Optimizing Segment Caching for Mobile Peer-to-Peer Interactive Streaming

Jingwei Li S-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: {ljxag, gchan}@cse.ust.hk

***Abstract*—With the penetration of broadband wireless access network and devices, interactive multimedia streaming to handhelds has become a reality. However, it is still challenging to offer such services to a large number of users in a cost-effective manner. With the increase of battery lifetime, memory capacity and processing capability, and the fact that many mobile devices nowadays are equipped with multiple interfaces (3G, Wi-Fi, bluetooth, etc.), we study wireless peer-to-peer (P2P) streaming for scalable interactive streaming. In the network, videos are divided into segments and collaboratively cached and accessed among mobile devices. The major challenge is then which segment to cache at each mobile to achieve efficient access (in terms of low segment access cost).**

We first formulate the problem of segment caching to minimize segment access cost. We show that the optimization problem is *NP-hard* and present OPSEC (Optimized Segment Caching), a distributed algorithm which achieves collaborative and efficient segment caching, given heterogeneous caching capacities of the participating users. Using simulation, we show that OPSEC achieves much lower network access cost as compared with some recent schemes for interactive wireless video streaming.

I. Introduction

With the advances in mobile technologies, handhelds nowadays have been equipped with much processing, memory and storage capacities. This has enabled multimedia streaming over broadband wireless (such as 3G). One of the important services is interactive streaming, which allows users to access and interact with the videos (news or movies) from a server over wireless networks. In the near future, more and more interactive contents will be accessed through wireless channel supporting random seeks, pauses and restarts. Clearly, as the number of users in the network increases, this "client-server" way of accessing content leads to high access cost and bottlenecks at both wireless channel and server.

Motivated by the fact that many handheld devices nowadays come with extra wireless interfaces (such as Wi-Fi or Bluetooth) which are often free and of high bandwidth, we consider in this paper a scalable mobile peer-to-peer (P2P) network formed via these *secondary* channels to support interactive streaming. The video stream is divided into fixed-sized segments, and the handhelds cache and share segments via the secondary channel. We show in Figure 1 the P2P mobile network under study. The users form a P2P network and access the segments in a multi-hop manner. During video playback, a user fetches his video segments for viewing in one of the following three ways:

1) Directly from its local memory, if the segment has been cached;
2) Fetches the segment from a peer within a certain diameter (following the shortest path); in this case, a certain cost depending on hop count is incurred;
3) If both cases 1) and 2) fail, the mobile gets the segment from the content server via the primary channel (such as 3G). In this case, a higher (fixed) channel cost is incurred.

If segments are cached properly, we expect that most of the time, peers can get their segments through case 1) or 2), which leads to a reduction in segment access cost in the network.

Given the scheme above, an important issue is which segment to cache at a peer node given its limited cache size, and how such decision can be made in a distributed manner (without a complete knowledge on the network). A simple approach is to either randomly cache pass-by data by applying LRU cache replacement policy. This is certainly not satisfactory because it does not make good decision on segment

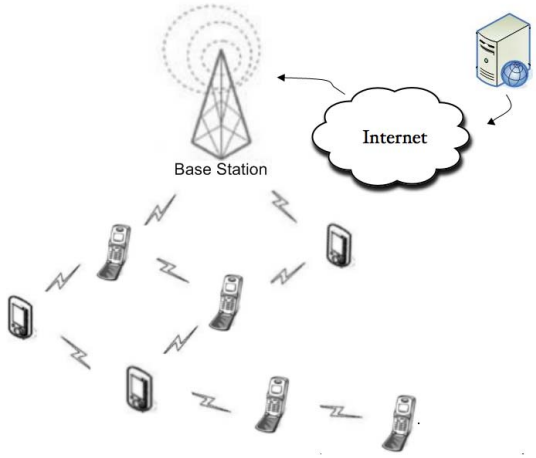

Fig. 1. A peer-to-peer network for mobile interactive streaming.

This work was supported, in part, by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209).

978-1-4244-6404-3/10/$26.00 ©2010 IEEE

caching; consequently the cache space is not efficiently used.

