A Six-Directional Static Searching Mechanism in a Distributed Virtual World

Jui-Fa Chen[1]*, Wei-Chuan Lin[2], Chih-Yu Jian[1] and Heng-Yi Chiou[1]

[1]Department of Information Engineering, TamKang University, Tamsui, Taiwan 251, R.O.C.
[2]Department of Information Technology, Tak-Ming University of Science and Technology, Neihu, Taiwan 114, R.O.C.

Abstract

Many connected distributed servers, each of them dominates a specific region, can construct a virtual world. In this interactive virtual world, users can connect to one of the servers at any place to access the needed information. This paper proposes two external static search mechanisms to obtain the needed information without passing duplicated commands to all the servers in the virtual world. The search command is passed by agents of the servers, which locations are already set, to decide whether the command is already received or not. In this way, the proposed search mechanisms can achieve three goals. First, the searching results fit the user's requirement. Second, the needed results provided by servers are bounded by specific conditions. Third, through the agents of servers to avoid passing duplicated command, the network bandwidth is reduced and also promoted the search performance.

Key Words: Interactive Virtual World, External Static Search, Search Command, Agent, Network Bandwidth

1. Introduction

With the prevailing development of internet and wireless network technology, the concept of a virtual world could be constructed by many location servers. Every location server in the virtual world controls a restricted range and data. When a user wants to obtain the needed information in this virtual world, there's a problem about searching a large-scale data in this world. The problem is that the needed information should be provided by the nearest location server. For example, if a user in the virtual world has made a query "How to find a dentist near my current location?", what should the location servers do? The server which the user located should pass this query command to its neighbor servers and they pass it again and again to the other neighbor servers. How to avoid sending the duplicated query command to the location servers is the major discussion of this paper. Before the searching steps, the topology of a virtual world should be defined first. The user's query command is passed by the agent of the location server to other agents of its neighbor servers. Through the communication among agents, the query command could be decided whether it was passed down to other agents or not. The level of location server should be defined to speed up passing the query command. Finally, the stop searching criterion can prevent a long waiting time for the user, and also promote the search quality of service. This paper proposes two external search mechanisms to let the user find out the needed information provided by the nearest location server.

In this paper, section 2 describes the related work of the search mechanisms. Section 3 is the proposed two search mechanisms in a virtual world. Section 4 is the simulation result which compares the proposed two search mechanisms. Section 5 is the conclusion and future research.

*Corresponding author. E-mail: alpha@mail.tku.edu.tw

2. Related Work

Generally speaking, data gathering, indexing and classification are the main functions of searching engines. The traditional search engines are centralized search architectures [1]. They are using one or more servers to act as a single search engine under centralized management. However, this approach has many problems as listed followed:

1. Cost: Due to central architecture, it is expensive for managing by a single organization.
2. Indexing: It is not guaranteed that the search information which is relevant to the interests of the user is indexed [2].
3. Non-Geographic property: The searching results which the server provides may not be near to the user.

The solutions to these problems lie in the distributed search architecture. In the distributed search architecture, each server makes indexes only a part of information which is available on the web and processes only a subset of user's queries. There is a distributed search system called "DESIRE" [3], which the database is distributed by geographic or network domain. However, most of the user queries for each server are equal likely to index relevant resources. This implies that most of the queries must be routed to all servers in the system and this leads to degrade the system performance.

The ADSA (Adaptive Distributed Search and Advertising) [4–6] project aims to develop a scalable Internet search system based on the concept of topic specific distributed search architecture. Query command only propagates to a small number of servers which contain the relevant topic. However, if the searching target is movable, such as searching someone with specific characteristics, ADSA cannot provide an effective search.

Another distributed search architecture is hierarchical location severs [7–9]. This architecture is based on the cellular network, which has better performance to search mobile units. However, this architecture cannot promise that all the search results provided by the servers are near to the user. If a user wants to search all the results which are near to him, it may cause too many searching results that are impracticable.

3. Virtual World

An interactive virtual world can be built by a large distributed architecture which is constructed by several location servers as shown in Figure 1. The cell region governed by a location server is a circle and each server should connect to at least one of other servers. Every server has its own location database which records the user's information and the dominated region data. User can login into one of the location servers in the virtual world at any place and time to find out the needed information by the agent of the server.

