Combined Path and Server Selection in Dynamic Multimedia Environments

Zhenghua Fu and Nalini Venkatasubramanian
Department of Information and Computer Science
University of California, Irvine , Irvine CA 92697-3425
{zfu,nalini}@ics.uci.edu

1. MOTIVATION

The evolving global information infrastructure consists of a wide-area networking backbone that provides connectivity among service providers and clients requesting multimedia services via different applications. As this infrastructure scales, service providers replicate data and resources on the network to serve more concurrent clients. Adaptive and intelligent scheduling techniques are required to increase the utilization of their resources and handle an increasing number of requests. Scheduling for multimedia applications must guarantee desired Quality-of-Service (QoS) from both the network path and the server. Furthermore, with the increasing amount of mobile clients and highly dynamic network topologies, optimizing resource utilization becomes complicated. In a highly dynamic and ad-hoc environment where clients are mobile, load sensitive routing and scheduling techniques must be able to tolerate some information imprecision. The information collection and scheduling processes must cooperate with each other, they cannot be viewed as independent components in the QoS provisioning framework. This paper deals with a framework in which scheduling decisions are based on path as well as server qualities.

Specifically, we address two problems in this paper.

1. Scheduling a request from a client with QoS constraints on (a) the path quality and the (b) server quality. This is framed as an optimization problem on the overall network and server loads while scheduling the request.
2. Information Collection to capture the current system state. We develop a model and algorithm for the parameter collection process that maximizes accuracy and minimizes traffic overhead.

We present an algorithm to combine the two problems into one optimization problem within a unified framework. The framework has two components. The first component implements the optimized scheduling algorithm, while the second component collects the network and server parameters used in the algorithm.

Our design objectives are: 1. Maximize the client request success probability. 2. Maximize the network and server utilization using load balancing techniques. 3. Tolerate the imprecision in network and server parameters 4. Minimize the overhead cost.

2. THE MODEL

We model the request R from client c as a triple: the path, the server and the end to end quality.

R:< $PATH_R, SERV_R, ETOE_R$ >

The requirement on the selected path is the available bandwidth, $PATH_R :< BW_R >$; the server requirements include the available capacities of CPU, memory buffer, disk bandwidth, and NIC (Network Interface card), [15],

$SERV_R :< CPU_R,\ BUF_R, DB_R,\ NIC_R >$; an End to End quality requirement is also postulated, e.g. end to end delay, $ETOE_R :< DL_R >$.

We model the network as a graph G<N, E> of nodes N and edges E. We assume the availability of a directory service that contains information about (a) S – the set of feasible replicas for a client request, and (b) available bandwidth and current delay of each link in the network and (c) available capacity of various resources and current response time on each server. Algorithms for data placement that provide a map of feasible replicas for a client request have been studied in previous work [15]. For a link *l*, we use a term BW^l_{avail} to note its available bandwidth and term DL^l to note its current delay (the delay includes the propagation delay and the queuing delay at the transmit end). By definition, for a path *p*, we have $BW^p_{avail} = \underset{l \in p}{Min}\left\{BW^l_{avail}\right\}$; $DL^p = \sum_{l \in p} DL^l$

We model the server using five parameters. The first four parameters correspond to server resources: available capacity of CPU, buffers, disk bandwidth and the network interface card bandwidth. In order to satisfy the end to end requirement, we are also interested in a fifth parameter, the response time of a server s, RSP^s. So given an assignment *X={p,s}*, We use the term