In this paper, we propose a distributed and efficient caching algorithm for mobile P2P interactive streaming called OPSEC (Optimized Segment Caching), which aims to minimize segment access cost in the network given segment access probabilities (which may be assessed beforehand or continuously). Our focus is on distributed caching algorithm; incentive issues are hence out of the scope of this paper.

We address the following in this paper:

- *Problem formulation and its complexity analysis:* We present the formulation of the caching problem to minimize overall segment access cost, and show that the problem is *NP-hard*;
- *OPSEC, a distributed caching algorithm:* We present a simple, distributed, efficient, and collaborative caching algorithm called OPSEC for mobile P2P interactive streaming. OPSEC achieves low segment access cost in the network;
- *Simulation study:* We conduct simulation study on OPSEC and compare it with other state-of-the-art schemes. OPSEC is shown to be scalable, efficient and of low streaming cost. As compared with state-of-the-art and traditional approaches, our scheme achieves close-to-optimal minimum cost.

We briefly discuss previous work as follows. Segment caching in a multi-hop wireless network have been discussed in [1]–[3]. As compared to them, we explicitly take segment popularity into consideration and hence achieve a much better optimization. This is shown in our simulation results. In *Zipf Caching* [4] and *VMesh* [5], interactive video caching algorithms based on segment popularity are discussed. However, unlike in *Zipf Caching*, our scheme does not assume any access pattern and therefore can adapt to any popularity distribution on the fly. Furthermore, we consider mobile handheld devices here and make use of broadcasting which is not possible in the wired Internet environment which *VMesh* is designed for.

Some other previous work on caching for VOD employs hierarchical structures like a tree [6] or a network with backbone nodes [7]. Hence, the schemes have not adequately considered peer churns; Some others require additional infrastructure such as local forwarders [8], a centralized monitoring server [9] or access points [10], [11]. We have a peer-to-peer wireless network which does not require any additional infrastructure.

This paper is organized as follows. In Section II, we present our problem formulation and its complexity analysis. In Section III, we discuss the distributed caching scheme OPSEC. We present illustrative simulation results in Section IV and conclude in Section V.

II. Problem Formulation and Complexity Analysis

We consider that the video file is divided into n video segments labeled as $S_1, S_2, \ldots, S_n$ with independent access popularity $\pi_1, \pi_2, \ldots, \pi_n$, respectively. The network is modeled as a graph $G = (V, E)$, where each user $v \in V$ has a certain caching capacity C_v (maximum number of segments the user can cache). Define d_i^v as the *cost* to fetch segment S_i for v. Clearly, if S_i is stored locally, d_i^v is 0. If S_i is supplied by a peer, d_i^v is given by the number of *hops* between v and the supplier(in appropriate unit). [1] Finally, if S_i is supplied by the content server, d_i^v equals the fixed server access cost. Given above, the total cost of accessing the video is given by $\sum_{i=1}^{n} d_i^v \pi_i$.

The problem is then which segments should be cached at each node under the constraint of C_v, where $v \in V$ so that the overall cost is minimized, i.e.,

$$\text{minimize } \frac{1}{|V|} \sum_{v \in V} \sum_{i=1}^{n} d_i^v \pi_i. \tag{1}$$

Summing up all terms for π_i, we re-write Equation (1) as

$$K = \sum_{i=1}^{n} \pi_i \left(\frac{1}{|V|} \sum_{v \in V} d_i^v \right) \equiv \sum_{i=1}^{n} \pi_i D_i, \tag{2}$$

where D_i is the average cost to access segment i for all the nodes in the network.

Denote X_{ivw} as a boolean variable where $X_{ivw} = 1$ if vertex v finds vertex w the supplier of S_i, and 0 otherwise. Denote δ_{vw} the cost given by the number of hops between v and w. Let α be the fixed server access cost for a single segment in the same unit. Let Y_{iv} be a boolean with $Y_{iv} = 1$ if vertex k gets segment S_i from the server and 0 otherwise ($1 \le i \le n$ and $v, w \in V$). With these, we can rewrite D_i as

$$D_i = \frac{1}{|V|} \left(\sum_{v,w} \delta_{vw} X_{ivw} + \sum_{v} \alpha Y_{iv} \right). \tag{3}$$

Since each vertex (user) will fetch a particular segment either from a vertex (including itself) or from the content server, we must have

$$\sum_{l} X_{ikl} + Y_{ik} = 1. \tag{4}$$

The cache optimization problem is then minimizing segment access cost K given by Equations (2) and (3), subject to Equation (4).

