International Journal of Computer Science & Applications
Vol. III, No. I, pp. 92 - 103

Intelligent Naming System: An Alternative for Enterprise Naming Management

Ladda Preechaveerakul and Pattarasinee Bhattarakosol
Department of Mathematics, Faculty of Science,
Chulalongkorn University, Bangkok, 10330, Thailand
E-mail: {ladda.pr@student.chula.ac.th, bpattara@sc.chula.ac.th}

Abstract

People use "name" in general to reference things easily. In addition, one name may refer to various types of things or objects (one name – many objects). For example, a convenience store named "Family Mart" uses this single name for every branch. Similarly, names are also important in all computer communication. Name services have been developed to map logical names to physical resources. The structure of the current name services is hierarchical for scalability. However, this limits one name mapping to many objects. This paper proposes the new naming system called the Intelligent Naming System that employs the concept of sets and trees. The technique called One Name – Many Objects – One Result (ONMOOR) is proposed to ensure that the result of the mapping is the intended object. Furthermore, the proposed solution has been proved by theoretical analysis.

1. Introduction

Although things or objects can be identified in several ways, "Name" is globally used to be an identifier. Names are also important in all computer systems. A name service is used to map logical names to any physical resources. Several name services have been developed for different purposes.

Global Name Service (GNS) [4] is used in an internetwork with a large naming database that stores resource location and mail addressing.

Handle System [11] is a general purpose name service that allows a name to persist over changes of location and serve for very large number of entities.

Network Information Service Plus (NIS+) [10], developed by Sunsoft engineering team, is a name service for supporting management of basic network resources such as workstation addresses, security information, mail information, etc.

Novell Directory Service (NDS) [2, 6] is a full-function directory service based on the 1988 X.500 standard and provides a single, logical tree-structured view of all resources on the network.

NIS+ and NDS are both proprietary name service. Their target supports within organization and management of basic network resources.

Domain Name System (DNS) [7, 8, 9] is a name service for a world-wide computer network that maps host names to IP addresses. In other words, DNS supports a variety of hosts running on various operating systems and makes communication simpler by using host names instead of addresses.

These name services have been structured hierarchically for scalability. However, we found some features of each name service as shown in Table 1 and revealed that most of the current name services obtain some limitations and drawback.

	Sharing unique name	Character Support	Anonymity
Global Name Service (GNS)	No	ASCII	No
Handle System	Yes	Unicode	No
Network Information Service Plus (NIS+)	No	ASCII	No
Novell Directory Service (NDS)	No	ASCII	No
Domain Name System (DNS)	No	{a-z, A-Z, 0-9, -}[a]	No

[a] DNS can use any octets, but the characters used in DNS labels which are a-z, A-Z, 0-9, and hyphen (-), form to DNS names.

Table 1: Features of current name services.

We classify the limitations of a variety of name services into 3 groups: uniqueness, character support, and anonymity.

- *Uniqueness*

The logical structure of each name service is hierarchical. Therefore, each name in a tree must be unique. The feature of sharing name is limited. This means that each node in the tree must be named differently and point to information of an individual object. Practically, using the same name to identify many objects often occurs.

- *Character support*

Most of the existing name services were designed for supporting only ASCII-based characters while the needs of those using non-ASCII characters have increased.

- *Anonymity*

Anonymity is normally used to protect the privacy of the objects. The name service structure is hierarchical, so that it is easily traced back to its ancestors in the tree.

Although, there are limitations as mentioned above, this paper will mainly focus on the limitation of uniqueness in the name of an organization that consists of many divisions. Hence we propose the new naming system called Intelligent Naming System (INS).

The INS is explained in Section 2. Some examples of the output from the simulation of INS are shown in Section 3. In Section 4, we prove this new architecture by mathematical theory. We discuss our work in Section 5 and finally, we give our conclusion.

2. Intelligent Naming System

The main purpose of designing INS, referred as SNS in [5, 3], are targeted to organizations with several faculties or divisions and each divisions is required to share the names which perform the same functions. Therefore, obtaining one accurate result from mapping one name to many objects is considered.

We suppose that a university name "Prototype University" or PU consists of 4 faculties: Engineering, Medicine, Science, and Nursing. The structure of PU with 4 faculties is shown in Figure 1. Each faculty consists of 3 offices: Academic Affairs (AA), Human Resources (HR), and Planning & Finance (PF). Though these offices have the same name, their faculties are different. Each office of every faculty performs the same tasks. For example, the office of Human Resources at Faculty of Science performs the same functions as the office of Human Resources of other three faculties. If we can refer to the name "Human Resources" for every faculty, it is clearly understand and easy to remember for everyone. Furthermore, using a single name for the same task in different faculty helps the organization maintains its unity.

