Efficient Search and Scheduling in P2P-based Media-on-Demand Streaming Service

Huicheng Chi, Qian Zhang, *Senior Member*, Juncheng Jia and Xuemin (Sherman) Shen, *Senior Member*

Abstract— **We are interested in providing a media-on-demand streaming service to a large population of clients using a peer-to-peer approach. Since the demands of different clients are asynchronous and the contents of clients' buffers are continuously changing, finding partners with expected data and collaborating with them for future content delivery are very important and challenging problems. In this paper, we propose a generic buffer-assisted search (BAS) scheme to improve partner search efficiency by reducing the size of index overlay. We have also developed a novel scheduling algorithm based on deadline-aware network coding (DNC) to fully exploit network resources by dynamically adjusting the coding window size. Extensive simulation results demonstrate that BAS can provide a faster response time with less control cost than the existing search methods, and DNC improves the network capacity utilization and provides high streaming quality under different network conditions.**

Index Terms— **Peer-to-peer, Media-on-Demand (MoD), buffer-assisted search, deadline-aware network coding.**

I. Introduction

With the widespread deployment of broadband access, Media-on-Demand (MoD) streaming on the Internet is recently receiving increasing attention. In an MoD streaming service, music and movies can be delivered to asynchronous users with low delay and VCR-like operation support (e.g., pause, fast-forward, and fast-rewind). However, it is a big challenge to provide streaming to a large population of clients due to the limited server capacity and little deployment of IP multicast [17]. Peer-to-peer (P2P) technology is one of the more promising solutions for providing streaming service over large-scale Internet users [4, 5, 7, 8, 13, 14, 20, 21, 23, 24].

Fig. 1 depicts the general framework of a typical P2P-based MoD system. In such a system, cooperative peers[1] are organized into an overlay network via unicast tunnels. The streaming content is split into a sequence of segments, each of which is a smallest playable unit, and the server distributes these segments among clients of asynchronous demands. Each client caches a limited number of segments around its current *play offset*, which is the in-sequence index of the playing segment. The client exchanges its available segments with *partners*, who have close play offset and thus can provide the expected data with high probability. The users in the MoD system need to search for such partners, when joining or taking a VCR operations. Since the users request the service asynchronously, and the contents buffered in one peer are continuously changing, an additional index structure is needed for the partner search upon peer joins or VCR operations. Without losing generality, we assume all peers have identical playing speed, so their partner relationship will not change unless they leave, fail, or take VCR-jump. After locating the partners, the user collaborates with them to schedule the data transmission and exchange the content. Therefore, *partner search* and *multi-partner* scheduling are the two main components in the P2P-based MoD system. It is noticed that the users form two types of overlay networks in the MoD system: the *index overlay* for partner search and the *data overlay* for media transmission. To avoid confusion, the term "*index neighbor*" denotes the neighbors in the index overlay, while "*data neighbor*" or "*partner*" represents the neighbors in the data overlay.

It is very challenging to develop an efficient partner search structure among a large population of collaborative peers. Due to scalability concerns, the search overlay should have a distributed structure with sub-linear search efficiency. Today, some distributed structures have been proposed with logarithmic search efficiency, for example, AVL tree [23], distributed hash table (DHT) [21], and skip-list [20]. In these structures, all peers are sorted by the play offset and maintained in the index overlay. However, indexing all the peers incurs a non-trivial maintenance overhead, especially considering the big insertion, deletion, and rebalancing cost due to frequent VCR interaction and dynamic node joining or leaving in P2P-based MoD systems.

Multi-partner scheduling is another challenging task. Traditional cooperative scheduling schemes, such as smallest-delay algorithm [8] and pull-based gossip algorithm [22, 23], assign requests to partners based on the local neighborhood information, for example, content in the local buffer and the available bandwidth among partners. These schemes suffer from inefficient use of network resources in large and heterogeneous populations (we elaborate on the details in Section II.B). Some recently proposed schemes leverage network coding to improve the throughput utilization and facilitate the design of an optimal solution for static file download applications [10, 15]. With network coding, each peer (including the server) performs a linear combination on available segments and relays the combined segments to the neighbors. When a node receives enough linearly independent combinations, the original media can be reconstructed. However, it is difficult to apply network coding to the MoD system. First, since media play starts before all segments are downloaded, some segments may miss the playback deadline before being decoded, which causes severe performance degradation. Second, the peers

Manuscript received Dec 15, 2005; revised June 27, 2006. Part of Q. Zhang's work was supported by the National Basic Research Program of China (973 Program) under Grant No. 2006CB303000.

H. Chi, Q. Zhang, and J. Jia are with Department of Computer Science and Engineering, Hong Kong University of Science and Technology (e-mail: addison@cse.ust.hk, qianzh@cse.ust.hk, jiajc@cse.ust.hk).

X. Shen is with Department of Electrical and Computer Engineering, University of Waterloo (e-mail: xshen@bbcr.uwaterloo.ca).

Digital Object Identifier 10.1109/JSAC.2007.070112.

[1]User, client, peer and node are used interchangeably in this paper.

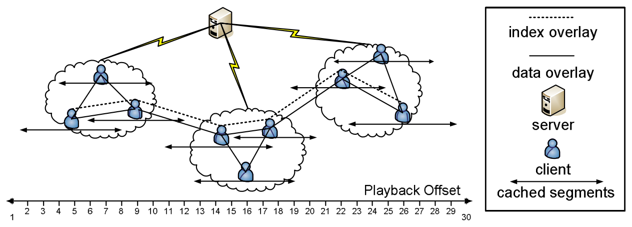


Fig. 1. Typical P2P-based MoD system.

have limited buffer capacity and maintain different parts of segments, so it is difficult for collaboration among partners due to the different ranges of encoded segments they own.

In this paper, we propose a novel P2P-based MoD system, where Buffer Assisted Search (BAS) structure and Deadline-aware Network Coding (DNC) scheduling are introduced to address the above challenges. As mentioned above, each client in an MoD system maintains a certain content buffer, it can be observed that there is buffer overlapping among different nodes, and removing the nodes whose buffer range is fully covered by other nodes does not reduce the total buffer coverage of the search structure. Therefore, we introduce BAS structure to exploit this buffer coverage redundancy so as to reduce the size of the index overlay. We also propose the DNC scheme for multi-partner scheduling. Since media segments have a play deadline, instead of encoding all available segments, the DNC scheme adjusts the *coding window* for each node, which is the number of segments to be encoded, based on its network condition and also play deadline so as to avoid segment miss.

Fig. 2 depicts the architecture of our proposed system, where the index overlay is used to look up partners for building connections in the data overlay upon new node arrival or VCR-operations. We denote the peers in the index overlay as *index peers* and the others as *non-index peers*. Upon node departure or failure, the system adjusts the connections in both the index overlay and data overlay. If the departing node is an index peer, it should find some non-index peers from its neighborhood to cover the remaining uncovered buffer range. In the data overlay, each peer calls DNC scheduling to exchange content with the partners.

By indexing a small subset of peers, BAS provides constant search efficiency and low control overhead. In addition, the BAS structure is generic and can be implemented based on existing data structures, for example, AVL-tree, DHT, and skip-list. Moreover, DNC improves the network throughput and reduces the total schedule time without missing the play deadline in high probability.

The rest of the paper is organized as follows. The background and motivation is reviewed in Section II. Section III presents the BAS structure. The DNC scheduling scheme is developed in Section IV. Simulation results are given in Section V, followed by the conclusions and future work in Section VI.

II. Background and Motivated Scenario

In this section, we briefly review the related works on MoD service and present two concrete examples that motivate our study.

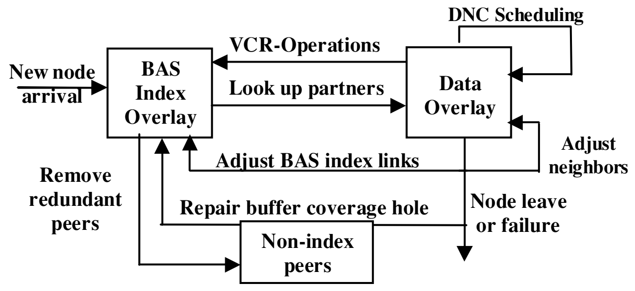


Fig. 2. Architecture of our MoD system.

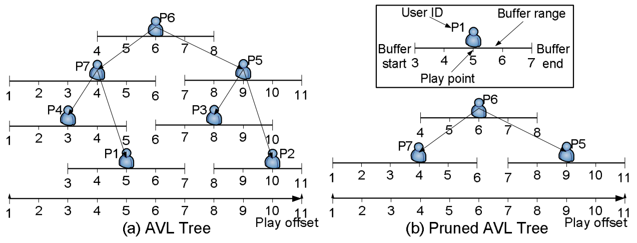


Fig. 3. A pruned example on an AVL tree.

