

A Scalable and Efficient End Host Multicast Protocol for Peer-to-Peer Systems—DSCT

Wanqing Tu

Department of Computer Engineering
and Information Technology,
City University of Hongkong,
Kowloon, Hongkong, P.R.China.
E-mail: tu.wanqing@student.cityu.edu.hk

Weijia Jia

Department of Computer Engineering
and Information Technology,
City University of Hongkong,
Kowloon, Hongkong, P.R.China.
E-mail: itjia@cityu.edu.hk

Abstract—We study a scalable and efficient end host multicast—DSCT (Dynamic Shared Cluster Tree) for P2P systems in this paper. In our design, we focus on making full use of underlying physical properties to improve both scalability and efficiency as data multicast in the overlay topologies of P2P systems. In this paper, we present the DSCT architecture, the dynamic scheme to select cluster cores according to the instantaneous network situations. These are all the key points to construct the basic DSCT (BDSCT) multicast. Through further studies, we provide DSCT improvement design (IDSCT) to make it more suitable for multiple-source P2P communications. Our simulation results indicate that it is possible to improve the scalability and efficiency of end host multicast in P2P systems by using our DSCT designs.

I. INTRODUCTION

Peer-to-peer (P2P) computing is the system of mutually exchanging information and services directly between the sender and the receiver. Based on the way to discover other peers, query peers for content and share content with other peers, P2P architectures are generally classified into two categories: the *centralized* and *distributed* P2P architectures. In the *centralized* architecture, a central server exists. It maintains an index of all available files in the system and coordinates the communications of the participating peers. In the *distributed* architecture, there is no such a central server. Each peer is only responsible for maintaining the index of files it stores. Query from a peer is flooded to other peers when it wants to locate a file in the system. In these two kinds of P2P architectures, the *centralized* one is fragile and not a real P2P architecture because it fully depends on the central server. The *distributed* one is relatively stable, but its query flooding mechanism makes it less scalable and incurs longer latency. If scalability concerns were removed from the distributed P2P systems, the *distributed* architecture might be the preferred choice for file-sharing and other P2P applications. Therefore, it is worthwhile to improve the scalability of distributed P2P architecture. Multicasting query can improve the scalability, moreover, efficient and simple multicast won't bring complexity to query. Furthermore, a key concept for P2P systems is to permit any two peers to communicate with one another in such a way that either ought to be able to initiate the communication, hence, P2P is a powerful tool for organizing cooperative

communications, which covers both one-to-many and many-to-many applications. To set up these applications, a multicast routing protocol is necessary. It is because that the multicast can improve much more scalability of these applications than the unicast. Therefore, multicast is necessary for both P2P cooperative systems and P2P file-sharing systems.

Since the end hosts in the P2P networking construct an overlay network, such overlay network structure is in the sense that each of its edges corresponds to an unicast path between two end hosts in the underlying Internet, we implement the multicast for P2P systems in the overlay network, which is called *end host multicast* by many researchers. Unlike the network layer multicast, the *end host multicast* has recently gained popularity as an alternate way to implement the multicast independent of the network architecture. Current proposed end host multicast protocols can be classified as the *mesh-first*, *tree-first* and *implicit* end host multicast protocols. NARADA [2] is a *mesh-first end host multicast protocol*. It demonstrates the feasibility of implementing the multicast functionality at the application layer. NARADA distributedly organizes group members into the overlay mesh topology, and runs the routing protocol, e.g. DVMRP, to compute the unicast paths between all pairs of members on the mesh. To guarantee robust, each group member periodically sends the refresh message to all other group members, which compromises the scalability of NARADA. Unlike NARADA, YOID [3] is a *tree-first end host multicast protocol*. It firstly constructs a distributedly shared data delivery tree among members. The tree structure achieves logarithmic scaling behavior. But YOID is neither efficient nor simple, because it has a direct control over various aspects of tree structure (e.g. the out-degree at the members). Moreover, expensive techniques of loop detection and avoidance, and resilience to tree partitions are required in YOID. HMTP [4] is another representative of the *tree-first end host multicast protocol*. Well-known NICE [5] and CAN-based multicast [6] adopt the *implicit* approach to conduct the multicast function, in which the mesh and the tree are simultaneously defined by the protocols. CAN-based multicast is based on an application layer infrastructure - Content-Addressable Network (CAN). It adopts the forward flooding scheme to multicast packets to all the neighbors who haven't received the packets before. NICE