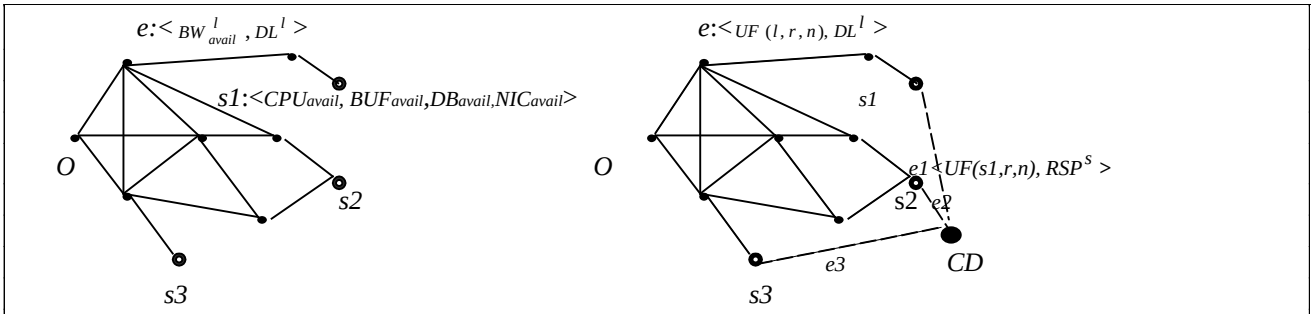

Figure 1. *Left: Graph G:<N,E> with the client requesting at point O and a set of feasible servers S: s1,s2,s3. Right: Graph extended from G, adding a point CD and artificial edges e1,e2 and e3*

EED^X to note the end to end delay of assignment X using path p and server s. By definition, we have

$$EED^X = DL^p + RSP^s,\ p,s \in X\ .$$

In order to deal with path and server selection in a unified way, we introduce the notion of a *Distance Function*, that is a measure of utilization of resources in a server or a path. The distance function represents the degree of congestion and is defined using the residue capacity after assigning a client request to the server or path. We initially define a utilization factor for links and servers to quantify the residue capacity. The utilization factors for a link l, given a request r and a parameter n, is defined as

$$\begin{cases} UF(l,r,n) = \left(\dfrac{1}{BW^l_{avail} - BW_r}\right)^n, \ if\ BW^l_{avail} > BW_r; \\ UF(l,r,n) = \infty, if\ otherwise. \end{cases}$$

The utilization factor for a server s, given a request r and a parameter is defined as

$$\begin{cases} UF(s,r,n) = \left(Max\left(\dfrac{1}{CPU^s_{avail} - CPU_r}, \dfrac{1}{MEM^s_{avail} - MEM_r}, \dfrac{1}{DB^s_{avail} - DB_r}, \dfrac{1}{NIC^s_{avail} - NIC_r}\right)\right)^n \\ if\ available\ CPU, BF, DB\ and\ NIC\ capacities\ greater\ than\ requested \\ UF(s,r,n) = \infty, \qquad Otherwise \end{cases}$$

The parameter n (n=0.5, 1, 2..) in the utilization factor represents the degree to which a lightly loaded resource is favored over a congested resource [11]. In a given schedule X{p,s}, we define the distance of the server s to be

$$Dist(s,r,n) = \sum_{l \in p, p \in X} UF(l,r,n) + UF(s,r,n),\ s \in X$$

The optimality condition: Given a client request $R:< BW_R, CPU_R, BUF_R, DB_R, DL_R >$, An assignment X*={p*, s*}, is optimal if and only if it satisfies all the following:

$$BW^{p^*}_{avail} \geq BW_R, \qquad (1)$$

$$CPU^{s^*}_{avail} \geq CPU_R, BUF^{s^*}_{avail} \geq BUF_R, DB^{s^*}_{avail} \geq DB_R, NIC^{s^*}_{avail} \geq NIC_R \qquad (2)$$

$$EED^{X^*} \geq DL_R \qquad (3)$$

$$Dist(s^*,r,n) = Min\{Dist(s,r,n)\},\ for\ all\ s\ in\ feasible\ set\ S \qquad (4)$$

3. COMBINED PATH AND SERVER SELECTION (CPSS) ALGORITHM

In this section, we propose an algorithm for the CPSS problem. Given a network topology *G*, a client request from point *O*, and a set of feasible servers *S*, we extend the existing topology *G<N,E>* to *G'<N', E'>* by adding one node called Common Destination, *CD*, and one artificial edge for each feasible server *s* in set *S*, denoted *Es*, from the server *s* to the common destination *CD[Figure 1]*.

