2013 International Conference on Computing, Networking and Communications, Multimedia Computing and Communications Symposium

Multi-Source IPTV Networks: Zap Time and Bandwidth Optimization

Daniel Bailey, Yuh-Rong Chen, Sridhar Radhakrishnan
School of Computer Science
University of Oklahoma
Norman, Oklahoma 73019
{dbailey, yrchern, sridhar}@ou.edu

Suleyman Karabuk
School of Industrial and Systems Engineering
University of Oklahoma
Norman, Oklahoma 73019
karabuk@ou.edu

***Abstract*—Broadcast television viewing over the Internet (IPTV) is becoming commonplace. Multicasting trees serve as an efficient mechanism to deliver streaming data as each internal node duplicates the packets it receives and sends it along to its children which eventually delivers them to the clients. Given a set of multicasting trees whose roots are servers capable of broadcasting a set of distinct channels, and a set of clients (which are not part of the multicasting trees) each with a set of requested channels, our goal is to determine for each client for each of its channel request, a node (contact node) in the appropriate multicast tree (that serves the channel). The contact nodes are determined in such a way that certain optimization constraints are taken into consideration and satisfied. We have provided Integer Programming (IP) models and heuristics to find these contact nodes in order to optimize constraints on zap time and bandwidth utilization. The proposed IP model is novel and the polynomial-time heuristic provide a fairly good solution in a short amount of time.**

I. Introduction

In recent years, Internet video streaming services have become increasingly prevalent and will continue to become even more so with the continued spread of devices capable of accessing the internet. These Internet Television (IPTV) services can be divided into two groups: video on demand (VOD) (eg. Netflix, Hulu, etc) and live broadcast (eg. Hulu-Live, ESPN, CNN, etc). As evidenced by Hulu, many of these content providers supply both types of service.

IPTV has a number of advantages over traditional broadcast mediums. Amongst these advantages is the diversity of the equipment that a subscriber may use to access the service. In traditional broadcast over the air or cable, each subscriber is required to have a set-top box (STB) which acts as an interpreter for the data stream and provides an interface for the end user to interact with. On the other hand, A STB is not required for IPTV as any device with Internet connectivity and an appropriate app installed may be used to view the channel.

One of many challenges that arises in an IPTV system is what is known as channel *zap time* which significantly impacts a clients Quality-of-Experience (QoE) [1]. Zap time is defined as the time it takes for a user to begin to receive a new channel after it is requested. In analog cable and broadcast television, this problem does not exist as a client only needs to receive the next video frame for the new channel to appear thus zap times are generally less than 200 ms [2]. Additionally, the number of demands for a particular channel has no effect on its zap time for a particular client [2]. In IPTV, there are a number of factors that contribute to a clients zap time, which make it larger than in traditional mediums. These factors include network latency, I-Frame (Intra-Frame) acquisition, buffering and multicast group join times [3]. It is desirable to keep zap time as minimal as possible to ensure the best possible QoE. Kooij et al. [4] suggest a time of less than $.43$ seconds for a good user QoE.

In this paper, we assume a network containing a set of multicasting trees has already been constructed and is provided as part of the input. These trees can be overlapping and the root of the each multicasting tree represents a server that is capable of distributing (or serving) a set of distinct channels. We are given a set of clients each of which has a set of channel requests. These clients are not part of the multicasting tree and a client can obtain a channels streaming data by connecting to an appropriate node of the multicasting tree that serves that particular channel. Once all the channel requests of all the clients are satisfied, the subtrees of each of the multicasting trees that do not serve any channels can be pruned. Our goal in this paper is to develop IP models and heuristics to determine the node(s) (contact node) in the multicasting trees that each of the client should connect to receive the channel data in such a way that it satisfies certain optimization constraints including zap time and bandwidth.

