{rf}(i)) \cdot p_{rf}(i)) \right) \tag{7}$$

and plot the transformed results in Fig. 5 for comparison. We see that regardless of the population size M and chunk importance granularity L are, the average content availability that rf part can supply is bounded by the range [1.88, 2.5], which is quite

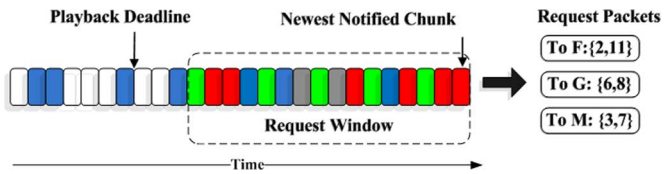

Fig. 6. Status of the request window and playback deadline in each peer. Numbers in each request packet means the required chunks' IDs of the peer.

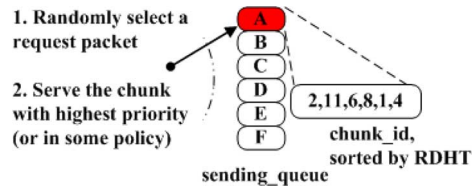

Fig. 7. Peer P's sending queue.

a narrow range compared with that of k^*, the range for which is [20], [35]. This indicates that there exists a unique evaluation metric that can be used to adapt simultaneously to M and L, simultaneously. This implies that if we can measure the data availability that each peer can sense and limit it to a specific range, we can effectively leverage the rf and *if(rf)* scheduling policies. With this in mind, we propose a new content-priority-aware scheduling policy that can easily be implemented in real systems.

IV. Implementation Issues

A. Buffer Space Concatenation Versus Dynamic Strategy Switch

Actually, to do this we need only modify slightly the buffer space concatenation approach; that is, we carefully adjust the size of k^* such that the data availability that each peer can sense can be maintained within a certain range. However, such adaptation is inflexible: it could make it difficult to respond quickly to content bottlenecks, because when a peer detects insufficient data availability, all it can do is to change the size of its rf part and wait passively for a rare chunk from a neighbor for it to exchange with its neighbors during the next scheduling time. The shortcoming of this approach is that if its neighbors cannot support useful data availability, it is useless for that peer to adjust k^* to improve its own data availability. To avoid this dilemma, our idea is to view each peer's neighbors as sensors that monitor the status of data availability, and to leverage the two policies according to the availability detected by the neighbors at different times. This approach is detailed below.

In general, in practice (as shown in Fig. 6), each peer periodically encapsulates the information on its required chunks into several request packets, in which each packet could contain one or more desired chunk IDs, and then delivers them to its different neighbors for the chance to be served. Our design will, by default, cause each request packet from its neighboring peers to be served in an *if(rf)* fashion. That is, when a peer receives a request packet, it first fetches the highest-priority chunk to transmit (in an RD sense) (as shown in Fig. 7). In our design, peers that serve as neighbors may reject this service style. That is, once a peer serving as a neighbor detects the potential shortage in data availability in the near future, that is, the number of available exchanging chunks is less than a threshold, it may immediately perform rf to serve all request packets to increase the content diversity of its neighbors and to prevent content bottlenecks. Thus, among the request packets delivered by a peer at scheduling time, some are request packets served in an rf fashion and some served in an *if(rf)* fashion. We use such a dynamic strategy switch to leverage the two policies.

B. Threshold Setting

The threshold value is the key to the proposed dynamic strategy switch. Fortunately, based on our abstract stochastic models, we have a proper range of data availability, approximately 2. When implementing our approach in practice, we use the following modifications.

Here, we consider the average download rate that each peer can measure as its capability in the abstract stochastic model. With the above information as well as our abstract stochastic models, we are able to set the value of the threshold in real systems. Suppose the average download rate that a peer measures is G and that each peer performs chunk scheduling every τ sec. Then the threshold is set to $2 * G * \tau$ chunks.

C. ϕ Selection

Although the above discussion and theoretical analysis are based on $\phi = rf$, there are many possible options that can be used to design the proposed dynamic strategy switch. Since the debate is still open as to which *cd-oriented* scheduling policy is better, we do not discuss this issue. For a fair comparison with existing content-priority-aware scheduling (such as the work in [30]), we use $\phi = \mathrm{RAND}$ to implement our dynamic strategy switch in the following evaluations.

V. Simulation Environment and Settings

The simulation environment used in this work is a modification of P2Pstrmsim[3] [2], [17], which has been widely used by other groups for P2P streaming-related research.