is a hierarchical multicast. Each layer has a set of clusters, each cluster has a cluster leader and includes $k \sim (3k - 1)$ (k is normally set as 3) group members. In NICE, packets are transmitted to cluster leaders which then forward them to all their cluster members, except in the source cluster where packets are sent to all the cluster members by the source. Compared with other protocols, NICE is more scalable for its hierarchy and cluster. However, the hierarchy and cluster is fully independent of the underlying topology. Actually, by the underlying network properties, we study an end host multicast for P2P systems — DSCT (Dynamic Shared Cluster Tree), which improves both scalability and efficiency of end host multicast. *Shared Tree* means that the multicast data path is a shared tree. *Cluster* is a “mini-group”. Peers who are physically close enough to one another can be assigned into the same cluster. *Dynamic* means that DSCT tree dynamically changes with the alterations of membership to guarantee the efficient communications.

The rest of the paper is organized as follows: Section II presents our motivation and the detail designs of BDSCT and IDSCT. Simulation evaluations and performance comparisons of CAN-based multicast, NICE, BDSCT and IDSCT are provided in Section III. Section IV concludes the paper.

II. DSCT DESIGN

A. Motivation

One of the dominant differences between the end host multicast and the network layer multicast is that packets are forwarded by the end hosts in the end host multicast instead of intermediate routers. End host forwarding incurs the identical packet traversing the same underlying link more than one time, which is one of the major reasons that the end host multicast is less scalable than the network layer multicast. Our studies also show that the end host forwarding causes not only less scalability but larger transmission delay as well. It is because of the excessive delay that the packets transmit from the application layer to the network layer in the end host forwarding. Moreover, end hosts have lower capacity to deal with packets than intermediate routers, which incurs longer processing delay. Hence, end host multicast is costly in terms of delay.

Our motivation is to design a scalable and efficient end host multicast for P2P systems based on the underlying topology characteristics. We first make the following definitions. 1) *Local domain*: It is composed of group members that attach to the same router directly or through several local network components (e.g. the hubs) and other local network resources (e.g. physical links, the hubs) used to connect these members. In this definition, the router mainly refers to the local router, not the router in the backbone network; 2) *Backbone domain*: It is composed of all the routers used to connect the multicast group members and other network resources (e.g. physical links) among these routers. The routers in this definition includes the local routers and the backbone routers; 3) *Unassigned end hosts*: Define the end hosts in a multicast group that have been assigned into any cluster as the *unassigned end hosts*. In our

design, the basic idea to improve the scalability and efficiency is to limit the packet transfer in the *backbone domain* links, which reserves the *backbone domain* resources to carry many more end hosts and decreases the chance that packets transmit in the long delay *backbone domain* links.

B. Basic DSCT Design (BDSCT)

In NICE, hierarchy has been proven to be effective in expanding scalability of end host multicast. DSCT follows the NICE hierarchy and make several improvements. It assigns group members into different layers, then partitions the members in the same layer into different clusters. The following is the regulation to construct DSCT.

- *Cluster formation*. In DSCT tree, each *local domain* has a *local core*. The selection of *local core* is introduced in the *layer formation*. Members in the same *local domain* form the “intra-clusters”. *Local cores* in different *local domains* form the “inter-clusters”. Different cluster sizes exist for these two kinds of clusters. As for the “intra-cluster”, the size is expressed as:

$$s_{ina} = \begin{cases} (k, 3k - 1) & , r_n > 3k - 1 \\ r_n & , r_n \leq 3k - 1 \end{cases} \quad (1)$$

where r_n is the number of *unassigned end hosts* in the same *local domain*, and the expression $(k, 3k - 1)$ represents a random constant between k and $3k - 1$. Like NICE, k is a constant, and in our experiment, we also use $k = 3$. Expression (1) guarantees that each “intra-cluster” only contains members of the same *local domain*. Hence, the packet transmission in the *backbone domain* links is limited. Similarly, as the “inter-cluster” contains end hosts of different *local domains*, the size is denoted as:

$$s_{ine} = (k, 3k - 1) \quad (2)$$

- *Layer formation*. *Local domain* members are mapped to different layers using the following scheme: Each cluster has a cluster core to coordinate the packet transmission between the inside and outside cluster members. The cluster core selection is based on the *dynamic core searching scheme*, which will be introduced in the next subsection. Cluster cores in layer L_i construct layer L_{i+1} , and are partitioned into different clusters in layer L_{i+1} by the *cluster formation*. New cluster cores in layer L_{i+1} are selected and form an immediate upper layer L_{i+2} . The hierarchy won't terminate until the number of members in the current layer is not greater than $3k - 1$. The core of these members is the *local core*. It constructs the upper layer of its *local domain*. By using the similar scheme, *local cores* are mapped into higher layers than the local upper layer. In these higher layers, they are partitioned into different clusters with the sizes of $(k, 3k - 1)$. The hierarchy won't terminate until the number of cluster cores in the current layer is not greater than $3k - 1$. The core of these *local cores* is the *DSCT core*. It constructs the highest layer of DSCT.

