Host-Aware Routing in Multicast Overlay Backbone

Author:
Guo, Jun; Jha, Sanjay

Publication details:
IEEE/IFIP NOMS 2008, Proceedings of
pp. 915-918
9781424420667 (ISBN)

Event details:
20th IEEE/IFIP Network Operations and Management Symposium
Salvador, Bahia, Brazil

Publication Date:
2008

DOI:
https://doi.org/10.26190/unsworks/382

Host-Aware Routing in Multicast Overlay Backbone

Jun Guo, Sanjay Jha
School of Computer Science and Engineering
The University of New South Wales, Australia
Email: {jguo, sjha}@cse.unsw.edu.au

Abstract— **To support large-scale Internet-based broadcast of live streaming video efficiently in content delivery networks (CDNs), it is essential to implement a cost-effective overlay multicast mechanism by exploiting peer-to-peer distribution capabilities among end hosts. This way, the access bandwidth demand on CDN servers in the multicast overlay backbone can be largely reduced. Such a streaming infrastructure gives rise to an interesting host-aware routing problem (HARP). For a live streaming video broadcast event, each participating CDN server is made aware of the largest delay from it to end hosts within its service area. The problem is to optimize routing among CDN servers in the multicast overlay backbone such that the de facto maximal end-to-end latency from the origin server to all end hosts is minimized subject to access bandwidth constraints on CDN servers. In this paper, we frame HARP as a constrained spanning tree problem which is shown to be NP-hard. We present a distributed algorithm for HARP. Simulation experiments confirm that our proposed algorithm converges to good quality solutions that are close to the optimum.**

I. INTRODUCTION

Live streaming video broadcast has become a killer application in content delivery networks (CDNs) [1]. For example, in July 2005, Yahoo!'s broadcast of NASA's Shuttle Discovery launch reached more than 335,000 simultaneous online viewers at its peak. For such a popular Internet multimedia application which is both resource-intensive and delay-sensitive, a challenging design issue is how to address large-scale live streaming video broadcast efficiently in CDNs.

Commercial CDNs such as Akamai Technologies [2] adopt the infrastructure-based overlay multicast mechanism to support Internet-based live streaming video broadcast. This approach relies on a set of geographically distributed and dedicated proxy servers with large processing power and high fanout capability. They are typically placed at co-location facilities with high-speed connection to the Internet. During a live streaming video broadcast session, these application-layer servers create an overlay tree among themselves via unicast connections and form a backbone service domain, which we call *multicast overlay backbone*, for the overlay multicast network.

Akamai's streaming solution largely benefits from the wide reach and large capacity of the Akamai platform, a CDN that consists of over 20,000 servers distributed in more than 70 countries. The solution is, however, a pure infrastructure-based approach, where each user is directly connected to one CDN server in the multicast overlay backbone to fetch the live streaming contents [2]. A more flexible and scalable approach is to exploit peer-to-peer distribution capabilities among end hosts [3], which leads to a two-tier overlay multicast architecture as depicted in Fig. 1. This way, the access bandwidth demand on CDN servers in the multicast overlay backbone can be largely reduced.

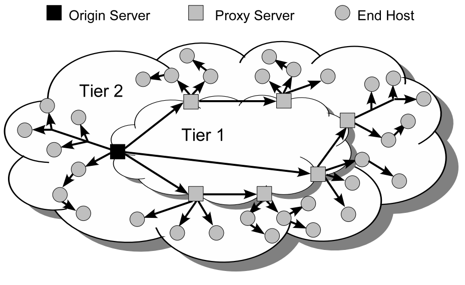

Fig. 1. Two-tier overlay multicast architecture.

The two-tier overlay multicast architecture gives rise to an interesting routing problem in the multicast overlay backbone. For a live streaming video broadcast event, each participating CDN server is made aware of the largest delay from it to end hosts within its service area. The problem is to optimize routing among CDN servers in the multicast overlay backbone so that the de facto maximal end-to-end latency from the origin server to all end hosts is minimized subject to access bandwidth constraints on CDN servers. We call such a problem as the host-aware routing problem (HARP) in the multicast overlay backbone. HARP is strongly motivated since the last-mile latency between end hosts and their corresponding proxy servers can be quite significant [3]. In this paper, we frame HARP as a constrained spanning tree problem and show that it is NP-hard. We present a distributed algorithm for HARP. We also provide a genetic algorithm (GA) to validate the quality of the distributed algorithm. Simulation experiments confirm that the distributed algorithm converges to good quality solutions that are close to the optimum.