When a client makes a channel change request (say to channel k), the client has to connect to the appropriate node a, that is part of the multicasting tree T (with root A) which serves that channel. Let x be the node in the multicasting tree T that is currently receiving k and is also closest to a during the upwards march from a to A (note that x can be A). Now the zap time is the time for the request packet to travel from the client to node a, and from node a to node x, and the time it takes for the client to receive the first packet corresponding to channel k from x via a. For the sake of simplicity, we only consider the network latency aspect of zap time. We also assume that each channel requires one unit of bandwidth on a link and the total bandwidth consumed on a link is equal to the number of channels that flow through that link.

The goal of this work is to extend previous work in client-channel-server assignment by Long et al. [5]. Two major

978-1-4673-5288-8/13/$31.00 ©2013 IEEE

differences between this research and previous work are as follows. First, in the original work all the channels originate from a single location, while we assume multiple sources provide distinct channels in this paper. Second, minimizing the total bandwidth usage on the network may result in high bandwidth utilization for some links. Hence, we consider the bandwidth consumption on a per link basis, i.e, *residual bandwidth* which is the left over bandwidth on each link. We are also interested in one of the side effects of this model, which is improving the load (client channel requests) distribution across the networks servers. By considering the residual bandwidth rather than load distribution we are able to model this problem using Linear Integer Programming rather than the more computationally intensive Nonlinear Integer Programming. We extend the Integer Programming (IP) model in [5] and develop new heuristics for this problem.

The rest of this paper is organized as follows. Related works are reviewed in Section II. We briefly describe the work by Long et al. [5] and our extension in Section III. IP formulations and heuristic algorithms are introduced in Section IV and Section V respectively. Finally, performance evaluation and results are presented in Section VI with conclusions drawn in Section VII.

II. Related Works

Several schemes have been proposed to improve zap time. Banodkar et al. [6] suggest a secondary channel change stream of a lesser quality to improve zap times. Whenever a new channel is requested, the client would access the lower quality stream before it begins buffering of the higher quality stream in the background. Zap time is decreased by 50% with the tradeoff being a reduced video quality at the beginning of the channel change. Begen et al. [2] presented a retransmission and caching scheme for improving zap time. A retransmission server is set to be joined to the appropriate multicast group so that it may cache key information (data scheme, I-Frames, encryption, etc) so that clients may have immediate access to said data. This retransmission server is contacted by clients prior to attempting a multicast join in order to attempt a *Rapid Synchronization* [2].

Multicast tree packing is a subject similar to the discussed topic which has been subject to a great deal of research. Ravindran et al. [7] developed a model and provided heuristics for the problem of merging two or more tree to minimize overall bandwidth consumption. Chen et al. [8] and Lee and Cho [9] developed IP models together with heuristics to merge multicasting trees for maximizing the minimum residual bandwidth. While these researches attempt to construct and pack the trees in the network, for the problem we consider, we are given a set of multicast trees (for different sets of channels) and our goal is to find a way to distribute data throughout the trees while optimizing zap time and bandwidth.

III. Client Assignment Problem (CAP)

Long et al. [5] defined the CAP as follows. Given a network $G = (V, E)$ where $V = \{v_i | 1 \leq i \leq m\}$ is the set of m vertices and E is the set of n edges. Each vertex v_i (we use this as equivalent to *server*) is associated with a capacity f_i (the total number of connections available) and each edge $e = (v_i, v_i')$ is associated with a delay value $\delta_{ii'}$ and bandwidth capacity $b_{ii'}$. A multicast tree t rooted at $s \in V$ which spans V is given where s is the source of the set of r IPTV channels K; each channel consumes a single unit of bandwidth. δ_i^t is the sum of delays on the path from v_i to s in tree t. There is a set of p clients $C = \{c_j | 1 \leq j \leq p\}$ such that (1) each c_j demands a set of channels $k_j \subseteq K$, (2) each c_j can access to each node $v_i \in V$ with the delay value σ_{ji}. The two criteria considered by the CAP are zap time and bandwidth consumption.