A. Environment Description

Members: The system is composed of peers, the tracker server, and the streaming server. Peers, which not only receive streaming content but also store and forward streaming content to other participants, are participants in a P2P streaming system. The tracker maintains a list of peers which participate in a specific channel or in the distribution of a streaming file, and also answers queries from peers for neighbor lists. The streaming server packetizes pre-encoded video streams into a sequence of 1040-byte chunks (including 40 byte headers) and disperses them into the system in a live fashion. The server's upload bandwidth is 2 Mb/s. To simulate the bandwidth heterogeneity of the peers, we used three different typical DSL nodes with upload capacities of 1 Mb/s, 384 kb/s, and 128 kb/s, and download capacities of 3 Mb/s, 1.5 Mb/s, and 768 kb/s, respectively. The fractions of different types of peers was set to 0.15, 0.39, and 0.46, respectively, and average upload and

[3]http://media.cs.tsinghua.edu.cn/~zhangm/

download capacities were set to 358.64 kb/s and 1.388 Mb/s, respectively.

Entry Phase: When a peer enters the system, it contacts the tracker and requests a neighbor list. For simplicity, we assume that the tracker gives the complete list of online peers to the peer. After obtaining the neighbor list, the peer randomly selects several peers (this could include the streaming server) as its neighbors and establishes neighbor relationships with them. After that, the peer periodically receives and sends buffermap packets from and to its neighbors. The buffermap packet is used to indicate which of the peer's chunks are cached and available in its streaming buffer. Then, the peer performs a chunk scheduling mechanism to retrieve chunks of interest from its neighbors, and waits for the first arrival chunk from the neighbors to initialize the start-up procedure. After a start-up delay, say $D = 32$ s, the peer commences playback, after which it is able to continue without interruption until the end of the simulation time. Any chunk that is received after the playback deadline is regarded as useless (or lost) and could result in serious error propagation. The playback speed of each peer is $F = 30$ fps and is semi-synchronized to the playback speed of the source peer.

Scheduling Protocol: In this simulation environment, each peer assumes the role of both video content receiver and supplier.

As a receiver, the peer periodically, say $\tau_r = 0.5$ s, checks its request window and determines which chunks should be requested. As shown in Fig. 6, there are four types of chunks in the request window: 1) (blue) chunks which have been stored in the streaming buffer, 2) (gray) chunks which have been requested but not yet received, 3) (green) chunks which are desired in the *request window* but not yet available in the neighbors, and 4) (red) chunks which are desired in the *request window* and available in the neighbors, but have not yet been requested. Only red chunks are considered in the scheduling process. Here we denote by π the set of the chunks: this is the data availability that a peer measures.

After the peer is aware of π, these chunks are encapsulated into several request packets according to a chunk scheduling policy. When a requested chunk does not arrive after the request has been sent out for a period of time, say $\tau = 0.5$ s, and is still in the request window, it will be included into the π again for next time request. Any absent chunk that falls out of the request window will not be requested further.

As a supplier, the peer periodically, say $\tau_s = 0.33$ s, checks its sending queue and determines which request packet chunk should be served first. As shown in Fig. 7, each peer collects request packets issued from its neighbors, and stores them in a sending queue. Each time, each peer randomly picks up a request packet and then chooses a chunk to serve according to the scheduling policy. Since each peer's upload bandwidth is different, the maximal number of chunks that each peer can serve within τ_s seconds is different. In the simulation environment, we assign a time-to-live (TTL) to each request packet. If the supplier cannot serve all chunks included in a request packet within the TTL, it simply removes this request packet from the sending queue and will not serve this request packet; this is to guarantee that each served chunk is able to arrive at the destination peer before the playback deadline.

Overlay: In the simulation environment, any pair of online peers may establish a bidirectional neighbor relationship. To control the overhead due to buffermap or request packets, we restrict the maximal number of neighbor relationships that each peer can maintain to a fixed number, say $m = 15$. On the other hand, we also enable each peer to replace its neighbors when some of its neighbors' contributions are too low. For this, each peer periodically replaces its lowest-contributing neighbor with some probability $p = 0.5$. In addition, we assume the end-to-end bottleneck is at the access links and not in the core network, in accordance with existing commercial swarm-based P2P systems [1]–[5] and academic research [9]–[22]. The round-trip-time (RTT) between every pair of peers p and q, denoted as rtt_{pq}, is taken into account and set as in [8], [17].

B. Video Trace and Metrics

We concatenated different types of CIF video sequences, including high-motion (e.g., Foreman and Football) and low-motion (e.g., Akiyo and News) video sequences, to generate approximately 1000 s of test video sequences. The concatenated video sequence was encoded using the H.264 encoder (JM16.0 [7]) with $F = 30$ fps. To achieve consistent video quality, we disabled the JM16.0 rate control and fixed the quantization parameters (QP) for I, P, B frames in encoding. The resulting variable-bitrate video had an average total bitrate of 328 kb/s, and its quality was 34.69 dB on average.