The remainder of this paper is organized as follows. Section II deals with the problem formulation. In Section III, we present the distributed algorithm. In Section IV, we provide the GA. Simulation experiments are reported in Section V. Finally, we provide concluding remarks in Section VI.

II. PROBLEM FORMULATION

We model the multicast overlay backbone as a complete directed graph $G = (V, E)$, where V is the set of N nodes and $E = V \times V$ is the set of edges. Each node in V represents a CDN server participating in the live streaming video broadcast

978-1-4244-2066-7/08/$25.00 ©2008 IEEE

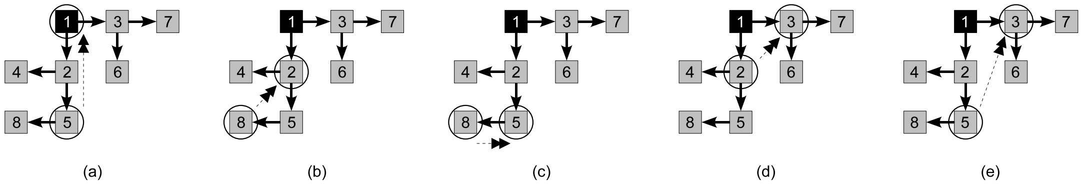

Fig. 2. Snapshot of the five different cases where an $i \rightarrow j$ transaction request from node i will be considered by node j. Both node i and node j are marked by circles. Node j is pointed by the double arrow.

session. Let node r be the origin server. All other nodes in $V - r$ are noted as proxy servers. The directed edge $\langle i, j\rangle$ in E from node i to node j represents the unicast path of latency $l_{i,j}$ from node i to node j. By $\{l_{i,j}\}$, we denote the matrix of unicast latency quantities between each pair of nodes in G.

An overlay backbone routing tree can be represented by a directed spanning tree T of G rooted at node r. For each sink node in the set $V - \{r\}$, we define $R_{r,v}$ as the set of directed edges that form the overlay routing path from the root node to node v in the multicast overlay backbone. Let $L_{r,v}$ denote the latency of the overlay routing path from the root node to node v. Let γ_v denote the maximum latency from node v to current end hosts within its service area. Let $L_{\max}$ denote the maximum host-aware end-to-end delay. Given the unicast latency matrix $\{l_{i,j}\}$, we readily have

$$L_{r,v} = \sum_{\langle i,j\rangle \in R_{r,v}} l_{i,j} \tag{1}$$

and

$$L_{\max} = \max_{v \in V - \{r\}} (L_{r,v} + \gamma_v) \ . \tag{2}$$

Let $\widehat{d_i}$ denote the residual degree of node i. Let d_i denote the out-degree of node i counted within the overlay backbone tree only. Since a spanning tree with N nodes has exactly $N - 1$ edges, the sum $\sum_{i \in V} d_i$ of out-degrees in the overlay backbone tree is $N - 1$.

Definition 1: Given the complete directed graph $G = (V, E)$, HARP is to find a constrained directed spanning tree T of G rooted at node r, such that $L_{\max}$ is minimized, and T satisfies the residual degree constraint, i.e. $d_i \leq \widehat{d_i}$, $\forall i \in V$.

Such a constrained spanning tree problem is NP-hard. This is shown by creating a dummy node for each node v in V and forming an edge of weight γ_v between node v and its corresponding dummy node. The resulting problem can be reduced to finding a Hamiltonian path within the augmented graph which is known to be NP-complete [4].

III. DISTRIBUTED ALGORITHM

Our proposed distributed algorithm for HARP requires each node i in V to maintain the following state information during the iterative tree update process except for those inapplicable to the root node:

- $\widehat{i}$: Parent node of node i in the current tree.
- Φ_i: Set of ancestor nodes in the overlay path from the root node to node i.
- Ω_i: Set of child nodes of node i in the current tree.
- d_i: Out-degree of node i in the current tree, which is equivalent to the cardinality of the set Ω_i.
- K_i: Number of aggregate descendant nodes in the subtree rooted at node i, which is given by

$$K_i = \begin{cases} 0, & \text{if node } i \text{ is a leaf node} \\ \sum_{j \in \Omega_i} (K_j + 1), & \text{elsewhere} \end{cases} \tag{3}$$

- $L_{r,i}$: Latency from the root node to node i, which is given by

$$L_{r,i} = L_{r,\widehat{i}} + l_{\widehat{i},i} \ . \tag{4}$$

- Γ_i: Maximum host-aware latency from node i to end hosts within the service areas of all nodes (inclusive of node i) in the subtree rooted at node i. This can be iteratively computed by

$$\Gamma_i = \begin{cases} \gamma_i, & \text{if node } i \text{ is a leaf node} \\ \max_{j \in \Omega_i} (l_{i,j} + \Gamma_j), & \text{elsewhere} \end{cases} \tag{5}$$

