A resource-search and routing algorithm within PCDN autonomy area

AUTHOR(S)

Ke Li, Wanlei Zhou, Shui Yu, Y Zhang

PUBLICATION DATE

01-01-2007

HANDLE

10536/DRO/DU:30008177

Li, Ke, Zhou, Wanlei, Yu, Shui and Zhang, Yunsheng 2007, A resource-search and routing algorithm within PCDN autonomy area*, in PDCAT 2007 : Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies : 3-6 December, 2007, Adelaide, Australia*, IEEE Computer Society, Los Alamitos, Calif., pp. 509-514.

A Resource-Search and Routing Algorithm within PCDN Autonomy Area

Ke Li, WanLei Zhou, Shui Yu
School of Engineering and Information Technology
Deakin University
{ktql, wanlei, syu}@deakin.edu.au

Yunsheng Zhang
School of Computer Science and Engineering
University of Electronic Science and Technology of China
zhgyunsheng@gmail.com

Abstract

This paper studied a new type of network model; it is formed by the dynamic autonomy area, the structured source servers and the proxy servers. The new network model satisfies the dynamics within the autonomy area, where each node undertakes different tasks according to their different abilities, to ensure that each node has the load ability fit its own; it does not need to exchange information via the central servers, so it can carry out the efficient data transmission and routing search. According to the highly dynamics of the autonomy area, we established dynamic tree structure-proliferation system routing and resource-search algorithms and simulated these algorithms. Test results show the performance of the proposed network model and the algorithms are very stable.

Key words: algorithm; PCDN; search and routing;

1. Introduction

The rapid development of peer-to-peer (P2P) technologies has led to a number of important application systems on the internet, and the system structure of these P2P application systems has changed gradually from Napster-like (centralized inquiry) to Kazaa-like (bias to the free connecting of strong nodes) structures. However, as the arbitrariness of free connecting in a P2P network, making inquiries must still rely on flooding, resulting in a waste of a lot of network resources, therefore the systematic distensible ability is severely restricted [7].

Content Delivery Network (CDN) is a technology committed to the distribution of content and resource. It is the virtual network formed by node server groups located in different areas, distributing dynamically the internet content to the edge, handling traffic according to the contents, and the visiting request will be transmitted to the optimal server, thus enabling users access to the information from the nearest place by the fastest speed [2]. The existing CDN routing technology is mainly based on DNS; it has the problem of high cost of hardware, and has limitations in terms of scalability, reliability and fault-tolerance, so the redirect server is very likely to become a new network bottleneck.

CDN with P2P technologies are highly complementary, and the research on the combination of the two technologies (PCDN) is still at the preliminary stage. For example, references [5, 6] reported applications in video streaming using CDN combined with P2P. In this paper we propose a new PCDN model, and construct the corresponding routing algorithm and search algorithm based-on the new PCDN model. Compared with existing PCDN models, the new model can balance the dynamics of the lower network and the service efficiency of the upper network. The new model can ensure the cost of maintaining the routing table is always kept within the affordable range by the dynamic network, and can achieve a higher routing algorithm efficiency than that of the Category I and Category II of the structural covered network, therefore its reliability and hardware expenses will be far less than the traditional CDN Network [1] [3].

The proposed PCDN network has the following features:

(1) The dynamic layering within an autonomy area;

(2) The level of node can be adjusted along with the changes of networks;

(3) A node can join the system, and without any regard to the node bandwidth, the node can provide the remaining resources to service the networks in accordance with its own capacity;

(4) The routing uses a tree-type design, and it can avoid producing too many redundant data within the routing; and

0-7695-3049-4/07 $25.00 © 2007 IEEE
DOI 10.1109/.50

(5) The entire network has high scalability.