Then for each edge *e* in Graph *G'*, the weight of *e* defined as *W(e):<UF,DL>*, according to the current collected parameters. *UF* corresponds to the distance (length) of the link, while *DL* is a constraint used in determining the restricted shortest path in the graph. Specifically, for *e(u,v)* in *E*, $W(e) = <UF(r,e,n), DL^e>$; for edge *e'(s,CD)* not in *E*, but in *E'*, $W(e') = <UF(r,s,n), RSP^s>$. To simplify the graph, we remove from G' those edges in which the available capacity is less than that requested. Finally we calculate a shortest path *P* from start point *O* to *CD*, subject to the constraint of *DL*. While the Restricted Shortest Path (RSP) problem is NP-hard, the algorithm can be implemented using dynamic programming by assuming an integer value of the Delay DL. For a detailed implementation of RSP in CPSS, see [7].

The CPSS algorithm *(G'<N', E'>, R, O, n,)*

```
(1)  /* initialization */
(2)  For each edge e(u,v) in G'
(3)      If e is in E,
(4)          if UF(e,R,,n) = INFINITY  then delete edge e
(5)          W(e).dist = UF(e,R,n); W(e).delay = DL^e
(6)      Else /* e is an artificial arc, e=(s,CD). */
(7)          if UF(s,R,n) = INFINITY then
(8)                  delete edge e
(9)          W(e).dist = UF(s,R,n); W(e).delay= RSP^s
(10)
(11) /* run the Restricted Shortest Path algorithm */
(12) P*{(O, v1), (v1,v2), .. , (s*, CD)} = RSP(G'<N',E'>, W, O,
     CD, DLR)
(13) return optimal assignment X*={ P* \ (s*, CD), s*}
```

Lemma 1: If path *P** is the RSP from origin *O* to Common Destination *CD*, there will be one and only one server *s* on path *P**.

Theorem 1 An assignment *X*={p*, s*}* is optimal if and only if path *P** = *p** ∪ *(s*,CD)* is the RSP from origin *O* to the Common Destination *CD*.

Theorem 2 The CPSS algorithm finds an optimal assignment in *O(|E|D)*.

The proofs of the above lemma and theorems can be found in [7].

4. PARAMETER COLLECTION IN A DYNAMIC ENVIRONMENT

The above CPSS algorithm is based on a given network topology and replica map, assuming knowledge of link and server load information. The network topology can be maintained by routing information exchange, and a replica map can be obtained from a distributed domain name service ([6], [10]). We propose an effective parameter collection method to deal with dynamic changes in network and server conditions. Probes that are distributed within the framework collect network and server information periodically; this information is used to update our directory service. The current state information together with the topology and replica map from the directory service is then used to make QoS server and path selections.

Delay and bandwidth information can be highly dynamic and often follows a heavy tailed distribution [2]. We assume that these values are changing discretely, in contrast to moving objects where the deviation function can be continuous[14]. However, it can be observed, that for a given period of time, the probability of the delay or available bandwidth taking certain values is high, and the trend of these "mean" values doesn't change dramatically. Hence we can use a predicted range of most probable values to approximate the state information with tolerable accuracy.

4.1 Parameter Collection Algorithms

A probe samples parameter values (e.g. delay) for a certain period *T(t).* We make this period a function of time to appropriately favor links rich in bandwidth over congested links. This sampling period does not necessarily correspond to the period with which the probe updates the directory. The update is made only when it is necessary, thus further reducing the overhead cost of our system model [1].