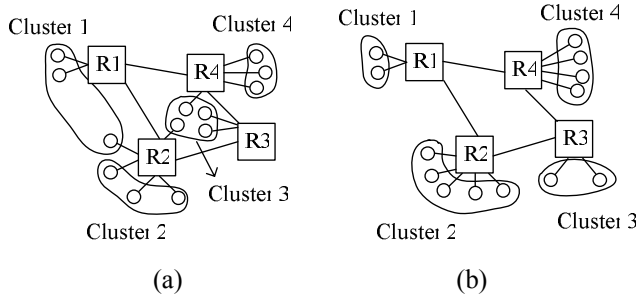


Fig. 1. Cluster formation in the lowest layer of NICE and DSCT. (a) is for NICE, (b) is for DSCT.

The following is the analysis of DSCT architecture. Through hierarchy, the whole large group management is divided into the management of several individual “mini-groups”. It greatly decreases the periodical control traffic among group members that belong to different “mini-groups”. Hence, many more network resources can be reserved for data transmission which improves the communication scalability. Apart from this, as shown in Fig. 1 and Fig. 2, DSCT decreases the data flow in the *backbone domain* links through the *cluster formation* and the definition of *local cores*, which reserves *backbone domain* capacities to carry more *local domains* and limits the chances that packets occupy the time-cost *backbone domain* links. Therefore, the scalability and efficiency are improved.

Fig. 1 is the comparison of DSCT and NICE *cluster formations*. Let $k = 3$, then the NICE cluster size is (3,8). Communications in Cluster 1 and Cluster 3 use the links between R_1 and R_2 , R_2 and R_3 , R_3 and R_4 . However, in (b), DSCT *cluster formation* guarantees that each “intra-cluster” only includes *local domain* end hosts. Hence, after message transmits to the cluster core, only local links are employed to forward the message to other cluster members. The data traffic in the *backbone domain* links of DSCT is lighter than the one of NICE.

Fig. 2 shows the difference of data transmission in DSCT and NICE architectures. In NICE, as there are no *local cores*, the data transmission in the cluster of the lowest layer must traverse the link between two routers three times. However, in DSCT architecture, each *local domain* has one *local core*, after the *local core* achieves data from their “*inter-cluster*” members, the data only need to transmit in local links.

C. Dynamic Core Searching Scheme

In DSCT, to guarantee that each group member achieves lower delay communication, cluster cores have the property that they have the minimum sum of unicast delay to other cluster members. Our solution of searching the core is based on the probe scheme. Probe scheme has been studied by many well-known systems [7][8][9][10]. In these systems, probes are either used to fetch server load information or used to record the paths/nodes’ availability that the probes traversed on the hop by hop basis. In DSCT, we propose a core selection scheme based on using *on-demand* light weight probes, which carry timestamps to collect delay properties, RTT, of logic links in the overlay network. It is novel to incorporate RTT

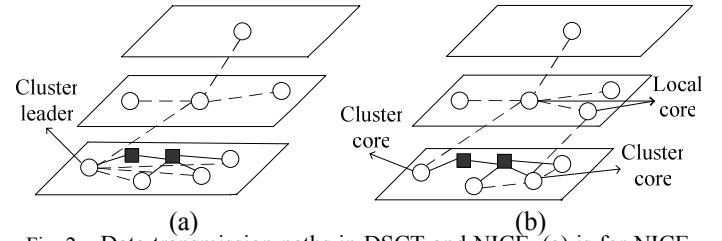


Fig. 2. Data transmission paths in DSCT and NICE. (a) is for NICE, (b) is for DSCT. The dotted lines represent the data multicast paths. The black blocks represent the routers. The small circles represent end hosts.

into the multicast core selection and the recovery of dynamic group alterations, although the probe scheme itself is not new.

In DSCT *on-demand* probe scheme, the request of probing is triggered by the dynamic alterations of cluster members. Once there is the alteration of membership, each cluster member injects probe packets with timestamp to the network that solicit responses from other cluster members. Once other cluster members receive the packet, they immediately send replies returning the timestamp value. When the initiator receives the reply, it measures the elapsed time. The elapsed time measures the RTT. The following notations will be used in order to describe the scheme.