There have been many application-layer solutions for MoD streaming, for example, application-layer-multicast (ALM) and P2P approaches. The broadband speed and massive buffer capacity at client side make these solutions readily deployable. These technologies have been well studied in the big-file download and live streaming applications. Originating from IP multicast, the ALM protocols often organize the users into a tree structure. Due to the single-parent nature, tree-based overlay is unbalanced and vulnerable. Thus many mesh-based and P2P proposals exploit multi-parents to balance the load and enhance the robustness, for example, mesh-based network (Bullet [14]), multiple-tree overlay (Split-Stream [2]), gossip partners (CoolStreaming [22]), as well as the schemes leveraging advanced source coding techniques such as layered coding (PALS [18]) and multiple description coding (CoopNet [16]). Our system utilizes multi-partner-based data delivering and targets the on-demand streaming, which handles asynchronous requests and VCR operations.

The P2P-based MoD system consists of two key parts: 1) partner search structure for accommodating the asynchronous requests and; 2) collaborative data scheduling among partners. In the following, we review the related works, respectively.

A. Partner Search

Recently a very active research area has been to develop the efficient search schemes for P2P-based MoD services [4, 5, 7, 8, 13, 14, 20, 21, 23]. CollectCast [14] and oStream [7] record the play progress of all the nodes in the centralized server. Though it is simple to implement and easy to manage, the server easily becomes overloaded in the presence of peers' frequent joining and leaving, and VCR operations. P2Cast [13], P2VoD [8], TAG [23], and DSL [20] logically organize all peers into a linear, tree, or skip-list structures. Given the play offset as the sorting key, these methods support random search and VCR interaction in a distributed manner. Each peer maintains a constant or logarithmic number of index neighbors and the search is performed by tracing the index connections hop by hop. Although some of them provide sub-linear search efficiency, the maintenance cost is not trivial since all peers need to be indexed in the structure. The frequent VCR operations and dynamic P2P environment further aggravate this maintenance overhead.

The next question is whether it is necessary to maintain all peers in the index structure to find all the potential partner candidates for each search. Recent studies show that having a small number of partners is sufficient for media streaming [8, 13, 20, 23]. Therefore, it is possible to prune the index structure without sacrificing the search performance. In the MoD system, each peer has a buffer to maintain content around its play offset and acts as a potential partner for the peers whose play offset lies within its buffer range. If a peer's buffer range is fully covered by other peers, removing it from the index structure will not cause much trouble as the peers who use this removed peer as a partner can still find other possible partners. Thus, in the ideal case those nodes with redundant buffer coverage can be safely removed from the index structure. Fig. 3 gives an example of how to illustrate the ideal case with pruning, where the AVL tree is used as the base structure. As shown in Fig. 3(a), *P5*, *P6* and *P7* collectively cover the playback offset range from 1 to 11, and *P1*, *P2*, *P3*, and *P4* are redundant. Thus after removing *P1*, *P2*, *P3*, and *P4*, only *P5*, *P6* and *P7* are indexed while the total buffer coverage is not affected, as shown in Fig. 3(b). We give an idea of how to figure out which peers should be removed to minimize the size of index overlay while maintaining the search effectiveness in Section III.

B. Cooperative Data Scheduling

The multi-partner scheduling for P2P MoD streaming is also difficult due to the limited bandwidth and the content heterogeneity of partners. Many heuristics have been proposed to address this issue, such as round robin, smallest-delay and pull-based gossip algorithm (PGA) [8, 13, 23]. PGA first counts the number of potential partners for each segment, and then schedules the segments one by one in the increasing order of this count. Among the multiple potential partners, it selects the one with the highest available bandwidth and enough available time (i.e., the remaining time before the deadline). Fig. 4(a) depicts an example of PGA. The streaming media is split into 6 segments *a*, *b*, *c*, *d*, *e*, and *f*. Peers *A*, *B*, *C* and the server form partner relationship, and each link can transfer at most one segment in one direction per time-unit. At Time 0 (T0), peer *A* gets *c* from the server; *B* gets *c* from the server and *b* from *A*; *C* gets *c* from the server, *b* from *A*, and *a* from *B*. After that, at T1, T2, and T3, all the three nodes *A*, *B*, and *C* have the same content and get *d*, *e*, and *f*, respectively from the server. Overall, PGA consumes 4 time-units and causes 12 traffic-units on the server. Though these heuristics are simple, their local decisions make inefficient use of network capacities. For example, the bandwidth among peer *A*, *B* and *C* has not been utilized since T1, while the server has to constantly provide data for the three peers.

Recently, network coding (NC) technologies have brought fundamentally new insights to the cooperative data scheduling problem [10, 15]. With NC, each node linearly combines all received segments with random coefficients before forwarding them, and the resulting encoding segment is the same in size as a single segment excluding a few bytes for coefficients. When the number of received combinations is no less than the size of *coding window* (*CW*), which is the number of segments in the combination, the media can be fully decoded with very high probability. Since a coding segment contains the information from multiple segments, it increases the probability of serving requests of partners and thus improves the network throughput utilization. Wang et al. proposed directly applying the linear NC into MoD streaming [20]. As an example shown in Fig. 4(b), at T0, *A* gets a linear combination among missing segments *c*, *d*, *e*, *f*, denoted as *c~f*, from the server; *B* gets *b~f* from the server and *b* from *A*; *C* gets *a~f* from the server, *a~b* from *A*, and *a* from *B*. At T1, *A*, *B* and *C* help each other with the received coding segments. At T2, *A*, *B* and *C* can recover the entire media. Overall, NC only consumes 2 time-units and generates 6 traffic-units on the server. Though NC improves the resource utilization, some segments may miss the play deadline since they are not playable until the last linear combination is obtained. For instance, at T1, *A* misses *c*, and this problem is further aggravated when the number of segments increases. Thus, the *CW* should be carefully decided instead of simply covering all available media at each node.

A better solution to address this segment-missing issue should consider more environmental parameters, for example, the play deadline and available bandwidth, when deciding the coding window. A simple and intuitive estimation of *CW* for each peer is the product of the number of partners that contains expected data and the remaining time-units before the most urgent play deadline. Fig. 4(c) depicts the deadline-aware case using this simple estimate. At T0, *A* has one partner and one time-unit before *c*'s deadline, so the *CW* is 1 and *A* gets *c* from the server; similarly, *B*'s *CW* is 2 and gets *b~c* from the server and *b* from *A*; *C* gets *a~c* from the server, *a~b* from *A* and *a* from *B*. At T1, *A* continuously gets *d* from the server; after recovering *b* and *c*, *B* has one partner, and the remaining time units before *d*'s deadline is 2, so it gets *d~e* from the server; similarly *C* gets *d~f* from the server. At T2, *A*, *B* and *C* exchange their data. At T3, peer *A* reconstructs the original media from *a*, *b*, *c*, *d* and *d~e*, *d~f*; similarly *B* and *C* decoded all segments. In summary, this deadline-aware solution only uses 3 time-units and costs 6 of the server's traffic-units. This not only reduces the schedule time and server overhead, but also avoids the situation where segments miss the play deadline as in the previous NC case. However, the simple estimation above may overvalue the *CW* when some partner can not utilize the full bandwidth due to its limited content, therefor we introduce a more elegant solution in Section IV.

In the next two sections, we describe our design schemes BAS for search and DNC for data scheduling, respectively.

III. Buffer-Assisted Search Overlay

The objective of BAS is to maintain as few index peers as possible for better search efficiency. In other words, we want to minimize the search overlay size without sacrificing the search effectiveness. We first formulate this as the minimum buffer cover problem and present a globally optimal solution. Concerning the scalability, we then design a distributed algo-

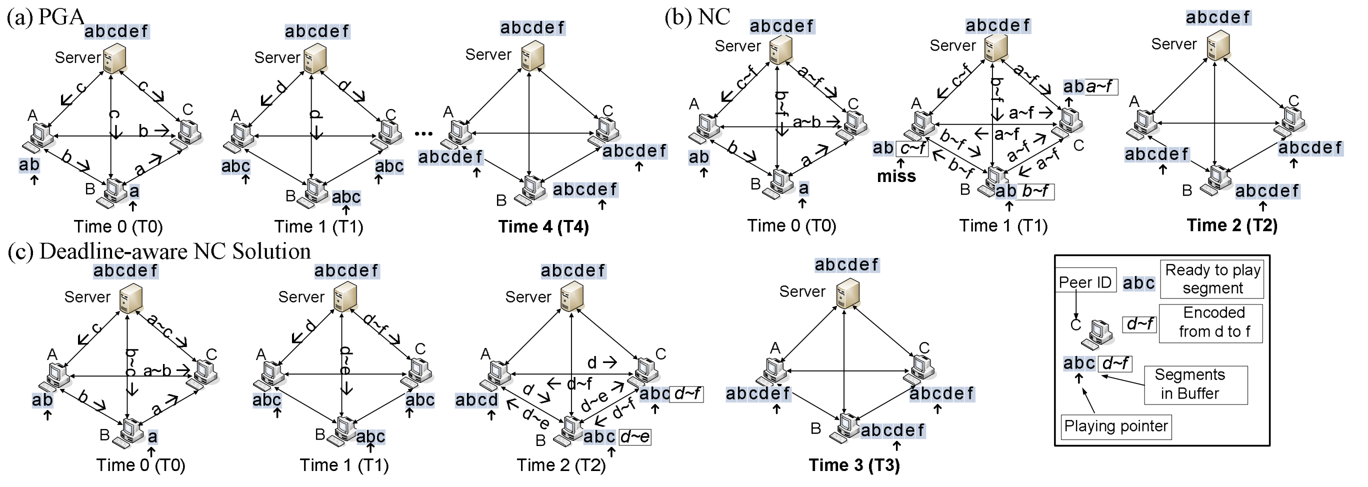


Fig. 4. Pull-based Gossip, Network Coding and Deadline-aware solutions.