In our experiments, each chunk in the video stream was packetized in equal sizes and only frames with the same frame_type were combined for packetization. Here, we used the one-order rate-distortion hint track (RDHT) mechanism [24]–[26] to assign a finer importance to each video chunk; this can be computed as $\mathrm{RDHT}(i) = \mathrm{PSNR}_{w/}(i) - \mathrm{PSNR}_{w/o}(i)$ where i is chunk ID;$\mathrm{PSNR}_{w/}(i)$ and $\mathrm{PSNR}_{w/o}(i)$ are the average PSNRs of the reconstructed video sequences with and without chunk$i$. The importance of chunk i increases with $\mathrm{RDHT}(i)$.

Two metrics are used to evaluate performance: the *delivery ratio* the ratio of the number of chunks that actually arrive before the playback deadline to the number of chunks that should arrive before playback deadline; and PSNR (dB) the rendered video quality compared with the raw video sequence. We used FFmpeg [6] for decoding. When a frame gets lost, FFmpeg uses its default error concealment tool to minimize the error propagation effect.

VI. Experiment Results

Here, we use the simulation environment in Section V to validate the robustness of DSW to different system parameters such as start up latency and population size, and demonstrate the adaptability of DWS to fluctuating video streaming rates as compared with the following approaches.

- RAND: Peers always serve the chunks in each request packet in a random fashion.
- IF: Peers always serve the chunk with highest RDHT.

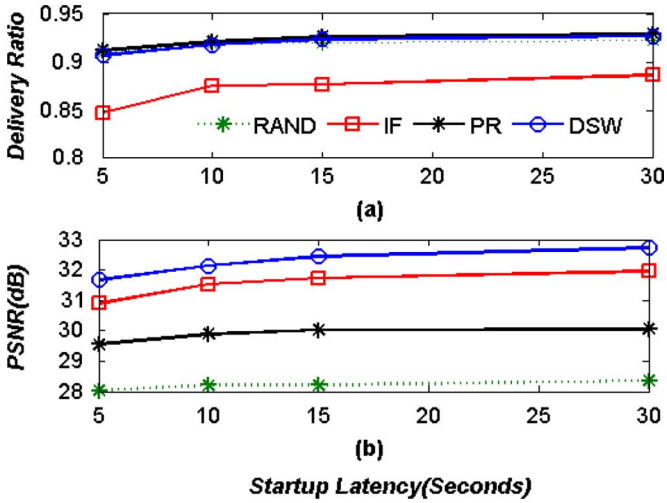

Fig. 8. (a) Delivery ratios for different startup latencies. (b) PSNR for different startup latencies.

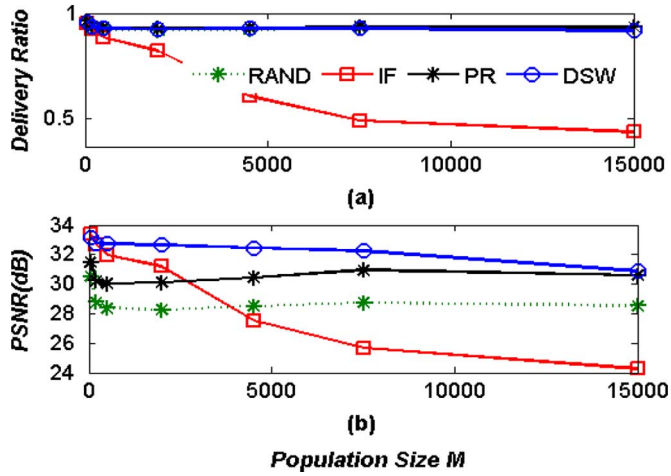

Fig. 9. (a) Delivery ratio for different peer counts. (b) PSNR for different peer counts.

- PR (the prioritized random scheduling in [30]): With this approach, π is partitioned into a *regular set* and a *probing set*. The chunks in the regular set, which are regarded as equal-important, are more important and are served with higher priority; the chunks in the probing set are less important and are served only when the suppliers have surplus upload bandwidth allocated to those chunks. Note that although chunks in the regular set are served in a RAND fashion, chunks in the probing set are served in an IF fashion. The size of the *regular set* is determined by $\#(\pi) \bullet \lfloor (R)/(\mathrm{BW}) \rfloor$, where BW[4] is the aggregate available bandwidth, R is the video streaming rate, and $\#(\pi)$ is the amount of data availability.

To ensure that all peers playback the same part of the given video sequence, all peers are forced to join the session initially and do not quit after they join the session. Unless otherwise stated, all settings follow those in Section V.

A. On Impact of Startup Latency

Startup latency is an important system parameter for swarm-based P2P streaming systems. In general, greater startup latencies lead to less chunk loss before the playback deadline. To investigate the impact of different startup latencies, we take $M = 500$ as an example and show in Fig. 8 several delivery ratios and PSNR plots.