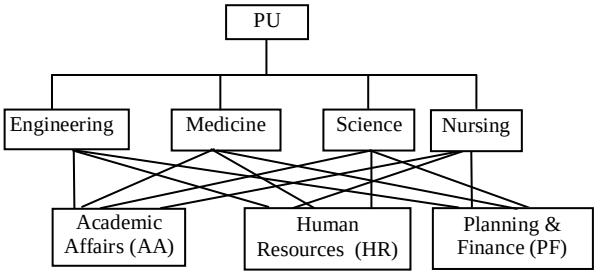

Figure 1: Prototype University and faculties.

2.1 The Structure of INS

The INS structure [5, 3] has been designed under the main consideration of each name is able to be shared. In summary, the global name space is a hierarchical structure with a single root at the top. The structure of the INS employs the concept of sets and trees. A global name space can be represented as labeled nodes. A global name may logically identify not only a single node, but several nodes in the tree. Each global name is a path in an inverted tree and may point to information of individual object, or a set of information of individual objects. Figure 2 illustrates the logical organization of the INS in the sense of a global name able to be shared. A single root at the top is named as "PU" and other global names are the path in an inverted tree. For example, "hr.pu" and "maths.sc.pu" are the global names. In Figure 2, the new structure provides a sharable name such as *hr.pu* from the left branch would map human-resource object at Faculty of Nursing (obj1) and *hr.pu* at the right branch maps human-resource object at Faculty of Science (obj2). Therefore, if we call *hr.pu*, the result provides a set of objects: {obj1, obj2}. This means that the INS maps each global name to a set of objects. In addition, the objects can be referred by using global names in 3 different manners as follows:

1. Full name is a path in the inverted tree such as *hr.pu*.

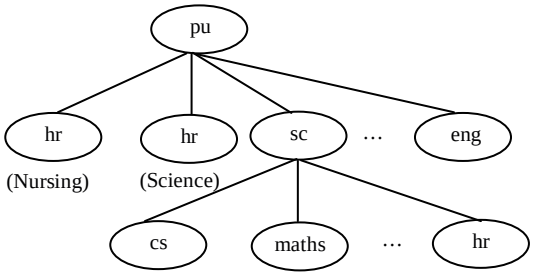

Figure 2: The organization of INS.

2. Combination of global names is a combination of some paths in the tree such as *hr.pu:hr.sc.pu*. Normally, it is supported to a global name which has mapped the object from one path in the tree and required to refer to other global name. For example, *hr.sc.pu* is a global name that maps the object (Obj_a) and for easily remembering, Obj_a maps to *hr.pu*. Hence, both global names map the same object.
3. Global names with a keyword are a global name following by a keyword such as loc for location. For example, *hr.pu:loc@science* is a global name located in Faculty of Science.

Additionally, the query name, which is a global name, has been defined in [5, 3] to support both ASCII and non-ASCII characters. Therefore, the limitation of character support mentioned in Section 1 has been solved.

2.2 Active Components

INS is a client/server model. Figure 3 shows the INS protocol design. There are 3 significant components: INS client, INS server and INS communication.

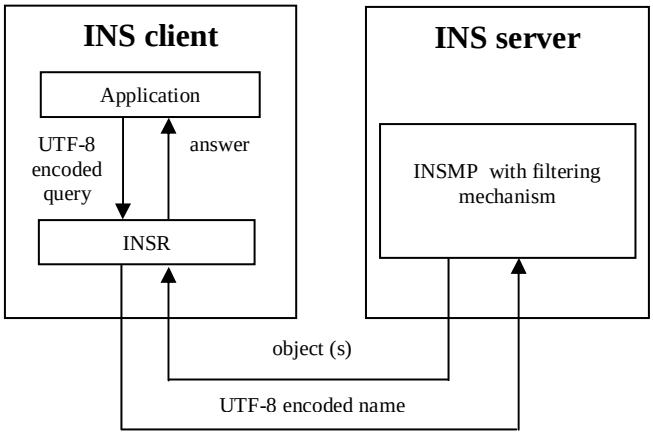

Figure 3: The INS protocol design.

1. INS client consists of an application and INS resolver (INSR). INSR is a function that accesses the INS server to query a name for translation. Normally, the INSR is a part of the application.

