6-1-1999

Optimal Video Distribution Using Anycasting Service

Zheng da Wu
Bond University, Zheng_Da_Wu@bond.edu.au

Chris Noble
Bond University

Dawei Huang

Optimal Video Distribution Using Anycasting Service

Z.D. Wu and C. Noble
School of Information Technology Bond University
Gold Coast, Queensland 4229,Australia
Telephone: +61 755 953311; Fax: +61 755 953320
E-mail: wz@bond.edu.au; chris_noble@bond.edu.au

D. Huang
School of Mathematics
Queensland University of Technology
Gardens Point Campus, GPO Box 2434 Brisbane Qld 4001, Australia
Telephone: +61 073 864 5195; E-mail: d.huang@fsc.qut.edu.au

Abstract

In the Internet, a group of replicated servers is commonly used in order to improve the scalability of a networked service. An anycasting service is the new network service that resolves an anycast address to one its member IP address from the group. The anycasting service uses some criteria to choose the one "best" server as a destination. In this paper we will present an optimal algorithm for mapping each anycasting query from clients into the one "best" video distribution server of replicas, such as Video-On-Demand servers on the Internet. The algorithm is developed at application-layer of the network based on economic models and queuing theory. By using this algorithm clients can get maximal satisfaction while system resources can be utilized most efficiently.

Keywords: *Anycasting; Economic Models; Optimal Load Distribution; VoD Server.*

1 Introduction

In order to increase service availability and provide efficient load distribution in a network, it is common practice to replicate servers on the network today. Examples of such services include World-Wide-Web "mirror" sites, Video-On-Demand, SOCKs servers [4], compute-servers and proxy servers in the Internet. This is often referred to as the scalability of a service. There have been different approaches proposed for improving the scalability of a network service, such as caching, batching of requests at the server, multicasting of server responses, and server replication over the network. In this paper we only deal with anycasting mechanism that can support server replication.

The notion of anycasting in the Internet was first introduced by C. Partridge, T. Mendez, and W. Milliken in [6]. Anycasting is also defined in IP version 6 (IPv6) [7]. Base on the definition of anycasting in [6][7], we may understand this service from several aspects as follows: (i) an IP anycast address (one of unicast IP addresses) is used to identify a group of servers that provide the same service; (ii) an anycast message is the one that should be delivered to one member of a group of designed recipients or servers and preferably to the "best" member. (iii) it is different from multicast in which a member must be sent to every member in the group.

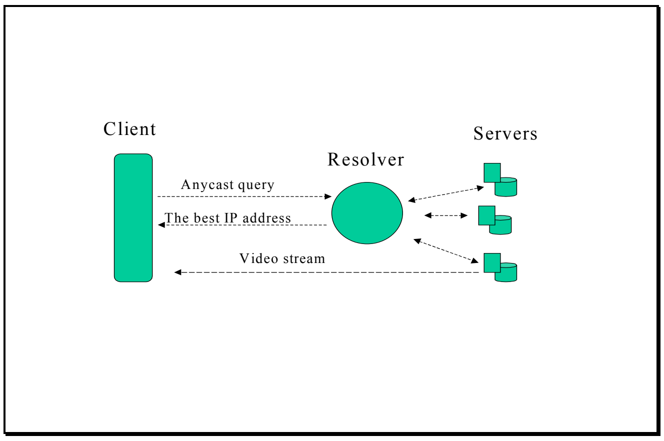
There are two notable research reports in anycasting, which represent two different approaches to working on this area. In report [5], the authors designed and implemented a network-layer anycasting service for load distribution in the context of the IBM Olympic Web site on the Internet. They investigated how the IP anycast service can be exploited by hosts connected to the Internet without significantly impacting the routing and protocol processing infrastructure already in place. The authors proposed possible enhancements to routing and forwarding to fully exploit the potential of anycast service. In another report [8], application-layer anycasting communication paradigm was proposed to support server replication. Specifically, the authors developed and evaluated an implementation based on the use of anycast resolver to map anycast domain names into one or more IP addresses. The 3W-server replication is chosen as a target to use anycasting service in both of the work. In this paper, we will present an optimal algorithm for mapping each anycast request (or query) into one "best" video distribution server of the replicas, for example, a group of Video-On-Demand servers which provide the same video stream service on the Internet. Our work is developed at application-layer and followed the notion presented in [8]. The algorithm design is theoretically based on economic models and queuing theory. By using this mechanism, clients or users with

Quality of Service (QoS) requirements can obtain maximal satisfaction, the resources of server replication can be utilised efficiently and therefore the whole system can achieve the best performance.