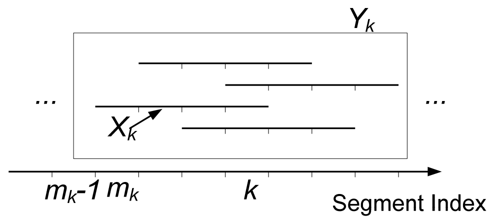


Fig. 5. Analysis of the recurrence of $A(k)$.

rithm and show how to apply it in the overlay construction, maintenance, and VCR operations accordingly.

A. Minimum Buffer Cover Problem and Optimal Solution

Each peer has a buffer range around its play offset and may overlap with other peers in buffer coverage. By noticing there is buffer coverage redundancy, BAS aims to select as few index peers as possible without reducing search effectiveness, that is, the total buffer coverage. The problem can be formulated as the Minimum Buffer Cover (MBC) problem as follows. We consider a collection C of buffers totally covering an integer range R $[1, 2, \ldots, M]$[2] , and every buffer is a consecutive subrange of R. A buffer cover for R is a subset $C' \subseteq C$ such that every element in R belongs to at least one member of C'. The goal is to find the buffer cover C' for R with the minimal cardinality $|C'|$[3]. Note that, at any time, every index peer should contain at least one *unique* number, which is not covered by other index peers, otherwise it will be removed from the index overlay.

MBC is a variation of the well-known NP-hard problem called Minimum Set Cover. In MBC, each buffer set contains continuous elements, so we can leverage it to design a polynomial-time optimal solution. We divide the problem into several sub-problems and use the dynamic programming method to solve them. Let $A(0) = \Phi$ and $A(k)$ be the minimum buffer set to cover $R_k[1, 2, \ldots, k]$ for $1 \le k \le M$.

[2]If the covering area consists of several discontinuous ranges, the problem can be deduced into some sub-problems, each for a range and the corresponding buffers.

[3]$|C'|$ is the size of the set C', and we will use $|\cdot|$ to denote the set size in this paper.

Let us figure out the relationship between these sub-problems. Obviously, the optimal solution is $A(M)$, and $|A(i)| \le |A(j)|$ for $i \le j$. We denote the set of buffers covering k as Y_k, the smallest segment index in the buffers of Y_k as m_k and the corresponding buffer containing m_k as $\chi_k \in Y_k$ (see Fig. 5). When there are more than one buffer containing m_k, χ_k is the one with the largest buffer capacity. Assume $A(i)$'s are known for $1 \le i \le k$, and Theorem 1 presents how to compute $A(k)$.

***Theorem** 1:* $A(m_k - 1) \cup \chi_k$ is the MBC for R_k.

Proof: In any buffer cover A' for R_k, there is at least one buffer $\chi' \in (Y_k \cap A')$. Assume the smallest number in χ' is m', then the set $A' - \{\chi'\}$ at least covers $R_{m'-1}[1, 2, \ldots, m' - 1]$, thus

$$|A' - \{\chi'\}| \ge |A(m' - 1)|, \text{i.e., } |A'| \ge |A(m' - 1)| + 1.$$

Recall that for $m_k \le m'$, we have

$$|A'| \ge |A(m'-1)|+1 \ge |A(m_k-1)|+1 = |A(m_k-1) \cup \{\chi_k\}|.$$

Thus, $|A(k)| \ge |A(m_k - 1) \cup \{\chi_k\}|$. From the optimality of $A(k)$ for R_k, $|A(k)| \le |A(m_k - 1) \cup \{\chi_k\}|$. Thus, we get $|A(k)| = |A(m_k - 1) \cup \{\chi_k\}|$, and $A(m_k - 1) \cup \{\chi_k\}$ is also a MBC for R_k. □

Theorem 1 indicates the recurrence relation of $A(i)$'s:

$$A(k) = \begin{cases} \Phi & \text{if } k = 0, \\ A(m_k - 1) \cup \{\chi_k\} & \text{if } 1 \le k \le M. \end{cases}$$

Note that $A(M)$ is the optimal solution to the MBC problem. There is a trick in computing $A(M)$. Instead of computing all $A(i)$'s for $i < M$, we only need to trace back from $A(M)$ to $A(0)$. For example, if $A(i) = A(j) \cup \{\chi_i\}$, the computation of entries $A(j+1), A(j+2), \ldots, A(i-1)$ can be avoided. With this observation, since each step requires $O(|C|)$ comparison time for m_k and the number of s is $O(|C'|)$, the total running time is $O(|C|^2)$. If the average buffer length of the peers is much smaller than $|C|$ (i.e., the average buffer length can be treated as a constant), we can further reduce the running time by pre-assigning the buffers to their containing numbers using $O(|C|)$ time. Thus upon computing $A(k)$, only the buffers containing k are compared. Since each buffer is compared only once, the total running time can be reduced to $O(|C|)$. The latter solution is described in Fig. 6.

```
1:  for i = 1 to $M$
2:      $Y_i = \Phi$ , $A(M) = \Phi$ ;
3:  for each buffer $\chi \in C$
4:      for each i$\in\chi$
5:          $Y_i = Y_i \cup \{\chi\}$;
6:  for (j = $M$; j > 0; j = $s_\chi$-1)
7:      $\chi = argmin_\chi$ {the smallest number $s_\chi$ in $\chi$, $\chi \in Y_j$ };
8:      $A(M) = A(M) \cup \{\chi\}$;
```

Fig. 6. The globally optimal Dynamic Programming (DP) algorithm.

However, this optimal solution requires global information about the buffer of all peers. This is not practical in a large scale system. We propose a distributed algorithm that makes decision only based on local information in the next subsection.

B. Distributed Algorithm

The large and distributed systems require a distributed algorithm which adjusts the index overlay based on local information. As shown in the above sections, not all peers need to be maintained in the index overlay. Here we design a distributed algorithm to calculate which peers should be indexed based only on the existing index peers plus the newcomer, upon each new peer insertion or VCR-jump. The basic flow is to divide the existing index peers into two groups according to whether they overlap with the newcomer, and then apply the dynamic programming (DP) algorithm proposed in the above section to the newcomer plus the group overlapping with the newcomer.

We denote the collection of buffers of existing index peers by C' and the newly added buffer by β. Let $LAP = \{\alpha | \alpha \in C'$ and α overlaps with $\beta\}$ and $UNLAP = C' - LAP$. Assume $UNLAP$ covers $[1, \ldots, i]$ and $[j, \ldots, M]$. Let LAP' = $\{\alpha \cup \{i+1, \ldots, j-1\} | \alpha \in LAP\}$ and record the mapping between LAP and LAP'. Let $D = \{\beta\} \cup LAP'$. The DP algorithm is to compute the optimal solution D' for D. Restore the buffers in D' according to the mapping between LAP and LAP'. Let $C" = D' \cup UNLAP$ and we prove soon that it is the optimal solution for $\{\beta\} \cup C'$. The complexity of the distributed algorithm is determined by the DP algorithm, which is $O(|D|) = O(|LAP|)$, that is, the number of existing index peers overlapping with the new peer, instead of the system size.

***Theorem** 2:* $C"$ is the optimal solution for $\{\beta\} \cup C'$.

Proof: Before the newly added buffer, β , joins, each buffer in C' contains at least one *unique* number, which is not covered by other buffers in C'. Since buffer β does not affect the unique numbers of the non-overlap buffers, they must remain in the optimal set. Note that $UNLAP$ covers $[1, \ldots, i]$ and $[j, \ldots, M]$, and the uncovered area left becomes $[i+1, \ldots, j-1]$. LAP is adjusted as LAP' for computing the minimum buffer cover D' for the uncovered range. Thus, the union of D' and $UNLAP$ is the minimum buffer cover for $\{\beta\} \cup C'$. □

As more peers join the system, the same operation is applied on $C"$ to get C^3, and then $C^4, C^5 \ldots$. Similarly, C^{k+1} is the optimal solution for $\{\beta\} \cup C^k$. Since the complexity of our distributed algorithm is determined by $|LAP|$ and C^k is the corresponding index overlay, we study the magnitude of $|LAP|$ and compare C^k with the optimal solution to the global system as follows in Lemma 3, 4, Theorem 5 and 6.

