Near-Optimal Service Facility Location in Dynamic Communication Networks

Antonio Liotta, *Member, IEEE,* Carmelo Ragusa, and George Pavlou, *Member, IEEE*

***Abstract*— Systems relying on increasingly large and dynamic communication networks must find effective ways to optimally localize service facilities. This can be achieved by efficiently partitioning the system and computing the partitions' centers, solving the classic *p*-median and *p*-center problems. These are NP-hard when striving for optimality. Numerous approximate solutions have been proposed during the past 30 years. However, they all fail to address the combined requirements of scalability, optimality and flexibility. Here we present a novel distributed algorithm that computes provably near-optimal *p*-medians in linear time, exploiting the properties of mobile software agents.**

***Index Terms*— Location on networks, mobile agents, network partitioning, p-median.**

I. Introduction

NETWORK location problems involve the optimal placement of *p* service facilities in networks of *N* nodes. Service facilities are traffic sources, traffic destinations or both and are part of a distributed system, application or protocol. The location problem can be defined as that of 1) partitioning the network in *p* sub-partitions; and 2) optimally placing those facilities within their respective partition. Optimal location involves the minimization of an objective function. The location problem is termed '*p*-median problem' when the objective is to minimize the overall traffic incurred by the service facilities while it is termed '*p*-center problem' when the objective is to minimize the maximum response time.

Because of their importance, the *p*-median and *p*-center problems have been subject of intensive study for over 30 years in a variety of scientific disciplines. In communication networks, location problems are fundamental to a number of other problems related to distance minimization. For example, scalability can be achieved for the effective monitoring and control through network partitioning, leading to reductions in traffic and response time. Partitioning problems assume particular relevance in the modern era of mobile ubiquitous computing, ad hoc networking and dynamically re-configurable networks. The challenge is to address the combined requirement of scalability, optimality and flexibility for dynamic, frequently changing environments.

Manuscript received February 19, 2005. The associate editor coordinating the review of this letter and approving it for publication was Christos Douligeris. This work was developed in the context of EPSRC POLYMICS - Grant GR/S09371/01.

A. Liotta is with the Dept. of Electronic Systems Engineering, University of Essex, Wivenhoe Park, Colchester, UK (e-mail: aliotta@essex.ac.uk).

C. Ragusa is with the IT Innovation Centre, University of Southampton, UK (e-mail: cr@it-innovation.soton.ac.uk).

G. Pavlou is with the Centre for Communication Systems Research, University of Surrey, UK (e-mail: g.pavlou@surrey.ac.uk).

Digital Object Identifier 10.1109/LCOMM.2005.09026.

The *p*-median and *p*-center problems are both *NP-hard* on general networks [1]. Numerous approximate polynomial algorithms have been proposed (see extensive survey presented in [1][2][3]) but none of them suits the aforementioned combined requirement. The main hurdle of existing approaches is their intrinsic centralized nature, *i.e.* they all require knowledge of the network topology in order to get the distance matrix according to a particular metric as input parameter. While this not a problem in offline calculations over static networks, it becomes a critical issue in the case of dynamic, large-scale networked systems. In this case, the task of collecting a real-time snapshot of the network topology is indeed ambitious. We present here a novel location algorithm that is distributed, does not require any direct knowledge of the network topology, and computes near-optimal *p*-medians in linear time. The algorithm exploits the key properties of Mobile Agents (MAs), *i.e.* autonomous software entities capable of roaming the network and cloning other MAs [4].

II. Location Algorithm

Our algorithm works on the basis of information that indirectly reflects the status of the network, produced by distributed IP routing algorithms, *i.e.* Dijkstra or Bellman-Ford, which specify – for any node – the estimated distance to any other remote destination and the related next-hop neighbor node. This information is externally accessible in network nodes through the SNMP IP routing Management Information Base (MIB)[5].

