Heuristic Algorithms for Server Allocation in Distributed VoIP Conferencing

R. Venktesha Prasad*, H. N. Shankar#
*Wireless and Mobile Communication,
Technical Univ. of Delft, The Netherlands.
vprasad@ewi.tudelft.nl
Tel: +31-15-2786795
TeleFax: +31-15-2781774

H. S. Jamadagni†, M. V. Rohith#, S. Vijay+
† CEDT, IISc, Bangalore.
# PES Institute of Technology,
Bangalore, India.
+Esqube Communication Solutions Pvt. Ltd.
Bangalore, India.

Abstract

To allocate Audio Conference Servers (CS) [1] for virtual conferencing over IP, we leverage Session Initiation Protocol (SIP) for signaling. We address here the problem of facilitating seamless conference amongst participants using CSs. This demands a proper allocation of CSs to clients to maximize the number of participants served and at a reduced cost. Seeking a more realistic approach, we avoid over-simplifying assumptions; hence the problem becomes relatively harder. We present three heuristic algorithms for these NP-hard problems and bring about the effectiveness of their performance.

Keywords: VoIP Conference, Conference Servers, SIP, Facility Location Problem, Heuristic Algorithms

1 Introduction

Traditional data differ from voice and video in aspects of delay-constraints and packet loss tolerance. Hence as time-sensitive voice and video applications are deployed on Internet Protocol (IP), the inadequacy of the Internet is exposed progressively. Virtual conference (teleconference) facility is at the cutting edge Internet Telephony. Audio and video conferencing on Internet have advantages and are popular. Clearly, most collaborative works demand audio conferencing more frequently than video interactions [2]. It makes sense to deal with audio conferencing first.

The bandwidth required for teleconference over the Internet increases rapidly with the number of participants. Audio "quality" in a conference includes facilitating interactivity, *i.e.,* allowing impromptu speech, and spatialism [3, 4]. The critical implementation issues are: 1.Packet delay; 2.Echo; 3.Customized mixing of audio from selected clients [3]; 4.Automatic selection of clients to participate in the conference; 5.Playout of mixed audio for every client; 6.Handling clients not capable of mixing audio streams (such clients are known as "dumb clients"); 7.Deciding the number of simultaneously active clients in the conference without compromising voice quality.

As participants increase, intermediate servers must serve to control, maintain and support a conference. Many of the issues listed above are closely related to server location *vis-à-vis* clients and assignment of clients to servers.

The problem we tackle is set in the backdrop of Virtual Conferencing Environment (VCE). The architecture for VCE is shown in Fig. 1. We focus on large audio VCE with several hundreds of users across the Internet. Key issues here are: (1) the front-end consisting of the application running on the computers of end users; and (2) the back-end that provides other applications that facilitate conferencing and conference service.

The domains and entities in a VCE are not fixed apriori. We identify clients and CSs of domains. Once a client is assigned to a domain the control messages for conference setup and maintenance are between SIPS and in turn CSs. We focus on finding a configuration of domains around possible locations of CSs seeking 'near optimality'. Optimality is governed by delay, connection cost, etc. (vide Section 2.1).

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**

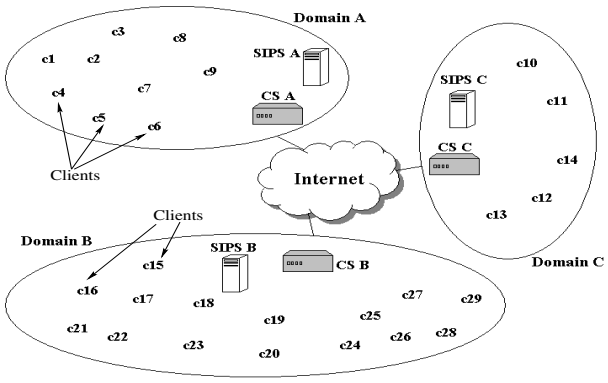

Figure 1: ***VCE* architecture showing domains, clients and servers**

1.1 Constraints

Necessary constraints for a conference are: (1) CSs locations are fixed; (2) CSs are software entities are enabled only when necessary; (3) Clients are assigned to a CS as and when conference service is requested; (4) A client is assigned to only one CS; and (5) Only one CS is opened at a location and if there is a need for another CS there, it is deemed to be another location with similar parameters with respect to the network.

