International Journal of Computational Intelligence Techniques, ISSN: 0976–0466 & E-ISSN: 0976–0474
Volume 1, Issue 1, 2010, PP-27-31

Distributed heuristic algorithm to optimize the throughput of data driven streaming in peer to peer networks

Velayutham A.S. and Chitra S.
Department of CSE, MKCE, India

Abstract- During recent years, the Internet has witnessed a rapid growth in deployment of data-driven (or swarming based) peer-to-peer (P2P) media streaming. In these applications, each node independently selects some other nodes as its neighbors (i.e., gossip style overlay construction) and exchanges streaming data with the neighbors (i.e., data scheduling). To improve the performance of such protocol, many existing works focus on the gossip-style overlay construction issue. However, few of them concentrate on optimizing the streaming data scheduling to maximize the throughput of a constructed overlay. In this paper, we analytically study the scheduling problem in data-driven streaming system and model it as a classical min-cost network flow problem. We then propose both the global optimal scheduling scheme and distributed heuristic algorithm to optimize the system throughput. Furthermore, we introduce layered video coding into data-driven protocol and extend our algorithm to deal with the end-host heterogeneity. The results of simulation with the real-world traces indicate that our distributed algorithm significantly outperforms conventional ad hoc scheduling strategies especially in stringent buffer and bandwidth constraints.

Introduction

The basic idea of data-driven streaming protocol is very simple and similar to that of Bit-Torrent [7]. The protocol contains two steps. In the first step, each node independently selects its neighbors so as to form an unstructured overlay network, called the gossip-style overlay construction or membership management. The second step is named block scheduling: The live media content is divided into blocks (or segments or packets), and every node announces what blocks it has to its neighbors. Then each node explicitly requests the blocks of interest from its neighbors according to their announcement. Obviously, the performance of data-driven protocol directly relies on the algorithms in these two steps. The scheduling methods used in most of the pioneering works with respect to the data-driven/swarming-based streaming are somewhat ad hoc. These conventional scheduling strategies mainly include pure random strategy [4], local rarest first (LRF) strategy [6] and round-robin strategy [5]. Actually, how to do optimal block scheduling to maximize the throughput of data-driven streaming under a constructed overlay network is a challenge issue. The media streaming is divided into blocks with the equal size, each of which has a unique sequence number. Every node has a sliding window, which contains all the up-to-date blocks on the node and goes forward continuously at the speed of streaming rate. We call the front part of the sliding window exchanging window. The blocks in the exchanging window are the ones before the playback deadline, and only these blocks is requested if they are not received. The unavailable blocks beyond playback deadline will be no more requested. The blocks that have been played are buffered in the sliding window, and they can be requested by other nodes. Every node periodically pushes all its neighbors a bit vector called buffer map in which each bit represents the availability of a block in its sliding window to announce what blocks it holds. Due to the announcement of the neighbors, each node periodically sends requests to its neighbors for the desired blocks in its exchanging window. We call the time between two requests a request period, or period for short, typically 1-6 seconds. Then, each node decides from which neighbor to ask for which blocks at the beginning of each request period. When a block does not arrive after its request is issued for a while and is still in the exchanging window, it should be requested in the following period again In this section, we introduce the block scheduling problem in data-driven P2P streaming. Figs. 1 and 2 give intuitive examples of block scheduling problem, BSP for short. The two numbers beside the pipe of each node represent the

maximum blocks that can be downloaded and uploaded in each request period, respectively, denoting the inbound and outbound bandwidth constraints of each node. The blocks close to each node illustrate what blocks the node currently holds. We compare the LRF scheduling strategy used in [6] and optimal scheduling in these examples. In Fig. 1a, node 4 asks for blocks from node 1, 2, and 3. In the LRF scheduling strategy used in [6], a block that has the minimum number of holders among the neighbors is requested first. If multiple neighbors hold this block, it is assigned to the one with the maximum surplus bandwidth in turn. As illustrated in Fig. 1a, using LRF, block 3 has only one holder, so it is assigned to node 3. Then, the surplus upload bandwidth of node 3 is reduced to 1. After that, block 2 is assigned to node 2 since the surplus bandwidth of node 2 is larger than node 1 and the surplus bandwidth of node 2 becomes 1. Next, LRF strategy assigns block 4 to node 1 whose surplus bandwidth then descends to 0. Finally, after node 2 gets block 5, no more blocks can be further assigned. Fig. 1a shows the scheduling result using LRF. Four blocks are delivered. On the other hand, one optimal scheduling solution is illustrated in Fig. 1b, and five blocks can be delivered with a gain of 25 percent compared to LRF. In fact, using LRF method usually cannot derive the maximum throughput. As shown in Fig. 1a, the upload bandwidth at node 3 and download bandwidth at node 4 are not fully utilized. Fig. 2a shows another scenario that node 3 and 4 may competitively request blocks from node 1, that is, their requests are congested at node 1. However, node 4 has more options. The optimal way is that node 4 requests blocks from node 2, while node 3 requests from node 1, as shown in Fig. 2b. In fact, compared to LRF, random strategies used in [4] would be even worse. Moreover, the real situation is more complex because the bandwidth bottlenecks are not only at the last mile but also different blocks have different importance. As a consequence, more intelligent scheduling algorithms should be developed to improve the throughput of data-driven protocol under bandwidth constraints.