The complexity of the problem is as follows. When a vertex w not yet cached S_i decides to cache segment S_i (which we refer to as a "caching decision"), D_i is reduced by some amount. Specifically, for each vertex v that now finds w cheaper (in terms of cost) to fetch segment S_i, their cost is reduced by simply fetching from w instead. We refer to the cost reduced by a caching decision the *gain* of this decision.

Denote $Z_v i$ the caching decision for node v on segment i. $Z_{vi} = 1$ if v caches S_i, and 0 otherwise. The gain associated with decision Z_{vi} is given by g_{vi}. Apparently g_{vi} contains the popularity factor π_i. It is clear that our caching problem effectively transforms to the caching decision of all vertices on all segments, and minimizing the total cost transforms to maximizing the total gain, subject to each vertex v's cache capacity C_v. For each vertex v we need to

[1] A supplier is the nearest neighbor of v that caches S_i.

TABLE I
SEGMENT ROUTING TABLE EXAMPLE.

Node 1		
SEGMENT	*NEXT_HOP*	*COST*
1	Local	0
2	Node 2	1
3	Server	6
4	Server	6

$$\text{maximize} \sum_i g_{vi} Z_{vi}, \tag{5}$$

$$\text{subject to} \sum_i Z_{vi} \leq C_v, \tag{6}$$

which is equivalent to a *0-1 knapsack* problem and known to be *NP-hard*. To do it for all the vertices is at least as hard because the single vertex version can be reduced to a special case of the optimization for the entire graph with only one active vertex.

III. OPSEC: A DISTRIBUTED SEGMENT CACHING ALGORITHM

In this section, we present OPSEC which caches segments in a distributed manner according to their popularity. Each node maintains a *Segment Routing Table* that resembles a network routing table where each table entry contains *SEGMENT*, *NEXT_HOP* and *COST*. An example of the segment routing table is shown in Table I. *SEGMENT* is the target segment ID to be fetched. *NEXT_HOP* is the next-hop node ID on the shortest path to the segment. If the segment is locally cached, the entry is *LOCAL*. It is *SERVER* if the segment cannot be found within the hop scope. *COST* is the cost associated with this route: when the next hop is *LOCAL* (segment is in local cache), *COST* is 0; when the next hop is a node, *COST* is the hop count to the supplier of the segment (nearest node that caches the segment); when the next hop is *SERVER*, *COST* is a constant representing server access cost. For example, in Table I, the cost to fetch segments 2 and 3 are 1 and 6, respectively.

Upon joining the network, a node (referred to as "joiner" hereinafter) performs the following steps:

- *Step 1:* Broadcasts a "join" message which initiates its direct neighbors to broadcast their segment routing tables.
- *Step 2:* Compile its own routing table by calculating the *shortest path* to each segment from all the known tables, according to what is similar to distance vector routing: if a direct neighbor can fetch S_i from another node or local cache with cost K, then the joiner can fetch the same segment with distance $K+1$ by setting the next hop to the direct neighbor. Besides, the joiner can always fetch a segment from the server by incurring a constant server access cost α. The joiner simply finds the cheapest cost out of all its options and creates the routing entry accordingly.
- *Step 3:* The joining may create new shortest paths between existing nodes. In the newly compiled table of the joiner, if a segment has associated cost K, then the joiner's direct neighbors will be able to get the segment with cost $K+1$ by setting the next hop to the joiner. Corresponding update messages should be sent to the neighbors.
- *Step 4:* Make greedy caching decisions that can benefit its direct neighbors and itself the most. Specifically, given the (updated) segment routing tables of its direct neighbors and itself, the joiner selects to cache segments that lead to the largest cost reductions in all the known tables. To account for segment popularity, we weight the cost reduction of fetching S_i by the segment's popularity π_i in this calculation.