- RTT_{ij} : The record stores the round trip time value of current probe from member i to member j .
- $EARTT_{ij}$: It is the exponential average value of RTT_{ij} . Considering the burst and fluctuation of data communications, the value of RTT_{ij} may be atypical as a metric to evaluate the network performance from member i to member j . The exponential average value, $EARTT_{ij}$, of the knowledge achieved directly from the network, RTT_{ij} , is used to estimate the network performance. The following equation is used to calculate $EARTT_{ij}$.

$$EARTT_{ij}(m) = \beta(EARTT_{ij}(m-1)) + (1-\beta)RTT_{ij}(m) \quad (3)$$

where $RTT_{ij}(n)$ is the current RTT measurement from member i to member j , $EARTT_{ij}(n-1)$ is the latest exponential average value of RTT_{ij} and β ($0 < \beta < 1$) is a smoothing factor that determines how much weight is given to the old value $EARTT_{ij}(m-1)$. We would like to give the greater weight to more recent instance RTT. Thus in our experiments, we use $\beta = 0.1$

- $SRTT_i$: It is the sum of $EARTT$ achieved by member i as it sends probe packets to other cluster members. Namely,

$$SRTT_i = \sum_{j=1, j \neq i}^n EARTT_{ij}, i \in [1, n] \quad (4)$$

where n is the number of cluster members.

The core selection is based on the metric $SRTT$. The core is the cluster member that satisfies: $SRTT_{core} = \min\{SRTT_i, i \in [1, n]\}$.

An interval I_{se} is defined to avoid the long time probing in case of network congestion. Once the probe action is activated, cluster members send probe requests to each of other cluster member and wait for the ack. After interval I_{se} , DSCT selects

a core from those cluster members who have achieved SRTTs. I_{se} is achieved based on *Jacobson* algorithm [11] that most TCP implementations use now.

D. Improved DSCT Design (IDSCT)

As P2P applications are characterized with multiple sources, “bottleneck” is easy to appear in the links around cluster cores, especially for “inter-clusters” whose links need to carry many other network services. Hence, considering of “bottleneck”, we expect that the cluster size should not be large. However, the delay property of end host multicast demands relatively fewer layers. Hence, with a fixed number of group members, a trade-off exists between the layer number and the cluster size. Formulation (5) illuminates the relationship among the number of group members n , the number of layers l and the cluster size s .

$$\frac{n}{s^{l-1}} < 3k - 1 \quad (5)$$

In IDSCT, we construct a distributed *Kruskal* tree in each “inter-cluster” which guarantees a lower packet transmission delay by dispersing the packet forwarding to other *local cores* in the “inter-cluster” to avoid the “bottleneck” around the “inter-cluster” core. The basic idea of *Kruskal* algorithm [12] is that the off-tree edge with the minimum weight has the priority to be the on-tree branch, if no loop is generated. It is the algorithm to calculate the shortest distance tree among the nodes.

The distance used to construct the *Kruskal* tree in IDSCT is the delay distance of the logic link connecting two end peers in the overlay topology achieved by the probe scheme. One-way transmission delay from member i to member j is $\frac{EARTT_{ij}}{2}$. IDSCT is designed to improve the BDSCT performance when heavy network load appears in the multiple-source P2P applications.

III. EXPERIMENTAL EVALUATION

A. Experimental Model

We use *ns-2* [13] to run our simulations on a group of SUN SOLARIS workstations. The simulation backbone network adopts the MCI ISP backbone. Fig. 3 shows the topology. End hosts directly or indirectly attach to the routers in the backbone. In our simulation, the link bandwidth in the backbone network and the local network are 1000Mbps and 100Mbps respectively. Constants between 20 and 40 and between 3 and 8 are set as the link costs of *backbone network* and *local network* respectively. The simulation traffic is the 1.5Mbps MPEG-1 video streams. Three criteria are used to evaluate the performances of different end host multicast protocols. 1) Average end-to-end delay (AD): AD of source s is define as the ratio of the sum of end-to-end delay from s to each group member to the number of group members; 2) Average cost stress (ACS): Define ACS as the ratio of the sum of the consuming costs of identical packet copies in each link to the sum of each link’s inherent cost; 3) Average link stress (ALS): ALS is defined as the ratio of the sum of numbers that identical packet copies traverse over the same underlying links to the number of links in the group.