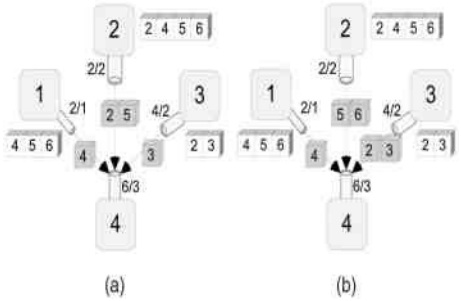

Fig. 1- Illustration of the block scheduling problem (I). (a) LRF scheduling. (b) Optimal scheduling.

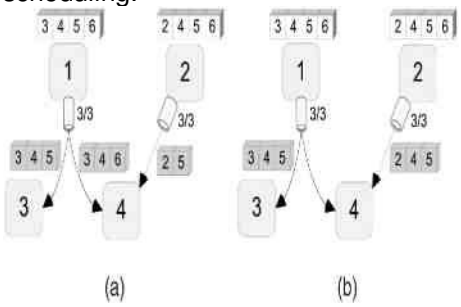

Fig. 2- Illustration of block scheduling problem (II). (a) LRF scheduling. (b) Optimal scheduling.

Performance Evaluation

As aforementioned, there are two key steps, i.e., the overlay construction and the block scheduling, in data-driven peer to-peer streaming, and we focus on the block scheduling step. For a fair comparison, all the experiments use the same simple algorithm for overlay construction: each node independently selects its neighbours randomly so that a random graph is organized. And, our simulation ensures that each node has the same set of neighbours at any simulation time for every method. Moreover, to evaluate the performance, We define a metric—delivery ratio formally here. The delivery ratio of a node is represented by the number of blocks that arrive at the node before playback deadline over the total number of blocks encoded in the stream. Since the total number of blocks in the stream is constant that only relies on the encoding and packetization, the delivery ratio of a node can represent the throughput from the source to this node. The delivery ratio of the whole session is measured as the average delivery ratio among all nodes, also representing the throughput of the session. For the underlying topology, we use the random model of GT-ITM [8] to generate

a topology with 2,000 routers and set delays proportional to the distance metric of the resulting topology within [5 ms, 300 ms]. In our experiment, we implement a discrete event-driven peer-to-peer simulator1 and use Goldberg's "CS2" library [7] to solve min-cost network flow problem. As suggested in [4] and [5], the request period should be several seconds. In all the experiments, we hence fix the request period to 3 seconds. And, the default group size of the whole session is 1,000 nodes. Previous study [3] has bandwidth of the source node to 2 Mbps. 0 2000 4000 8000 10000 12000 shown that there is a sweet range of neighbour count or peer degree roughly between 6 to 14 in data-driven/swarming-based streaming where the delivered quality to the majority of peers is high, and actually, it is the overlay construction issue. Therefore, in our simulation, each node randomly selects 14 other nodes as its neighbours. Each block has the same size of 1,250 bytes, i.e., 10 Kbits. Each node estimates the bandwidth allocated from a neighbour with the traffic received from it in the past five periods, namely, M ¼ 5. Moreover, the default aggressive coefficient used is _ ¼ 1:5. We set the default exchanging window size to 10 seconds and the sliding window to 1 minute. The cumulative distribution of the user online time is shown in Fig. 3. Hence, we can employ the realistic arrival/leave patterns in the traces to simulate the churn of the participating nodes. In our experiment, each run of the simulation is driven by the same part of traces on that night, i.e., the 30-minute traces. As our default group size is set to 1,000, new node joining request is refused in our simulation, when the total online nodes exceed 1,000. Besides, in all of our experiments, we set the outbound