Let $N = |C^k|$ and the buffers in C^k, denoted as $X_1, X_2, \ldots, X_N$, be ordered by their left numbers. l_i and r_i are used to represent the left and right end number of X_i. From the definition, we have $l_i < l_j$ for $i < j$. The symbol B_i denotes the length of X_i, and the average length of all buffers is B.

***Lemma** 3:* $r_i < r_j$ for $i < j$.

Proof: According to the definition, we have $l_i < l_j$ for $i < j$. If $r_i > r_j$, then X_j can be fully covered by X_i, which contradicts the assumption that every index peer should contain at least one unique number. □

***Lemma** 4:* $l_{i+2} - l_i > B_i$.

Proof: Assume $l_{i+2} - l_i \leq B_i$, i.e., $l_{i+2} \leq B_i + l_i = r_i$. Thus, the range from l_i to r_{i+2} is fully covered. According to the definition and lemma 3, we get $l_i < l_{i+1}$ and $r_{i+1} < r_{i+2}$, i.e., X_{i+1} is fully covered by X_i and X_{i+2}, which is contradicting the assumption. □

***Theorem** 5:* $N \leq 2M/B$.

Proof: Assume N is an odd integer.

$$\begin{aligned} M &\geq B_N + l_N - l_1 = B_N + \sum_{i=1}^{(N-1)/2} (l_{2i+1} - l_{2i-1}) \\ &> B_N + \sum_{i=1}^{(N-1)/2} B_{2i-1} = \sum_{i=0}^{(N-1)/2} B_{2i+1} \quad (1) \\ M &> B_{N-1} + l_{N-1} - l_2 = B_{N-1} + \sum_{i=2}^{(N-1)/2} (l_{2i} - l_{2i-2}) \\ &> B_{N-1} + \sum_{i=2}^{(N-1)/2} B_{2i-2} = \sum_{i=1}^{(N-1)/2} B_{2i} \quad (2) \end{aligned}$$

From (1) and (2), we get

$$\begin{aligned} 2M &> \sum_{i=0}^{(N-1)/2} B_{2i+1} + \sum_{i=1}^{(N-1)/2} B_{2i} \\ &= \sum_{i=1}^{N} B_i = NB, \text{ i.e. }, N < 2M/B. \quad (3) \end{aligned}$$

The proof is similar when N is an even integer. □

***Theorem** 6:* The expected size of LAP is a constant.

Proof: The overlapped buffers can be categorized into two types: the ones left the index overlay and the ones remained after inserting the new buffer β. From Theorem 5, the index size is bounded by 2M/B, so β's arrival removes at most one buffer in the old index list on average. Lemma 4 ensures that any buffer in the index list X_i can only overlap with X_{i-1} and X_{i+1}. Thus on average, there are at most three overlapped buffers found by the new buffer. □

Theorem 5 and 6 demonstrate that the distributed algorithm consumes constant amortized time and bounds the index overlay size within $O(M/B)$, where M is the media length and B is the average buffer length of the peers. If every buffer is equal in size-B, then we get $\frac{|C^k|}{|OPT|} \leq \frac{2M/B}{M/B} = 2$, where OPT is the globally optimal solution. Our simulation results in Section V show that $|C_k|$ is even closer to the optimal value.

C. Overlay Construction, Maintenance, and VCR Operations

This subsection presents how the distributed algorithm is applicable to the index overlay to improve search efficiency. The protocol design consists of four parts: 1) node join; 2) node leave; 3) failure recovery; and 4) VCR-interactions. In this paper, we reuse the index structure implemented using AVL tree, DHT, or skip-list, which can provide sub-linear search efficiency. We apply BAS scheme on top of those structures so as to reduce their index overlay size.

1) Node Join Operation: When a new client joins the overlay, it first looks for the closest index neighbor in $O(\log N)$ hops. After that I partners are selected from this found peer's data neighbors, where I is the number of initial partners. The selection criteria can be based on the end-to-end network delay, available bandwidth, node workload, or a combination of those metrics. There have been many research works on how to use these metrics to achieve network-aware selection and load balance [14, 21, 22], however we do not discuss them in detail in this paper. Then the newcomer finds out the index peers with buffer overlapping by tracing backward and forward along the closest index peer's predecessor and successor. From theorem 6, the expected number of hops to be traced is a constant. Finally the dynamic programming algorithm is applied to the found peers plus the new peer to figure out which peers should be pruned from the index overlay. The expected number of nodes a new client should contact is a constant.

2) Node Leave Operation: When a peer leaves, its neighborhood in the index overlay and the data overlay should be adjusted for system resilience. A peer scheduled to leave notifies its neighbors, such that they can select new partners and form new neighboring relations. If the departing peer is in the index overlay, it first notifies its index neighbors to rebuild the index links in each level; and then chooses the least number of peers among the neighbors to cover the index coverage hole left by the departing node. This hole-recover problem can be transformed into the MBC problem. Assume the hole left covers $H = [i, \ldots, j]$ and the set of non-index neighbors' buffers is denoted by $NB = \{\beta_1, \beta_2, \ldots, \beta_n\}$. If we adjust NB to $NB' = \{\alpha \cap \{i, \ldots, j\} | \alpha \in NB\}$, the hole-recover problem is to find the minimum buffer covering from NB' for range H. Thus, we can apply the the MBC optimal algorithm to compute the result of NB' for H.

Since the number of neighbors maintained at each node is a constant, the cost of the leave operation is constant. Note that the leave operation does not change the index overlay's property: every index peer contains at least one unique number. Thus the theorems in the last subsection still hold because they are based on this property.

3) Failure Recovery: Different from the leave operation, the failure recovery is detected and handled passively. Each peer sends a heartbeat packet to its neighbors at the beginning of each schedule round or each common default period. If the peer does not receive any response from a neighbor in default consecutive (e.g., 3) rounds, it considers the neighbor as failed. The heartbeat packet is small in size, so the probability of its loss in several consecutive rounds due to network congestion is low.

Upon detecting a failed neighbor, the peer decides to find one more neighbor or only depend on its current data neighbors. If the number is less than the default minimum partner number (e.g., 3), the peer will find more data neighbors from both its two-hop data neighborhood and the index overlay. If the failed node is an index peer, a backup-node scheme is

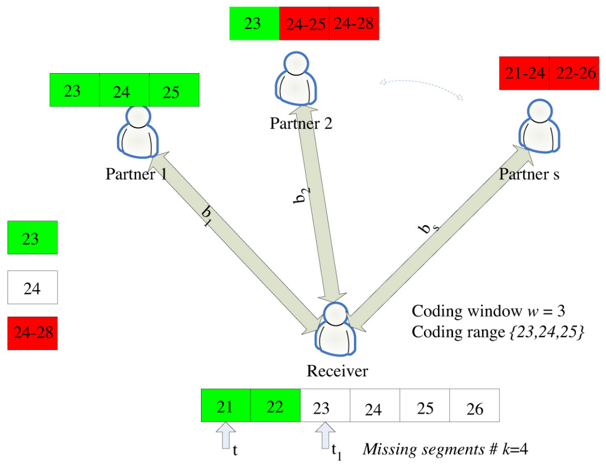


Fig. 7. Analysis of CW assignment problem.

applied for failure handling. We introduce one more type of node, backup node, which acts as a backup of the index peer. Each index peer selects a backup node from its neighbors and notifies them the backup node. The backup node serves to collect information from the index peer's neighbors when they detect the failure of index peer and decide which nodes should rejoin. The computation is the same as the algorithm described in the Node Leave Operation subsection.

4) VCR Operations: In general, the typical VCR operations, for example, fast-forward or rewind, can be implemented with the combination of a leave and re-join operation. However, a common VCR operation may consist of a series of such movements, which are not far from each other, so frequent re-join operations are not efficient. If the index structure supports horizontal shortcuts to other index peers with logarithmically-increasing distances, the VCR operation can jump to the new offset by skipping the most unnecessary nodes. DHT and skip-list are such examples. Compared with searching from the root node or top layer, following the horizontal shortcuts can reduce the search cost significantly. BAS further reduces the number of horizontal hops with a smaller index overlay.

In summary, the BAS scheme applies a distributed pruning algorithm to existing mature index structures such that the index overlay size is bounded within a small and stable scale of $O(M/B)$. A small structure ensures a fast response time and low maintenance overhead.

