This is the published version:


Available from Deakin Research Online: http://hdl.handle.net/10536/DRO/DU:30008190

Every reasonable effort has been made to ensure that permission has been obtained for items included in DRO. If you believe that your rights have been infringed by this repository, please contact drosupport@deakin.edu.au

Copyright: 2007, CIRAS
Novel Dynamic Routing Algorithms in Autonomous System

Ke Li, WanLei Zhou, Shui Yu, RongHua Tian
School of Engineering and Information Technology
Deakin University, 221 Burwood HWY, VIC 3125, Australia
{ktql, wanlei, syu, rthia}@deakin.edu.au

Abstract
This paper proposed two novel dynamic routing algorithms in autonomous system, and studied a new type of network model. This model is formed by the P2P 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 algorithms and simulated these algorithms. Test results show the performance of the proposed network model and the algorithms are very stable.

Keywords: routing algorithms; autonomous system;

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 a waste of a lot of network resources, therefore the systematic distensible ability is severely restricted [4] [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; which 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 these two technologies, namely the Peer-assisted Content Delivery Network (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 algorithms based-on the new PCDN model. Compared with existing PCDN models, the new model can balance the dynamics of the lower level network and the service efficiency requirement of the upper level 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 rest of this paper is organized as follows: In section 2 we describe the topology of PCDN; in section 3, after a brief analysis of the problems of a routing algorithm in a traditional network, we propose our new routing algorithms in PCDN; we present the implement of the new routing algorithms in section 4; in section 5 we simulate these new algorithms in PCDN and present an analysis of the results; finally, section 6 summarizes the paper.

2 The System Structure
PCDN architectures not only reflect the centralized management of resources of the CDN model, but also highlight the use of resource sharing, equality of data access and other characteristics of the peer-to-peer model.

2.1 The PCDN Topology
The logic structure of the entire PCDN network is formed by three layers, as shown in Figure 1 and is described as follows:

The first layer is the CDN source-server layer; it is located at the top of the entire system structure, and is formed by the source servers who provide content.

The second layer is the Proxy layer, which is formed by proxy servers who receive the share resources from different source servers.
The third layer is the Autonomy area, which is formed by many different size P2P autonomy areas.

Figure 1: Hierarchy of Logic Graph

2.2 The P2P Autonomous System Topology

A P2P autonomous system, as shown in Figure 2, is formed by many user-nodes, and user-nodes are composed to form many small P2P autonomy areas within a certain region.

Figure 2: The P2P Autonomous System Topology

At the initial construction of network, there is only one P2P autonomy area in every autonomous system. With the number of users gradually increasing, those ordinary nodes with fine-line performance will be upgraded to as a super-node, and go down re-building a new level.

3 Routing Algorithm in Autonomous System

Based on the dynamics of the lower layer network (through testing Napster and Gnutella networks, the average online time of a node is 60 min [4]), the use of 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.

3.1 The Relevant Concepts and Routing Algorithm

Definition : Node $M$ Series features are:
$$k+NodeD_{ip}+port+i$$, where $NodeD_{ip}$ is identification, $k$ is the node level, $i$ is the node layer, and the keyword is $NodeD_{ip}$.

The nodes within the autonomy area need to maintain four storage structures:

1. `typedef struct {
   char Father-NodeID;
   char Brother-NodeID;
} OrdinaryRoutingTable;`

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.

Here is the definition of the data structure for each node within the autonomy area:
```c
typedef struct {
   Element NodeID;
   Element OrdinaryRoutingTable;
   Element LocalResourceTable;
   Element SuperRoutingTable;
   Element PrecinctResourceTable;
};
```
With the above definitions, we can now describe the following algorithm:

**The Routing Algorithm within Autonomy Area:**

Input: the routing search towards node \( S \) from node \( M \).