Cost formulation and the problem definition comprise Section 2. Section 3 has three heuristic algorithms. Results are discussed in Section 4. We conclude in Section 5.

2 The Allocation Problem

Frequently asked questions and those addressed here are: (a) If a client is in a remote domain, say, a dial up link, how can that client be served? (b) Should a CS be always opened in a SIPS domain or can a client be assigned to a CS in another domain? (c) With several clients in a small network neighborhood, is it cost-effective to assign a new CS to that group?

Though a distributed conference system based on CS is scalable [4, 1, 5] it requires allocating CSs to clients. We consider choosing of CSs and CS group formation. Scalability depends on the number of CS domains. With more CS domains, communication cost between CSs goes as $O(\nu^2)$, with ν domains. The costs of client connecting to a CS and of CS opening and maintaining are pertinent.

A conference is controlled by distributed SIPS [6], or by a centralized controller [4]. The servers can allocate CSs to clients at the start of a conference if the list of participants is known apriori. Else SIPS has to reallocate CSs. The options are: (1) Only the new client is allocated to that CS which reduces the cost; (2) The new client is allocated to a nearest CS and periodically, all clients are allocated afresh to CSs available. We propose the second approach to avoid frequent reallocations. In VoIP, the User Datagram Protocol (UDP) packets carrying voice are sent to the CS whose address is available with the client at allocation time. Addresses can be changed as UDP is not connection oriented.

This problem of allocation of clients to CSs, a Facility Location problem, is NP-hard [7], to which heuristics offer attractive solutions.

2.1 Cost

Two components of cost, viz., of connection between a client and a server, and of opening a CS, are pertinent here. They may be estimated by direct measurement on networks. This paper is focused on demonstrating the efficacy of our heuristic algorithms, we refrain from cost estimation here.

2.2 Problem Formulation

Let F_S be the set of m CSs and D_C, the set of n clients; locations of CSs and clients are known. Let the cost of connecting a client j to CS i be c_{ij}, d_j the demand from Client j, f_i the cost of opening CS i and l_i the upper bound on the units of computation by CS i.

$y_i = 1$, if CS i is opened, else 0. $x_{ij} = 1 or 0$, according as Client j is assigned to CS i or not. The minimization problem ($\mathrm{P_{min}}$) is formulated as an integer program as

$$\min : \quad \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

subject to

$$\sum_{j=1}^{n} d_j x_{ij} \quad \leq \quad l_i, \qquad (2)$$

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**

$$x_{ij} \leq y_i, \quad (3)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad (4)$$

$$x_{ij} \in \{0,1\} \quad \text{and} \quad y_i \in \{0,1\} \quad (5)$$

Constraint 2 limits the number of clients connected to a CS depending on its limit. Constraint 3 ensures that if there is an assignment of a client to a CS then that CS must have been running. Constraint 4 ensures each Client j is assigned to a CS i, $i \in F_S$ and $j \in D_C$. Constraints 5 are part of integer programming. Another constraint that a client is allocated to only one CS is captured in the above constraints and hence not mentioned explicitly.

3 Solution by heuristics

Two operations are involved here: (a) assigning clients to CSs with less cost; and (b) opening least number of CSs so that there is saving in supporting the conference service. When demands are splittable, *i.e.,* they can be met by more than one server, solutions by transhipment algorithms are suggested. In case of splittable demands, solution by approximation algorithms exist [8, 9, 10, 11].

In this problem of allocating CSs, there are instances in which a solution may not exist. A case in point is when the total capacity of CSs is less than the total demand by clients. In these cases heuristic algorithms definitely fail. Heuristic algorithms try to find a near optimal solution, if there exists one.

3.1 Algorithms: Phase-I

The algorithm suggested here is divided into two phases. The first one is the assignment phase. Here the clients are assigned to CSs in set $F_S^G \subseteq F_S$. CSs only in F_S^G are considered; the algorithm considers CSs outside this set as nonexistent. In the second phase, the number of CSs is minimised by closing some of them and clients previously assigned to the recently closed CSs are reallocated to other CSs. Algorithms of the first phase are given below.