This paper is organised as follows. Following the description of application-layer anycasting in Section 2, we introduce an optimal algorithm for video distribution using the anycasting service in Section 3. In Section 4, we discuss economic models and optimization technique, the theoretical base on which the algorithm is proposed and then, demonstrate some numerical results. Finally, a conclusion and further discussion about this subject is given.

2 System Description

In this section we will follow the notion presented in [8] to briefly describe application-layer anycasting. An anycast domain name (ADN) is used to identify a group of servers that provide the same service. We consider a collection of VoD servers that may be located in different places on the network and offer video stream service. An anycast domain name uniquely identifies a set of IP addresses, which forms an anycast group. Such group can be made up entirely of unicast IP addresses or entirely of multicast IP addresses. In both cases, anycasting service provides a mapping from the AND representing the group to the "best" IP address in the collection according to some criteria. Conceptually, each of the IP address in the group represents the address of one replicated server and reaching any one of them is acceptable, but in practice one of important problems is how to realize an anycasting service to find out the "best" IP address or memeber. The basic motivation of this paper is to try to answer this question by developing an optimal algorithm for making such decision in the context of video distribution service. The anycasting request/response cycle is shown in Figure 1 as below.

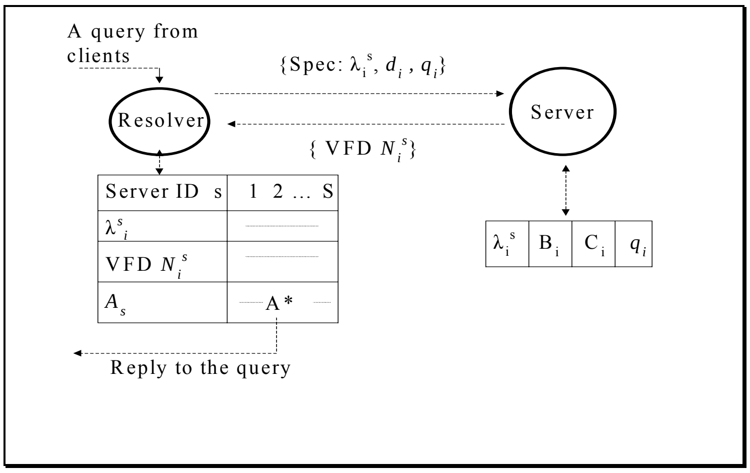


A client initiates an anycast query with an anycast domain name and its additional request-specification, such as QoS parameters, the resolver processes the query by executing a decision algorithm and replies with the "best" IP address as its anycast response. The anycasting architecture is like the existing Domain Name System (DNS) infrastructure in the Internet [10]. Logically, we consider issues related to anycast resolver design separately from the DNS. In reality, the functions of anycast resolver could be integrated with operation of the DNS[8].

Now, let us discuss what is the meaning of the "best" server from among multiple replicated servers. We may investigate this problem from two aspects as follows: (i) from the point of view of clients, they wish to obtain their maximal satisfaction for accessing the service, for example, they wish to gain the minimal call blocking probability under their QoS requirements; (ii) from the point of view of a whole system, it should achieve its best performance whenever a response IP address mapping is resolved. In other words, the system resources should be utilised most efficiently under the "best" service chosen. The quantitative expressions for the user satisfaction and the system performance will be given in Section 4.

3 Algorithm

In this section, an optimal algorithm for mapping each anycast request into the "best" video distribution server of the replicas is given. Its theoretical base and details of optimization procedure will be discussed in the next section. The algorithm consists of two modules, which are implemented in the resolver and each server of a group respectively, and is illustrated in Figure 2 as below:



For the resolver, a table or data structure is constructed to contain system measures and characteristics referenced by each server ID, *s* in the table. λ_i^s represents the average arrival rate of the ADN query at server *s* with type of service, *i*. A_s denotes an unicast IP address assigned to server *s*. N_i^s represents the value for decision (VFD) during the mapping process which is generated from server *s* for each of service type *i*. q_i^s denotes a QoS parameter of service type *i* which can be linked to λ_i^s (not shown in the diagram). Whenever an ADN query arrives at the resolver, it will be in a queue. The resolver will execute the mapping process for the queries waiting in the queue under the FCFS discipline. For each query of type *i*, the resolver will find out the maximum of N_i^s among all of servers, $N_{s,i}^{\max}$, and the corresponding server, *s**. Consequently, the IP address, A^*, can be chosen as the "best" response to its client.

Another function of the module in the resolver is to update the values of entries in the table. At each time of updating, the module will determine a new value of λ_i^s and send them to each server. Once each server receives this value, the server is triggered and immediately computes its new value of the VFD, N_i^s, as its response to the resolver. The resolver will update the VFD in the table immediately after each of them arrives in.