3.1 Topology

The proposed architecture [10–12] for the distributed servers to set up the virtual world is defined as Figure 2. Each server has at most 6 neighbor servers and a unique number was assigned to each of them. The user's position in the virtual world is identified by the server number. Assuming that the server 1 is the position where the user located, the query command is passed by server 1 to other neighbor servers gradually accord-

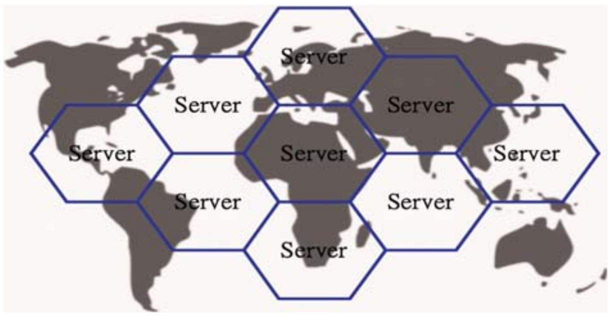

Figure 1. Virtual world architecture.

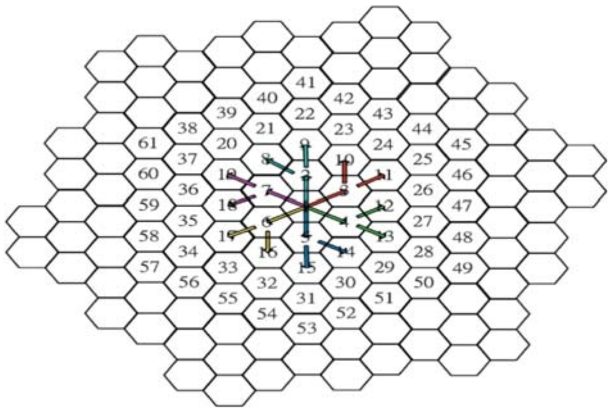

Figure 2. The topology of location servers.

ing the boldface line in Figure 2. In this way, there is no duplicated command passing down to the neighbor location servers.

3.2 Search Mechanism

The proposed search mechanisms are separated into four parts such as level definition, external search, level calculation, and the strategy of stop searching.

3.2.1 Level Definition

Figure 3 is a hierarchical tree which defines the level of each server. The server level means that the distance from the current position to the original query server. If the level number is large, it means that the distance from here to the original query server is long. The search level should be limited to speed up the searching result. For example, a virtual world is separated into several cities. Each city is governed by a location server and the level of the server is relative to the server which the user located. The user can find out the nearest airport or dentist by limiting the search level in the virtual world. The function *f(L)* for calculating the node numbers of each level is defined as followed:

$f(L) = \begin{cases} 1, L = 1 \\ 6 * (L-1), L > 1 \end{cases}$, where L is the level number in the hierarchical tree.

3.2.2 External Search

Two external search mechanisms are proposed to pass the query command from the agent of server which the user located to the agents of neighbor servers. These proposed mechanisms such as 5_way_Passing and Assigned_path_Passing mechanisms are explained as followed:

3.2.2.1 5_way_Passing mechanism

When a server receives a user's query command, it checks the search level in the content of the command before executing the query command. If the server level is not matched the stop criterion, the query command is executed. After executing the query command and the search results match the user's requirement, this server returns the results to the user and stop passing the query command to its neighbor location servers. If the search results are not matched the user's requirement, the query command will be passed to the server's first neighbor server. This server's neighbor list is sorted according to the delivery time from current server to its neighbor servers and the search level limitation. Because the servers in the list would duplicate with other server's sorted list, a server may receive duplicate query commands from different location servers. Except the original query server, each server need to check the query command whether it had been executed or not. If the query command had not been executed, the server executes the query command or stop searching because the criterion is matched. Query command will be passed to other server until the search results match the user's requirement or it is stop because the criterion is met. When some of the servers complete the search, the results are returned to original query server. The user can decide that which information provided by the server is the nearest one.