Algorithm 3.1 ***Assignment Phase: Assignment of clients to given set of CSs*** (F_S^G)

Sort c_{ij} *in list* $L(k)$ *in increasing order* $k = \{1, 2, \ldots, mn\}$. *Consider assignment of clients to a CS as* $A_C[\cdot]$, *a row vector that holds the CS identity* i *for each Client* j.

$A_C[j]$ *is zero vector to start with.*

$k = 1$. *REPEAT*

{

Step 1: *Assign the Client j to CS i if*

- *For CS i, $l_i \geq d_j$, and*
- *Client j is not already assigned;*

Step 2: *$l_i = l_i - d_j$, if Client j is assigned to CS i. Update $A_C[j] = i$ and $x_{ij} = 1$.*

Step 3: *Stop if all clients are assigned; Return $\sum_{i,j} c_{ij} x_{ij}$ and $A_C[j]$. Break;*

Step 4: *Return 'Infeasible' if any client is not assigned when $k = mn$.*

Step 5: $k = k + 1$

}

The complexity of this algorithm is $O(m^2 n^2)$. The advantage of this algorithm is that it is very easy to implement. The next algorithm is an intelligent assignment solution.

Algorithm 3.2 ***Assignment Phase: Assignment of clients to given set of CSs*** (F_S^G)

$A_C[j]$ *is zero vector to start with.*

Step 1: *Find $b(j) = \min_i\{c_{ij}\}$ for each j. That is, finding the CS $i \in F_S^G$ that results in minimum cost of assignment to Client j.*

Step 2: *Update $A_C[j] = i$ by assigning Client j to CS i found above and set $x_{ij} = 1$.*

Step 3: *Find $F_S^{(OL)} \subseteq F_S^G$, that are overloaded taking into account $A_C[j]$ and l_i. If no CS is overloaded, goto Step 7.*

Step 4: *Find difference matrix $[a_{ij}]$ where $a_{ij} = c_{ij} - b(j)$, $i \in F_S^G$. Mark 'X' in $[a_{ij}]$ if Client j is allocated to CS i in Step 2.*

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**

Step 5: *For each CS $r \in F_S^{(OL)}$, for clients $s \in D_C^{(r)}$, i.e., clients assigned to the CS r, find new CS-client pair (p, q) corresponding to* $\text{argmin}_{(i,s)}\{a_{is}\}$, *$s \in D_C^{(r)}$, $i \in F_S^G$ and $i \notin F_S^{(OL)}$ and CS p is able to serve this new client without overloading itself (a_{ij}'s marked with 'X' are not to be considered)*

Step 6: *If a new CS is not found in Step 5, Return 'Infeasible'. Else, assign Client q to CS p, i.e., $A_C(q) = p$ and $x_{pq} = 1$. Subtract the load d_q from CS r. Update $A_C[j]$ and $F_S^{(OL)}$ taking into account this change. Mark a_{rq}, the previous position of Client q, with 'X'. Repeat Step 5 till $F_S^{(OL)} = \{\}$.*

Step 7: *Return $A_C[j]$ and cost of assignment $\sum_{i,j} c_{ij}x_{ij}$.*

Unlike Algorithm 3.1, Algorithm 3.2 does an intelligent CS selection. It can be found that the worst case complexity of this algorithm is $O(mn^2)$.

Algorithm 3.3 ***Assignment Phase: Assignment of clients to CSs in*** (F_S^G)

Algorithm 3.3 is a minor variant of Algorithm 3.2. It is blind to overloading of CS in Step 5. Eventually an overloaded CS transfers its client to another CS such that transfer incurs minimal extra cost. Since a client can move to all the m servers in worst case this algorithm has worst case complexity of $O(m^2n^2)$.

Remark: The relative merits of the algorithms here is problem-dependent. The strategy is to invoke all the algorithms and carry forward the best at each iteration. We can also find the solution with each first phase algorithm – without mixing them together in the iterations – individually. The solution can be compared later to select the best one. We have selected the former.

3.2 Algorithm: Phase-II

Here we try to close CSs one by one using the cost function

$$T_i = \frac{f_i}{l_i} - \sum_i \sum_j I_C(j)c_{ij} \qquad (6)$$

The first term in the RHS of Eq. (6) pertains to the cost of operating for unit capacity of the server. Later we close the Server-L corresponding to highest T_L.