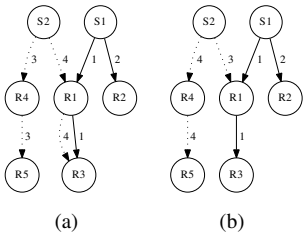


Fig. 1. Non-Optimal and Optimal Solutions

Using this model, each client can access a channel κ by connecting to any $v_i \in V$ and zap time is defined as $\sigma_{ji} + \delta_i^t$. Bandwidth consumption of a link is simply the number of channels flow through it when multicast is used. Note that it is sufficient to deliver a channel κ to a node v_i when (1) there are clients requesting κ that are connected to v_i or (2) there is some node v_i' in v_i's subtrees that requires κ.

The goal of CAP is to find a mapping from each client $c_j \in C$ to $v_i \in V \backslash \{s\}$ so that some constraints are satisfied. The problem variants discussed in [5] are: Minimize total bandwidth consumption (CAP-MB); minimize maximum zap time (CAP-MZT); minimize maximum zap time, constrain bandwidth (CAP-MZCB) and minimize bandwidth, constraint zap time (CAP-MBCZ). CAP is an offline model and assumes that all channel demands are known in advance, are unordered and that there is no distribution of channels in the network.

A. Multi-Source Extensions (MCAP)

We extend the CAP by assuming that there are multiple sources with each providing distinct channels. The real world analogy for this would be multiple IPTV providers operating under the scope of the same Internet Service Provider. In addition to the notations used in the original CAP problem, we use S to denote the set of q sources and $S \subset V$. Each $s_i \in S$ provides a subset of channels $K_i \subset K$ and $K_i \cap K_i' = \phi \ \ \forall \ i \neq j$ (no overlapping). This allows multiple multicast trees to simultaneously exist within a single network. We use T for the set of q multicast trees and each $s_i \in S$ is the root of multicast tree $t_i \in T$. Each t_i also spans $V \cap s_i$ as the original CAP model and we use the same notation for the delays. Note that for any $s_i \in S, i \neq j$, $s_i \notin t_j$. This means the sources only serve channels but not help distribute channels provided by other sources. We also reuse the notations and values that are defined previously for the CAP.

MCAP is interesting due to the fact that the various multicast trees share edges in the same network. Hence we can not simply solve each CAP instance and then aggregate the results as shown by the example in Fig. 1.

$S1$ provides channels $K_1 = \{1, 2\}$ and $S2$ provides channels $K_2 = \{3, 4\}$ we also have 5 servers, $\{R1 \cdots R5\}$, that may be used as contact servers by clients. The number of demands each server may deliver is $\{3, 2, 3, 2, 2\}$ respectively. Assume that all link capacities are equal. We are given 7 clients $\{c_1 \cdots c_7\}$ who demand the following channel sets $\{\{1, 3\}, \{2, 4\}, \{2, 3\}, \{4, 1\}, \{4\}, \{3, 1\}, \{4\}\}$. Since all edge capacities are equal in this scenario our goal is to minimize the number of channels transmitted over each link. As shown in Figure 1(a) if we allocate the demands for $T2$ first we can achieve an optimal result for that tree however that is only half of the full solution. We must then assign the demands for $T1s$ channels which results in the following assignments. $c_{i,k}$ is client i with demand k. $R1 = \{c_{2,4}, c_{4,4}, c_{3,4}\}$, $R2 = \{c_{2,2}, c_{c_{3,2}}\}$, $R3 = \{c_{7,4}, c_{1,1}, c_{4,1}\}$, $R4 = \{c_{6,3}, c_{3,3}\}$, $R5 = \{c_{1,3}, c_{6,1}\}$. Notice that this results in a nonoptimal solution, with 2 channels being sent over $\{R1, R3\}$, even though both of the CAP instances were solved optimally. If we instead solve $T1$ first and then $T2$ than we can arrive at Figure 1(b) with the following contact server assignments $R1 = \{c_{1,1}, c_{4,1}, c_{6,1}\}$, $R2 = \{c_{2,2}, c_{3,2}\}$, $R3 = \{c_{1,3}, c_{3,3}, c_{6,3}\}$, $R4 = \{c_{2,4}, c_{4,4}\}$ and $R5 = \{c_{5,4}, c_{7,4}\}$. This solution is optimal as there is only one channel transmitted over any link. This shows that the *order* in which the CAP instances are solved individually affects the overall systems solution