IV. Deadline-Aware Network Coding Scheduling

After getting partners from the BAS overlay, a peer should collaborate with these partners to dynamically schedule data transmission according to their buffer content information and network conditions. In this section, we propose a DNC based approach for this multi-partner scheduling. We first introduce the DNC scheme and formulate the scheduling problem using DNC. We then present an optimal algorithm and a distributed protocol accordingly.

A. DNC Scheme and Scheduling

As illustrated in Section II, the basic idea of DNC is to consider the play deadline of data segments when doing network coding so that it can improve the network throughput and at the same time satisfy the deadline requirement.

Assume there are k expected segments $\{m_i\}$ as $m_1, m_2, \ldots, m_k$. A random linear coding on a w-size subset $\{m_i'\}$ ($|m_i'| = w$) of $\{m_i\}$ is a vector $c_j = \sum_{i=1}^{w} \alpha_{ji} m_i'$, where w is called the coding window, and the coefficient vector α_{ji} is randomly generated in a finite field F_q of size q (q is determined by certain coding scheme). If all c_j are linearly independent, once a node has a subset of c_j that spans w, it can recover the w expected segments in $\{m_i'\}$ by solving a set of linear equations. In conventional network coding, the *CW* is always the same as the number of expected segments, that is, $w = k$, and thus the coding range covers all expected segments. Though a larger *CW* makes better utilization of resource, it causes a higher possibility of segments being missing upon the play deadline due to the longer waiting time before decoding. Thus the DNC scheme tries to use as large a *CW* as possible for better coding efficiency while controlling the *CW* size so that no segment will miss its play deadline.

DNC estimates the *CW* to be the maximum number of segments that can be retrieved from the partners before the most urgent deadline. Thus we formulate the *CW* assignment problem with DNC as follows. The segment size is assumed to be 1 unit for simplicity. As illustrated in Fig. 7, at time t, suppose a peer has k data segments $\{m_i\}$ missing in the buffer each with a play deadline t_i, $i = 1, 2, \ldots, k$, and s partners each with a subset of data segments D_i (available or coded) and available bandwidth between partner i and this peer is b_i, $i = 1, 2, \ldots, s$. Then the problem is to find an assignment $A = \{(i, j)\}$ for retrieving segment j from partner i, such that the number of received segments $\{m_i'\}$ is maximized before the most urgent deadline t_1. The problem formulation is shown in Fig. 8. We have two rules on the assignment.

1) A *valid* assignment means the same segment request can not be assigned to more than one partner, since it makes no sense in receiving multiple copies of the same segment.
2) The number of segments assigned to each partner can not exceed the quantity that can be provided by the partner, that is $f(i) = \lfloor b_i * (t_1 - t) \rfloor$.

B. Max-flow based Optimal Solution

Interestingly, we found that we can construct a flow network $G = (V, E)$ to transform the DNC problem to the max-flow problem. Nodes S and T are the source and sink, respectively. Each subset D_i and each element m_j correspond to node D_i and node m_j, respectively. The capacities of edges between S and D_i are $f(i)$. Between D_i and m_j, there is an edge with capacity 1 only if $m_j \in D_i$. The capacities of edges between T and m_j are all 1. Then the complexity of the node number and edge number is $|V| = s + k + 2 = O(s + k)$ and $|E| = s + k + \sum_{i=1}^{s} |D_i| = O(s * k)$, respectively. Fig. 9 depicts the flow network G. For a flow g in G, its flow value is denoted as $|g|$, and the flow value from node x to node y as $g(x, y)$.

***Theorem** 7:* If A is a valid assignment in the coding window problem, there is a flow g in G with $|g| = |\{m_i'\}|$. Similarly if g is a flow in G, there is a valid assignment A with $|\{m_i'\}| = |g|$.

Input: *s* subsets D_i of a finite set $\{m_i\}$, with $|\{m_i\}|=k$.
Output: A set $\{m'_i\} \subseteq \{m_i\}$ that consists of at most $f(i)= \lfloor b_i*(t_1-t) \rfloor$ elements from each subset D_i.
Goal: Maximize the cardinality of $\{m'_i\}$, i.e., $|\{m'_i\}|$.

Fig. 8. Formulated problem statement.

Proof: For any valid assignment $A = \{(i, j)\}$, we can construct a flow g in this way. If $(i, j) \in A$, then let $g(D_i, m_j) = 1$. The two rules presented in the last subsection ensure that this constructed flow g does not exceed the limits on the edges between S and D_i and those between T and m_j. Since each segment m_j assigned in A corresponds to a value-1 flow from node m_j to T, we have $|g| = |\{m_i'\}|$. A similar proof can be done for the reverse direction. □

Thus our coding window assignment problem can be mapped to the max-flow problem in G. The max-flow solution not only gives the size of the coding window, but also specifies which segments should be included inside the coding range. The max flow value is the coding window size. The coding range includes the segments whose corresponding nodes in G have value-1 flows to node T. Since there may be more than one optimal solution and the max-flow algorithm does not consider different play deadlines of the missing segments, a post process is needed to make sure that we select the solution with the most urgent segments. We use a k-bit data B to measure the urgency of a solution. The highest bit of B corresponds to the missing segment m_1, and the i-th highest bit corresponds to m_i. If a solution contains m_i, then the i-th highest bit is set to 1, otherwise 0. The higher value of B, the urgency is higher. The post process aims to find the solution with the highest urgency.

The post process (PP) checks whether the missing segments that are not inside the solution can replace the less urgent segments that are inside the solution without changing the flow value. For each of them m_i, PP traces back to its preceding nodes in G, that is, those partners containing it, and examines whether they have value-1 flow to the nodes whose play deadline is less urgent than m_i. If there is one such node m_j, exchange m_i with m_j in the solution set, and change the corresponding flows in G.

***Theorem** 8:* (1) PP does not change the network flow value; (2) PP guarantees that the final solution has the highest urgency value.

Proof: Interested readers are referred to [3] for details. □

We can use the well-known max-flow algorithm to compute the coding window w and corresponding coding elements $\{m_i'\}$. Since the fastest known max-flow algorithm has time complexity of $O\big(|V| * |E| * \log(|V|^2/|E|)\big)$ [11], the running time for a coding window assignment is also $O\big(s^2 k * \log((s + k)^2/(s * k))\big)$, where k is the number of missing segments and s is the number of partners.

C. Distributed Protocol

The distributed protocol consists of four parts: 1) full or partial decoding; 2) determine the new coding window; 3) request from the partners; 4) respond to the requests from partners. Each peer uses the max-flow algorithm to calculate

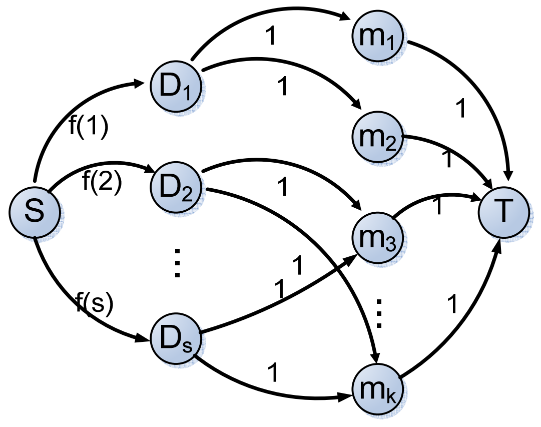


Fig. 9. Constructed flow network G.

```
1: for each node n in the system
2:   if(any received combinations can be decoded)
3:     decode and update buffer;
4:   if(all the w expected segments in {m'_i} are decoded)
5:     recalculate the coding window w and {m'_i};
6:   for each idle partner s_i with expected data in {m'_i}
7:     request length-w combinations among {m'_i} from s_i;
8:   if (any requests arrive) and (n contains the requested data)
9:     responds the requests with encoded segments;
```

Fig. 10. DNC scheduling algorithm.

the coding window w and corresponding coding elements $\{m_i'\}$, and then sends this request to all the partners. After decoding all requested segments in $\{m_i'\}$, the peer starts a new round of computation of the coding window and corresponding coding elements. The DNC scheduling algorithm is shown in Fig. 10.

1) Full or partial decoding: Upon receiving new segments, the peer examines whether the received combinations can be decoded. If they are decodable, the segments are reconstructed and the buffer is updated immediately. We then have to decide whether the received segments are decodable or which are decodable. The simplest way is to test all combinations of the X existing segments, that is, 2^X rounds of decoding. However, this incurs exponentially-growing overhead and thus is unfeasible. We propose a novel algorithm called Enhanced Gaussian Elimination (EGE), which can efficiently decide which segments can be decoded by extending the Gaussian Elimination algorithm [1]. Due to the limited space, interested readers are referred to our technical report [3].