2. INS server contains an INS mapping process (INSMP). The INSMP is a process with a filtering mechanism to determine a mapping from a query name to an object.
3. INS communication is an INS message (INSM) which transmits queries and responses between client and server using TCP. The INSM proposed in [5, 3] using TCP is guaranteed that packets will be delivered to the destination in the proper order. Thus, transmitting data is reliable. The detail of INSM can be found in [5, 3] referred as SNSM.

The INSMP with filtering mechanism ensures that the result to a client would be the right unique object. This technique is called One Name – Many Objects – One Result (ONMOOR).

2.3 One Name – Many Objects – One Result (ONMOOR) Technique

2.3.1 Global Name Properties (GNP)

The GNP is closely related to the filtering mechanism. Therefore, we defined a global class (Gclass) and user class (Uclass) in [5, 3]. Gclass stores the information of each global object. Each global object contains 6 attributes: <*Gname*, *Gobj*, *Glocality*, *GcrDate*, *Gcount*, *Gtype*>. *Gname* is a global name. *Gobj* is an object which relates to a global name. *Glocality* is a location that stores the object. *GcrDate* is a creation date of the object. *Gcount* is a total number of the global name queried. *Gtype* is a type of the global name whether being a leaf or a node. Uclass stores some information such as location. The relationship between the global class and the user class is shown in Figure 4 [5, 3].

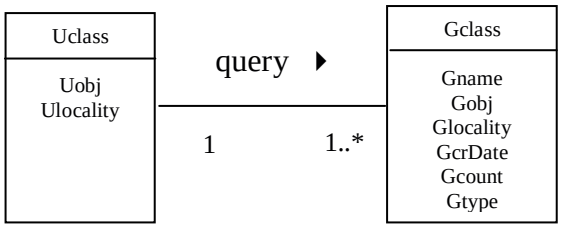

Figure 4: Class diagram of User and Global and their relationship.

The example below refers to Figure 2.

Example : A local name server is "pu" and stores Gclass.

Suppose that Gclass contains six attributes: <*Gname*, *Gobj*, *Glocality*, *GcrDate*, *Gcount*, *Gtype*>. The following table shows the data stored in Gclass

Gname	*Gobj*	*Glocality*	*GcrDate*	*Gcount*	*Gtype*
hr.pu	obj1	Nursing	2002-01-12	5	Lf
hr.pu	obj2	Science	2001-06-28	2	Lf
sc.pu	obj3	Science	1999-08-11	8	Nd
. . .					
nurse.pu	objx	Nursing	2000-09-07	6	Nd

In addition, Uclass contains two attributes: <*Uobj*, *Ulocality*> and the example is shown below:

Uobj	*Ulocality*
Uobj1	pu

2.3.2 The INS Mapping Process

The responsibility of the INSMP is to find the solution. This can be a unique object, or a set of objects. When a client queries a global name (g_1:g_2: …:g_k) at a local name server (*X*), and *X* does not know the answer, a process to decompose the query into subqueries ({g_1, g_2, …, g_k}) is invoked. Then, *X* asks these subqueries to a root name server (*T*). *T* refers to its child for each subquery, or each name server which has been delegated. The name servers are queried to determine a mapping from each subquery name to a set of objects. When each subquery returns, a process to intersect all subqueries is invoked, and gives the answer (*ANS*).

The algorithm to obtain the solution is elaborated as follows, and illustrated in Figure 5.

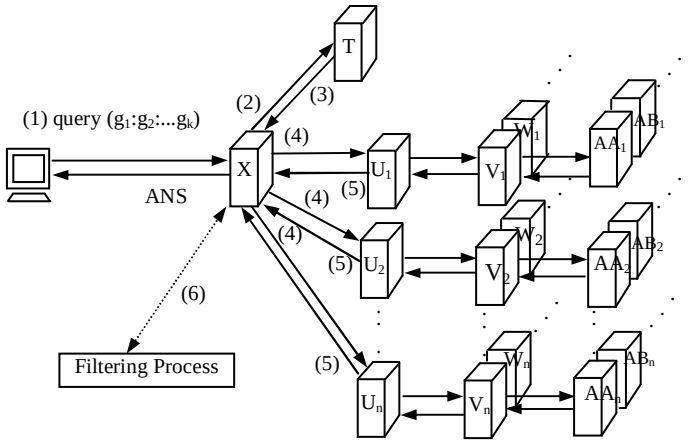

Figure 5: The INS mapping process (INSMP).

Algorithm of INSMP

Let *X* be a local name server which is located nearby the client *C*.
Let *Y* be a *C*'s location. *Y* obtains from *X*'s location.
Let *T* be a root name server which stores information of other name server been delegated.