2. Resource Search and Routing Algorithm within Autonomy Area

Based on the dynamics of the lower network (through testing Napster and Gnutella networks, the average online time of a node is 60 min [4]), using structural Hash algorithms is unwise between the frequently on-line and off-line users. Also each node in and out of the networks will destroy the topological structure of the algorithm, so the cost to calculate frequently the topological structure is staggering. We also need to consider the difference of the network bandwidth of different users -- users with narrow bandwidth are very likely to become data transfer bottlenecks when reconstruction of network topology. Therefore, the routing algorithm of the lower network will be similar to the unstructured mixed P2P model -- to select certain nodes as a super-node (Super Peer); it can manage the sub-node identification and routing. Meanwhile, a security strategy is to select one node from sub-nodes as a slave-node (Slave Peer), and copy the corresponding forms to it. The slave-node will become the super-node while its father node is invalidated, and then will select a new one as the slave-node from its own sub-node. Every node will maintain their own table items and modify according to the dynamic changes of the network structure.

2.1 The Relevant Concepts

Definition：Node *M* Series features are:
k+$NodeID_M$+*ip*+port+*l*, where $NodeID_M$ is dentification, *k* is the node level, *l* is the node layer, and the keyword is $NodeID_M$.
The nodes within the autonomy area need to maintain four storage structures:

① *typedef struct*{
  char Father-NodeID;
  char Brother-NodeID;
} *OrdinaryRroutingTable*;
② *typedef struct*{
  char Resource-fileID;
} *LocalResourceTable*;
③ *typedef struct*{
  char Son- LowerSuperNodeID;
  char Son-NodeID;
} *SuperRoutingTable*
④ *typedef struct*{
  char Resource-fileID;
}*PrecinctResourceTable*;

Every node (whether the ordinary node or the super node) are required to maintain their own Ordinary routing tables and Local resource tables. If a node is also the super node of the under layer, then it is required to extra maintain Super routing tables and Precinct resource tables. This four storage structures are used as the storage method for Binary Sort Tree in addition to the *Father-NodeID* in Ordinary routing Table.
To Definite the data structure of each node within the autonomy area:

typedef struct{
  Element NodeID;
  Element OrdinaryRroutingTable;
  Element LocalResourceTable;
  Element SuperRoutingTable;
  Element PrecinctResourceTable;
  } *Node*;

2.2 Resource Search Algorithm and Routing Mechanism

The application of resource search within autonomy area always searches in their own region at beginning, and then submits the request to brother nodes and eventually to the regional super node, and even to proxy servers, until the source servers and proxy servers run another search algorithm.

Algorithm 1:
Resources Search Algorithm within Autonomy Area
Input: the request of node *M* regarding the *Resource-fileID=key*;
Output: the *NodeID* which has the *Resource-fileID=key*.

Node FileSearchTree(*Node X* ,*char key*)
{*IF*(*SearchBST*(*X.PrecinctResourceTable,key)*!= *NULL*) *Return the NodeID which has the resources*;
ELSE For Every N_i=*X. SuperRoutingTable.Son-LowerSuperNodeID, in Parallel DO*
FileSearchTree(N_i , *key*);
Until(*Every*N_i*.SuperRoutingTable.Son-LowerSuperNodeID==NULL*);
}
Node FileSearchBrother(*Node Y* ,*char key*)
{*ForEvery* R_i=*Y.OrdinaryRroutingTable.Brother-NodeID, in Parallel DO*
{*IF*(*SearchBST*(R_i*.LocalResourceTable,key*)!= *NULL*) *Return* R_i*.NodeID*;
ELSE IF (R_i*.SuperRoutingTable*!=*NULL*) *FileSearchTree*(R_i, *key*);}
Until (*Every* R_i *Not Found Resource-fileID ==key*);
}