3.2.2.2 Assigned_path_Passing mechanism

Assigned_path_Passing mechanism is to pass the

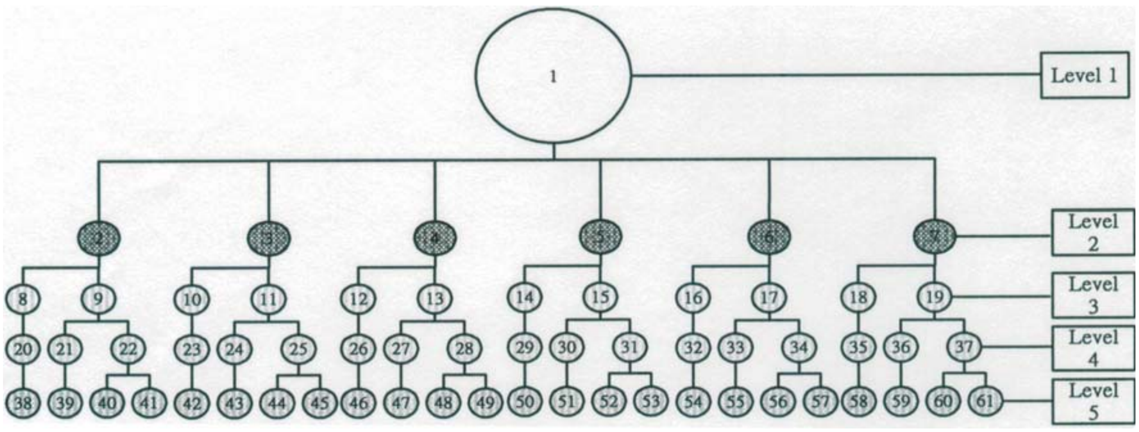

Figure 3. Hierarchical tree.

query command according to the server number in hierarchical tree as shown in Figure 3. All the passing paths are assigned and the query command is not duplicated. The server in level 1 passes the query command to all of the servers in level 2. After processing by the servers in level 2, the query command is passed to level 3 and so on. When passing query command, the sender will tell the receiver which direction should be passed at the next time. For example, as shown in Figure 3, server 1 passes query command to server 2, 3, 4, 5, 6, 7 and tells server 2 the direction to pass the query command is server 8 and server 9 at the next time. When the servers complete the search, the results are returned to the original query server.

Because Assigned_path_Passing mechanism passes the query command by a fixed path, if the upper level server was failed, the successive servers will not know where the searching commands from. To solve this problem, the hierarchical tree of Assigned_path_Passing mechanism is modified as shown in Figure 4. The red dotted line is the substitution path between servers of different level. The green dotted line is the substitution path between servers of same level. For instance, in the hierarchical tree of Figure 3, if server 3 was failed, the server 10, 11, 23, 24, 25, 42, 43, 44, 45 and etc. would not receive search commands. However, in the new hierarchical tree of Figure 4, if server 3 was failed, server 1 would ask server 2 send search commands to server 10, and server 10 will send search commands to server 11, thus the failure of server 3 will not affect the whole process.

3.2.3 Direction Definition

Passing direction is denotes by 3 bits as shown in Figure 5. Inverse direction is the bits inversion. Adjacent direction is the data different in one bit. The distance can be calculated by direction bits.

3.2.4 Level Calculation

(1)5_way_Passing

Query command packet only needs to record at most two different passed directions and the numbers of servers. The next passing direction would be compared with the recorded direction. If the direction is the same, the total number of this direction is increased by 1. Finally, the sum of recorded direction number is the level limitation of searching results.

(2)Assigned_path_Passing

When the query command is passed through one server, the level of searching results is increased by 1. Receiver's level is informed by the sender server.

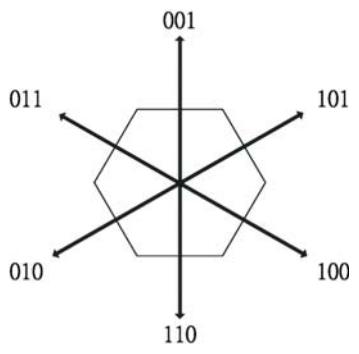

Figure 5. Passing direction.