The MA-based location algorithm is depicted in Fig. 1. The process can start at any node by injecting a single MA that is initialized with the list of nodes to be partitioned, N; the number of target partitions, p; and the tolerance margin admissible on p, $\varepsilon_p = (\Delta_p/p) \times 100$ (Δ_p is the absolute variation admitted on p). Starting from its initial location, the MA initiates an iterative, distributed procedure that subsequently creates new partitions and clones new agents (one per new partition). Every MA has exactly the same logic, *i.e.* they all go through the process depicted in Fig. 1. Our stratagem is to parallelize the partitioning process through a clone-and-migrate procedure where agent clones take full responsibility for disjoint subsets of N and operate independently from each other. We describe below the individual agent behavior that leads to the computation of p-medians in linear time.

The main objective of the cloning phase is to determine whether or not, and how, the agent partition ($N \subseteq N_t$) will be sub-partitioned. Through a "matching operation" of N_t against the local routing table, the MA discovers, for each target node, the next-hop neighbor node and the distance. Nodes

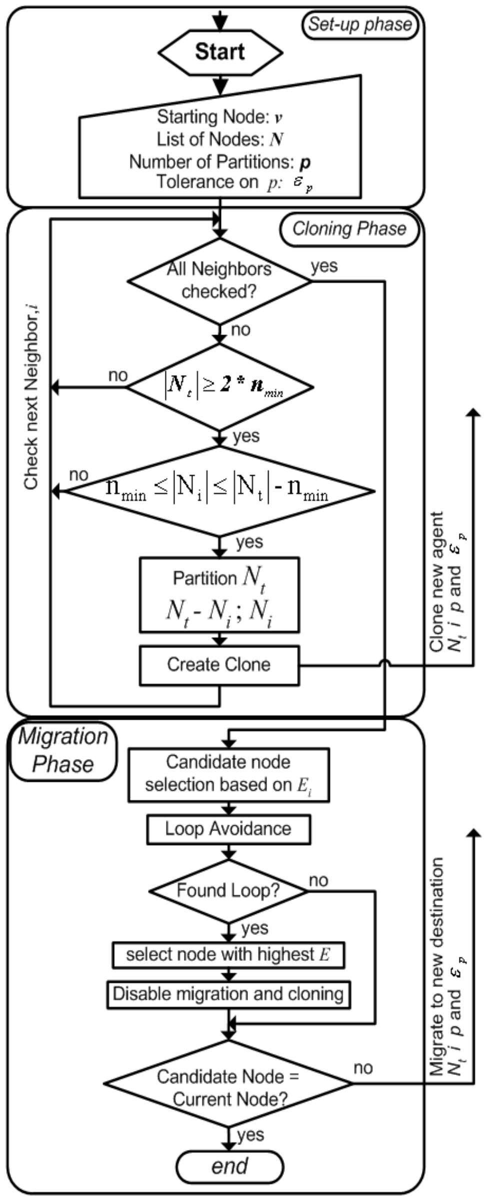

Fig. 1. Flow-chart diagram of the partitioning algorithm.

are then classified by next-hop address, i and each list N_i (which includes the subset of cluster nodes reachable through neighbor i) is submitted to a set of cloning decision conditions (the agent goes to the migration phase only upon checking all classes). Intuitively, by satisfying the cloning conditions of Fig. 1 the system pursues the constraints on the number of acceptable target partitions (given by p and ε_p). The condition $|N_t| \geq 2 \cdot n_{\min}$, where $n_{\min} = round\left(|N|/p + \Delta p/2\right)$, prevents cluster fragmentation (we do not want partitions having less than $n_{\min}$ elements). Then, we have to decide whether or not the sub-partition N_i is large enough ($N_i \geq n_{\min}$); but we also have to make sure that, should N_i form a new partition, the remaining nodes in N_t – *i.e.* $\{N_t - N_i\}$ – would still constitute a sufficiently large partition. The latter holds when $|N_t| - |N_i| \geq n_{\min}$. When all conditions are satisfied, N_t is split in two partitions, $\{N_t - N_i\}$ and N_i. New clones start from the beginning but are initialized with $N_t \equiv N_i$ and starting node $v = i$. The thread of execution of the main clone

TABLE I
SIMULATION PARAMETERS AND VALUES

Parameter	Values
$\|N\|$	51 102 147 198 258 303 (20 repetitions)
$p/\|N\|$	0.05 0.1 0.15 0.2 0.25 0.3
Diameter [hops]	11 12 13 14 15 16 17
ε_p	0 5 10 15 20 25
Root Node ID	All nodes

subsequently tests the other sub-partitions, which may lead to further cloning. After that, it enters the migration phase.