First, we can see that IF yields a poor delivery ratio for small startup latencies; this is consistent with Result 2 in Section II. We see that the PR delivery ratio outperforms that of IF, because in comparison with IF, PR has a larger regular set, which supports greater content diversity for swarming. Thus, compared with IF, PR suffers from less content bottleneck and better utilizes each peer's available bandwidth. In addition, PR yields a delivery ratio comparable even to RAND. This because PR maintains a regular set for swarming, which is similar to the case with $L = 2$ in Section II. As indicated in the numerical results, the case of $L = 2$ is not sensitive to content bottleneck. In addition, the coverage of PR's regular set is larger than that of ρ_0 for $L = 2$: the PR delivery ratio is thus similar to that for RAND. As shown, DSW also yields a delivery ratio comparable to RAND. This validates the feasibility of our data-availability driven approach.

In Fig. 8(b), we further investigate the corresponding PSNR of Fig. 8(a). We see that although the IF delivery ratio is worse than that of PR, its PSNR gain is much better than that of PR. This is because 1) small population sizes lead to less content bottleneck, shrinking the delivery ratio gap between IF and PR; 2) IF essentially protects chunks with higher RDHT. This implies that the delivery ratio gap is mostly due to the loss of low-priority chunks. Thus, on the whole, the PSNR gain of IF is higher than that of PR.

Last, we can see that our DSW consistently outperforms other approaches by several dBs. This is because the proposed DSW, when choosing when to switch between IF and RAND, is driven by the data availability measured by each peer. In general, the greater the startup latency, the more measurable the data availability. This implies that DSW adapts well to different startup latency settings to strike a good balance between content diversity and content importance, that is, the greater the startup latency, the more frequently IF can be performed. In contrast, IF can be performed infrequently.

Therefore, compared with IF and PR (no adaptability to startup latency) our data availability driven DSW yields performance gains in terms of either delivery ratio or PSNR.

B. On Impact of Population Size

We further investigate the delivery ratio and PSNR for various population sizes (from $\mathrm{M} = 50$ to $\mathrm{M} = 15\,000$) for the different scheduling strategies in Fig. 9.

We see that when population sizes are small, both the delivery ratio and PSNR of all approaches are better. Clearly, the PSNR of IF and DSW are the best, almost the same in fact; the second best is PR and the worst is RAND. This is because for very small

[4]The estimation of aggregated available bandwidth in a P2P system is not trivial: it depends not only on network core congestion but also on the data availability of each peer [32]. Here, for simplicity, we assume that the aggregated available bandwidth is the sum of the average upload bandwidth that its neighbors can supply, that is, $\mathrm{BW} = \sum o_i/m_i$ where o_i is the upload bandwidth of neighbor i and m_i is the number of neighbors of neighbor i.

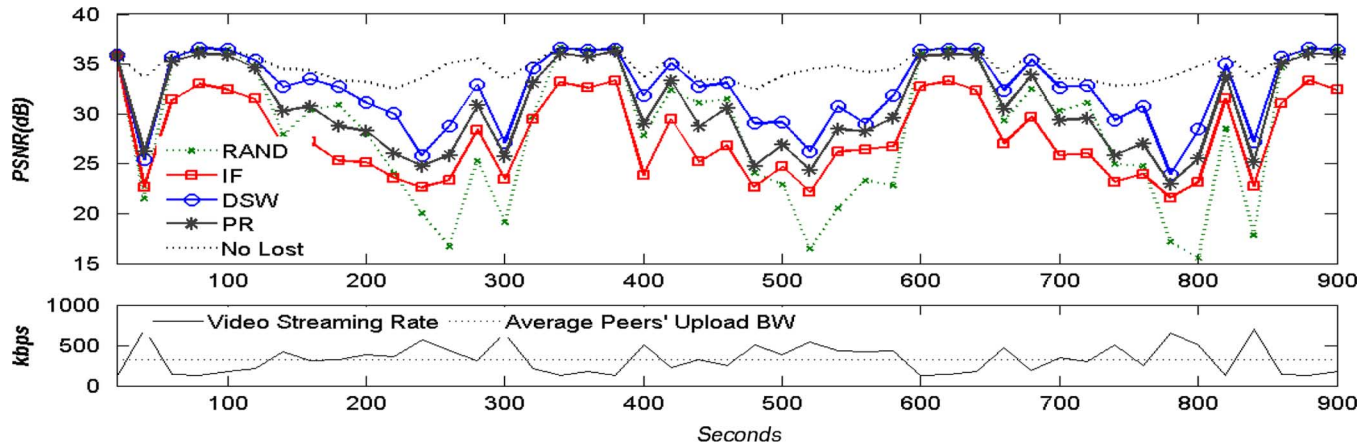

Fig. 10. PSNR over time.

population sizes (say $\mathrm{M} = 50$), the performance of each peer is easily influenced by the server's capability. Since the server's upload bandwidth is set to 2 Mb/s in our simulation, which is several times the video streaming rate, both data availability and available bandwidth are sufficient. Under such conditions, all approaches work well. However, when population size is increased to $\mathrm{M} = 200$, the server's impact disappears. As shown, there is a rapid drop in both delivery ratio and PSNR. We can see that for populations larger than $\mathrm{M} = 3\,000$, the IF delivery ratio rapidly falls, due to content bottleneck. We also observe that the IF PSNR becomes worse even than that of PR.