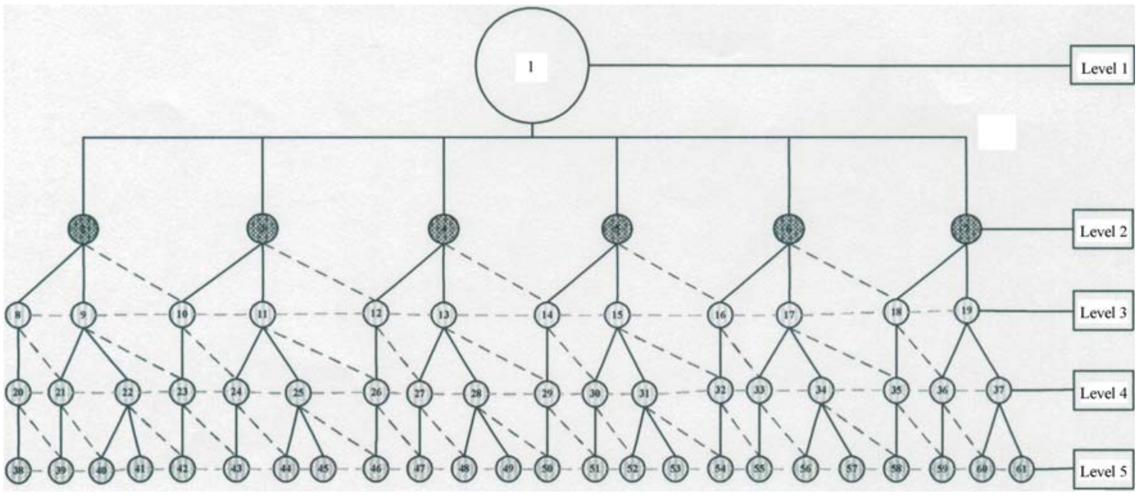

Figure 4. New hierarchical tree.

3.2.5 The Strategy of Stop Searching

The searching requirement should be defined clearly before search. It may cause too many searching results if the description is obscure. The better way for a user to find out the search result is provided by the nearest server. There are two kinds of strategies of stop searching as listed followed.

(1)Limit the amount of results

When the amount of results is matched the limited number, the searching is stopped.

(2)Limit the search level

When the original server passes the query command to other servers, the level of neighbor servers is limited.

The first stop strategy ensures the quality of the searching results. The second stop strategy is focus on the performance. However, the search results provided by the server may be far away from the user's location.

3.3 Algorithm

The algorithm of 5_way_Passing is shown as below:

```
Procedure 5_way_Passing ()
Comments:
**query_command**: a query command includes the description information of what the user wants
**neighbor_server_level**: the level of neighbor server's
**limit_level**: the limited searching level to stop processing the query command
**Internal_Search ()**: search in this server
**check_server_receive**: check the query command whether it had been received or not
**received**: the query_command had been received by the server
BEGIN
    While (neighbor_server_level <= limit_level)
    BEGIN
        If (check_server_receive != received) then
        BEGIN
            Internal_Search()
            If the searching result is not fit the user's requirement then
                Check the neighbor server's level and sort the neighbor servers according to the delivery time from current server to its neighbor servers
                Pass the **query_command** to the first neighbor server in the previous sorted list.
            Else
                Stop searching and break out of the while loop
            End if
        END
    END
END
```

The algorithm of Assigned_path_Passing is shown as below:

```
Procedure Assigned_path_Passing ()
Comments:
**query_command**: a command include the description information of what the user wants
**current_level**: current server's level
**limit_level**: the limited searching level to stop processing the query command
**Internal_Search ()**: search in the server
BEGIN
    While (**current_level< limit_level**)
    BEGIN
        **Internal_Search()**
        If the searching result is not fit the user's requirement then
            Pass the **query_command** to other servers by an assigned direction and telling the received server which direction should be passed at the next time
        Else
            Stop searching and break out of the while loop
        End if
    END
END
```

3.4 Compare the Searching Time

Two stop strategies as described above are compared by the searching time when search level is limited and the amount of searching results are limited. Assume the search level is limited to the n-th level and the total node number of server is m, the relation between level and number of servers is shown as below:

$$m = 3n(n-1) + 1 \qquad (3.1)$$

The complete searching time is calculated by the following time parameters:

CKCT	The time to check query command whether it had been received or not.
ET	The time to execute query command
CKPT	Check the priority of the path to deliver the query command
CKLT	Check the current level of the server
DT	The deliver time to pass the query command (The load of the passing path)
DDT	The delay of deliver time to pass the query command (the servers are in a sorted queue while passing the query command)
RT	The return time of the search results

There are two restrictions while comparing the two proposed search mechanisms which are listed followed:

(1) Assume that the execution time (ET) in different server is the same

(2) Discard the quality of search result in a single server

The complete time means that the time from the query command is passed to all of the servers in limited searching level, completes searching, and finally returns the results to the query server.