1. *C* queries a global name (g_1:g_2: …:g_k) to *X*. While *X* receives a message from *C*, *X* also stores *Y*.
2. *X* decomposes the query and forwards each subquery g_1, g_2, …, g_k to *T*.
3. *T* returns a name server, U_i, for each g_i to *X*.
4. *X* queries g_1, g_2, …, g_k to name servers: U_1, U_2, …, U_k, respectively.

5. U_i resolves g_i by either returning data held on U_i or querying the set of name servers that has been delegated. Additionally, each U_i returns the sets of objects found (ans_i) to *X*.
6. *ANS* is the intersection of all sets of objects (ans_1, ans_2, …, ans_k).
   If (*ANS* = {Obj_1, Obj_2, …, Obj_n} then
      Call FilterProcess
   Else if ANS is a unique object then
         X returns *ANS* to *C*
      Else *X* returns "no result" to *C*.

Algorithm of FilterProcess

1. Use Quicksort algorithm to sort *ANS* ({Obj_1, Obj_2, …, Obj_n }) based on counting factors: location and creation date, respectively.
2. *X* takes *Y* comparing to Obj_i in sorted *ANS* (*sANS*).
3. If one object (Obj_i) in *sANS* has a location as *Y* then
      Return Obj_i to *C*
   Else if no object in *sANS* has a location as *Y* then
         For each object in *sANS*
            Find Obj_j with a latest creation date
      Return Obj_j to *C*.

Using Quicksort in FilterProcess algorithm reveals that the expected time of sorting n elements is $O(n \log n)$ [1], and the comparison between the location or latest creation date and *ANS* is $O(n)$. Thus, this algorithm requires the expected time in $O(n + n \log n)$. So, the complexity is $O(n \log n)$. The reason for choosing object with the latest creation date is based on obtaining the new and up-to-date object. However, if the answer after filtering does not satisfy the client, the whole answer list on the basis of the number of accessing each global name, can be presented. We define HIT to be the frequency of accessing each global name. The HIT comes from the *Gcount* attribute in Gclass. Then, the higher the value of *Gcount* is, the more objects are queried.

Using the example of Gclass and Uclass described in Section 2.3.1, the above algorithm gains the following results:

Example 1: Suppose a client at pu queries "hr.pu" to its local name server "pu", the *ANS* returns {*obj1, obj2*}. Therefore, the FilterProcess is invoked and the final result is ***obj1*** because no object in *ANS* has the same location as the client. Then, the algorithm chooses the latest creation date of the object in *ANS* and returns only one object back to the client.

Example 2: Suppose a client at Faculty of Science queries "hr.pu" to its local name server "sc.pu", the *ANS* returns {*obj1, obj2*}. Therefore, the FilterProcess is invoked and the final result is ***obj2*** because *obj2* has the same location as the client.

3. Implementation

We have simulated a client/server program as a prototype of INS and show that INS can be worked properly. We have written an INS finder program used to resolve queries. Suppose that "pu" and "sc.pu" are name servers. The following examples show that the global names map the documented file(s).

1) A client requests a query name that a local name server “pu” can resolve it.

When a user or client requests a query: “*hr.pu*”, the user can choose whether “Result” or “All Results” button to get the result. If the user chooses “Result”, the output will be only one result as shown in Figure 6. If the user chooses “All Results”, the output will be a list of results as shown in Figure 7.

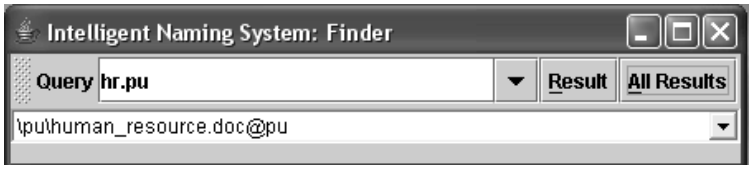

Figure 6: A single result of “hr.pu”.

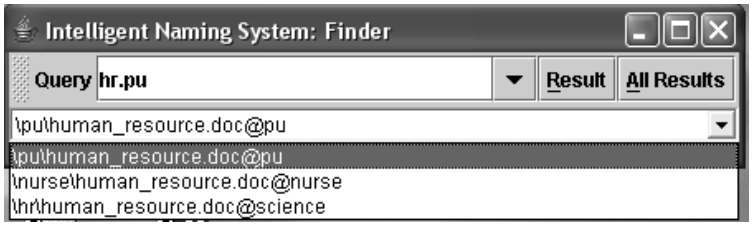

Figure 7: The multiple results of “hr.pu”.