2) Determine the new coding window: If the w expected segments decided in the last schedule have been fully recovered, the coding window and coding range are recomputed.

3) Request from the partners: Then the node requests combinations spanning the w expected segments in the coding range from each idle partner that contains unavailable data.

4) Respond to the requests from partners: Upon receiving a request from some neighbors, the node checks whether it is able to response an encoded segment computed from its available segments or received encoded segments or both. There are three cases that the node is able to provide useful segments for the receiver (see Fig. 7). First, there are some available segments within the requested window range, for example, partner 1 has the expected segments 23, 24, 25. Second, the node has some encoded segments whose coding range is within the requested window range, for example, partner 2 has a coded segment 24~25 that is within the requested window range {23, 24, 25}. Third, there are some encoded segments that have some portion of coding range within the requested window range and the other portion corresponding to available segments in the requested node, for example, 21~24 on partner 3.

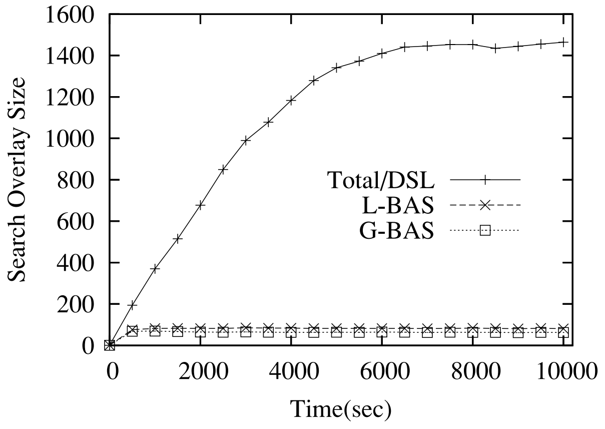


Fig. 11. The size of index structure.

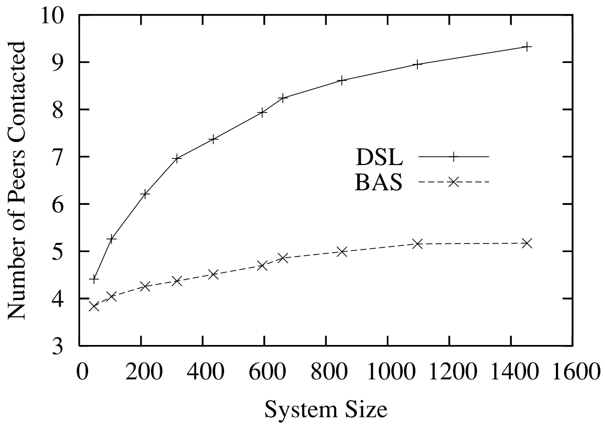


Fig. 12. Join/Search cost for DSL and BAS.

In summary, this section studied the data scheduling among partners. We proposed the deadline-aware network coding scheme and a max-flow-based optimal solution. DNC exploits network coding to improve peer cooperation, and thus increases the network throughput. Different from traditional network coding, DNC provides a dynamic coding window for each peer to avoid segment missing upon the play deadline. A distributed scheduling protocol using DNC scheme was proposed.

V. Simulation Results

We evaluate the performance of our MoD system by comparing it with some existing systems.

A. Simulation Configuration

We use the Sprint ISP topology collected by Rocketfuel engine [19] for system setup. It consists of 112 backbone

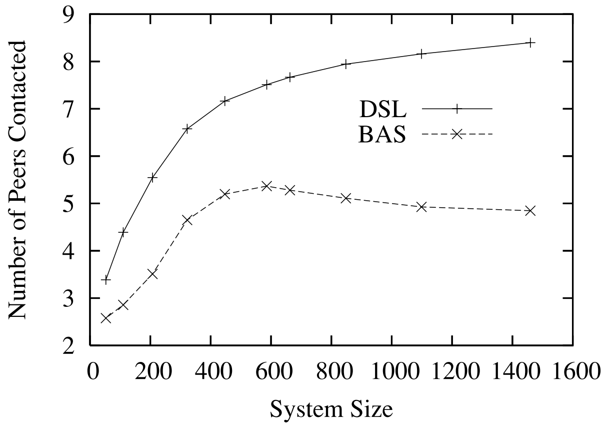


Fig. 13. Leave cost for DSL and BAS.

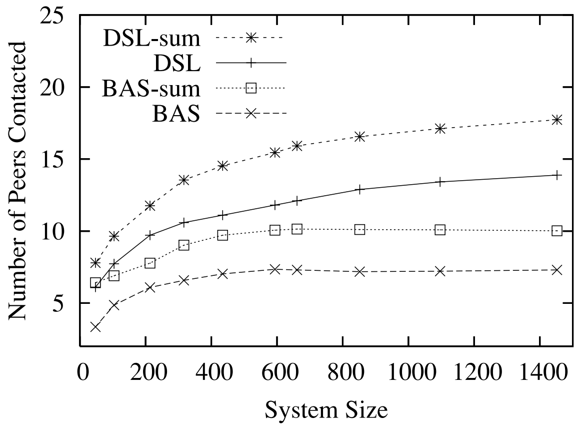


Fig. 14. 30%-VCR-jump cost.

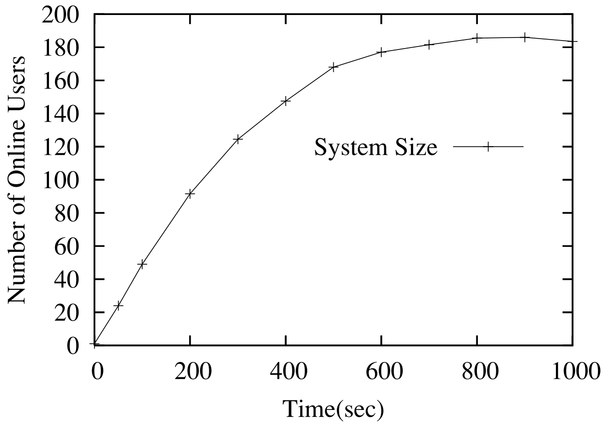


Fig. 15. System size during 1000-sec simulation.

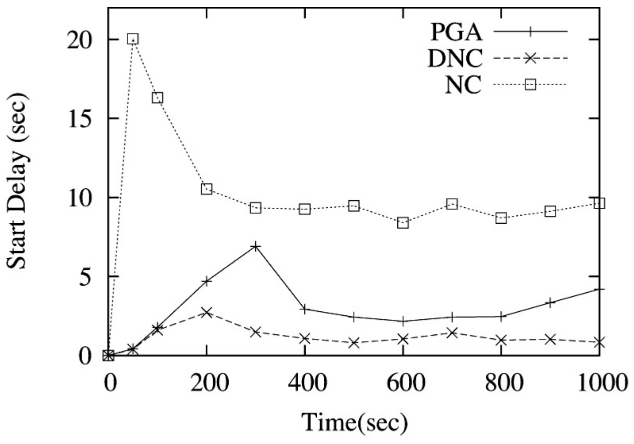


Fig. 16. Start delay for PGA, NC and DNC.

nodes and 242 access nodes; 5 stub nodes are attached to each access node to simulate a local area network. Thus the network size is about 1,500. The default bandwidth settings between two backbone nodes, a backbone and an access node, two access nodes, two stub nodes within the same LAN are 10 Mbps, 5 Mbps, 3Mbps, and 10 Mbps, respectively.

In our simulation, the server and users are located at randomly selected stub nodes. The communication path between any two nodes follows the shortest path. The bit rate of the streaming media is 500Kbps and its length is 2 hours. The length of a segment is 1 second, and the default size of the user buffer is 7.5 Mbytes, which can accommodate 120 segments, that is, less than 2% of the entire stream. There is no user in the system at the beginning, and users join the system following a Poisson process with a mean inter-arrival time of 2.5 seconds. The start offset of each user is evenly distributed between 0 and 2 hours. The users leave the system upon the end of the video. For each set of configuration, 100 simulation runs have been performed to mitigate the effect of randomness.

B. Search Efficiency and its Control Overhead

We first investigate the search performance for the BAS scheme. Due to the space limitation, only the BAS using the skip-list structure is shown in this section. We compare BAS with the DSL [20], which also uses the skip-list structure.

Fig. 11 depicts the size of search structure for DSL and BAS during a 10,000 sec simulation. L-BAS represents the case where the distributed algorithm is used, while G-BAS is the size computed by the globally optimal algorithm. Since DSL has to maintain the play progress of all online peers in the MoD system, the size of DSL increases as the system grows up with time until the system size k relatively stable around 1,400 after 7,000 sec. In contrast, BAS only needs to maintain less than 100 peers regardless of the magnitude of system expansion. The curve of L-BAS is very close to G-BAS, indicating that the performance of our distributed algorithm is close to the optimal one.