3.4.1 The Searching Time Limits by Level

The comparison is focus on ensuring the performance of searching results. Assume that the search level is limited to n-th level and he total node number of server is m, the complete time is computed in worst case. The computation of complete time shows as below:

5_way_Passing:

P_1^k: The shortest path of server 1 to sever k (sum of DT)

Complete search time (CT) is computed by the following function:

$$CT=\begin{cases} CKLT+ET+RT, m=1 \\ (m-1)CKPT+5(m-1)DDT+4CKCT+(m-2)CKCT \\ +\max\left(\sum_{k=2}^{m}P_1^k\right)+mCKLT+ET+RT, m\geq 2 \end{cases} \tag{3.2}$$

Equation 3.2 shows that if the virtual world is constructed by one server, the CT is CKLT+ET+RT. If the virtual world is constructed by two or more servers, except the last server, each server has 1 CKPT and 5 DDT. The query command should be passed through m servers in worst case. Every server needs to check the level limitation. The last server has 4 CKCT, the first server needs not have 1 CKCT and other server needs 1 CKCT. The shortest distance should be found out from the original server to other servers. To obtain the maximum of these distances, all the DTs would be summed up. Finally, the complete time is obtained by adding the ET and RT.

Assigned_path_Passing:

$\sum_{k=2}^{b}L_{k-1}^{k}$: The path of level 1 to level b pass through the servers of level 2, level 3…, level b-1

Complete search time (CT) is computed by the following function:

$$CT=\begin{cases} CKLT+ET+RT, n=1 \\ nCKLT+\max\left(\sum_{k=2}^{n}L_{k-1}^{k}\right)+5DDT \\ +(n-2)DDT+ET+RT, n\geq 2 \end{cases} \tag{3.3}$$

Equation 3.3 shows that if the virtual world is constructed by one server, the CT is CKLT+ET+RT. If the virtual world is constructed by two or more servers, the query command should be passed to n servers in worst case. Every server needs to check the level limitation. To obtain the shortest distance from level 1 to level n, all the DTs would be summed up. The level 1 to level 2 has 5 DDTs and there is 1 DDT in each level except the last level. Finally, the complete time is obtained by adding the ET and RT.

No controlled

P_1^k: The shortest path of server 1 to sever k (sum of DT)

Complete search time (CT) is computed by the following function:

$$CT=\begin{cases} CKLT+ET+RT, m=1 \\ (m)CKLT+\max\left(\sum_{k=2}^{m}P_1^k\right)+6(m-1)DDT \\ +ET+RT, m\geq 2 \end{cases} \tag{3.4}$$

No controlled means that the system will not control to pass duplicated commands, but only control the stop timing. Thus each server will pass search commands to its nearby six neighbor servers. In this case, the duplication of search command will affect the efficiency and it is shown as equation 3.4.

Limited controlled (No backward pass and pass by the order of direction)

P_1^k: The shortest path of server 1 to sever k (sum of DT)

Complete search time (CT) is computed by the following function:

$$CT = \begin{cases} CKLT + ET + RT, m = 1 \\ (m)CKLT + \max\left(\sum_{k=2}^{m} P_1^k\right) + 5(m-1)DDT \\ +ET + RT, m \geq 2 \end{cases} \quad (3.5)$$

In this case, each server passes search commands to its nearby servers except the source server. However, the root is server 1 still passes search commands to its nearby six neighbor servers. The search command passing order should be clockwise or anti-clockwise. The result can be shown as equation 3.5.

Limited controlled (No backward pass and pass by speed)

P_1^k: The shortest path of server 1 to sever k (sum of DT)

Complete search time (CT) is computed by the following function:

$$CT = \begin{cases} CKLT + ET + RT, m = 1 \\ (m-1)CKPT + (m)CKLT + \max\left(\sum_{k=2}^{m} P_1^k\right) \\ +5(m-1)DDT + ET + RT, m \geq 2 \end{cases} \quad (3.6)$$

In this case, each server passes search commands to its nearby servers except the source server, but the root is server 1 still passes search commands to its nearby six neighbor servers. The search command passing order is decided by the deliver speed among servers. The result can be shown as equation 3.6.