At a given point of time, a probe sends out a sampling packet and waits for the result. Upon receiving the result, it first determines whether the load of this link is changing *dramatically*, based on history data. If so, it could be because the link is admitting/releasing a load, or it could be because of a transient burst. In either case, the probe enters a "look-into" state. In the "look-into" state, the probe sends a fixed number of continuous packets to verify the "dramatic change". If verified, the probe updates the directory at once, and then updates the local history information. The probe doesn't update the directory until a new change is confirmed. We present the algorithms for the processes of the probe –the sampling process simply sends sampling packets according to period *T(t),* therefore is omitted here. Implementation details of the parameter collection process such as threshold and range manipulation are discussed in [7].

The look-into process

```
(1) Begin
(2)     Send out a series of packet to link
(3)     if the change confirmed
(4)         Update the history inventory
(5)         Enlarge the range value for link L
(6)         Update the directory
(7)     If the link is now getting congested
(8)         Enlarge T(t) for link L
(9)     If the link is now getting relieved
(10)        Shorten T(t) for link L
(11) END
```

The reading process *(Link L, Value v)*

```
(1)  Loop
(2)  Wait and read sample result (L,value)
(3)      If the value for link L is changing dramatically
(4)          Signal the look-into process to look into link L
(5)      Else
(6)          if the steadyTime(l) for link l >= sTHRD
(7)              Shorten the range value
(8)              Update the directory
(9)              steadyTime(l) = 0
(10)         Else
(11)             steadyTime(l)++
(12) END
```

5. ENHANCING THE CPSS ALGORITHM WITH UNCERTAINTY PARAMETERS

In order to perform effective path and server selection based on the uncertainty parameters that we introduced in section 3, we enhance the basic CPSS algorithm by incorporating a probability model for network link and server parameters. Our parameter collection process guarantees that for a short period of time interval, say *T(t)*, the possible values of a parameter will be distributed in the range *(L,H).* Since many existing traffic distributions are convex, e.g exponential, heavy-tail and Zipf, we use a uniform distribution, *U(L, H),* to make reasonable approximations of traffic distribution in the given time interval *T(t).* Now for link *l*, we have

$$\Pr^{l}_{BW}(b) = \Pr\{BW^{l}_{avail} \geq b\} = U(L^{l}_{BW}, H^{l}_{BW}) \quad ,$$

$$\Pr^{l}_{DL}(d) = \Pr\{DL^{l} \leq d\} = U(L^{l}_{DL}, H^{l}_{DL})$$

By definition, the probability that a the bandwidth on a path *p* from origin *O* to a server *s* is at least BW_R is

$$\Pr^{p}_{BW}(BW_R) = \prod_{l \in p} \Pr\{BW^{l}_{avail} \geq BW_R\}$$

However, given the probability distribution of the delay on each individual link, calculating the probability that the delay of a given path *p* from origin *O* to a server *s* is at most DL_R is much more difficult. The solution to this problem is to find an Optimal Partition of DL_R along the path *p* [12]. Assuming a uniform distribution of the delay value on each link, calculating the Optimal Partition of a given path needs *O(|p|).* *|p|* is the total

number of links of path *p*. We omit the server parameter discussion here due to space limitations. We briefly describe two methods to enhance the basic CPSS: the *Simple* and *Statistical* methods. See [7] for details.

5.1 The Simple Method

The objective is to derive values from a chosen probability distribution according to a policy, and use them to fit into the overall structure of the CPSS calculation. For each edge *e* in *G'*, we developed three polices to define the two weights in *W(e):<UF,DL>*. The *Pessimistic* Policy uses a lower bound of a uniform distribution, while an *Optimistic* Policy uses an expected value. The third policy, *Optimistic Favor Stable* Policy uses an expected value multiplied by a *"Degree of Stability"* calculated as $\frac{H^e - L^e}{Capacity^e}$.