It is interesting that although PR shows poor PSNRs for small scales, increased populations (from $\mathrm{M} = 500$ to $\mathrm{M} = 7500$), yield increased PSNRs. This is a server-side effect and can be explained as follows. In general, in pure swarm-based P2P streaming systems, the server's duty is to diffuse at least a copy of video streaming to its neighboring peers (of course, the more powerful the server, the more copies it can send out). To better utilize each peer's available bandwidth, the server still follows the content diversity principle to disperse video content to each neighboring peer, after which the peers reciprocate among themselves with their received video content. Hence, under such conditions, if strict IF is not performed (in comparison with PR of course), it is difficult for each peer to concentrate on receiving high-RDHT chunks. Hence the PSNR of PR is not guaranteed. However, as the population increases, the server effect disappears and the PR PSNR rises.

When the population is increased to $\mathrm{M} = 15\,000$, we observe that although the delivery ratios of PR and DSW remain high and the DSW PSNR still outperforms that of PR, their PSNRs decline. We explain this as follows.

Recall the buffer concatenation approach and the discussion on the smallest k^* in Fig. 5. We know that the larger M is, the larger k^* is. This indicates that large population sizes make it difficult to guarantee the rate-distortion performance for any content-priority-aware chunk scheduling approach because weight of if becomes small. Therefore, when population keeps increasing, the PSNR of DSW and PR converge theoretically. Nevertheless, they still can provide better rate-distortion performance than RAND can do in mostly situations.

C. On Impact of Streaming Rate Fluctuation

In Fig. 10, we plot the average PSNR and video streaming rate over time to demonstrate the adaptability of different scheduling mechanisms to fluctuating video streaming rates for the case of 4500 peers. Each plot in the figure is the average PSNR of 600 frames. Clearly, our approach outperforms other approaches consistently, whether video traffic overwhelms the average peer's upload bandwidth or not.

We see that the PSNR of IF is still quite poor even for bandwidth-sufficient regimes (320–380 s and 580–650 s). This is because IF is inherently vulnerable to a serious content bottleneck effect for large populations. Thus, in such situations, if peers still apply IF, they waste system-wide available bandwidth. On the other hand, we can see that when bandwidth supply is insufficient (467–575 s and 200–300 s), RAND performs poorly. This is because RAND offers no protection for high RDHT chunks. Hence a well-designed chunk scheduling mechanism should simultaneously take into account both content diversity (among peers) and content importance. As shown, PR yields consistent PSNR improvement for RAND and IF, and thus could be a way to simultaneously achieve these two objectives. However, compared with the proposed DSW, it does not work well when the video streaming rate exceeds the average peer's upload bandwidth.

This is because when video streaming rate is large, the server generates more chunks than usual, and thus injects more diverse chunks into the system than usual. As we have discussed, this case is very disadvantageous to PR because it is difficult for peers to concentrate the received chunks in regular set. Thus, the PSNR of PR is poor.

However, DSW, by default, performs IF, and switches to RAND only in the case of serious content bottleneck (that is, data availability less than a threshold). Therefore, when video streaming rate is large, DSW adaptively performs IF more frequently. Thus, compared with PR, DSW strikes a better balance between content diversity and content importance, yielding greater PSNR gains.

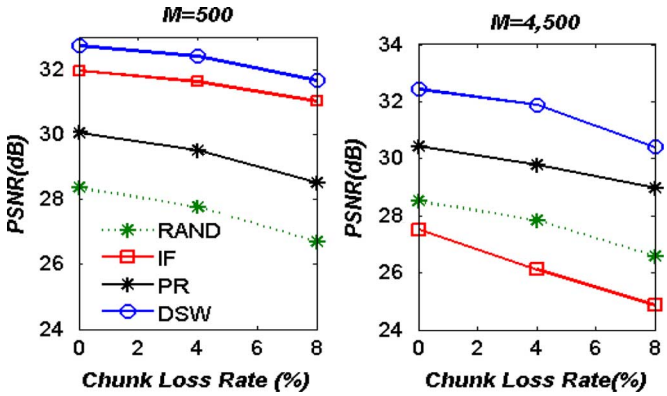

Fig. 11. PSNR for different peer counts and chunk loss rate.