The best of Phase-I and Phase-II algorithms is used in the comprehensive Algorithm 3.4 below.

Algorithm 3.4 ***Allocation Phase: A heuristic solution to the problem*** $(\boldsymbol{P_{min}})$

Let $F_S^{(O)}(i)$ be a vector that represents whether a CS is open. It is found using client assignment vector $A_C[j]$.

$$F_S^{(O)}(i) = \begin{cases} 1 & \text{if CS } i \text{ is serving any client} \\ 0 & \text{Otherwise} \end{cases}$$

Total cost SOLN of any assignment is found using

$$SOLN = \sum_i F_S^{(O)}(i)f_i + \sum_{\forall i,j} I_C(j)c_{ij} \qquad (7)$$

where $I_C(j) = \begin{cases} 1 & \text{if Client } j \text{ is served by CS } i \\ 0 & \text{Otherwise} \end{cases}$

Note: The second sum is nothing but $\sum_{i,j} x_{ij}c_{ij}$

Step 1: *Initialise $A_C[j]$ and $F_S^{(O)}(i)$ $\forall i, j$ to zero, $F_S^G = F_S$ and a set $Z = \{\}$.*

Step 2: *Find $A_C[j]$, $F_S^{(O)}(i)$ corresponding to a minimum SOLN, after using the three Algorithms 3.1, 3.2 and 3.3 of first phase.*

Step 3: *Find T_i, $i \in F_S$, as defined below and arrange in descending order.*

$$T_i = \frac{f_i}{l_i} - \sum_i \sum_j I_C(j)c_{ij}$$

Step 4: *For $k = 0$*
Do {
Temporarily close CS u that corresponds to the highest T_i, $u \notin Z$
(a) Find the best iSOLN using Equation 7 (Here iSOLN means

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 IEEE

intermediate solution) out of three returned assignments $A_C[j]$'s of three first phase algorithms with $\{F_S^G - \{u\}\}$ as CSs open for assignment. If ($iSOLN < SOLN$) then permanently close CS u and $F_S^G = F_S^G - \{u\}$; $SOLN = iSOLN$; Recalculate T_i as above with the CS u closed and arrange in descending order; Else, $Z = Z \cup \{u\}$; (b) $k = k + 1$; } Repeat till $k = m$;

Step 5: *$A_C[j]$ of $SOLN$ corresponds to the best assignment found by this algorithm. The total cost is $SOLN$.*

For example if we use all three Phase-I algorithms, and taking best allocation returned by them, the resulting complexity is $O(m^3n^2)$.

4 Results and Discussions

LP formulation, hard but not NP, provides best solution relaxing the constraints. It is interesting to see how Algorithm 3.4 fares for some test samples.

We generated the test cases randomly. The cost $c_{ij} \in [1, 1000]$ is generated arbitrarily. So also, $d_j \in [1, 5]$ and $f_i \in [1, 1000]$ are chosen. l_i are changed across test samples as per Table 1 (shown here a sample of many test cases). As higher densities of clients can be seen in geographical proximities, and each client may be billed differently, we refrain from the use of triangular inequality on the physical distance between CS and a client even though it may reduce the complexity of the problem.In such cases our algorithm converges faster than those with random inputs

Table 1 is meant to give an overall picture for different test scenarios. It shows the ratio of performance of LP to that of Algorithm 3.4. Algorithm 3.4 achieves results very close to the optimum solution compared to using Algorithm 3.1 alone in assignment (first) phase. The heuristic may not give a solution, though one exists. Yet heuristic algorithms serve as a good starting point to group clients. Moreover, with many heuristics the possibility of finding a solution, when one exists, is enhanced.

Now we shall get back to the question of how these algorithms are used in our allocation of CSs. We suggest that these algorithms be implemented on SIPS in each domain. Once they receive information about a new client added to the conference, the algorithm is invoked. This requires information regarding the new column in $[c_{ij}]$ and the demand from the new client. Since, this information can be exchanged across SIPS while supporting the conference [5] unique allocation can be found at every SIPS. If the conference is booked earlier this algorithm can be used only in the beginning.