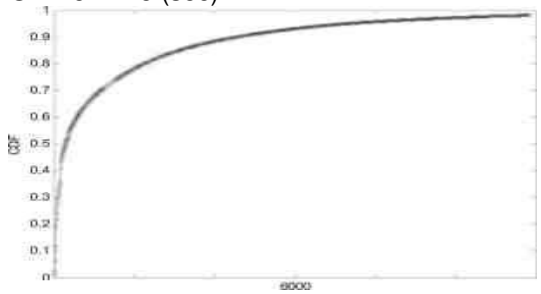

Fig. 3- CDF of user online time obtained from a real-world peer-to-peer streaming system [1].

Performance Comparison for Single Rate

We show the performance of our proposed algorithms, including the global optimal and distributed algorithm, and give the comparison with the following conventional ad hoc methods in block scheduling: Pure random method. Each node will assign each desired block randomly to a neighbor that holds that block. LRF method. A block that has the minimum owners among the neighbors will be requested first. Round-robin (RR) method. All the desired packets will be assigned to one neighbor in a prescribed order in a round-robin way. If the block is only available at one sender, it is assigned to that sender. Otherwise, it is assigned to a sender that has the maximum surplus available bandwidth. When a block is assigned to a neighbor, the surplus bandwidth of that neighbor will be recalculated by subtracting the amount the block consumes. These steps are repeated till there is no surplus bandwidth or no blocks can be assigned. We first use a simulation-based approach to show that the proper value of _ in priority definition (1) should be 1. We set all nodes to DSL/Cable nodes and set the streaming rate to 500 Kbps. Fig. 4 shows the delivery ratio under different value of _ 2 ½0; 1_. In this section, we set _ ¼ 1 in all of our experiments. In Fig. 5, we study the performance of each method when all the nodes are DSL/Cable nodes. This scenario is frequent. For example, in the online classes of some distant education institute in China, such as CRTVU [8], most of the students access Internet through DSL from their home. Moreover, the exchanging window size and the sliding window is set to 10 seconds and 1 minute, respectively. We see that when the streaming rate is 250 Kbps, all the methods except Narada have very high delivery ratio usually above 90 percent. As the streaming rate increases, the delivery ratio the global optimal solution keeps around 100 percent. Even when the streaming rate reaches 500 Kbps, its delivery ratio still remains above 97 percent. This reveals that the network capacity is sufficient to support the multicast session with 250-500 Kbps.

We note that the performance of the three compared methods (LRF, Round-Robin, and Random) goes down fast with the increase of the streaming rate; however, the delivery

ISSN: 0976–0466 & E-ISSN: 0976–0474, Volume 1, Issue 1, 2010

ratio of our proposed heuristic distributed algorithm is fairly good. At the rate of 500 Kbps, the distributed algorithm outperforms the LRF, Round-Robin, and Random methods by gains of 21 percent, 52 percent, and 62 percent, respectively. The gap between the global optimal solution and the heuristic distributed algorithm is 9 percent. We can also see that the delivery ratio of Narada protocol is low because the traditional single tree-based protocol cannot effectively utilize the outbound bandwidth of all the peers.

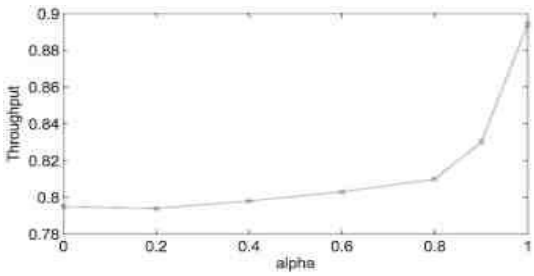

Fig. 4- Delivery ratio for different value deviation of a cluster is 10 percent of its mean Streaming Rate (Kbps)

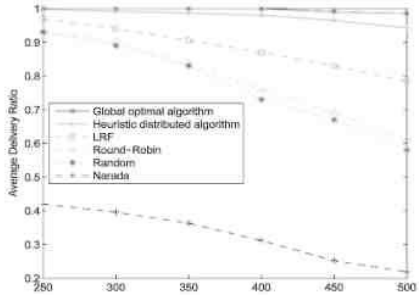

Fig. 5- Delivery ratio for different value of streaming rate.