D. On Impact of Chunk Loss

The above experimental results are based on the assumptions that the end-to-end bottleneck is at the access links and not in the core network, and that peer churn is ignored. In truth, though, both exercise substantial influence over the received video quality of each peer. In these experiments, we simulated these effects by adding random chunk loss to each peer's uplink to demonstrate the resilience of DSW. Fig. 11 shows the PSNR performance for various chunk loss rates under $\mathrm{M} = 500$ and $\mathrm{M} = 4\ 500$. The trend is consistent with the results in Fig. 9(b); that is, IF yields greater PSNR gains in small scale, but PR works better in large-scale conditions. As shown in Fig. 11, DSW works well even with different chunk loss rates and population sizes.

VII. Related Works

Although there have been several studies that take into account rate-distortion characteristics for swarm-based P2P streaming, they focus mostly on layered P2P streaming. We briefly survey the relevant literature here. In [31], [37], [44] each chunk is assigned an empirical score as a function of the chunk's rarity, emergency, and importance (in an RD sense). Then the peers diffuse higher-priority chunks to achieve a better rate-distortion benefit. Such an approach can be categorized as $if(\phi)$. As shown in previous work [29], however, this approach does not scale. In [38], a heuristic buffer space concatenation approach is proposed where the request window is partitioned into three disjoint regions for different functionalities. Each region plays different roles for better video quality. Although this work seems good at small scales, many settings are empirical and heuristic. The work in [30], [39], which is the extended version of [31], introduces the concept of prioritized random regions to solve the scalability problem in [31]. Although the idea is simple and solves the scalability problem, it is still a heuristic and cannot guarantee rate-distortion performance even when finer rate-distortion characteristics are considered. To solve the problem, we introduce the concept of data availability and show theoretically that there exists a sweet data availability that can be used effectively to control the tradeoff between content diversity and content importance. Simulation results show that the proposed DSW outperforms PR significantly by several dBs.

VIII. Conclusion

In this work, we modify the abstract model in [18] to analyze and compare the *cd-oriented* and *if-oriented* chunk scheduling policies. Based on the simplified model, we successfully devise a way, from the data availability domain, to leverage the two scheduling policies. The proposed data-availability driven DSW not only retains the advantages of the two scheduling policies but also adapts to different system parameters. Simulation results show that DSW outperforms other approaches and significantly enhances the received video quality of each peer. In particular, the proposed scheduling method is quite simple: it requires only the integration of a simple data-availability detector into each peer. We believe that it is applicable to existing commercial systems.

Appendix

$p_{rf}(i)$: By intention, a peer using rf first pulls the chunk which has the fewest number of copies in the system. From (1), we know that $p(i)$ is an increasing function of i. Hence, rf conceptually selects chunks with priority $B(0) \succcurlyeq B(1) \succcurlyeq \ldots \succcurlyeq B(n-1)$. Hence, $\mathcal{A}$ in $s_{rf,\mathcal{A}}(i)$ is set to the buffer positions before $B(i)$, which is denoted as $\mathcal{B}(j<i)$, and thus is modeled as the probability that the selected peer cannot provide any desired chunk before $B(i)$

$$s_{rf,\mathcal{B}(j<i)}(i) = \left(1-\frac{1}{M}\right) \prod_{j\in\mathcal{B}(j<i)} (p_{rf}(j) + (1-p_{rf}(j))^2). \tag{A1}$$

Substituting (A1) into (1) yields the following difference equation for $p(i)$:

$$p_{rf}(i+1) = p_{rf}(i) + (1-p_{rf}(i))p_{rf}(i) \times \left(1-\frac{1}{M}\right) \prod_{j\in\mathcal{B}(j<i)} (p_{rf}(j) + (1-p_{rf}(j))^2) \quad i = 1,\ldots,n-2 \text{ and } p(0) = u/M. \tag{A2}$$

$p^{\rho_l}_{\phi}(i)$: From the above we know that $\mathcal{A}$ in $s_{rf,\mathcal{A}=\mathcal{B}(\rho_l)}(i)$ is reduced to $\mathcal{B}(j<i) \cap \mathcal{B}(\rho_l)$. Thus, the corresponding $p^{\rho_l}_{rf}(i)$ can be given by

$$\begin{aligned} p^{\rho_l}_{rf}(i+1) &= p^{\rho_l}_{rf}(i) + \left(1-p^{\rho_l}_{rf}(i)\right) p^{\rho_l}_{rf}(i) \\ &\times \prod_{i=0}^{l-1} \prod_{j\in\mathcal{B}(\rho_i)} \left(p^{\rho_i}_{\phi}(j) + \left(1-p^{\rho_i}_{\phi}(j)\right)^2\right) \\ &\times \left(1-\frac{1}{M}\right) \prod_{j\in\mathcal{B}(j<i)\cap\mathcal{B}(\rho_l)} \left(p^{\rho_l}_{rf}(j) + \left(1-p^{\rho_l}_{rf}(j)\right)^2\right) \\ &\quad i = 1,\ldots,n-2 \text{ and } p(0) = u/M. \end{aligned} \tag{A3}$$

After obtaining each $p^{\rho_l}_{\phi}(i)$, we compute $p_{if(\phi=rf)}(i)$ using (3).