5.2 The Statistical Method

We define an ε - **Optimality Condition**: Given a request R:< $BW_R, CPU_R, BUF_R, DB_R, NIC_R, DL_R$ >, and a threshold value ψ and a parameter ε, an assignment *X*={p*, s*},* is ε - optimal if and only if it satisfies all the following:

$$\Pr_{BW}^{p^*}(BW_R) \cdot \Pr_{CPU}^{s^*}(CPU_R) \cdot \Pr_{BUF}^{s^*}(BUF_R) \cdot \Pr_{DB}^{s^*}(DB_R) \cdot \Pr_{NIC}^{s^*}(NIC_R) \geq \psi \quad (1)$$

$$\Pr_{EED}^{X^*}(DL_R) \geq Max\{\Pr_{EED}^{X}(DL_R)\} - \varepsilon, \ \forall X = \{p, s\}, satisfies(1) \quad (2)$$

The optimality condition implies a two-step algorithm. Initially we have an extended graph *G':<N',E'>* where each network link has two probability distributions corresponding to the bandwidth and delay. Our mission is to find an ε -optimal path from the origin *O* to the common destination *CD*. The first step of the algorithm tries to satisfy the bottleneck typed constraints, such as bandwidth, by deleting the links from network topology and servers from the feasible server set that are less likely to satisfy the client request. From the reduced graph, we then try to satisfy the additive typed constraint, the End to End delay, by running an *OP-MP* algorithm [12] to find out the most probable paths from *O* to *CD* that have delay less than *DLR*. The set of paths P* that are ε - optimal can be used for multi-path connection setup[4],[5] or could be subject to further load balancing optimizations. The OP-MP problem is NP-Hard, by assuming an integer delay and uniform distribution, a heuristic is developed with complexity of $O(|E|^2(|V| + D_R))$. So our two step enhanced algorithm will have a complexity of $O(|E|^3(|V| + D_R))$.

In this short paper, we have briefly described our approach to providing combined path and server selection in a dynamic MM environment. We are currently evaluating the performance of the proposed algorithms and heuristics via detailed simulations

6 REFERENCES

[1] G.Apostolopoulos, R.Guerin, S.Kamat, S.K.Tripathi, Quality of Service Routing: A Performance Perspective.

[2] R.L.Carter and M.E.Crovella, "Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks", in Proceedings of Infocom '97.

[3] S.Chen, K.Nahrstedt, On Finding Multi-Constrained Paths, in Proceedings of IEEE International Conference on Communications (ICC 98), June 1998, Atlanta.

[4] S.Chen and K.Nahrstedt, Distributed Quality-of-Service Routing in Ad-Hoc Networks, IEEE Journal on Special Areas in Communications, Special Issue on Ad-Hoc Networks, 1999.

[5] I.Cidon, R.Rom, and Y.Shavitt. Multi-path routing combined with resource reservation. Infocom'97

[6] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service". Proc. IEEE INFOCOM '99, March 1999.

[7] Z.Fu, N.Venkatasubramania. Combined Path and Server Selection In Dynamic Multimedia Environments, Technical Report, ICS Dept., UC Irvine.

[8] J.D.Guyton and M.F.Schwartz, "Locating nearby copies of replicated internet servers," in Proceedings of ACM SIGCOMM, August 1995

[9] R.Hassin. Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research, 17(1):36-42, February 1992

[10] S.Keshav, R.Sharma, and R.Siamwalla, "Project octopus: Network Topology discovery", url:http://www.cs.cornell.edu/cnrg/topology/Default.html, May 1998

[11] K.Lang and S.Rao. Finding Near-Optimal Cuts: An Empirical Evaluation. In Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 212-221, 1993, Austin, Texas

[12] D.H.Lorenz and A.Orda. QoS Routing in Networks with Uncertain Parameters. INFOCOM '98

[13] Q.Ma, P.Steenkiste and H.Zhang. Routing high-bandwidth traffic in max-min fair share networks. Proceedings of SIGCOMM'96, August 1996

[14] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, G. Mendez, "Cost and Imprecision in Modeling the Position of Moving Objects", Proceedings of the 4th International Conference on Data Engineering (ICDE14), Feb. 1998.

[15] N.Venkatasubramanian and S.Ramanathan, " Load Management for Distributed Video Servers", Proceedings of the Intl. Conference on Distributed Computing Systems (ICDCS '97), May 1997.