Starting from the initial tree including the root node only, we let nodes in $V - r$ join the tree in a random order, which is likely to be realistic in an actual live streaming video broadcast session. Should multiple nodes arrive concurrently, they are added to the tree in the decreasing order according to their node IDs. When node i wishes to join the tree, it obtains the list of all nodes in the current tree from the root node. For each node j in the current tree, node i requests and obtains the $L_{r,j}$ value as well as the measurement of $l_{j,i}$ from node j, given that $\widehat{d_j} > 0$. Node i chooses node j with the smallest value on $L_{r,j} + l_{j,i}$ as its parent node in the initial tree.

Let Δ denote the time period of each iteration, during which nodes contact each other to make *transaction* requests for tree update. This also allows each node i to measure the unicast latency $l_{j,i}$ from each other node j to node i. Let $i \rightarrow j$ denote a transaction request from node i to node j. Following our distributed algorithm, if $i \in \Phi_j$, $i \rightarrow j$ will be rejected by node j. Node j considers $i \rightarrow j$ only if the current positions of node i and node j in the tree match one of the five different cases illustrated in Fig. 2.

a) If $j = r$, $j \neq \widehat{i}$ and $d_j > 0$, a Type-C transaction (see Section III-C) will be attempted by node j.
b) If $j \in \Phi_i$, $j \neq \widehat{i}$ and $d_{\widehat{j}} > 0$, a Type-A transaction (see Section III-A) will be attempted by node j. If $d_{\widehat{j}} = 0$ but $d_j > 0$, node j will instead attempt a Type-C transaction.

c) If $j \in \Phi_i$, $j = \widehat{i}$ and $d_{\widehat{j}} > 0$, a Type-A transaction will be attempted by node $\widehat{j}$.

d) If $\widehat{j} = \widehat{i}$, a Type-C transaction will be attempted by node j provided $d_j > 0$. However, if $d_j = 0$, node j will instead attempt a Type-D transaction (see Section III-D).

e) In all other situations, node j will subsequently attempt a Type-A, B (see Section III-B), C or D transaction depending on if the condition $d_{\widehat{j}} > 0$, $d_{\widehat{j}} = 0$, $d_j > 0$, or $d_j = 0$ holds true.

A. Type-A transaction

Since $d_{\widehat{j}} > 0$, node i (together with its subtree if any) is switched to be a child node of node $\widehat{j}$ given that

$$\breve{L}_{r,i} = (L_{r,\widehat{j}} + l_{\widehat{j},i}) - (L_{r,\widehat{i}} + l_{\widehat{i},i}) < 0 \ , \tag{6}$$

where $\breve{L}_{r,i}$ denotes the amount of variation on $L_{r,i}$. This transaction reduces the end-to-end latency of all nodes in the subtree rooted at node i.

B. Type-B transaction

Since $d_{\widehat{j}} = 0$, if (6) holds true, node j will check if

$$\breve{L}'_{r,j} = (L_{r,\widehat{i}} + l_{\widehat{i},j}) - (L_{r,\widehat{j}} + l_{\widehat{j},j}) < 0 \ . \tag{7}$$

It will also check if

$$\breve{L}''_{r,j} = (\breve{L}_{r,i} + l_{i,j}) - (L_{r,\widehat{j}} + l_{\widehat{j},j}) < 0 \ , \tag{8}$$

given that $d_i > 0$, so that it is possible that node j can reduce its latency by switching itself to be a child node of node i after node i is connected to node $\widehat{j}$.

If neither (7) nor (8) holds true, but

$$(K_i + 1) \cdot \left|\breve{L}_{r,i}\right| > (K_j + 1) \cdot \min(\breve{L}'_{r,j}, \breve{L}''_{r,j}) \ , \tag{9}$$

node j will proceed with the transaction so long as $L_{\max} = \Gamma_r$ of the current tree will not be increased after the tree update.

C. Type-C transaction

This transaction is similar to the Type-A transaction by instead checking if node i can obtain

$$\breve{L}_{r,i} = (L_{r,j} + l_{j,i}) - (L_{r,\widehat{i}} + l_{\widehat{i},i}) < 0 \tag{10}$$

by being connected to node j.