During migration, agents are driven towards their respective partition central location. Each agent operates only within the scope of its sub-partition N_t, computing the weighed routing costs E_i for each of the neighbor nodes i, where $E_i = \left(C_i \times |N_i|\right) / \left(C_t \times |N_t|\right)$; C_i is obtained by adding the individual routing costs of N_i; and C_t is the sum of all C_i. E_i provides an estimate of the global distance (in terms of routing costs) associated to each neighbor (so the agent elects as new candidate location the neighbor i that has the maximum value of E_i). Intuitively, once the agent reaches the migration stage its central location is pursued by aiming at reducing the largest weighed cost, which is in turn achieved by migrating towards higher-cost locations.

Agents can avoid migrating in loops by retaining the highest value of E_i for all previously visited nodes. So, as soon as the agent is presented with a candidate node that has previously been visited, it simply elects the node having the highest value of E_i (among all historical values) and disables migration and cloning before finally migrating to the target node. After that, the agent starts the whole process of Fig. 1 all over again, as it may have to migrate again because of changing conditions, *e.g.* faults, topology changes, congestion, etc.

III. EVALUATION METHODOLOGY

The algorithm has been evaluated through an extensive set of simulations providing an insight into computational complexity, sensitivity to starting point, optimality, stability and correctness. The algorithm was run over a set of realistic Inter-network topologies created with the GT-ITM topology generator [6], following a well-established methodology by Calvert and Zegura [7][8]. IP network and protocol behavior were simulated using the NS-2 simulator [9] that was extended with MA capabilities. Near-optimality was demonstrated by comparing our p-medians with those computed by the *Lagrangian* algorithm [3]. The latter is a classic provably near-optimal algorithm – *i.e.* it computes near-minimal total hop-distances in polynomial time – but does not satisfy the combined requirements of scalability, optimality and dynamicity because it relies on prior knowledge of full network topology and the relevant distance matrix.

The simulation parameters are depicted in Table 1. Most combinations were simulated, involving over 60,000 simulation runs that took over 70,000 hours. We have also verified that the partitioning process always exits and that agent migration does not create oscillatory or looping conditions.

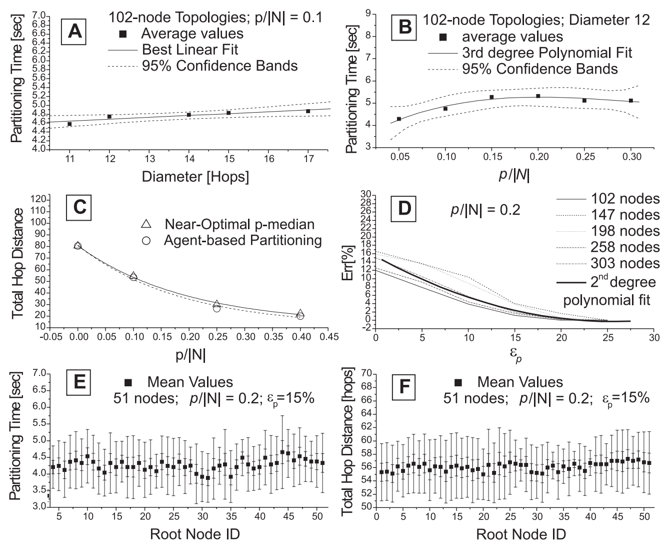

Fig. 2. Sample of results.

Due to space limitations we present below a representative selection of results (Fig. 2).