The second module will be implemented in each server. The server also needs a data structure to keep information for generating the VFD value once it is triggered by the resolver. Besides λ_i^s, q_i^s and N_i^s defined as the same characteristics as above, two additional measures are needed and they are: B_i^s - the current free buffer space available for each type of service at server *s* and C_i^s - the current available bandwidth of network output link for server *s* with type *i*. Each server will run an optimization procedure which takes B_i^s, C_i^s, λ_i^s and q_i^s as input values and produces the VFD, N_i^s, as its output value, which will be sent to the resolver immediately after it is generated each time. The details of how to determine the VFD within each server and how and when to update λ_i^s within the resolver will be given in the next section.

4 Model and Optimization

In order to find out the "best" server for each anycast query from among multiple replicas, an optimal decision algorithm is proposed in the last section. The algorithm is theoretically based on analytic techniques in economics and queuing theory. By using this algorithm, the system resources are utilized most efficiently, meanwhile the users (or clients) can gain their maximal satisfaction. The detail of mathematical models used in the algorithm is now presented as the following.

Suppose that a video distribution system on the network consists of *S* servers, each of them may be identified as *s*, *s* = 1, 2,..., *S*. Each video server *s* provides concurrent video streams or connections with different type of service *i* , *i = 1, 2,..., I.* Each type of video streams is associated with its display speed at d_i frames per second, which is requested by its end-user or client. We define the value for decision (VFD), N_i^s, as the number of potential connections which can be additionally supported by server s. It notes that N_i^s does not represent the number of connections the server have set up. Assuming that anycast query arrivals of service type *i* is a Poisson process with average rate λ_i which will be further divided by the algorithm into arrival sub-rate λ_i^s for each server s. Thus, $\lambda_i = \sum_{s=1}^{S} \lambda_i^s$. The rule of this division will be discussed at the end of this section. We denote b_i^s, measured in frames, as buffer size to be needed for establishing connections of new users of service type *i* with server s and c_i^s in frames per second as bandwidth requirement for the same reason. Thus, $\sum_{i=1}^{I} b_i^s \leq B^s$ and $\sum_{i}^{I} c_i^s \leq C^s$, where B^s and C^s represent the total of free buffer space and the total of available bandwidth of network output link with server s respectively. Suppose that the video streams are sent over the network from the server at the same rate as they display on the end-user device. Therefore, $c_i^s = N_i^s \, d_i$.

In practice, users prefer to minimize their call blocking probability. Thus, user satisfaction or preference may be measured by the call blocking probability, q_i. As a result, a utility function representing the user satisfaction is expressed as $U_i^s = q_i^s = q_i$. It is reasonable to assume that q_i^s is independent of the identifier of a replicated server. The call blocking is due to the shortage of system resources, such as buffer and bandwidth with a server. For a server the client request, which was refused by the server, is often referred to as "lost" due to limited resources available in the server. We may employ *M/M/1/B* queuing model [11] to evaluate the loss rate, δ_i^s, as follows:

$$\delta_i^s = 1 - \sum_{k=0}^{b_i^s} p_k^{i,s}$$

where

$$p_k^{i,s} = \begin{cases} \dfrac{1-\lambda_i^s / c_i^s}{1-(\lambda_i^s / c_i^s)^{b_i^s+1}} (\lambda_i^s / c_i^s)^k & 0 \le k \le b_i^s \\ 0 & otherwise \end{cases}$$

Given the preferences of all users with different types, we can aggregate their utility functions into a social welfare function, which may represent the performance of a server as whole [1]. There are two methods to define the social welfare function for this optimization problem: (i) we use a social welfare function to directly reflect the call blocking probability of users requested for the service; (ii) we employ a social welfare function to represent the total number of potential connections, $\sum_{i=1}^{I} N_i^s$, which can be additionally supported by server *i*. Logically, these two methods are equivalent in terms of the meaning of choosing the "best" server as discussed in Section 2. In this paper we will use the second method and thus, we may define the social welfare function for each server s as follows:

$$W_s(U_1^s, U_2^s, \cdots, U_I^s) = \sum_{i=1}^{I} N_i^s$$

Then, we can pose the welfare maximization problem:

$$Max \qquad W_s(U_1^s, U_2^s, \cdots, U_I^s)$$

$$\begin{aligned} &Such\ \ that \\ &U_i^s = q_i \le \delta_i^s \\ &\sum_{i=1}^{I} b_i^s \le B^s \\ &\sum_{i=1}^{I} c_i^s \le C^s \end{aligned}$$