IV. IP Models

To construct the IP Models, we compute the following values during pre-processing:

- $Z_{ijk} = \sigma_{ji} + \delta_i^{\hat{t}_k}$, which is the zap time when c_j accesses channel k through v_i in tree $\hat{t}_k$.
- $M = \max(\ p \times r, m \times r)$ where p is the number of clients, r the number of channels, and m the number of servers
- $d_{jk} = 1$ if client c_j demands channel k, 0 otherwise.

For convenience, we use $\hat{t}_k$ for the tree associated with channel $k \in K$. We use p_i^k to denote the parent of $v_i \in V$ in the tree $\hat{t}_k$, and use w_i^k for the successors (children) of $v_i \in V$ in the tree $\hat{t}_k$.

The decision variables are defined as follows. Y_{ijk} is a binary variable that takes the value of 1 if $v_i \in V$ sends channel $k \in K$ to $v_j \in V$, 0 otherwise. X_{ijk} is a binary variable that takes the value of 1 if $v_i \in V$ sends channel $k \in K$ to $c_j \in C$, 0 otherwise. B is a continuous variable that is used to capture the minimum residue bandwidth.

Due to the space limitations, we do not provide the extended IP models for the variants in [5] and only consider the two new variants which focus on residual bandwidth.

A. Maximize Residual Capacity (MCAP-MR)

In MCAP-MR, the goal is to maximize the minimum residual capacity of the networks links on a per link basis rather than minimizing the summation of bandwidth of the whole network in CAP-MB. While Ravindran [7] showed that it can be efficient to attempt to utilize shared links as much as possible, we take a different approach here. All networks will have some upkeep associated with them (maintenance, rental costs, etc). Due to this cost it is better to spread out the usage of the networks links. The model is presented below.

Model MCAP-MR

$$\text{Maximize:} \quad B \tag{1}$$

Subject To:

$$\sum_{v_i \in (V \setminus S)} X_{ijk} = d_{jk} \qquad \forall c_j \in C,\ k \in K \tag{2}$$

$$\sum_{l \in w_j^k} Y_{jlk} \leq M \cdot Y_{p_j^k jk} \forall v_j \in (V \setminus S),\ k \in K \tag{3}$$

$$\sum_{c_j \in C} X_{ijk} \leq M \cdot Y_{p_i^k ik} \forall v_i \in (V \setminus S),\ k \in K \tag{4}$$

$$\sum_{k \in K} Y_{ijk} \leq b_{ij} \qquad \forall (i,j) \in (E) \tag{5}$$

$$\sum_{c_i \in C} \sum_{k \in K} X_{ijk} \leq f_j \qquad \forall v_j \in (V \setminus S) \tag{6}$$

$$b_{ij} - \sum_{k \in K} Y_{ijk} \geq B \qquad \forall (v_i, v_j) \in E \tag{7}$$

The Objective (1) is the minimum residue bandwidth of a link in the network. Constraint (2) forces each client to receive any channel that it demands from a node in G that is not a source. Constraint (3) enforces that a node receives a channel from its parent if any of its children in the corresponding tree request this channel. Constraint (4) forces any node that sends a channel to a client to have received it from its parent in the appropriate tree. Constraint (5) ensures that all links are not carrying more than there maximum capacity. Constraint (6) restricts the number of demands that a node may service, sources are assumed to have enough capacity to provide all channels to all of their children (in G) but cannot provide channels to any client. Constraint (7) is used to capture the residue bandwidth for each link.