Node FileSearchFather(*Node Z* ,*char key*)
{*A=Z.OrdinaryRroutingTable.Father-NodeID*;
IF(*SearchBST*(*A.LocalResourceTable,key*)!=*NULL*) *Return A.NodeID*;
ELSE DO
{*FileSearchBrother*(*A*, *key*);
A=A.OrdinaryRroutingTable.Father-NodeID;
}
Until (*A.OrdinaryRroutingTable.Father-NodeID is Proxy*);
}
Node FileSearch (*Node M*, *char key*)
{*IF* (*SearchBST*(*M.LocalResourceTable,key*)!= *NULL*) *Return M.NodeID*;
ELSE{ *FileSearchTree*(*M*, *key*);
FileSearchBrother(*M*, *key*);
FileSearchFather(*M*, *key*);
}
SendFileRequest (*M*, *key*) *to Proxy*;
}

The *SearchBST*( ) is to call the Binary Sort Tree algorithm; *For …in Parallel DO…Until* is that the implementation of a parallel algorithm until the constraint conditions coming into existence. The following is similar.

Algorithm 2:

The Routing Algorithm within Autonomy Area

Input: the routing search towards node *S* from node *M*.

Output: the *NodeID* of the node *S*.

Element RouteSearchTree(*Node X* , *Node S*)
{*IF*(*SearchBST*(*X.SuperRoutingTable.SonNodeID*, *S.NodeID)*!=*NULL*) *Return S.NodeID*;
ELSE For Every N_i=*X. SuperRoutingTable.Son-LowerSuperNodeID, in Parallel DO*
RouteSearchTree (N_i, *S*);
Until(*Every*N_i.*SuperRoutingTable.SonLowerSuper NodeID*==*NULL*);
}
Element RouteSearchBrother(*Node Y* , *Node S*)
{*ForEvery* R_i=*Y.OrdinaryRroutingTable.Brother-NodeID*, *in Parallel DO*
IF(R_i.*SuperRoutingTable*!=*NULL*)*RouteSearch Tree*(R_i, *S*);
Until (*Every* R_i *Not Found S*);
}
Element RouteSearchFather(*Node Z* , *Node S*)
{*A=Z.OrdinaryRroutingTable.Father-NodeID*;
IF(*SearchBST*(*A.OrdinaryRroutingTable.Brother-NodeID,S.NodeID*)!=*NULL*) *Return S.NodeID*;
ELSE DO
{*RouteSearchBrother*(*A*, *S*);
A=A.OrdinaryRroutingTable.Father-NodeID;
}
Until (*A.OrdinaryRroutingTable.Father-NodeID is Proxy*);
}
Element Dynamic-Ira (*Node M*, *Node S*)
{*IF*(*SearchBST*(*A.OrdinaryRroutingTable,S.Node*)!=*NULL*) *Return S.NodeID*;;
ELSE{ *RouteSearchTree* (*M*, *S*);
RouteSearchBrothe(*M*, *S*);
RouteSearchFather(*M*, *S*);
}
SendRouteRequest (*M*, *S*) *to Proxy*;
}

If the resource search and routing request can be completed within its autonomy area, then we have the following analysis about the time complexity:

Algorithm 1: the time complexity:

$$O(\frac{Num}{\overline{\beta}} \cdot (1 + \log_2^L)) = O(\frac{Num}{\overline{\beta}} \cdot \log_2^L)$$

Algorithm 2: the time complexity:

$$O(\frac{Num}{\overline{\beta}} \cdot (1 + \log_2^{\overline{\beta}})) = O(\frac{Num}{\overline{\beta}} \cdot \log_2^{\overline{\beta}})$$

In which, *Num* is the node number within an autonomy area; $\overline{\beta}$ is the average service rate of each Super node; and L is the average of the resource number provided by each node.

2.3 The Nodes Join In Autonomy Area

The initial level when a new node X joins the network is the lowest ($\max\{k\}$+1) among the brothers with the common father, its actual level will be increased or decreased with the running gradually and to achieve the best. This process is known as "slow start."