A small size index structure generates low control overhead and provides a fast search time. We use the number of peers contacted (*peer hops*) during an operation as the metric for controlling overhead and execution time. Fig. 12 depicts the mean peer hops during the partner search of a node join for both DSL and BAS. In DSL, though the number of peers contacted during a search is logarithmic to the system size, about 9 peer hops are required when the system size is about 1,400. In contrast, BAS requires about 5 peer hops and the search cost is not sensitive to the system expansion, since the size of BAS index overlay is small and stable.

Fig. 13 depicts the mean peer hops during a node leave operation in both DSL and BAS. A node leave needs to inform both index neighbors and data neighbors. BAS outperforms DSL in reducing the control overhead related to the index neighbors. Since many peers are not index peers in the BAS, the average adjustment cost due to node leave is much smaller

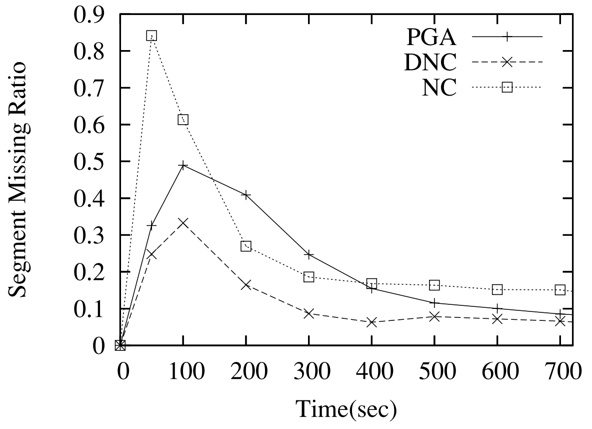


Fig. 17. SMRs for PGA, NC and DNC.

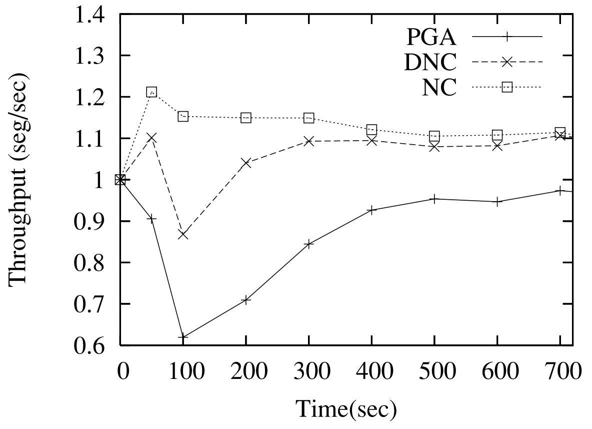


Fig. 18. Throughput comparison.

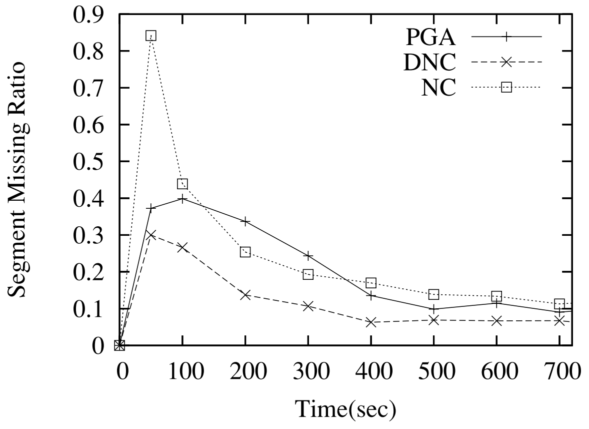


Fig. 19. SMRs with 3%-loss.

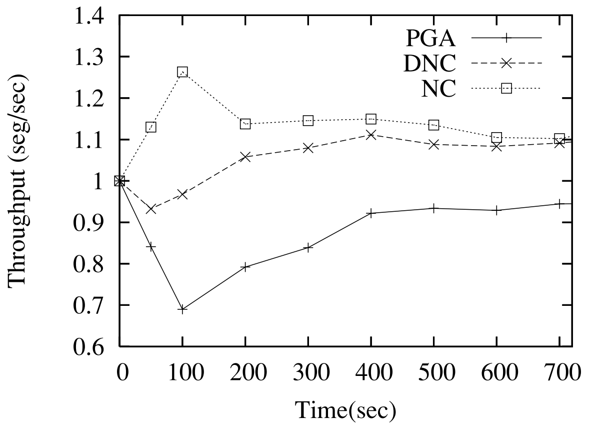


Fig. 20. Throughput with 3%-loss.

than that of DSL. This is more significant when the system size increases. As shown in Fig. 13, the average cost for BAS first increases, because the number of index neighbors is increasing. When the system grows large enough, the size of BAS overlay becomes stable. Then the average leave cost decreases since the portion of index nodes decreases.

A VCR-jump operation consists of two steps: leave the current neighborhood, and find the partners close to the new play offset. The first step is similar to the common node leave, while the second step can be implemented more efficiently than the new node join operation if we exploit the horizontal shortcuts supported by the skip-list. Fig. 14 depicts the peer hops of a 30%-VCR-jump operation (jump offset is 30% of the streaming length) in both DSL and BAS. Let "DSL-sum" and "BAS-sum" denote the cases simply combining leave and rejoin operation in DSL and BAS, respectively, while "DSL" and "BAS" denote those exploiting horizontal shortcuts to realize VCR-jump interactions. It can be seen that the VCR-jump cost is lower than the sum of join cost and leave cost. This demonstrates the effect of jumping from a current play point using horizontal shortcuts. BAS outperforms DSL due to its smaller size of index overlay, thus generating fewer control messages and providing faster response time.

C. Streaming Quality

In the second set of our simulation, we examine the streaming quality of the MoD system. The start delay, which is the waiting time before the first playable segment is ready, and Segment Missing Ratio (SMR), which is the ratio of segments that are not available upon their play deadline, are used as the performance evaluation metrics. Three scheduling schemes,PGA, NC and DNC, are implemented for comparison. In a pure NC scheme, the coding window size is set to the default user buffer length, instead of the whole media length.

In the following simulations, some settings are changed for simplicity. The video length is 720 seconds and the default capacity of the user buffer can accommodate 20 segments. Fig. 15 plots the number of online users in the system during a 1000-sec simulation. It is seen that the system k increasing to about 180 users during the first 720 seconds. After that, the system size k relatively stable, since the number of new users is close to that of the departing users.

Fig. 16 plots the start-up delay for MoD systems based on the three scheduling methods, PGA, NC and DNC, under the same network configuration and user setting. The start-up delay for NC is obviously larger than those of PGA and DNC because the first segment could not be reconstructed until enough coded segments are received. In the dynamic changing network, PGA may suffer from the congestion due to the greedy and local decisions, so the start-up delay is not stable. In contrast, the start-up delay in DNC is smaller and more consistent because the coding window is adaptive to the network condition and thus small upon peer arrival, and the network throughput utilization is improved with network coding.

Fig. 17 plots SMRs for PGA, NC and DNC systems. We observe SMR increased dramatically in the first 100 sec. Since the server can serve at most 10 (5Mbps/500Kbps) simultaneous connections and there are at most 36 (streaming-len/buffer-size) non-overlapping online users, the users are congested at the server side at the early stage when the system has more than 10 non-overlapping users and there are few users filled with segments. Compared with PGA, DNC and NC quickly reduce the SMRs to reasonable degree due to the power of network coding in improving the network throughput. With the content play deadline in mind, DNC can obtain the smallest SMR.

Fig. 18 depicts the network throughput comparison for these three schemes. The throughput metric is the average number of segments a peer received per second. From the figure, the throughput of DNC and NC outperforms PGA more than 20%. The NC always get a high throughput, but due to the changing and large coding window, many segments can not be decoded when they required to be played and thus even though the segments eventually can be received, some of them are wasted. DNC leverages the throughput improvement and also considers the deadline to change the coding window to make as many segments decodable upon their deadline as possible. So even though DNC gets an uncomfortable SMR at the initial stage, it can adaptively reduce SMR into around 5% soon and such loss can be effectively recovered by applying Forward Error Correction (FEC) [9].

To evaluate the effect of packet loss on the network coding, we assume each link will drop the packet at a certain probability even though there is enough available bandwidth at the link. Fig. 19 and 20 depict the SMRs and network throughput comparison at 3% packet loss ratio. We observe that the performance results are similar to the previous set of experiments. Thus, the packet loss does not affect the DNC's performance.

We also perform simulation for 500 and 1000 users and similar results have been obtained [3].