B. Maximize Residual Capacity while Controlling Zap Time (MCAP-MRCZ)

In MCAP-MRCZ, the goal is to maximize the minimum residual capacity of each link while constraining the zap time to be less than a bound τ. Thus the following constraint is added to *Model MCAP-MR*.

$$X_{ijk} \cdot Z_{ijk} \leq \tau \qquad \forall c_i \in C,\ v_j \in V,\ k \in K$$

V. Heuristic Algorithms

While Integer Programming provides an optimal solution, it generally has a lengthy execution time. We present two heuristic algorithms to approximate the optimum in a short amount of time. In both heuristics we allow link capacity constraints to be violated if necessary, but make attempt to minimize such occurrences. By allowing link capacity constraints to be violated if necessary we are also able to identify portions of

the network that might require more resources to better serve the clients.

Algorithm MCAP-HMR

Algorithm HMR HMR is a simple greedy best-fit heuristic that functions on the assumption that the links connecting the root nodes of a tree to their associated children form a *chokepoint* for the distribution of channels. First, HMR groups each demand by channel. For each set of demands we do the following. Locate the subtrees rooted at the children of the root of the current multicast tree. Choose the subtree with the largest residual capacity on the link connecting it to its parent (the trees root), if there is more than one choose the subtree whos *server* capacity is the best fit for the set of demands. Assign client demands throughout to the selected subtrees servers, favoring links with higher residual capacity. Repeat until all demands for the current channel are distributed and then continue with the next set of channel demands. This heuristic works to ensure that there is an even distribution of channels at the top level of every tree and that interior links are shared as infrequently as possible. It will always succeed unless the total server capacity is less than the number of client demands.

A. Algorithm MCAP-HMRCZ

Algorithm HMRCZ attempts to maximize the minimum residual capacity of a networks links while keeping all client zap time under a given bound. It functions in a similar manner to MCAP-HMR. First, all clients are grouped by there demand. For each subtree that is formed by the children of the root of each multicast tree the number of demands that it may satisfy while adhering to the zap time bound is calculated. Like MCAP-HMR, this algorithm then finds the subtrees that have the most residual bandwidth on the associated edge. From these, the subtrees that can best satisfy the remaining demand set is chosen and demands are assigned to the deepest node within the subtree as possible. By assigning demands to deeper nodes we reserve the nodes that are closest to the root (source) node whose zap times are likely to be better due to the fewer number of hops. These steps are repeated until all of the demands for each channel have been satisfied. This heuristic will fail when there is no feasible solution or all servers with a feasible zap time have had there capacity exhausted by previously assigned demands. The latter case can be addressed by either loosening the bound or ignoring it and allocating any unsatisfiable demands to a server providing the demand with the best zap time. The inputs to H-MRCZ (Algorithm 1) are G: a graph of the network, T: the multicast tree set, C: the list of clients, D: the list of all client demands, X, Y: the variable sets that control channel propagation from routers to clients and routers to routers respectively and B the zap time bound that all demands must meet when satisfied.