In the last step, if the joiner decides to cache S_i, then it can access the segment with no cost, and all its direct neighbors can access the segment with cost 1. Suppose the joiner has m direct neighbors. Denote the associated cost of S_i in the joiner and its direct neighbor's tables as $K_{i0}, K_{i1}, \ldots, K_{im}$. After weighting with segment popularity, the cost reduction of the joiner's caching decision can be written as $\pi_i \left(K_{i0} + \sum_{j=1}^{m}(K_{ij}-1)\right)$.

However, if a neighbor has already cached S_i, its cost can not be reduced further. As a result the cost reduction for this neighbor should be 0 instead of -1. Denote the number of such neighbors as θ_i, then the actual cost reduction R_i of caching S_i should be

$$R_i = \pi_i \left(K_{i0} + \sum_{j=1}^{m}(K_{ij}-1) + \theta_i \right). \tag{7}$$

Because all the items in Equation (7) can be obtained from the segment routing tables, the cost reduction of caching each segment can be calculated. Note the joiner's capacity as C, it then selects to cache segments $S_{i_1}, \ldots, S_{i_C}$ where $R_{i_1}, \ldots, R_{i_C}$ are the largest C elements in R_i $(0 \leq i \leq n)$.

In terms of computation complexity of the distributed algorithm, suppose there are m direct neighbors. Step 2 requires a traversal of m table entries (to find minimum cost) for each segment, thus complexity is nm. With the newly compiled routing table, Step 3 needs one cost comparison of each neighbor table, thus complexity is also nm. In Step 4, for the calculation of cost reduction of each segment, a traversal of $m+1$ table entries is required, thus its complexity is $n(m+1)$. In order to find the largest cost reductions, a sort of n elements is needed, thus the complexity is $n \log n$. The total computation complexity for the joining process and the caching decisions is hence $nm + nm + n(m+1) + n\log n = O(n(m + \log n))$.

After joining and determining which segments to cache, a node periodically advertises its segment routing table. Upon hearing such advertisements, a node updates its local table in a way similar to Step 3 in the joining process by cross-comparing its own table and the received table, so that others can make caching decisions independently.

A node is determined failed or left the network if a peer

TABLE II
BASELINE PARAMETERS.

Parameter	Value
Network space	200 m * 200 m
Node signal coverage	33 m
Number of segments per video	15
Cache capacity	3 (constant)
Popularity of Segment i	proportional to 0.95^i
Server access cost (α)	6
Advertisement interval	1 second
Average inter-arrival time	60 s
Average node lifetime	100 min

have not heard from the node for a time substantially longer than the advertisement period. If the node is the *NEXT_HOP* for some entries in the peer's table, the peer invalidates the original entries and replaces them with the best known option in a way similar to Step 3 in the joining process.

Node leave and fail can cause conflicts in segment routing tables. For example, when node v hears an advertisement of node w. For an entry in node v's own table the *NEXT_HOP* is node w and the *COST* is K, but in the advertised table of node w, the same segment is associated with *COST* more than $K-1$. In this case, node v invalidates the entry in conflict and replaces it with the best choice known.

It can be seen that both in the joining and the advertising phase, each caching decision of a node will have a cost reduction of at least 1 to the entire network.[2] Thus, as nodes join and communicate, the network cost will reach a local minimum. We will show the network cost converging quite fast in the simulation study in Section IV.

Although our distributed scheme does not rely on any underlying routing protocol, when routing to a segment, the message containing the segment routing table entries can be piggybacked on AODV route request and route reply packets so as to reduce communication overhead.

IV. ILLUSTRATIVE SIMULATION RESULTS

We have conducted event-driven simulation of OPSEC to study its performance. Users join the network at random locations in a 2D space, connections are symmetric with a certain signal coverage range (same for all nodes). Unless otherwise stated, we use the baseline parameters shown in Table II. User inter-arrival time and lifetime are exponentially distributed. Users advertise their segment routing table regularly. Each user accesses a segment with probability proportional to the segment popularity.