Output: the NodeID of the node \( S \).

```c
Element RouteSearchTree(Node X, Node S)
{ IF(SearchBST(X.SuperRoutingTable.Son-NodeID, S.NodeID) = NULL) Return S.NodeID;
 ELSE For Every \( N_{i} \) = X.SuperRoutingTable.Son-LowerSuperNodeID, in Parallel DO
 RouteSearchTree(Ni, S);
 Until (Every Ni.SuperRoutingTable.Son-LowerSuperNodeID = NULL);
 }
 Element RouteSearchBrother(Node Y, Node S)
{ For Every \( R_{j} \) = Y.OrdinaryRoutingTable.Brother.NodeID, in Parallel DO
 RouteSearchTree(Rj, S);
 Until (Every Rj. Not Found S);
 }
 Element RouteSearchFather(Node Z, Node S)
{ A = Z.OrdinaryRoutingTable.Father.NodeID;
 IF(SearchBST(A.OrdinaryRoutingTable.Brother-NodeID, S.NodeID) = NULL) Return S.NodeID;
 ELSE DO
 RouteSearchBrother(A, S);
 A = A.OrdinaryRoutingTable.Father-NodeID;
 Until (A.OrdinaryRoutingTable.Father-NodeID is Proxy);
 }
 Element Dynamic-Ira(Node M, Node S)
{ IF(SearchBST(A.OrdinaryRoutingTable.SonNodeID) = NULL) Return S.NodeID;
 ELSE RouteSearchTree(M, S);
 RouteSearchBrother(M, S);
 RouteSearchFather(M, S);
 }
 SendRouteRequest(M, S) to Proxy;
}
```

The function `SearchBST()` calls the Binary Sort Tree algorithm; The `For ... in Parallel DO... Until` is the implementation of a parallel algorithm until the constraint conditions come into existence.

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

\[
O\left(\frac{\text{Num}}{\beta} \cdot (1 + \log \frac{1}{\beta})\right) = O\left(\frac{\text{Num}}{\beta} \cdot \log \frac{1}{\beta}\right)
\]

Where \( \text{Num} \) is the node number within an autonomy area; and \( \beta \) is the average service rate of each Super node.

**3.2 The Node Insertion Algorithm for Nodes Joining the Autonomy Area**

The initial level when a new node \( X \) joins the network is the lowest \((\max\{k|k+1\})\) among the brothers with the common father, and 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:

\[
\alpha = k_X + \log_2 \left( \frac{W_x}{C_x} \frac{M_x}{T_{\text{total}}} \right)
\]

Where \( k_X \) is the initial-level of \( X \); \( W_x \) is the statistical bandwidth of the node \( X \) occupied in recent running; \( C_x \) is the network provides for \( X \)-usable bandwidth; \( M_x \) is the calculation ability of \( X \), \( 1 \leq i \leq n \) is the CPU resources occupied by each process and there is \( n \) process; \( M_X \) is the storage capacity of \( X \), \( m_j, 1 \leq j \leq \gamma \) is the storage capacity occupied by each documents and there are \( \gamma \) documents; \( T_{\text{online}} \) is the online time until now.

The constant 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 \( \text{Load}_{X_{-\text{res}}} \) of node \( X \) is that it can also undertake additional load capacity besides the current load.

\[
\text{Load}_{X_{-\text{res}}} = \log_2 \left( \frac{(W_x \cdot \text{Load})}{C_x} \cdot \frac{M_x}{T_{\text{total}}} \right)
\]

In general, it can be considered the node \( X \) has sufficient residual load capacity when the residual value of \( W_x \cdot C_x \cdot M_x \) are more than 20% of the total, \( T_{\text{online}} \geq 5 \) hours. And then, it can be provided to the network as a super-node or slave-node, its residual load capacity is: \( \text{Load}_{X_{-\text{res}}} = \log_2^{18} \). Therefore the number 18 is usually used as the turning-point to measure the residual load capacity of a node.
The Node Insertion Algorithm within Autonomy Area:

Input: the new node $X$;
Output: $X$ joined an autonomy area.

1. Void InsertNode (Node $X$, Node $A$)
   a. Send($X, A$, JoinMessage);
   b. If (Load of $X$ is greater than or equal to 18)
      c. InsertNode ($X$, $A$);
   d. Else If (Load of $X$ is less than 18)
      e. ReduceLoad ($X$, $A$);
      f. Send ($X$, $A$, JoinMessage);
   g. Else
      h. Reject ($X, A$);

2. Z. OrdinaryRoutingTable ($X$)
   a. Send ($X, A$, JoinMessage);
   b. While (ClockTime < Autonomy)
      c. InsertNode ($X$, $A$);
      d. If (Node $X$ is not in the other's Autonomy)
         e. If (Node $X$ is not in the other's Autonomy)
            f. InsertNode ($X$, $A$);
   g. Else
      h. Reject ($X, A$);