VI. Conclusion and Future Work

In this paper, we have presented a novel P2P MoD system, where Buffer Assisted Search (BAS) structure and Deadline-aware Network Coding (DNC) scheduling schemes are proposed to address the two key issues of an MoD system, that is, partner search and peer collaboration. The BAS scheme exploits the buffer coverage redundancy to reduce the index overlay size for constant search efficiency and low control overhead. The BAS structure is generic and can be implemented based on existing data structures, for example, AVL-tree, DHT, and skip-list. The DNC scheme, which extends network-coding techniques in the data exchange, adaptively adjusts the coding window for each node based on its varying network conditions and play deadline. DNC improves the network throughput and reduces the total schedule time without missing the play deadline at high probability. Extensive simulation results have shown that our proposed MoD system outperforms existing systems in both search efficiency and streaming quality.

In the future, we plan to deploy a prototype system on the planet-lab for a better understanding of the packet-based content delivery in the real distributed network environment. Moreover, in this paper, the BAS scheme deals with one-cover problem, that is, each segment is covered by at least one index peer. For better resilience, the multiple-cover problem is worth further studying.

Acknowledgment

We would to thank Dr. Xueyan Tang for his helpful suggestions on the optimization problem. The work at Hong Kong University of Science and Technology was supported in part by grants from RGC under contracts DAG05/06.EG05, a grant under the contract MRA05/06.EG03.

References

[1] `http://en.wikipedia.org/wiki/Gaussian_elimination`

[2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments", in *Proceedings of the 19th ACM SOSP*, Bolton Landing, NY, October 2003.

[3] H. Chi, Q. Zhang, J. Jia, and X. Shen, "Efficient Search and Scheduling in P2P-based Media-on-Demand Streaming Service", *HKUST Technical Report*, May 2006.

[4] H. Chi and Q. Zhang, "Efficient Search in P2P-based Video-on-Demand Streaming Service", accepted to appear in the *IEEE International Conference on Multimedia & Expo (ICME2006).*

[5] H. Chi and Q. Zhang, "Deadline-aware Network Coding for Video on Demand Service over P2P Networks", accepted to appear in the *15th International Packet Video Workshop (PV06).*

[6] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast", in *Proceedings of ACM SIGMETRICS*, Santa Clara, CA, USA, June 2000.

[7] Y.Cui, B.Li, and K. Nahrstedt, "oStream: asynchronous streaming multicast", *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[8] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment", *Proc. IEEE ICC'04*, Paris, June 2004.

[9] P. Frossard and O. Verscheure, "Joint Source/FEC Rate Selection for Quality-Optimal MPEG-2 Video Delivery", *IEEE Transactions on Image Processing*, vol. 10, n. 12, December 2001, pp.1815-1825.

[10] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution", in *INFOCOM 2005*, Miami, March 2005.

[11] A.V. Goldberg and R.E. Tarjan, "A New Approach to the Maximum-Flow Problem", J. ACM, vol. 35, no. 4, pp. 921-940, Oct. 1988.

[12] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer patching scheme for VoD service", in *Proceedings of the 12th World Wide Web Conference*, Budapest, Hungary, May 2003.

[13] M. Hefeeda, B. Bhargava, and D. Yau, "A hybrid architecture for cost effective on demand media streaming", *Journal of Computer Networks*, 44(3), pages 353-382, 2004.

[14] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proc. of the 19th ACM Symposium on Operating Systems Principles*, 2003.

[15] Z. Li, B. Li, D. Jiang, and L.C. Lau, "On Achieving Optimal Throughput with Network Coding", in *INFOCOM 2005*, Miami, March 2005.

[16] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking", in *Proceedings of ACM NOSSDAV*, 2002.

[17] B. Quinn and K. Almeroth, "IP multicast applications: Challenges and solutions", *Internet Engineering Task Force (IETF) Internet Draft*, work in progress, March 2001.

[18] R. Rejaie and A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming", in *Proceedings of ACM NOSSDAV*, June 2003.

[19] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel", in *Proceedings of ACM SIGCOMM'02*, pages 133-145, August 2002.

[20] D. Wang and J. Liu, "A Dynamic Skip List based Overlay Network for On-Demand Media Streaming with VCR Interactions", *IEEE International Conference on Multimedia & Expo (ICME)*, Toronto, ON, Canada, July 9-12, 2006.

[21] W. Yiu, S. Chan, Y. Xiong, Q. Zhang, "Supporting Interactive Media-on-Demand in Peer-to-Peer networks", Technical report, 2005.

[22] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming", *IEEE INFOCOM'05*, Miami, USA, Mar 2005.

[23] M. Zhou and J. Liu, "Tree-Assisted Gossiping for Overlay Video Distribution", to appear in *Kluwer Multimedia Tools and Applications*, 2005.

[24] X. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance", *IEEE Journal on Selected Areas in Communications*, Vol. 22, Issue 1, pp. 18 - 28, Jan. 2004.

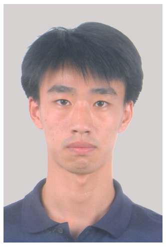
Huicheng Chi received the BEng degree in computer science and technology from Tsinghua University, Beijing, China, in 2003.

He is currently a MPhil candidate in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include Peer-to-Peer Streaming service, wireless networking, and distributed systems.

Qian Zhang (M'00-SM'04) received the B.S., M.S., and Ph.D. degrees from Wuhan University, China, in 1994, 1996, and 1999, respectively, all in computer science.

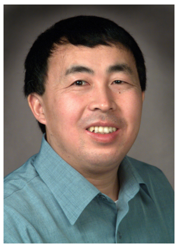
Dr. Zhang joined Hong Kong University of Science and Technology in Sept. 2005 as an Associate Professor. Before that, she was in Microsoft Research, Asia, Beijing, China, from July 1999, where she was the research manager of the Wireless and Networking Group. Dr. Zhang has published about 150 refereed papers in international leading journals and key conferences in the areas of wireless/Internet multimedia networking, wireless communications and networking, and overlay networking. She is the inventor of about 30 pending patents. Her current research interests are in the areas of wireless communications, IP networking, multimedia, P2P overlay, and wireless security. She also participated many activities in the IETF ROHC (Robust Header Compression) WG group for TCP/IP header compression.

Dr. Zhang is the Associate Editor for IEEE Transactions on Wireless Communications, IEEE Transactions on Multimedia, IEEE Transactions on Vehicular Technologies, and Computer Communications. She also served as Guest Editor for special issue on wireless video in IEEE Wireless Communication Magazine and is serving as Guest Editor for special issue of cross layer optimized wireless multimedia communication in IEEE JSAC. Dr. Zhang has received TR 100 (MIT Technology Review) world's top young innovator award. She also received the Best Asia Pacific (AP) Young Researcher Award elected by IEEE Communication Society in year 2004. She received the Best Paper Award in Multimedia Technical Committee (MMTC) of IEEE Communication Society. Dr. Zhang is the vice-chair of Multimedia Communication Technical Committee of the IEEE Communications Society. She is also a member of the Visual Signal Processing and Communication Technical Committee and the Multimedia System and Application Technical Committee of the IEEE Circuits and Systems Society.



Juncheng Jia received the B.E. degree in computer science and technology from Zhejiang University, Hangzhou, China in 2005. He is currently working towards the Ph.D degree in computer science at Hong Kong University of Science and Technology.

His research interests include peer-to-peer streaming and spectrum management.



Xuemin (Sherman) Shen received the B.Sc. (1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. (1990) degrees from Rutgers University, New Jersey (USA), all in electrical engineering. From September 1990 to September 1993, he was first with the Howard University, Washington D.C., and then the University of Alberta, Edmonton (Canada). Since October 1993, he has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where he is a Professor and the Associate Chair for Graduate Studies. Dr. Shen's research focuses on mobility and resource management in interconnected wireless/Internet interworking, UWB, WiFi/WiMAX, sensor and ad hoc wireless networks. He is a coauthor of three books, and has over 200 publications in wireless communications and networks, control and filtering. Dr. Shen received the Outstanding Performance Award in 2004 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario for demonstrated excellence of scientific and academic contributions, and the Distinguished Performance Award from the Faculty of Engineering, University of Waterloo, for outstanding contributions in teaching, scholarship and service. He serves as one of the General/Technical Chairs, for QShine'06, IWCMC'06, IEEE ISSPIT'06, IEEE Broadnet05, QShine05, IEEE WirelessCom05, IFIP Networking 2005, ISPAN04, and IEEE Globecom'03 Symposium on Next Generation Networks and Internet; He also serves as the Editor, for IEEE Trans. Wireless Communications, and DCDIS - Series B: Application and Algorithm; the Associate Editor, for IEEE Trans. Vehicular Technology, ACM Wireless Networks, Computer Networks, WCMC (Wiley), and Int. J. Computer and Applications. He also serves as Guest Editor for IEEE Wireless Communications, IEEE JSAC, and IEEE Communications Magazine. Dr. Shen is a senior member of the IEEE, and a registered Professional Engineer of Ontario, Canada.