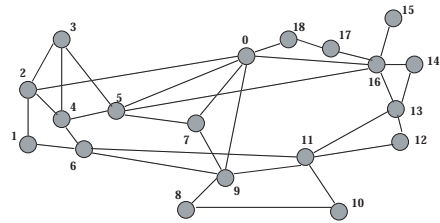


Fig. 3. The experimental backbone network.

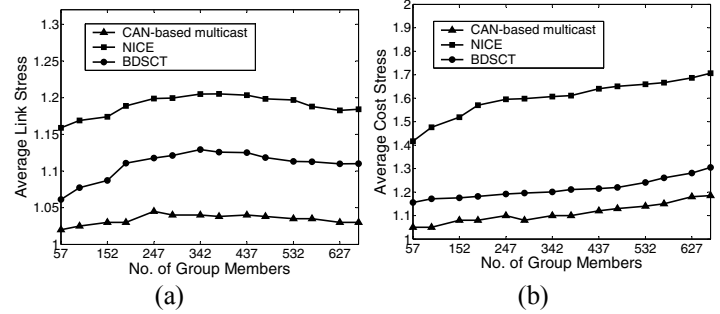


Fig. 4. The average stress performance with the increasing group members. (a) average link stress, (b) average cost stress.

B. Simulation Results and Observations

We have done two groups of simulations to compare the performances of well-known CAN-based multicast, NICE, our BDSCT and IDSCT along the ALS, AD and ACS metrics. In the first group simulation, 5 sources coexist, we observe the performances of proposed multicast schemes with the number of group members increasing from 57 to 665. Fig. 4 (a) shows that BDSCT achieves a lower ALS than NICE. In BDSCT and NICE, the worst link stress appears in the links around cluster cores (or cluster leaders in NICE). BDSCT *cluster formation* disperses the worst link stress to some degree because different cluster sizes exist. For CAN-based multicast, the flooding distributes the link stress much more evenly over all links, hence, the worst link stress of CAN-based multicast is lower than BDSCT and NICE. It introduces lower ALS to CAN-based multicast. However, the flooding scheme uses all the links in the multicast group, which consumes a lot of network resources. The decline of BDSCT and NICE curves after 247 members is because that the increment of the total link stress cannot overtake the increment of group members. Fig. 4 (b) illuminates that BDSCT achieves much lower cost stress than NICE. We have analyzed in Section II that BDSCT avoids the unnecessary packet visit to the costly links. It is the major reason that BDSCT incurs much lower cost stress than NICE. For the same reason of ALS, the ACS of CAN-based multicast is lower than BDSCT and NICE. The average stress performances (i.e. the ALS and the ACS performances) have shown that, compared to NICE, BDSCT has more capacity to carry many more end hosts, namely, BDSCT is more scalable than NICE. The AD curves are shown in Fig. 5. The simulation results show that BDSCT is the most efficient in terms of AD. The AD of BDSCT is always the lowest one in the three multicast protocols. NICE achieves much lower AD than CAN-based multicast. The delay performances of BDSCT

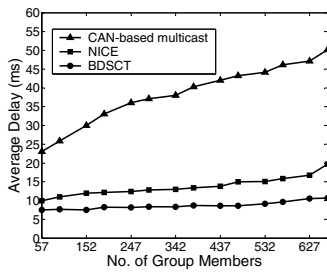


Fig. 5. The average delay performance.

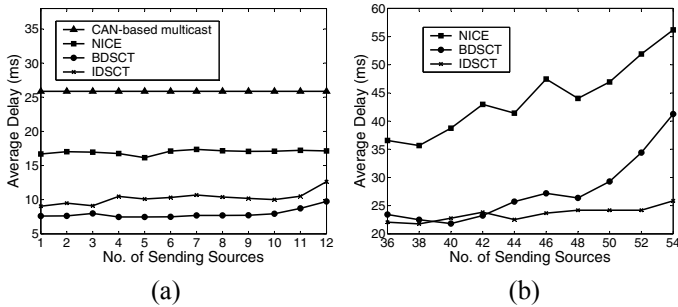


Fig. 6. The average delay performance. (a) is for the low traffic situation, (b) is for the high traffic situation.

and NICE can satisfy the delay demands of real-time streams. The forward flooding scheme brings much larger transmission delay to CAN-based multicast.