D. Type-D transaction

Given that $d_j = 0$ in this case and (10) holds true, for all $c \in \Omega_j$, node j will check if node c can obtain either

$$\breve{L}'_{r,c} = (L_{r,\widehat{i}} + l_{\widehat{i},c}) - (L_{r,j} + l_{j,c}) < 0 \ , \tag{11}$$

or

$$\breve{L}''_{r,c} = (\breve{L}_{r,i} + l_{i,c}) - (L_{r,j} + l_{j,c}) < 0 \ , \tag{12}$$

given that $d_i > 0$. Let c' denote the node that minimizes $\min(\breve{L}'_{r,c}, \breve{L}''_{r,c})$.

If neither (11) nor (12) holds true, but

$$(K_i + 1) \cdot \left|\breve{L}_{r,i}\right| > (K_{c'} + 1) \cdot \min(\breve{L}'_{r,c'}, \breve{L}''_{r,c'}) \ , \tag{13}$$

node j will approve the transaction so long as $L_{\max} = \Gamma_r$ of the current tree will not be increased after the tree update.

IV. GENETIC ALGORITHM

GAs are population-based stochastic search and optimization approaches inspired by the mechanism of natural selection which obeys the rule of "survival of the fittest" [5]. GAs have been extensively used for solving various real-world complex optimization problems due to their broad applicability, ease of use and global perspective. In particular, they have found successful applications in tree-based network optimization problems (see e.g. [6] and references therein). In all cases, it was observed that GAs can achieve near-optimal solutions for fairly large-size problem instances with reasonable computational effort. In this paper, we have chosen to implement a weight-coded GA [7] for HARP, since it uses a node-based encoding approach for chromosome representation of candidate solutions to the problem, rather than an edge-based encoding approach which is inefficient in the context of a complete graph.

Implementing a weight-coded GA for HARP requires a heuristic method for computing the overlay backbone tree. The following low complexity tree computation algorithm suffices for this purpose. Starting from the initial tree including the root node only, at any iteration, for each unconnected node v, we find node u with $\widehat{d_u} > 0$ in the partial tree constructed so far which minimizes $\delta_v = L_{r,u} + l_{u,v}$. We then identify node v with the smallest value on such δ_v, and add it to the tree by creating a directed edge from node u to node v.

V. SIMULATION EXPERIMENTS

The network topologies used in our experiments are obtained from the transit-stub graph model of the GT-ITM topology generator [8]. All topologies are with 12,000 nodes. CDN servers are placed at a set of nodes, chosen uniformly at random. Here we report experiment results for three topologies ($N = 100,\ 200,\ 300$). Unicast latencies between different pairs of nodes in these graphs vary from 10 to 1000 msec. For each i, the value γ_i of node i is randomly generated. In all experiments, we conduct the weight-coded GA with ten independent runs and evaluate the performance of the distributed algorithm against that of the weight-coded GA.

Since HARP is NP-hard and our proposed distributed algorithm uses a random approach for tree initialization, independent runs of the distributed algorithm for the same experiment setting converge to different solutions. In order to examine the quality of the distributed algorithm, we randomly select 20 nodes as the root nodes for each topology. For each such experiment setting, we solve the distributed algorithm for 100 independent runs each of which for 100 iterations. We compare $L_{\max}$ of each solution against that of the weight-coded GA. The data points are presented in the form of percentage deviation. We also note the last iteration where tree update occurs. This is for us to investigate the convergence property of the distributed algorithm. All data points are presented in the form of cumulative percentage in Fig. 3 for the solution quality on $L_{\max}$ and in Fig. 4 for the convergence time.

To reduce the control message overhead at any time instant in the multicast overly backbone, we adopt a probabilistic

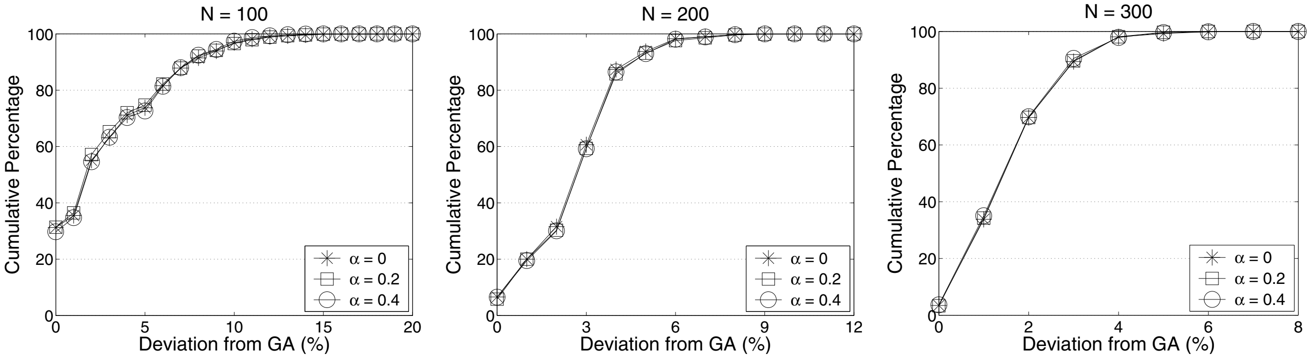

Fig. 3. Solution quality on $L_{\max}$ of the distributed algorithm.