The formula to calculate the level k of the node X is:

$$\alpha = k_X + \log_2^{\frac{w_X}{W_X} \cdot \frac{\sum_{i=1}^{n} c_i}{C_X} \cdot \frac{\sum_{j=1}^{\gamma} m_j}{M_X} \cdot \frac{5}{T_{online}}} \quad ; \quad k = \begin{cases} \lceil \alpha \rceil, & \alpha > 0 \\ 0, & \alpha \le 0 \end{cases}.$$

In which, k_X is the initial-level of X; w_X is the statistical bandwidth of the node X occupied in recent running, W_X is the network provides for X-usable bandwidth; C_X is the calculation ability of X, $c_i, 1 \le i \le n$ is the CPU resources occupied by each process and there is n process; M_X is the storage capacity of X, $m_j, 1 \le j \le \gamma$ is the storage capacity occupied by each documents and there are γ documents; T_{online} is the online time until now, the number 5 in the formula is determined according to reference [8].

The node will continually adjust its own level and notify the changes to its father node in the course of operation, the higher the level, the more likely to upgrade to a slave-node till a super-node when its father node failed. Meanwhile, the higher-level nodes have the higher ability, are more easily to accept new nodes, and the larger the sub-tree of these nodes.

Definition: the residual load capacity ($Load_{X\text{-}rest}$) of the node X is that it can also undertake additional load capacity besides the current load.

$$Load_{X-rest} = \left\lceil \log_2 \frac{(W_X - w_X)\cdot 10^2}{W_X} \frac{(C_X - \sum_{i=1}^{n} c_i)\cdot 10^2}{C_X} \frac{(M_X - \sum_{j=1}^{\gamma} m_j)\cdot 10^2}{M_X} \frac{T_{online}}{5} \right\rceil$$

In general, it can be considered a node X has the sufficient residual load capacity when the residual value of W_X,C_X,M_X are more than 20% of the total, $T_{online} \geq 5$ hours. And then, it can be provided to the network as a super-node or slave-node, its residual load capacity is: $Load_{X-rest} = \left\lceil \log_2^{80^3} \right\rceil = 18$ .Therefore the number 18 is usually used as the turning-point to measure the residual load capacity of a node.

Algorithm 3:

The Node Insertion Algorithm within Autonomy Area

Input: the new node X;

Output: X joined an autonomy area.

VOID InsertNode (*Node X*, *Node A*)
{*Send*(*X*, *A*, *JoinMessage*);
IF (*Load*$_{A\text{-}rest} \geq 18$)
{*X.OrdinaryRroutingTable.Father-NodeID=A*;
X.OrdinaryRroutingTable.BrotherNodeID=A.SuperRoutingTable.Son-NodeID;
X.NodeID(*l*)= *A.NodeID*(*l*)+1;
IF(*A.SuperRoutingTable*==*NULL*)*X.NodeID*(*k*) = *A.NodeID*(*k*)+1;
ElseX.NodeID(*k*)=*max*(*A.SuperRoutingTable.Son-NodeID.NodeID*(*k*))+1;
APPEND(*A.SuperRoutingTable.Son-NodeID*,*X*);
APPEND(*A.PrecinctResourceTable*,*X.LocalResourceTable*);
Every A_i=*A.SuperRoutingTable.Son-NodeID, in Parallel DO*{
APPEND(A_i.*OrdinaryRroutingTable.Brother-NodeID*, *X*);}
}
ELSE IF(*A.SuperRoutingTable*!=*NULL*)
{*For Every* A_i=*A.SuperRoutingTable.Son-LowerSuperNodeID, in Parallel DO*
IntersertNode (*X*, A_i);
Until (*Every* A_i *Reject X*);}
}
VOID InsertTree (*Node X*)
{*Broadcast*(*JoinMessage*);
Z=min(*time*(*ReturnMessage*));
While(*clock*<$t_{deadline}$ ||*Z.OrdinaryRroutingTable.Father-NodeID is not Proxy*)
{*A= Z.OrdinaryRroutingTable.Father-NodeID*;
InsertNode (*X*, *A*);
Z= A;
}
Y= Z.OrdinaryRroutingTable.Father-NodeID;
Send(*X*,*Y*, *JoinMessage*);
}

The time complexity of algorithm 3 is $O(Num)$, *Num* is the amount of the nodes within autonomy area.