In the second group simulation, there are 100 group members, we observe the performances of different multicast protocols as the number of coexisting sources increases. Two simulations are done separately. Fig. 6 plots the performance curves. Fig. 6 (a) is the simulation results when the number of sources changes from 1 to 12. In this situation, the AD performances vary smoothly. Two DSCTs achieve lower AD than NICE and CAN-based multicast. Furthermore, as the traffic load is light, BDSCT achieves better delay performance than IDSCT. Fig. 6 (b) is the performance observations as the number of coexisting sources increases from 36 to 54. We only plot NICE, BDSCT and IDSCT curves for the large CAN-based multicast delay. The performances show that the heavy traffic load causes relative larger delay variations. The AD of NICE is much larger than the AD of two DSCTs. Furthermore, IDSCT achieves better delay performance than BDSCT. This group simulation results illuminate that our DSCT is more efficient than NICE and CAN-based multicast and IDSCT begins to work when the network traffic becomes heavy.

Table I gives the comparison of link properties of the four protocols in this group of simulation.

IV. CONCLUSION

In this paper, we addressed the problem of improving scalability and efficiency for P2P systems. We study a new scalable and efficient end host multicast, DSCT, for P2P systems. In our design, we make full use of the underlying physical properties to implement the *cluster formation*, the cluster core searching and the *local core* selection. Furthermore, IDSCT is designed

TABLE I

THE COMPARISON OF LINK PROPERTIES IN THE FOUR PROPOSED PROTOCOLS.

Protocol	Link stress	Cost stress	No. of actual link used
CAN-based multicast	1	1	261 (average value)
NICE	1.26	1.52	209
BDSCT	1.16	1.28	202
IDSCT	1.15	1.21	197

to improve the bad communication performance caused by “bottleneck” when the network load is heavy. IDSCT uses the *Kruskal* tree to forward packets instead of only cluster cores to avoid the possibility of the appearance of “bottleneck”. Our simulation results basically match our theoretic analysis. The simulation data and curves show that it is possible to improve scalability and efficiency of P2P communications by using our DSCT designs.

V. ACKNOWLEDGMENT

The work is supported by Research grant council (RGC) Hong Kong, SAR China under CityU Strategic Grant No. 7001587.

REFERENCES

- [1] Stephanos Androutsellis-Theotokis, “White Paper: A Survey of Peer-to-Peer File Sharing Technologies,” Athens University of Economics and Business, 2002.
- [2] Y.H.Chu, S.G.Rao and H.Zhang, “A case for end system multicast”, In *Proc. of ACM SIGMETRICS*, June 2000.
- [3] P.Francis, “Yoid: Extending the multicast internet architecture”, White paper <http://www.aciri.org/yoid/>, 1999.
- [4] B.Zhang, S.Jamin and L.Zhang, “Host multicast: A framework for delivering multicast to end users,” In *Proc. of IEEE INFOCOM*, June 2002.
- [5] S.Banerjee, B.Bhattacharjee and C.Kommareddy, “Scalable application layer multicast,” *Proc. of ACM SIGCOMM*, August 2002.
- [6] Sylvia Ratnasamy, Mark Handley, Richard Karp and Scott Shenker, “Application-level multicast using content-addressable networks”, *Proc. of NGC*, 2001.
- [7] E.W.Zegura, M.H.Ammar, Z.Fei, S.Bhattacharjee, “Application-layer Anycasting,” *IEEE/ACM Transactions on Networking(TON)*, August 2000, Volume 8 Issue 4.
- [8] L.Breslau, E.Knightly, S.Schenker, I.Stoica and H.Zhang, “Endpoint Admission Control: Architecture Issues and Performance”, In *Proc. of SIGCOM 2000*, pp.57-69.
- [9] Weijia Jia, Wanqing Tu and Lidong Lin, “Efficient Distributed Admission Control for Anycast Flows”, *Proc. of The 2003 International Conference on Computer Networks and Mobile Computing (ICCNMC-03)*, IEEE Computer Society Press.
- [10] Weijia Jia, Dong Xuan, Wanqing Tu, Lidong Lin and W.Zhao, “Distributed Admission Control for Anycast Flows”, *IEEE Transaction on Parallel and Distributed Systems*, vol.15, no.6, June 2004.
- [11] Andrew S. Tanenbaum, “Computer Networks (The Third Edition)”, Prentice-Hall International, Inc.
- [12] Eric W. Weisstein, “Kruskal’s Tree Theorem”, <http://mathworld.wolfram.com/KruskalTreeTheorem.html>.
- [13] UC Berkeley,LBL,USC/ISI,and Xerox PARC, “ns Notes and Documentation,” October 20, 1999.