3. Z. OrdinaryRoutingTable ($X$)
   a. Send ($X, A$, JoinMessage);

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

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

In which, $m(t) = \int p(t_0) dt$. The mean value of the nodes having joined the network within $(t_0$, $t_0 + \Delta t)$ is:

$$m(t_0 + \Delta t) - m(t_0) = \int \lambda(t) dt - \int \lambda(t) dt$$

$\lambda(t)$ is decided by the sub-function. Therefore while $\Delta t$ is extremely tiny (assume $\Delta t = 1$), the number of the new nodes joined the network within $\Delta t$ from $t_0$ is:

$$N_{in} = \frac{3600}{\lambda}$$

In which, $M_{in} = \frac{N_{in}}{\beta}$ new nodes becoming the super-nodes.

For the same reason, the number of the nodes left the network within $\Delta t$ from $t_0$ is:

$$N_{out} = \frac{3600}{\beta}$$

In which, there are $M_{out} = \frac{N_{out}}{\beta}$ super-nodes left the network.

3.4 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 maintains the resource index belonging to its own, and processing the received inquiries. The average number needs to be processed per second is:

$$Q = \frac{\beta}{q}$$

The message needed for each super-node to transmit per second is:

$$Z^* = q \cdot h = q \cdot M$$

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.

4 Simulation Results

4.1 Simulation Environment:

IBM-PC: AMD Athlon (tm) 64 processor 2800+ 1.81GHz; 512Mb; 120GB.
4.2 Simulation Results:

(a) This simulation tests the network performance of CDN model with various network loads and in different network nodes.

In figure 3: Application 1 represents the network of 100 light load nodes; Application 2 represents the network of 100 heavy load nodes; Application 3 represents the network of 200 heavy load nodes.

![Figure 3: Compare with the Load of Servers](image)

From the experimental results, we can see that there is a very serious dependence on the sever's performance in CDN network, with the network's load increases, the load of servers increased sharply, and it can result in servers' capacity being declined and the entire network's delays increased greatly, until the network is congested. The servers usually become the whole network bottlenecks in multi-user instantaneous peak.

(b) This simulation tests the network performance of traditional P2P network model in different network sizes (to save space we only present the network performance of heavy load nodes).

![Figure 4: Traffic of Point to Point in P2P Network in Heavy Load](image)

From the experimental results, we can see that there is a greater reliance on the nodes' performance in the traditional unstructured P2P network, with the increase of the number of nodes and the network load, it can result in the rapid increase of traffic for the entire network, or even lead to network congestion.

(c) This simulation tests the network performance of our new PCDN network model with various network loads and in different network nodes.

- Topology $N_1$: the topology of least size nodes (entire tree is pruned)

![Figure 5: Average Throughput of Super-Node in Requesting Resource in 1-5/s for Every Node in $N_1$](image)

From above figure 5, we can see the request volume sent by Super-Node has an increased process at the beginning, and then stablized on 11/s around.

- Topology $N_2$: the topology of medium size nodes (entire tree is part pruned)

![Figure 6: Average Throughput of Super-Node in Requesting Resource in 1-5/s for Every Node in $N_2$](image)

From above figure 6, we can see the request volume sent by Super-Node has an increased process at the beginning, and then stablized on 15.8/s around.

- Topology $N_3$: the topology of large size nodes (entire tree)
From above figure 7, we can see the request volume sent by Super-Node has an increased process at the beginning, and then stabilized on 32/s around.

From the figure 8, we can see the request volume sent by Super-Node has an increased process at the beginning, and then stabilized on 32/s around.

The figure 9 shows that the request volume sent by Super-Node has an increased process at the beginning, and then stabilized on 19/s around.

5 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 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 model can also solve the users' dynamic features according to the layered tree-type autonomic area used by the PCDN. Apparently, this PCDN model can run in any environment, and is not limited to the size of the system, the strength of the nodes' capabilities and the frequency of the nodes' in and out. This PCDN model can form a wide area distributed system which can ensure routing efficiency by dynamic regulation. The performance analysis of this model showed that it is very stable.

6 References


[8] Gnutella: To the Bandwidth Barrier and Beyond.