where $0 \le q_i < 1$. The first constraint implies that the use preference of each type must be satisfied under a certain limitation. Specifically, q_i represents the call loss rates those users of type *i* may be tolerated. In optimization problems in economics, a very neat technique called "the method of Lagrangian multipliers" is generally used. For the case of two types of users, *i = 1, 2*, we seek to maximize the social welfare function above, by using the Lagrangian theorem we can obtain the following equations:

$$\frac{\Phi_1 - b_1^s}{c_1^s \log \rho_1^s} = \frac{\Phi_2 - b_2^s}{c_2^s \log \rho_2^s}$$

$$B^s = b_1^s + b_2^s$$

$$C^s = c_1^s + c_2^s$$

Where $\rho_1^s = \lambda_1^s / c_1^s$, $\rho_2^s = \lambda_2^s / c_2^s$ and

$$\Phi_i = \frac{\rho_i^s (1 - q_i)}{1 - \rho_i^s (1 - q_i)}$$

Through these equations we can achieve a set of Pareto allocation of buffer and bandwidth for users of each type *i*, $\{c_i^s, b_i^s\}$ [1][2][3]. Based on this optimal resource allocation we can easily estimate the maximum number of potential connections, $N_{s,i}^{\max}$, for each type of users *i*, that can be additionally supported by server *s* using $N_{s,i}^{\max} = c_i^s / d_i$.

A simulation is carried out based on the algorithm for optimal resource allocation with server *s*, as discussed above. In the simulation, we express the utility function for each type *i* as $U_i^s = q_i$ and *i = 1, 2*. It means that we assume two types of users compete for resources (buffer space and network output link capacity) in the numerical analysis. Suppose that the buffer space and the network output link capacity of server *s* are $B^s = 80$ frames and C^s = 100 frames/s respectively. The request arrival rates from clients to server *s* are λ_1^s and λ_2^s while the frame output rates or frame display rates are $c_1^s = d_1$ and $c_2^s = d_2$. Certainly, the display rates depend on their service types. The typical examples of video data types include 15 frames/s animation, 30 frames/s NTSC video, and 60 frames/s HDTV video. Based on theses specifications we can now determine optimal buffer allocation, b_i^s, under call loss constraints, $U_i^s = q_i \le \delta_i^s$. We let $q_1 = 10^{-3}$ and $q_2 = 10^{-4}$ or 10^{-2}. Once we get resource allocations satisfying the call loss constraints we can estimate the maximum number of streams or connections the server can support additionally. We assume $d_1 = 15$ frame/s and $d_2 = 30$ frames/s which represent two typical video display rates, and remain all assumptions above. The numerical results are first shown in Table 1 and Table 2 where they have different values of q_i. In these two tables, $B_{\min}^s$ (certainly, $B_{\min}^s \le B^s$) represents the minimum buffer space required so as to achieve the maximum number of the additional connections denoted as N_i^s for type *i*. Thus, $N_1^s + N_2^s$ is the total number of connections that can be added to the server. According to the numerical results, we remark that (i) if the video demand (or request) arrival rate λ_i^s increases, the buffer size to be allocated also increases; (ii) the call loss constraints impact on the buffer size to be allocated but they do not affect the maximum number of additional connections very much. Let us examine Table 3 where we remain buffer space $B^s = 80$ frames and increase the network output link capacity up to $C^s = 200$ frames/s. Comparing with Table 1 and Table 2, this change leads the maximum number of additional connections increased significantly meanwhile the allocated buffer size becomes smaller.

Table 1: Assuming $B^s = 80$, $C^s = 100$, $d_1 = 15$ and $d_2 = 30$.

λ^s_1	λ^s_2	q_1	q_2	B^s_{min}	N^s_1	N^s_2	$N^s_1 + N^s_2$
10	*30*	10^{-3}	10^{-4}	*35*	*4*	*1*	*5*
10	*50*	10^{-3}	10^{-4}	*50*	*2*	*2*	*4*
30	*50*	10^{-3}	10^{-4}	*65*	*2*	*2*	*4*
40	*40*	10^{-3}	10^{-4}	*70*	*3*	*1*	*4*
40	*50*	10^{-3}	10^{-4}	*>80*	Φ	Φ	Φ

λ^s_1	λ^s_2	q_1	q_2	B^s_{min}	N^s_1	N^s_2	$N^s_1+N^s_2$
10	*30*	10^{-3}	10^{-2}	*20*	*4*	*1*	*5*
10	*50*	10^{-3}	10^{-2}	*25*	*2*	*2*	*4*
30	*50*	10^{-3}	10^{-2}	*30*	*2*	*2*	*4*
40	*40*	10^{-3}	10^{-2}	*35*	*3*	*1*	*4*
40	*50*	10^{-3}	10^{-2}	*>80*	*Φ*	*Φ*	*Φ*

Table 2: The call loss constraints are set to different values from Table 1.