REFERENCES

[1] CoolStreaming [Online]. Available: http://www.coolstreaming.us/

[2] GridMedia [Online]. Available: http://www.cctv.com/html/Gridmedia_index.html

[3] PPTV [Online]. Available: http://www.pptv.com/

[4] PPStream [Online]. Available: http://www.ppstream.com

[5] UUSee [Online]. Available: http://www.uusee.com

[6] FFmpeg [Online]. Available: http://ffmpeg.org/

[7] JM16.0 [Online]. Available: http://iphome.hhi.de/suehring/tml/

[8] Meridian [Online]. Available: http://www.cs.cornell.edu/People/egs/meridian/data.php

[9] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P Vod systems," in *ACM SIGCOMM*, Seattle, WA, Aug. 2008, pp. 375–388.

[10] X. Hei, Y. Liu, and K. W. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 10, pp. 1640–1654, Oct. 2007.

[11] D. Ren, Y. H. Li, and S. G. Chan, "On reducing mesh delay for peer-to-peer live streaming," in *IEEE INFOCOM*, Phoenix, AZ, Apr. 2008, pp. 1732–1740.

[12] M. Piatek, A. Krishnamurthy, A. Venkataramani, R. Yang, D. Zhang, and A. Jaffe, "Contracts: Practical contribution incentives for P2P live streaming," in *USENIX NSDI*, San Jose, CA, Apr. 2010, pp. 81–94.

[13] A. Habib and J. Chuang, "Incentive mechanism for peer-to-peer media streaming," in *Proc. 12th IEEE Int. Workshop IEEE Qual. Service*, Montreal, Canada, Jun. 2004, pp. 171–180.

[14] F. Wang, J. Liu, and Y. Xiong, "Stable peers: Existence, importance, and application in peer-to-peer live video streaming," in *Proc. IEEE 27th Conf. Comput. Commun.*, Phoenix, AZ, Apr. 2008, pp. 2038–2046.

[15] X. Zhang, J. C. Liu, B. Li, and P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient media streaming," in *Proc. IEEE 27th Conf. Comput. Commun.*, Miami, FL, Mar. 2005, vol. 3, pp. 2102–2111.

[16] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver-driven mesh-based streaming," in *Proc. IEEE 27th Conf. Comput. Commun.*, Anchorage, AK, May 2007, pp. 1415–1423.

[17] M. Zhang, Q. Zhang, and S. Q. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 1678–1694, Sep. 2007.

[18] Y. Zhou, D. M. Chiu, and J. C. Lui, "A simple model for analyzing P2P streaming protocols," in *Proc. IEEE Int. Conf. Netw. Protocols*, Beijing, China, Oct. 2007, pp. 226–235.

[19] S. Shakkottai, R. Srikant, and L. Ying, "The asymptotic behavior of minimum buffer size requirements in large P2P streaming networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 928–937, May 2011.

[20] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: Optimal performance trade-offs," in *Proc. ACM SIGMETRICS*, Annapolis, MD, Jun. 2008, pp. 325–336.

[21] C. Liang, Y. Guo, and Y. Liu, "Is random scheduling sufficient in P2P video streaming?," in *28th Int. Conf. Distrib. Comput. Syst.*, Beijing, China, Jun. 2008, pp. 53–60.

[22] B. Q. Zhao, J. C. Lui, and D. M. Chiu, "Exploring the optimal chunk selection policy for data-driven P2P streaming systems," in *Proc. IEEE 9th Int. Conf. Peer-to-Peer Comput.*, Tarragona, Spain, Sep. 2009, pp. 271–280.

[23] J. Chakareski, J. Apostolopoulos, S. Wee, W. Tan, and B. Girod, "Rate-distortion hint tracks for adaptive video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 10, pp. 1257–1269, Oct. 2005.

[24] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[25] Z. Miao and A. Ortega, "Expected run-time distortion based scheduling for delivery of scalable media," in *Proc. Int. Packet Video Workshop*, Pittsburgh, PA, 2002, pp. 687–693.

[26] J. Chakareski and J. Apostolopoulos, "Rate-distortion optimized distributed schediling of multiple video streams over shared communication resources," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 207–218, Apr. 2006.

[27] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[28] C. Y. Chang, C. F. Chou, T. C. Chiu, Y. M. Chen, and Y. C. Chou, "Conducting rate-distortion optimization in data-driven P2P video streaming," in *ACM SIGCOMM*, Barcelona, Spain, 2009.

[29] C. Y. Chang, C. F. Chou, and M. H. Chen, "Striking the balance between content diversity and content importance in swarm-based P2P streaming system," in *Proc. IEEE 13th Int. Conf. High Performance Comput. Commun.*, Banff, Canada, Sep. 2011, pp. 653–660.