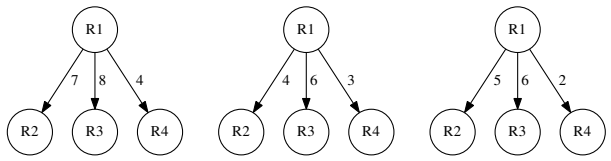
Algorithm 1 H-MRCZ

Input: $G,\ T,\ C,\ D,\ X,\ Y, B$
  for all $k \in K$ **do**
    L[k] = list of clients that demand k
  end for
  for all $t \in T$ **do**
    for all Child v of root t **do**
      Find and add subtree $a_{t,v}$ rooted at v to A
    end for
  end for
  sort L in descending order by size of sublist $L[k]$, let the resulting list be $L = \{L[\hat{1}], \cdots L[|\hat{K}|]\}$
  for $i \leftarrow 1 \cdots |K|$ **do**
    for $a \in A$ and a can provide demand k of $L[\hat{i}]$ **do**
      for all clients c in $L[\hat{i}]$ **do**
        for all servers s in a **do**
          if $Z_{sck} < B$ **then**
            $H[a] = H[a]+$ capacity of s
          end if
        end for
      end for
    end for
    for all $H[a] \in H$ **do**
      $r[a] =$ Residual Link Capacity to subtree $H[a]$
    end for
    while $|L[\hat{i}]| > 0\ and\ |H| > 0$ **do**
      find the subtrees t largest residual capacity $r[a]$
      assign clients to feasible nodes starting at leaf nodes
    end while
    if clients remain with demand $L[\hat{i}]$ **then**
      Return Fail
    end if
  end for
Output: X, Y assignments and Residual Bandwidth

VI. Performance Evaluation

To evaluate the extended CAP models performance we generated a set of test instances of various sizes. We used the BRITE Topology Generator [10] to generate our network topologies. The following parameters were provided to BRITE: Barabasi-Albert Topology [11] (a Scale-Free Network), $n \in [10, 20 \cdots 100]$(number of nodes). Additionally, Main Plain (HS) $= 500$, Inner Planes (LS) $= 100$, Node Placement = Heavy Tailed, Growth Type = Incremental, m (Number of connected nodes) $= 4$ and a heavy tailed bandwidth distribution where $b \in [12, 25]$.

We generate 10 sets of 30 graphs of size $n \in [10, 20 \cdots 100]$. BRITE provides the link capacities while latencies are calculated as $1/5$ of the euclidean distance between a given pair of servers. Node capacities are calculated as $[(\sum_{j=1}^{p} k_j)/m] \times r$, where r is randomly chosen from the range $[.75, 1.75]$, $\sum_{j=1}^{p} k_j$ is the total demands and m is the total number of servers. We call $(\sum_{j=1}^{p} k_j)/m$ a servers *fair share* in this paper. For each set, $1, 2$, and 4 sources are randomly placed within the network and connected to their closest neighbors. Shortest path trees are then created for each of these source nodes. In each instance, there are 20 channels which are evenly divide between the sources. Clients are randomly placed within the network and the latency



(a) Base Bandwidths (b) Good Residuals (c) Bad Residuals

Fig. 2. Examples of Bandwidth Allocation

between each client c and server v is determined by the distance between the client and its closest node in G. For MCAP-MRCZ and MCAP-HMRCZ, the optimal zap time is calculated and then multiplied by 1.2 as the bound. The time limit of the Gurobi solver is set to 600 seconds and retrieve the best feasible solution found and current upper objective bound (gap) if not optimal.

Besides the residual bandwidth, we are also interesting in how evenly the servers are utilized in the solutions. Fig. 2(a) shows a multicast tree with the associated link bandwidth capacities. If we have 6 channels and 2 clients demanding each of them, than there are many possible ways the channels could be distributed throughout the network. For example, 3 channels could be sent over link (R1,R2), 2 over (R1,R3) and 1 over (R1,R4). As a result $R2$ serves 6 demands, $R3$ serves 4 demands and $R4$ serves 2 demands. This assignment leads to a minimum residual bandwidth of 3 as in Fig. 2(b). Alternatively, we could distribute 2 channels over each link as in 2(c). While the first example has a higher minimum residual bandwidth, it does not utilize the servers equally. So, in addition to using the residual bandwidth to evaluate the performance of our heuristics, we also use Residual Sum of Squares (RSS) which can be loosely defined as $\sum_{\forall i}(expected_i - observed_i)^2$. This gives a measure of the error between expected and observed values. In our case the *expected value* is a servers *fair share* while the *observed value* is the number of demands actually served by a given server i, as determined by either an IP Model or heuristic approximation. In the previous example, the *fair share* value is 4 and the RSS in Fig. 2(b) is $(4-6)^2+(4-4)^2+(4-2)^2 = 8$ while it is 0 in Fig. 2(c) as each server serves its *fair share* of the channel demands. This indicate that even though the residual bandwidth was lower in Fig. 2(c), the actual distribution of channels to our servers was better. By using RSS, we examine a side effect of our objective function in order to gain a better idea of the models performance. RSS was not used as an objective function as we prefer to examine a linear model.