The system can automatically adjust the level and layer of the nodes to adapt to the network's changes.

2.4 The Number of the nodes within Autonomy Area

Theorem: the number of the nodes to join and leave the new network within a unit time is subordinate to the integral function regarding λ , β , and there are:

$$N_{in-t_0} = \frac{\int_{t_0}^{t_0+\Delta t} \lambda(s)\,ds}{3600}, \quad N_{out-t_0} = \frac{\int_{t_0}^{t_0+\Delta t} \beta(s)\,ds}{3600}.$$

Proof: according to the *Poisson* distribution, the probability of having n nodes joined the network within $\triangle t$ from t_0 is:

$$P\{[N(t_0+\Delta t)-N(t_0)]=n\} = \frac{[m(t_0+\Delta t)-m(t_0)]^n}{n!} e^{-[m(t_0+\Delta t)-m(t_0)]},$$

In which, $m(t) = \int_0^t \lambda(s)\,ds$ .The mean value of the nodes having joined the network within (t_0, t_0+$\triangle t$) is:

$$m(t_0+\Delta t) - m(t_0) = \int_0^{t_0+\Delta t} \lambda_j(s)\,ds - \int_0^{t_0} \lambda_i(s)\,ds \quad ;$$

$\lambda_i(t), \lambda_j(t)$ is decided by the sub-function. Therefore while $\triangle t$ is extremely tiny (assume $\triangle t$ =1s), the number of the new nodes joined the network within $\triangle t$ from t_0 is: $N_{in-t_0} = \frac{\int_{t_0}^{t_0+\Delta t} \lambda(s)\,ds}{3600}$, in which, there are $M_{in-t_0} = \frac{N_{in-t_0}}{\bar{\beta}}$ new nodes becoming the super-nodes. For the same reason, the number of the nodes left the

network within $\triangle t$ from t_0 is: $N_{out-t_0} = \frac{\int_{t_0}^{t_0+\Delta t} \beta(s)ds}{3600}$, in which, there are $M_{out-t_0} = \frac{N_{out-t_0}}{\overline{\beta}}$ super-nodes left the network.

2.5 The Network Costs for Inquiring Message within Autonomy Area

An inquiring message to resources or routing is mainly transmitted among the super-nodes; their costs are approximately the same. Each super-node is maintaining the resource index belongs to its own, and processing the received inquiries. The average number needs to be processed per second is: $Q = \overline{\beta} \cdot q$. The message needs each super-node to transmit per second is: $Z'' = q \cdot h = q \cdot \tilde{M}$, where h is the number of transmitted hops. The transmitted bandwidth is: $W = Z'' \cdot c$, where c is the data packet's size of the inquiries message.

3 Simulation Results

3.1 Simulation Environment:

IBM--CPU: AMD Athlon（tm）64 processor2800＋ 1.81GHz; 512Mb; 120GMb.
Dell--CPU: Pentium(R)4 + 2.66GHz; 512Mb; 80GMb.
Windows XP professional; OPNET Modeler10

The Relevant Parameter Set:

(a) The parameter set of local statistics

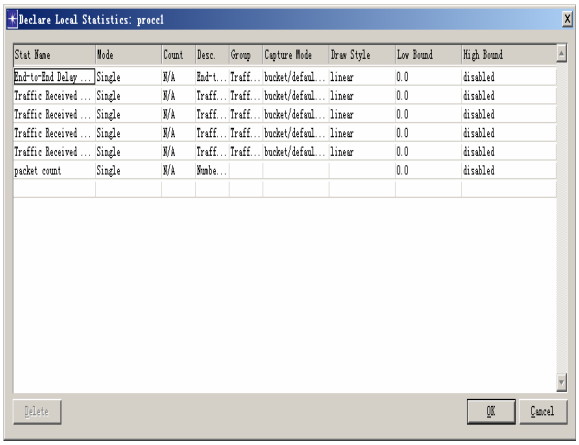

Table 1 Declare Local Statistics

(b) The parameter set of node in requesting resource

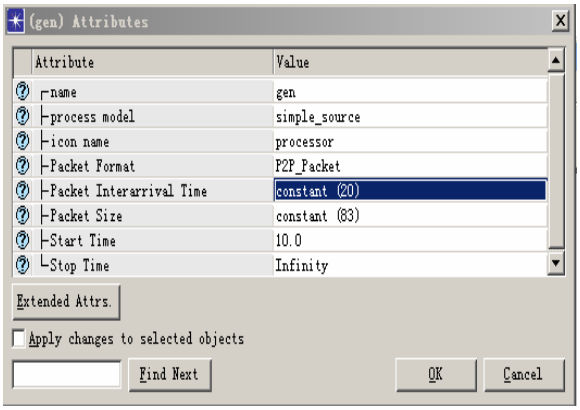

Table 2 Attributes of Node in Requesting Resource in 20/s

The Process Proc Model of Super-Node:

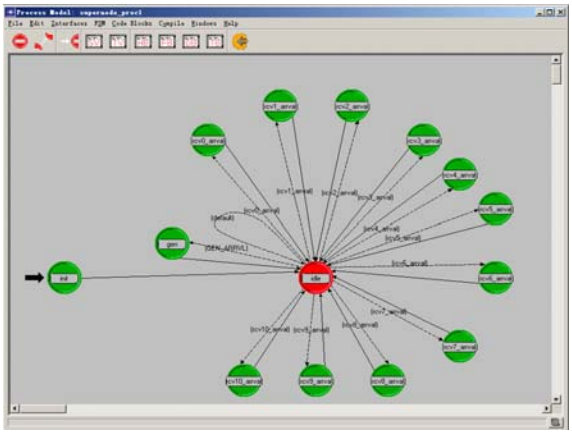

Figure 1 The Process Proc Model of Super-Node

The Topology Graph:

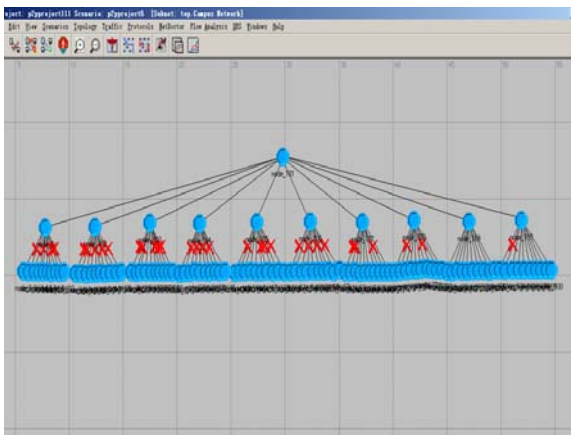

Figure 2 The Topology Graph of Medium Size Nodes (Entire Tree is Part Pruned)