λ^s_1	λ^s_2	q_1	q_2	B^s_{min}	N^s_1	N^s_2	$N^s_1+N^s_2$
10	*30*	10^{-3}	10^{-4}	*20*	*9*	*2*	*11*
30	*50*	10^{-3}	10^{-4}	*50*	*9*	*2*	*11*
50	*70*	10^{-3}	10^{-4}	*65*	*8*	*2*	*10*
70	*90*	10^{-3}	10^{-4}	*70*	*5*	*4*	*9*
80	*90*	10^{-3}	10^{-4}	*>80*	*Φ*	*Φ*	*Φ*

Table 3. Assuming $B^s = 80$, C^s=200, d_1=15 and d_2=60.

In order to update the metrics in the mapping table, we must have an algorithm that the resolver will use to determine when the arrival rate of anycast query for each type of service, λ_i is estimated based on the traffic measurements from clients. As suggested in [8], we also borrow and adopt the link state update algorithm used in the ARPANET [9]. The update algorithm is parameterized by a measurement interval *L*, a maximum threshold *T* and a reduction factor *R*. The algorithm maintains a current threshold *M*, initialized to *T*. The resolver measures its state over each interval *L*. If the state changes from the previous measurement by at least *M,* the state is pushed and *M* is reset to *T*. If the state does not changes by at least *M, M* is reduced by *R*. The algorithm will produce at least every *T/R* time units and at most every *L* units. Based on this updating algorithm,

λ_i can be first estimated at each time of updating. Then, we can compute λ_i^s for each server s by using the expression as follows:

$$\lambda_i^s = \frac{\lambda_i N_i^s}{\sum_{i=1}^{i=I}(N_i^1 + N_i^2 + \cdots + N_i^s)}$$

It means that if a server can support more number of potential connections, the resolver will return its address to more number of clients.

Conclusions

We have presented an optimal algorithm for mapping each anycast request (or query) into one "best" video distribution server of the replicas in the context of a group of Video-On-Demand servers, which provide the same video stream service on the Internet. Our work is developed at application-layer. The algorithm design is theoretically based on economic models and queuing theory. By using this mechanism, clients or users with Quality of Service (QoS) requirements can obtain maximal satisfaction, the resources of server replication can be utilized efficiently and therefore the whole system can achieve the best performance. The numerical results provide useful insights in the performance behaviors of a video server. It is noted that if the number of service types is large, the cost of computation for optimal resource allocation will be high. However, this will not cause too many overheads in the system since the server is usually equipped with a high-performance computer.

Acknowledgments

The work described in this paper was funded by the Australian Research Council Grant.

References

[1] Hal R. Varian, "Intermediate Microeconomics - A Modern Approach", (Second *Edition), 1990.*

[2] S. Charles Mailrice and Owen R. Phllips, "Economic Analysis: Theory and Applications", *(Fifth Edition), 1986.*

[3] D. F. Ferguson, C. Nikolau and Y. Yemini, "An Economy for Flow Control in Computer Networks", *Proc. of the IEEE INFOCOM, 1990.*

[4] M. Leech, M. Granis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS protocol version", *RFC 1928, 5 April 1996*

[5] Erol Basturk, Robert Engel, Robert Haas, Vinod Peris and Debanjan Saha, "Using Netivork Layer Anycast for Load Distribution in the Internet", IBM *Research Report, RC 20938 (07129197), IBM Research Division, T. J. Watson Research Center, New York.*

[6] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service", *RFC 1546, November 1993*

[7] R. Hinden and S. Deering, "IP version 6 addressing architecture", *RFC 1884,* December *1995.*

[8] S. Bhattacharjee, M. Ammar, E. Zegura, V. Shah, and Z. Fei, "Application Level Anycasting", *Tech. Rep. 96-25, College of* Computing, Georgia Institute *of Technology.*

[9] E. C. Rosen, "The updating protocol of ARPANET's new routing algorithm", Computer *Networks, No. 4, pp.11-19, 1980.*

[10] P. Mockapetris, "Domain names - concepts and facilities ", *RFC 1034,* November *1987.*

[11] L. Kleinrock, "Queucing Systems", Vol. *1 2, Vol.1, Wiley, 1976.*

[12] Z.D. Wu, "Dynamic Sessions Control for Multimedia Distributed Systems", *Proc. of IEEE ICC'97, pp. 1148-1152.*