[30] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "LayerP2P: Using layered video chunks in P2P live streaming," *IEEE Trans. Multimedia*, vol. 11, no. 7, pp. 1340–1352, Jul. 2009.

[31] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in P2P live streaming," in *Proc. SIGCOMM P2P-TV*, Kyoto, Japan, Aug. 2007, pp. 311–316.

[32] D. C. Tomozei and L. Massoulié, "Flow control for cost-efficient peer-to-peer streaming," in *Proc. IEEE INFOCOM*, San Diego, Mar. 2010, pp. 1–9.

[33] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Portland, OR, Aug. 2004, pp. 367–378.

[34] B. Fan, D. Andersen, M. Kaminsky, and K. Papagiannaki, "Balancing throughput, robustness, and in-order delivery in P2P VoD," presented at the ACM CoNEXT, Philadelphia, PA, Jun. 2010.

[35] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," in *Proc. IEEE ICDCS*, Vienna, Austria, Jul. 2002, pp. 363–371.

[36] A. P. C. da Silva, E. Leonardi, M. Mellia, and M. Meo, "A bandwidth-aware scheduling strategy for P2P-TV systems," in *Proc. 8th Int. Conf. Peer-to-Peer Comput.* , Tarragona, Spain, Sep. 2008, pp. 279–288.

[37] K. L. Hua, G. M. Chiu, H. K. Pao, and Y. C. Cheng, "An efficient scheduling algorithm for scalable video streaming over P2P networks," *J. Comput. Netw.*, vol. 57, no. 14, pp. 2856–2868, Oct. 2013.

[38] X. Xiao, Y. C. Shi, Q. Zhang, J. H. Shen, and Y. Gao, "Toward systematical data scheduling for layered streaming in peer-to-peer networks: Can we go farther?," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 685–697, May 2010.

[39] H. Hu, Y. Guo, and Y. Liu, "Peer-to-peer streaming of layered video: Efficiency, fairness and incentive," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 8, pp. 1013–1026, Aug. 2011.

[40] D. Xu, C. Rosenberg, S. Kulkarni, and H.-K. Chai, "Analysis of a CDN-P2P hybrid architecture for cost-effective streaming distribution," *Multimedia Syst. J.*, vol. 11, no. 4, pp. 585–599, 2006.

[41] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid CDN-P2P: Why limelight needs its own red swoosh," in *Proc. ACM NOSSDAV*, Braunschweig, Germany, May 2008, pp. 75–80.

[42] H. Yin, X. N. Liu, T. Y. Zhan, V. Sekar, F. Qiu, C. Lin, H. Z. , and B. Li, "Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with livesky," in *Proc. ACM MM*, Beijing, China, Oct. 2009, pp. 25–34.

[43] Y. Zhou, D. M. Chiu, and J. C. S. Lui, "A simple nodel for chunk scheduling strategies in P2P streaming," *IEEE/ACM Trans. Network*, vol. 19, no. 1, pp. 42–54, Feb. 2011.

[44] J. Chakareski and P. Frossard, "Utility-based packet scheduling in P2P mesh-based multicast," in *Proc. SPIE Vis. Commun. Image Process.*, San Jose, CA, Jan. 2009, p. 72571S1.

Chun-Yuan Chang (M'06) is currently working toward the Ph.D. degree at the Department of Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei City, Taiwan.

His current research interests are video coding and P2P content distribution.

Cheng-Fu Chou received the M.S. and Ph.D. degrees from the University of Maryland, College Park, MD, USA, in 1999 and 2002, respectively.

After his graduation, he joined the Computer Science and Information Engineering Department at the National Taiwan University. He has been a visiting scholar in the Computer Science Department at the University of Southern California. His current research interests are in distributed multimedia systems, peer-to-peer computing, wireless networks, and their performance evaluation.

Kwang-Cheng Chen (M'89–SM'94–F'07) received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, MD, USA, in 1987 and 1989, respectively, all in electrical engineering.

From 1987 to 1998, he worked with SSE, COMSAT, IBM Thomas J. Watson Research Center, and the National Tsing Hua University, in mobile communications and networks. Since 1998, he has been with National Taiwan University, Taipei, Taiwan, and is the Distinguished Professor and Associate Dean for academic affairs in the College of Electrical Engineering and Computer Science, National Taiwan University. He is a SKKU Fellow Professor, Korea (2013–2014). His research interests include wireless communications and network science. He has authored and co-authored around 300 technical papers and more than 20 granted U.S. patents.

Dr. Chen has been actively involved in the organization of various IEEE conferences as General/TPC Chair/co-Chair. He has served editorship with a few IEEE journals and many international journals and served various positions in IEEE. He also actively participates and has contributed essential technology to various IEEE 802, Bluetooth, and 3 GPP wireless standards. He has received a number of awards including 2011 IEEE COMSOC WTC Recognition Award and co-authored a few award-winning papers published in the IEEE ComSoc journals and conferences.