3.4.2 The Searching Time Limit by Amount of Searching Results

Assume that the amount of search results is α, the searching results in a single server is β, the total search number of servers would be γ which is equal to α/β. The best and worst case search time are computed and explained as followed:

(1)Best Case

As the passing tree spreads quickly and the DT is constant, it is like to search to the δth level. The δth level can be calculated by the equation 3.7.

$$\gamma \leq 3\delta(\delta - 1) + 1, \gamma, \delta \geq 1, \delta \text{ is the minimum integer} \quad (3.7)$$

5_way_Passing:

Complete search time (CT) is computed by the equation 3.8:

$$CT = \begin{cases} CKLT + ET + RT, \delta = 1 \\ (\delta-1)CKPT + 5(\delta-1)DDT + 4CKCT + (\delta-2)CKCT \\ +(\delta-1)DT + \delta CKLT + ET + RT, \delta \geq 2 \end{cases} \quad (3.8)$$

Assigned_path_Passing:

Complete search time (CT) is computed by the equation 3.9:

$$CT = \begin{cases} CKLT + ET + RT, \delta = 1 \\ \delta CKLT + (\delta-1)DT + 5DDT \\ +(\delta-2)DDT + ET + RT, \delta \geq 2 \end{cases} \quad (3.9)$$

(2)Worst Case

The query command is passed through the total search number of servers (γ level).

5_way_Passing:

P_1^γ: The shortest path of server 1 to sever γ (sum of DT)

Complete search time (CT) is computed by the equation 3.10:

$$CT = \begin{cases} CKLT + ET + RT, \gamma = 1 \\ (\gamma-1)CKPT + P_1^\gamma + 4CKCT + (\gamma-2)CKCT \\ +(\delta-1)DT + \gamma CKLT + ET + RT, \gamma \geq 2 \end{cases} \quad (3.10)$$

Assigned_path_Passing:

$\sum_{k=2}^{b} L_{k-1}^k$: The path of level 1 to level b pass through the servers of level 2, level 3…, level b-1

Complete search time (CT) is computed by the equation 3.11:

$$CT = \begin{cases} CKLT + ET + RT, \gamma = 1 \\ \gamma CKLT + \min \sum_{k=2}^{b} L_{k-1}^{k} + ET + RT, \gamma \geq 2 \end{cases} \quad (3.11)$$

4. Simulation Result

By the equations described in section 3, the complete search time can be compared by setting the parameters. The RT is ignored and the stop strategy is using the limitation of the search level. The DT value is set in different range value to compute the complete search time. The time parameters are set as Table 1:

(1)Set DT value as a constant range

In Figures 6, 7, 8, the DT value is set to 1, 50 and 1000, Assigned_path_Passing shows better performance. However, when the DT value is 1000, the difference between Assigned_path_Passing and 5_way_Passing is not obvious.

(2)Set DT value as a range

When the time parameters are set as Table 2, Figure 9 shows that 5_way_Passing has better performance than Assigned_path_Passing when DT ranges between 1 and 50. The range broadens 1 to 1000 as shown in Figure 10, 5_way_Passing is also better than Assigned_path_Passing.

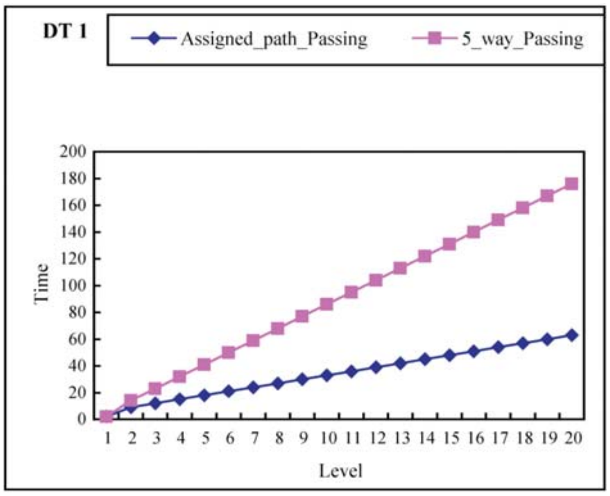

Figure 6. DT const to 1.