We have *GroupCache* [1] and *HybridCache* [2], two recent segment caching algorithms to compare OPSEC with. For *HybridCache*, we have used threshold parameters that achieves best performance shown in their work (T_h = 40, T_{TTL} = 5000 seconds). In addition, a simple random caching scheme is included as a comparison scheme.

[2]The number of caching decisions a node can make is subject to its cache capacity

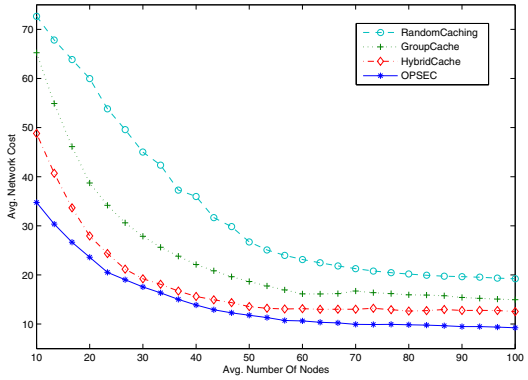

Fig. 2. Comparison of network cost versus number of nodes for different caching algorithms.

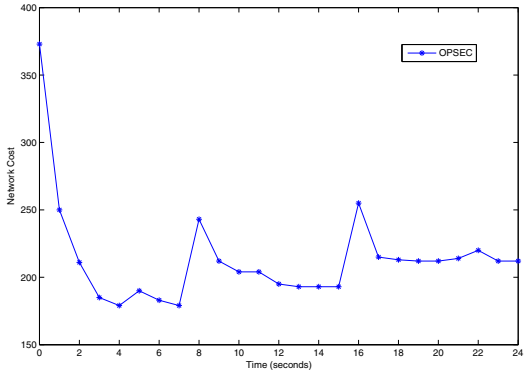

Fig. 3. The convergence of network cost over time.

We mainly focus on two performance metrics: network cost, whose calculation has been specified in Section II; hop count, which is the average hop distance to fetch a segment. [3]

In Figure 2, we plot average network cost for users against network size. Average number of users in the network increases from left to right as we set longer user lifetimes. Cost decreases because more users cache and share segments in the peer-to-peer network. In general, our scheme outperforms all benchmark schemes in reducing cost. With a small network, *HybridCache* also has quite a good performance due to a stronger locality effect that the scheme takes advantage of. However, as the network gets larger our scheme achieves a significantly better optimization of cache selections.

We plot total network cost against time in Figure 3. Cost converges to a local optimal value very quickly as users communicate. Although two temporary rise in cost can be observed due to users leaving (which causes route breaks in the *segment routing table*), it recovers quickly as users reestablish alternative routes. The new stabilized cost is slightly higher than before because total user cache capacity drops.

[3]The "distance" to server is defined as a constant (6 hops away as a baseline parameter).

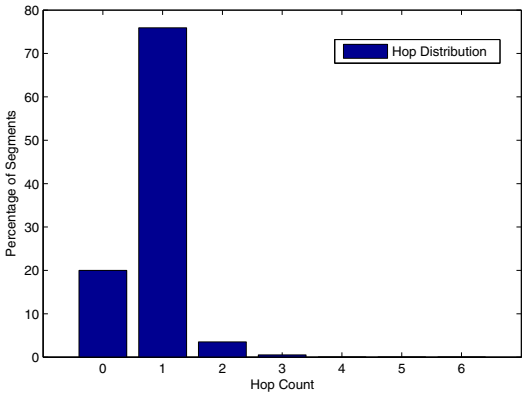

Fig. 4. Distribution of the hop distance to get a segment.

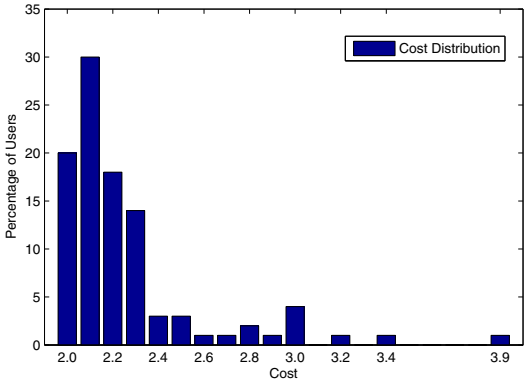

Fig. 6. Cost distribution of individual users.