Execution Times

Fig. 3(a) and 3(b) show the optimization times for MCAP-MR and MCAP-MRCZ using IP models (IP-MR, IP-MRCZ) as well as heuristics H-MR and H-MRCZ, each of which is evaluated using 1, 2 and 4 overlaid multicast trees. Note that these graphs use a logarithmic scale for the y-axis (Time). It is interesting to note that instances with a larger number of trees can be much easier for the Gurobi Optimizer to solve. The difficulty of the problem varies greatly between instances, up to several orders of magnitude for the same size problem. This is an interesting anomaly and is likely due to the networks topology, but we leave further study to the future.

Residual Bandwidth

Fig. 3(c) and 3(d) show the average residual bandwidths for each set of instances from different approaches. For the IP approach, the best feasible solutions were used when the time limit was encountered. For the heuristic approach, the residual may be negative as they are allowed to violate the link capacity constraints. It is easy to see instances with larger numbers of trees and servers are more likely to have higher residual bandwidths. As the number of trees grows, the number of channels that each individual tree must distribute lessens. This simplifies the distribution of the channels at the top level of the tree which contains the links most likely to be shared by a larger number of channels. Additionally, when the number of nodes is higher, it is likely that the root of the trees will have more children which will also helps to simplify the distribution of channels. The heuristics perform well for both 2 and 4 trees.

Residual Sum of Squares

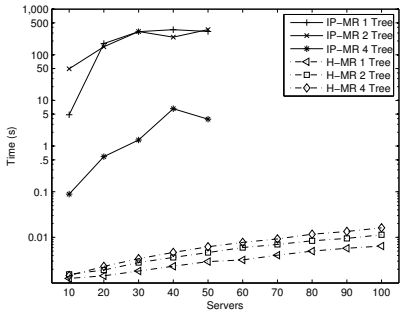
The RSS results are shown in Fig.3(e) and 3(f). The RSS for all variants grows linearly with the problem size. The performance of IP approach is consistently better than our heuristics except for MR and MRCZ which is an interesting outlier for the 2-tree case.

VII. Conclusion

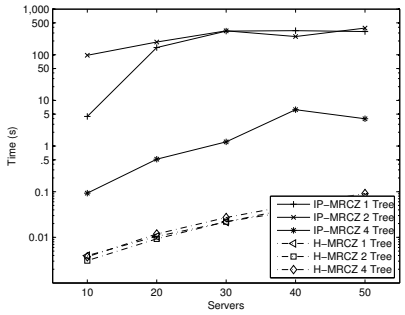
We have introduced an extended CAP Model (MCAP) that allows for a forest of multicast trees within a single network, as well as two new variants that make more efficient use of the networks resources. Heuristics for each of these new variants are introduced and their performance is analyzed on a variety of test networks of various sizes.

A further expansion of this model to allow for non-unit costs for channels is relatively straightforward and would be especially helpful in better modeling the multi-commodity aspect of this problem. Additionally, there are several alternative IP Model formulations that could be used to attempt to better utilize overall server usage such as the metric used in our performance evaluation. However this would require a quadratic model and would be much more computationally intensive.

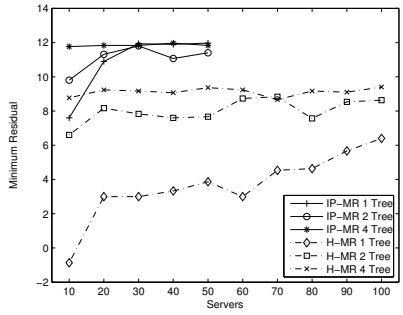
Currently the CAP is an offline model, an online version of this model would be a much better model of a production system. This would also be a relatively straight forward alteration to the model. Instead of all channel demands being known prior to execution, they would be introduced in sequence. The model would be forced to account for channels that are already distributed through the system (potentially reducing zap times) and ensure that all previous channel requests are satisfied until their associated user changes channels again.