The results show that 5_way_Passing suitable to the case when transmission speed (DT) varies largely among servers. While the Assigned_path_Passing suitable to the case when transmission speed (DT) varies little among servers, and with little distance among the servers.

5. Conclusions and Future Research

In this paper, a virtual world is built by several distributed servers. As the virtual world is a distributed architecture, the load is shared by these connected servers. Two external static search mechanisms are proposed to reduce the bandwidth of network and speed up the ob-

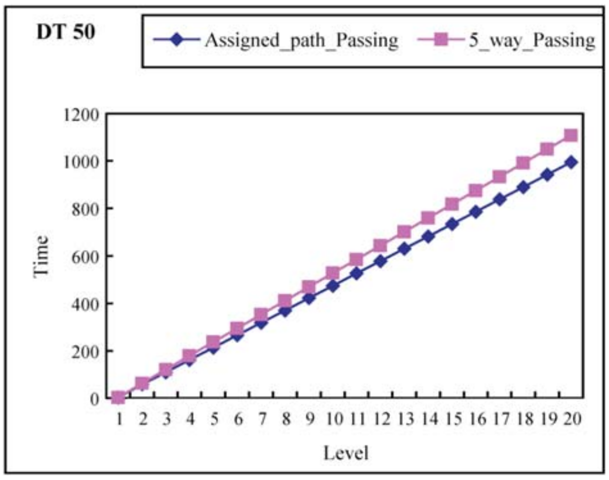

Figure 7. DT const to 50.

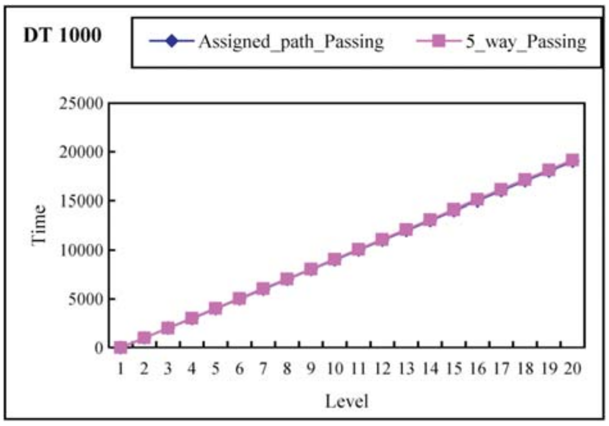

Figure 8. DT const to 1000.

Table 1. Time parameters in simulations

Time parameter	CKPT	CKLT	CKCT	DDT	ET	DT	RT
Value	1	1	1	1	1	variable	Don't care

Table 2. Time parameters in simulation

Time parameter	CKPT	CKLT	CKCT	DDT	ET	DT	RT
Value A	1	1	1	1	1	1~10	Don't care
Value B	1	1	1	1	1	1~50	Don't care
Value C	1	1	1	1	1	1~1000	Don't care

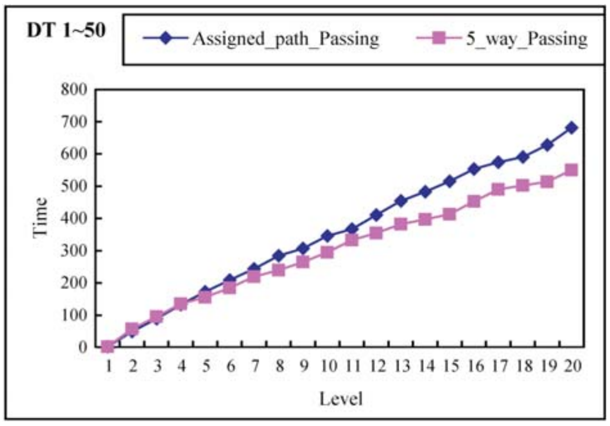

Figure 9. DT ranges between 1 and 50.