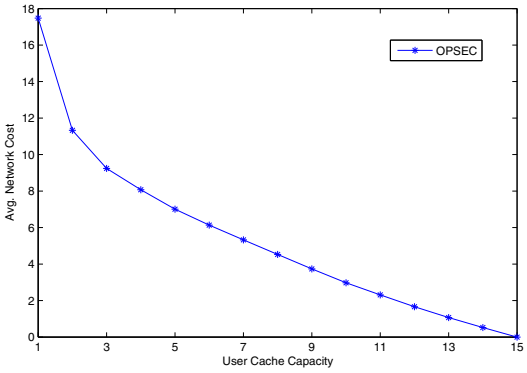

Fig. 5. Network cost versus user cache size.

In Figure 4, we show the distribution of the average number of hops needed to get a segment over all users. It can be observed that most video segments are available either locally or in a one-hop neighborhood. As a result, delay and network resource consumption in viewing the video is minimized.

We plot the average network cost against cache capacity in Figure 5. As cache size increases, a significant gain in network performance indicated by the drop in network cost is witnessed. This demonstrates the sensitivity of our scheme to cache space allocation.

At last, we explore cost distribution in Figure 6. While 90% percent of users experience low cost, there are some participants with cost almost twice as high as the lowest cost. This is partly due to their locations in the network (low connection degree) and partly due to the scheme which does not compensate for users that has been marginalized. How to achieve better fairness in our scheme, however, is out of the scope of this paper and may be pursued as a potential future topic.

V. Conclusion

In this paper, we have presented OPSEC, a distributed algorithm to collaboratively cache video segments in a wireless peer-to-peer network. We have formulated the caching problem as a combinatorial optimization problem and proven it *NP-hard*. We then propose OPSEC, a distributed caching algorithm as an efficient heuristic to address the problem given segment access probabilities.

We compare OPSEC with other recent schemes using simulation. OPSEC is shown to achieve substantially lower network cost with low hop-distance to each segment. It also has fast convergence time when the network is dynamic.

References

[1] Y.-W. Ting and Y.-K. Chang, "A novel cooperative caching scheme for wireless ad hoc networks: Groupcaching," July 2007, pp. 62–68.

[2] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," vol. 4, March 2004, pp. 2537–2547 vol. 4.

[3] F. R. Dogar,, A. Phanishayee,, H. Pucha,, O. Ruwase,, and D. G. Andersen,, "Ditto: a system for opportunistic caching in multi-hop wireless networks," in *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2008, pp. 279–290.

[4] J. Yu, C. T. Chou, X. Du, and T. Wang, "Internal popularity of streaming video and its implication on caching," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1, Apr. 2006, pp. 6pp.–.

[5] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications Special Issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1717–31, Dec. 2007.

[6] H. Badis, "A QoS-aware multicast overlay spanning tree protocol for multimedia applcations in MANETs," Feburary 2008, pp. 242–247.

[7] Y. Abdelmalek, A. El Al, and T. Saadawi, "Collaborative content caching algorithms in mobile ad hoc networks environment," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, pp. 311–314.

[8] D. A. Tran, M. Le, and K. A. Hua, "MobivoD: a video-on-demand system design for mobile ad hoc networks," 2004, pp. 212–223.

[9] H. Yoon, J. Kim, F. Tan, and R. Hsieh, "On-demand video streaming in mobile opportunistic networks," March 2008, pp. 80–89.

[10] Y. Zhu, H. Liu, Y. Guo, and W. Zeng, "Challenges and opportunities in supporting video streaming over infrastructure wireless mesh networks," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, July 2009, pp. 1548–1549.

[11] Y. He, I. Lee, X. Gu, and L. Guan, "Centralized peer-to-peer video streaming over hybrid wireless network," July 2005, pp. 550–553.