The allocation found using the above heuristics should answer the questions raised in Section 2. Likelihood of success of allocation can also be increased by some pragmatic assumptions. For example, in case of an isolated client, the nearest CS can serve it and multiple streams from CS to client can be mixed at that CS to reduce the bandwidth. There are some open issues here. One such issue concerns the method of implementing this algorithm in the proposed distributed conference setup [6, 5].

5 Conclusions

The problem considered here is a hard problem and not attempted in literature without relaxing some of the constraints. The heuristic algorithms presented here address problems involving (i) assignment of clients to CSs that do not have unit demands and hence not solvable using transportation problem; and (ii) reducing the total cost by opening an 'optimum' number of CSs. Their performance is comparable to LP solution, thus they are nearer to the optimal solution given by IP. They can be easily deployed. They are portable to the generic class of Facility Location problems as well as a host of End System Multicast or Overlay Network applications. Custom tuning these algorithms for online applications – when participants join and leave conference at will similar to the case of participants in a real conference hall – is one of the next logical steps.

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**

Table 1: **Comparison of Heuristic Algorithms with LP solution**

Number of CSs	Number of Clients	Range of l_i	LP Solution	Heuristic Algorithm 3.4 with Phase-I Algorithm(s)			Ratio of Heuristic to LP
				3.1	best of (3.1 & 3.2)	best of (3.1, 3.2 & 3.3)	
10	50	1-50	9707	10309	10309	10309	1.062
10	100	1-50	13000.4	13207	13037	13037	1.003
10	150	1-150	20439.6	22620	21155	21015	1.028
10	500	1-500	53541.5	57047	53876	53852	1.005
10	1000	1-1000	99698	102049	99951	99951	1.003
10	1000	1-800	100376	112422	101813	101734	1.014
10	1000	1-1000	105684	127217	111214	111420	1.052
10	1200	1-1000	174134	179141	174443	174443	1.002
12	1000	1-1000	93761.8	102985	94732	94732	1.011

References

[1] R. Venkatesha Prasad, Richard Hurni, H S Jamadagni, "A Scalable Distributed VoIP Conferencing using SIP", *Proc. of the eigth IEEE Symposium on Computers and Communications*, Antalya, Turkey, June 2003.

[2] E. Doerry, "An Empirical Comparison of Copresent and Technologically-mediated In-teraction based on Communicative Breakdown", *PhD thesis*, Graduate School of the University of Oregon, 1995.

[3] M. Radenkovic and C. Greenhalgh, "Multi-party distributed audio service with TCP fairness", in Proceedings of the *10^{th} ACM International Conference on Multimedia (MM 02)*, Juan-les-Pins, France, pp. 1120, December 2002.

[4] R. Venkatesha Prasad, "A New Paradigm for Audio Conferencing on Voice over IP (VoIP)", *Ph.D Thesis, Indian Institute of Science*, Bangalore, India, 2003.

[5] R. Venkatesha Prasad, Richard Hurni, H S Jamadagni, "A Proposal for Distributed Conferencing on SIP using Conference Servers", *Proc. of MMNS 2003*, Belfast, UK, September 2003.

[6] R. Venkatesha Prasad, Richard Hurni, H. S. Jamadagni, H. N. Shankar, "Deployment Issues of a VoIP Conferencing System in a Virtual Conferencing Environment", *ACM symposium on Virtual Reality and Software Techniques*, Osaka, Japan, October 2003.

[7] S. Guha and S. Khuller, "Greedy strikes back: Improved facility location algorithms," *Journal of Algorithms*, vol. 31, pp. 228–248, 1999.

[8] Martin Pal and Eva Tardos and Tom Wexler, "Facility location with hard capacities," in *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science*, 2001.

[9] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani, "A greedy facility location algorithm analyzed using dual-fitting," in *In Proceedings of 5th International Workshop on Randomization and Approximation Techniques in Computer Science*, vol. 2129, pp. 127–137, 2001. Lecture Notes in Computer Science.

[10] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani, "Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP," *Journal of ACM*, 2002.

[11] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a local search heuristic for facility location problems," *ACM symposium on Discrete Algorithms*, pp. 1–10, 1998.

Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)
1530-1346/05 $20.00 © 2005 **IEEE**