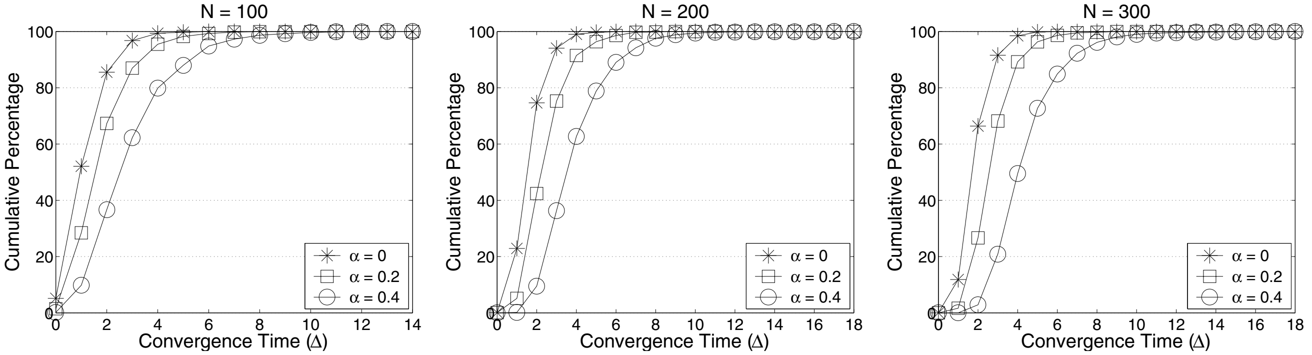

Fig. 4. Convergence time of the distributed algorithm.

approach to restrain the amount of transaction requests in each iteration. To that end, we introduce a parameter α in the range of $[0, 1)$. When a node wishes to make a transaction request for tree update, it randomly draws a probability β within the range of $[0, 1)$. The node will deliver the transaction request only if $\beta > \alpha$. In our experiments, we consider $\alpha = 0,\ 0.2,\ 0.4$ to study the solution quality on $L_{\max}$ and convergence time of the distributed algorithm.

We observe in Fig. 3 that the majority of the solutions from the distributed algorithm fell within 8% of those from the weight-coded GA. Interestingly, the $L_{\max}$ results are not affected by the parameter α. Even under rather high probability of restraining the amount of transaction requests in each iteration, our proposed distributed algorithm statistically converges to solutions of comparable quality in all circumstances, albeit at the expense of slightly larger convergence time as shown in Fig. 4.

VI. CONCLUSION

We have identified and addressed a strongly motivated host-aware routing problem to support large-scale Internet-based live streaming video broadcast in CDNs. Utilizing the knowledge of the largest delay from each participating CDN server to end hosts within its service area, HARP optimizes the overlay routing among CDN servers in the multicast overlay backbone, so that the de facto maximal end-to-end latency from the origin server to all end hosts can be significantly reduced subject to access bandwidth constraints on CDN servers. We have proposed a distributed algorithm for solving HARP in a distributed iterative way. Simulation experiments have confirmed that our proposed algorithm can yield sufficiently good quality solutions with small convergence time.

ACKNOWLEDGMENT

This project is supported by Australian Research Council (ARC) Discovery Grant DP0557519.

REFERENCES

[1] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Proc. ACM IMC 04*, Taormina, Sicily, Italy, Oct. 2004, pp. 41–54.

[2] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep./Oct. 2002.

[3] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in *Proc. Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW 2006*, Edinburgh, Scotland, May 2006 2006.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman, 1979.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[6] S.-M. Soak, D. W. Corne, and B.-H. Ahn, "The edge-window-decoder representation for tree-based problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 124–144, Apr. 2006.

[7] J. Guo and S. Jha, "Placing multicast proxies for Internet live media streaming," in *Proc. IEEE LCN 07*, Dublin, Ireland, Oct. 2007, pp. 149–156.

[8] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM 96*, vol. 2, San Francisco, CA, USA, Mar. 1996, pp. 594–602.