3.2 Simulation Results:

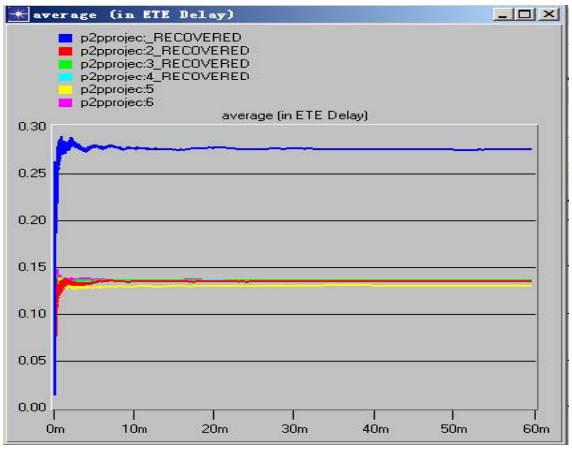

Table 3 Delay of Net in Difference Requesting Resource in Vary Topology

Here we describe the curves in Table 3:
P2Pproject_RECOVERED; P2Pproject2_RECOVERED; P2Pproject3_RECOVERED; P2Pproject4_RECOVERED. These curves represent the network topology; all are N_1-10-trees, that is, they represent the case of the largest number of nodes. P2Pproject5 and P2Pproject6 represent the network topology of the covered partly pruning, which represent the case of the middle and less number of nodes, as listed below:
P2Pproject_RECOVERED Each node sent 20 resource-searches per second;
P2Pproject2_RECOVERED Each node sent 5 resource-searches per second;
P2Pproject3_RECOVERED Each node sent 1 resource-search per second;
P2Pproject4_RECOVERED Each node sent 1-5 resource-searches per second randomly by the function 'Uniform_int';
P2Pproject5 Each node sent 1-5 resource-searches per second randomly by the function 'Uniform_int';
P2Pproject6 Each node sent 1-5 resource-searches per second randomly by the function 'Uniform_int';

From the above data, we can see that there is a 0.27's delay in the N_1 network which has the largest number of nodes and it gives each node the most requests of resource-searches (P2Pproject_RECOVERED, resource-search 20/s) compulsively. However there are the stable delays with an average of 0.14s in other different network topologies and different random resource-searches.

4 Conclusions

The proposed new PCDN model, together with the algorithms described in this paper, has a good adaptability for the dynamic changes of network, whereas the network delay and super-nodes' throughput are not changed significantly along with the change of the number of nodes, and each node can serve the network based on their own abilities. At the same time, this PCDN can also solve the users' dynamic features according to the layered tree-type autonomic area used the PCDN. Apparently, this PCDN can run in any environments, and is not be limited to the size of the system, the strength of the nodes' capabilities and the frequency of the nodes' in and out; it is a wide area distributed system which can ensure the routing efficiency by dynamic regulation. The performance of this network showed that it is very stable.

5. References

[1] Dong, Yingfei ; Kusmierek, Ewa; Duan, Zhenhai; Du, David .A hybrid client-assisted streaming architecture: Modeling and analysis. Proceedings of the Eighth IASTED International Conference on Internet and Multimedia Systems and Applications, Proceedings of the Eighth IASTED International Conference on Internet and Multimedia Systems and Applications, 2004, p 217-222.
[2] Tran, Minh ; Tavanapong, Wallapak .On using a CDN's infrastructure to improve file transfer among peers. Lecture Notes in Computer Science , v 3754 LNCS, Management of Multimedia Networks and Services - 8th International Conference on Management of Multimedia Networks and Services, MMNS 2005, Proceedings, 2005, p 289-301.
[3] Hefeeda, Mohamed M. ; Bhargav, Bharat K.; Yau, David K.Y. .A hybrid architecture for cost-effective on-demand media streaming.Computer Networks, v 44, n 3, Feb 20, 2004, p 353-382.
[4] Saroiu S., Gummadi P.k., Gribble S.D.. A measurement study of Peer-to-Peer file sharing systems[J]. In: Proceedings of the Multimedia Computing and Networking Conferences, San Jose, California, USA, 2002.
[5] Khan, Shoaib ; Schollmeier, Rudiger; Steinbach, Eckehard .A performance comparison of multiple description video streaming in peer-to-peer and content delivery networks.2004 IEEE International Conference on Multimedia and Expo (ICME), v 1, 2004 IEEE International Conference on Multimedia and Expo (ICME), 2004:503-506.
[6] Xu, Dongyan; Kulkarni, Sunil Suresh; Rosenberg, Catherine; Chai, Heung-Keung .Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. Multimedia Systems, 2006,11(4): 383-399.
[7] Z.Xu and Y.Hu, SBARC: A supernode Based Peer-to-Peer File Sharing System .in (ISCC), Kemer- Antalya, Turky, June 2003.
[8] Genutella: To the Bandwidth Barrier and Beyond. http://lambda.cs.yale.edu/cs425/doc/ gnutella.html.2001.