Performance Comparison for Multiple Rates

In this section, we check the performance of each method when we encode the video into multiple rates using layered video coding.With the assistance of layered coding, the video rate can adapt to different bandwidth capacity, and all types of users can be supported in one session. As a consequence, in terms of the fractions shown in Table 4, we add an additional cluster of low-capacitated DSL/Cable users whose inbound and outbound bandwidth are 384 Kbps and 128 Kbps, respectively, with a fraction of 0.1. We assume that the percentage of Ethernet users is 10 percent. The group size is set to 1,000. As aforementioned, the bandwidth of each cluster follows a Gaussian distribution, and the standard

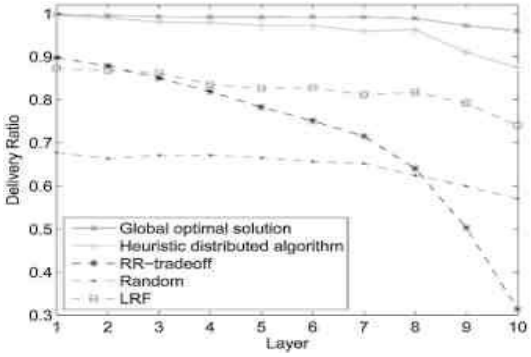

Fig. 6- The video is encoded into 10 layers.

The group size is 1,000. Fig. 6 gives the delivery ratio at each layer. We note that the global optimal solution has the best performance, and the delivery ratio in all layers is nearly 1. This demonstrates that the generated topologies have sufficient capacity to support all the nodes to receive all layers that they can achieve. The performance of distributed algorithm is fairly good. Most of the delivery ratio in lower layers has nearly 1 and most in higher layers is also above 0.9.

For the priority in (7), we set _ as a large value 1,000 and define function PLðIÞ ¼ 102ðL_IÞ to ensure the lower layers have much larger priority than the upper layers. And, the compared methods include the following ones: Random method and LRF method . RR method On the contrary, aggressive block ordering scheme requests blocks of all layers with lowest sequence number (or time stamp) pre-emptively. Since the first two schemes evidently have its own limitations [5]. We only compare with the third scheme (we call it RR-trade-off for short).

Conclusion and Future Enhancements

In this paper, we study the scheduling problem in the datadriven/ swarming based protocol in peer-to-peer streaming. The contributions of this paper are twofold. First, to the best of our knowledge, we are the first to theoretically address the scheduling problem in data-driven protocol. Second, we give the optimal scheduling algorithm under different bandwidth constraints, as well as a distributed heuristic algorithm, which can be practically applied in real system and outperforms conventional ad hoc strategies

by about 10 percent-70 percent in both single rate and multirate scenario. For future work, we will study how to maximize the blocks delivered over a horizon of several periods, taking into account the interdependence between the periods, and will analyze the gap between the global optimal solution and the proposed distributed algorithm.

We also would like to study how to adapt the parameter _ in terms of the network condition in our proposed distributed algorithm. Besides, based on our current results, we are interested in combing the video coding technique to our algorithm to further improve the user perceived quality.

References

[1] V. Pai et al., "Chainsaw: Eliminating Trees from Overlay Multicast," Proc. IEEE INFOCOM '05, Feb. 2005.

[2] V. Agarwal and R. Rejaie, "Adaptive Multi-Source Streaming in Heterogeneous Peer-to-Peer Networks," Proc. Multimedia Computing and Networking (MMCN '05), Jan. 2005.

[3] X. Zhang, J. Liu, B. Li, and T.-S.P. Yum, "Coolstreaming/Donet: A Data-Driven Overlay Network for Efficient Media Streaming," Proc. IEEE INFOCOM '05, Mar. 2005.

[4] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A Peer-to-Peer Network for Live Media Streaming Using a Push-Pull Approach," Proc. ACM Multimedia, Nov. 2005.

[5] N. Magharei and R. Rejaie, "Prime: Peer-to-Peer Receiver-Driven Mesh-Based Streaming," Proc. IEEE INFOCOM '07, May 2007.

[6] B. Cohen, http://bitconjuer.com, 2008.

[7] V. Venkataraman and P. Francis, "Chunkyspread: Multi-Tree Unstructured End System Multicast," Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '06), Feb. 2006.

[8] Venkataraman and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks," Proc. IEEE INFOCOM '06, Apr. 2006.

[9] J. Jiang and K. Nahrstedt, "Randpeer: Membership Management for QoS Sensitive Peer-to-Peer Applications," Proc. IEEE. INFOCOM '06, Apr. 2006.

ISSN: 0976–0466 & E-ISSN: 0976–0474, Volume 1, Issue 1, 2010