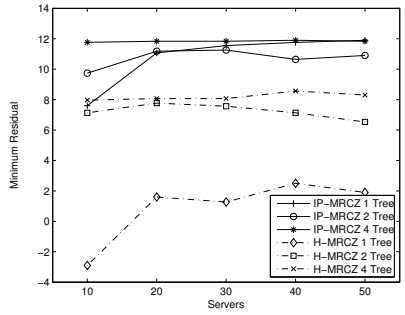
(a) Runtimes for MCAP-MR



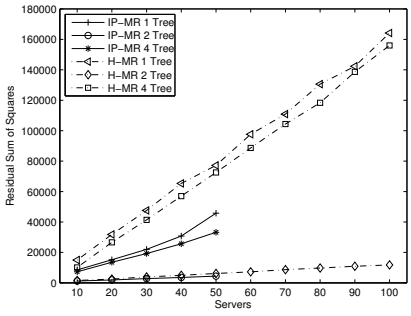
(b) Runtimes for MCAP-MRCZ



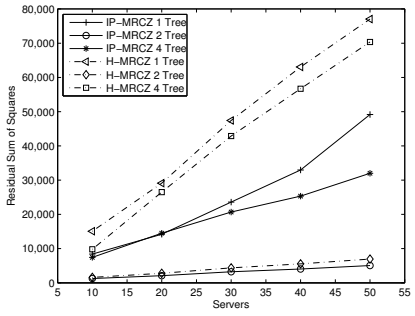
(c) Average Minimum Residual Bandwidths MCAP-MR



(d) Average Minimum Residual Bandwidths MCAP-MRCZ



(e) Average Residual Sum of Squares MCAP-HMR



(f) Average Residual Sum of Squares MCAP-HMRCZ

Fig. 3. Experimental Results

REFERENCES

[1] ITU-T FG IPTV. http://www.itu.int/md/T05-FG.IPTV-C-0545/en.

[2] A. Begen, N. Glazebrook, and W. Ver Steeg, "A Unified Approach for Repairing Packet Loss and Accelerating Channel Changes in Multicast IPTV," in *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, Jan. 2009, pp. 1 – 6.

[3] H. Fuchs and N. Farber, "Optimizing Channel Change Time In IPTV Applications," in *Broadband Multimedia Systems and Broadcasting, 2008 IEEE International Symposium on*, 2008.

[4] K. A. R. Kooij and K. Brunnstrom, "Perceived Quality Of Channel Zapping," in *Proceedings of the Fifth IASTED International Conference Communication Systems and Networks*, 2006.

[5] M. Long, S. Radhakrishnan, S. Karabuk, and J. Antonio, "On zap time minimization in IPTV networks," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, Feb 2012, pp. 713 – 718.

[6] D. Banodkar, K. Ramakrishnan, S. Kalyanaraman, A. Gerber, and O. Spatscheck, "Multicast instant channel change in IPTV systems," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, Jan. 2008, pp. 370 – 379.

[7] K. Ravindran, A. Sabbir, D. Loguinov, and G. Bloom, "Cost-optimal multicast trees for multi-source data flows in multimedia distribution," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2001, pp. 966 – 975.

[8] S. Chen, O. Günlük, and B. Yener, "The multicast packing problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 311 – 318, Jun. 2000.

[9] C. Y. Lee and H. K. Cho, "Multiple multicast tree allocation in IP network," *Computer Operations Research*, vol. 31, no. 7, pp. 1115 – 1133, Jun. 2004.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 346–.

[11] A. Barabasi and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509 – 512, 1999.