IV. Results

The first important result is linearity with network diameter (Fig. 2A), which makes the algorithm suited to large-scale systems. The algorithm is typically computed in fractions of a second, whereas the slope is determined by the efficiency of the MA platform (typical time-to-hop between two nodes is in the order of hundreds of msecs). We have also studied scalability *vs.* number of nodes, number of partitions, and average node degree, finding that computational time is not significantly affected by them when the network diameter is kept constant. Fig. 2B is a representative result demonstrating a very high level of parallelism (observe the plateau over a large range). Intuitively, independent processes are forked at each level of the network routing tree as required. Hence, computational time is upper-bound by the tree depth.

Fig. 2C illustrates near-optimality indipendently of the number of service facilities - the total-hop distance (the sum of the distances between partition centres and their respective nodes) is upper-bound by the one achieved by the *Lagrangian* algorithm. We also studied the ability of the system to control the actual number of target partitions. Fig. 2D illustrates a level of controllability that is perfectly acceptable for typical p-median problems. Finally, Fig. 2E and 2F demonstrate the algorithm independence from the initial conditions.

V. Concluding Remarks

This article motivates the use of MAs to provide a ‘distributed’ solution to the p-median problem that is traditionally tackled only by ‘centralized’ algorithms. Efficiency is achieved through a combination of distribution, code mobility and simple use of network routing information. We proposed a near-optimal algorithm that does not require the reconstruction of the network distance matrix and is characterized by linear computational complexity. Our approach outweighs existing algorithms which are polynomial of 2^{nd} and 3^{rd} degree and do require a complete knowledge of the network topology. In addition, our approach is adaptive to the dynamics of communication networks since agents keep operating even after their initial deployment and can reposition themselves when the conditions change, *e.g.* because of faults, congestion and so forth.

We have also performed an in depth viability study in the context of IP networked systems, assessing the level of maturity of the required infrastructure. Lightweight code mobility frameworks are being standardized by the IETF (*e.g.* Script MIB defines an SNMP-compliant MIB for code pushing [10]) so code transport in routers is not problematic nowadays. Also our system does not dictate code mobility support in every node. Code execution in routers is also viable – *e.g.* Cisco routers run Tcl scripts that have SNMP MIB object access [11]. Regarding the size of those agents we have a full implementation of the algorithm of Fig. 1 in Tcl – agent size is 50Kbytes and 1-hop migration time is 400msec. We have not specifically tackled the security and safety issues related to code mobility given our relaxed requirements (our algorithm merely needs read-only access to routing table) and the ample literature available in the subject.

We are currently applying our system to practical case studies including distributed monitoring, peer-to-peer systems, network overlays, application-level multicast, and content adaptation networks.

References

[1] O. Kariv and S. L. Hakimi, “An algorithmic approach to network location problems,” *SIAM J. Appl. Math.*, vol. 37, pp. 539-560, Mar. 1979.

[2] F. Buckley and F. Harary, *Distance in Graphs*. Addison-Wesley, 1990.

[3] M. S. Daskin, *Network and Discrete Location*. Wiley, 1995.

[4] A. Fuggetta *et al.*, “Understanding code mobility,” *IEEE Trans. Software Engineering*, vol. 24, pp. 342-361, May 1998.

[5] W. Stallings, *SNMP, SNMPv3 and RMON*. Addison Wesley, 1998.

[6] GT-ITM code, available at www.cc.gatech.edu/projects/gtitm

[7] E. W. Zegura *et al.*, “A quantitative comparison of graph-based models for Internet topology,” *IEEE Trans. Networking*, vol. 5, pp. 770-783, Dec. 1997.

[8] K. L. Calvert, M. B. Doar, and E. W. Zegura, “Modeling Internet Topology,” *IEEE Commun. Mag.*, vol. 35, pp. 160-163, June 1997.

[9] NS code and manual, available at www.isi.edu/nsnam/ns/

[10] J. Schönwälder *et al.*, “Building distributed management applications with the IETF script MIB,” *IEEE J. Select. Areas Commun.*, vol. 18, pp. 702-714, May 2000.

[11] Cisco IOS Scripting with Tcl, http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_2/gt_tcl.htm