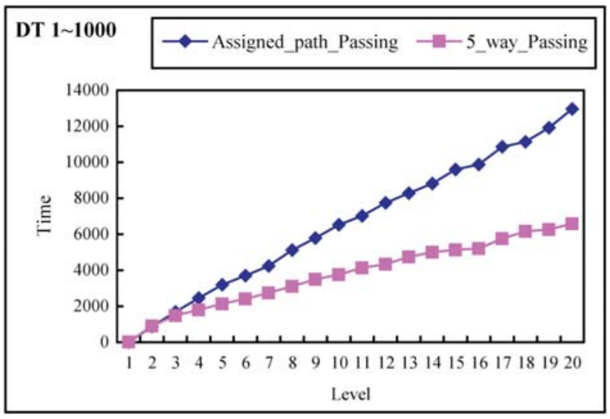

Figure 10. DT ranges between 1 and 1000.

taining of needed information. From the experiments as shown above, 5_way_Passing searching mechanism suitable to the case when transmission speed varies largely among servers. While the Assigned_path_Passing searching mechanism suitable to the case when transmission speed varies little among servers.

Since the regions of the proposed virtual world, governed by the distributed servers, are defined as a circle, there should be a critical region dominated by different servers. It needs to make sure that the data in such a critical region are consistent and the duplicated search command should be avoided. Furthermore, this proposed architecture can be applied to wireless network. This can let user login to this virtual world in anywhere he goes and quickly finds out the needed information.

References

[1] Brin, S. and Page, L., "The Anatomy of a Large-Scale Hyper-Textual Web Search Engine," *Proceedings of the Seventh International World Wide Web Conference,* Brisbane, Australia (1998).

[2] Lawrence, S. and Giles, C. L., "Accessibility of Information on the Web," *Nature,* Vol. 400, pp. 107–109 (1999).

[3] Anders Ardo and Sigfrid Lunberg, "A Regional Distributed WWW Search and Indexing Service — the DESIRE Way," *Computer Networks and ISDN System,* Vol. 30, pp. 173–183 (1998).

[4] Rinat Khoussainov, Tadhg O'Meara and Ahmed Patel, "Independent Proprietorship and Competition in Distributed Web Search Architectures," *Proceedings of the Seventh IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2001), (University of Skövde, Skövde, Sweden)*, pp. 191–199, IEEE Computer Society Press, Los Alamitos, California, USA (2001).

[5] Rinat Khoussainov, Tadhg O'Meara and Ahmed Patel, "Advance Distributed Search for the Web," *Web Computing Introduction,* (E. Leiss, N. Callaos and J. Aguilar, eds.), IIS Society (2002) (in press).

[6] Nikita Schmidt and Ahmed Patel, "Distributed Search for Structured Documents," *Proceedings of AusWeb02 — The Eighth Australian World Wide Web Conference,* pp. 256–273, A. Treloar and A. Ellis (eds.), Southern Cross University, ISBN 1-863844-53-8 (2002).

[7] DasBit, S. and Mitra, S., "Query Processing in a Cellular Network-A Database Approach," *Vehicular Technology Conference,* pp, 2560–2564 (2001).

[8] Dolev, S., Pradhan, D. K. and Welch, J. L., "Modified Tree Structure for Location Management in Mobile Environments," *14th Annual Joint Conference of*

IEEE Computer and Communication Societies (INFOCOM), pp. 530–537 (1995).

[9] Imielinski, T. and Badrinath, B. R., "Querying in Highly Distributed Mobile Environments," *Proceedings of the 18th VLDB,* pp 41–52 (1992).

[10] Chen, J.-F., Lin, W.-C., Jian, C.-Y., Chiou, H.-Y. and Chen, J.-W., "Two Enhanced Searching Mechanisms on a Distributed Virtual World," *Journal of Takming College,* Vol. 22, pp. 139–151 (2003).

[11] Chen, J.-F., Lin, W.-C., Jian, C.-Y. and Chiou, H.-Y., "A Searching Mechanism on a Distributed Virtual World: Six-Direction Simultaneous Search," *10^{th} IEEE International Symposium Pacific Rim Dependable Computing (PRDC 2004),* pp. 13–14, (2004).

[12] Chen, J.-F., Lin, W.-C., Jian, C.-Y. and Chiou, H.-Y., "Creating Non-Duplicated Simultaneous Web Search by Six-Direction Search Mechanisms," *2005 International Conference on Computer Networks and Mobile Computing (ICCNMC'05),* pp. 1–5 (2005).

Manuscript Received: May. 24, 2006
Accepted: Jun. 8, 2007