2) A client requests a query name that a local name server “pu” cannot resolve it. Then a local name server sends this query to other related name server that has been delegated. The result shows in Figure 8.

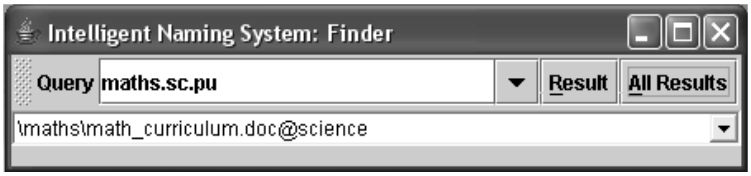

Figure 8: The result of “maths.sc.pu” located at Faculty of Science.

3) A client requests a query name with specific location and the output shows in Figure 9.

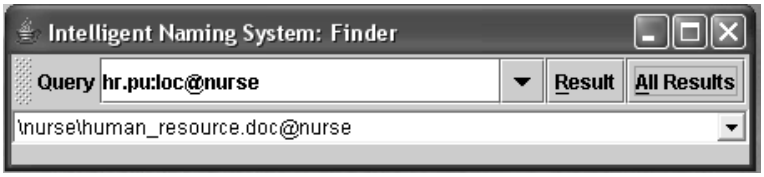

Figure 9: The result of a specific location.

4) A client requests a query name with a combination of global names and the result shows in Figure 10.

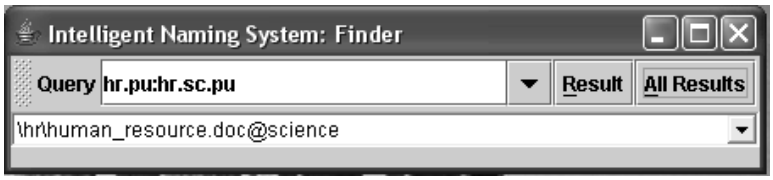

Figure 10: The result of a combination of global names.

4. Theoretical Analysis

As we proposed INS to provide the sharable name property, we also prove this new naming system that can be implied theoretically. The structure of INS is explained by the following definitions.

DEFINITION 4.1 Let α be a set of characters and τ is a string defined by

$$\tau = \alpha - \{":", "."\}.$$

DEFINITION 4.2 Let *tName* (a special node) be the top of the tree and let *vName* be any nodes labeled by τ. Let N be a hierarchical name space consisting of *tName* and *vName* as shown in Figure 11.

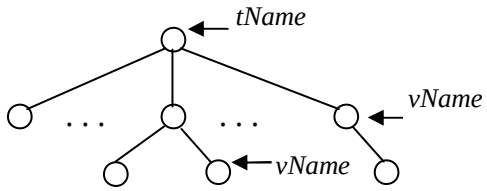

Figure 11: A Name space N.

DEFINITION 4.3 A global name g is a path in a name space N such that

$$g = <vName_0><dl><vName_1><dl>\ldots <tName><dl>$$

where *tName* is the top of the tree,
$vName_i$ is any node in N ($1 \le i \le n, n \in I$),
$vName_0$ is a leaf node in N,
dl is a delimiter character ".".

DEFINITION 4.4 Let f_l be a function of $G \times L$ into O and let T be a subset of $G \times L$ and $f_l(T)$ called " f_l of T ", is defined to be the set of image of elements in T. In other words,

$$f_l(T) = \{x \mid x = f_l(g, l), (g, l) \in G \times L, x \in O\}$$

DEFINITION 4.5 Let f_d be a function of $G \times D$ into O and let V be a subset of $G \times D$ and $f_d(V)$ called " f_d of V ", is defined to be the set of image of elements in V. In other words,

$$f_d(V) = \{x \mid x = f_d(g, d), (g, d) \in G \times D, x \in O\}$$

Remark: It is obvious that $f_l(T), f_d(V) \subseteq O$.

DEFINITION 4.6 Let $S = (N, G, O, L, D, Z)$ be a naming system
Where N is a name space,
G is a set of global names: $\{g_1, g_2, \ldots, g_n\}$,
O is a set of objects: $\{Obj_1, Obj_2, \ldots, Obj_m\}$,
L is a set of locations: $\{l_1, l_2, \ldots, l_o\}$,
D is a set of creation date of the objects: $\{d_1, d_2, \ldots, d_p\}$,
Z is a set of function of T into O or a set of function of V into O denoted by

$$Z = f_l(T) \cup f_d(V).$$

Remark: If $S \neq \varnothing$, then $Z \neq \varnothing$.

DEFINITION 4.7 Let Q be a combination of global names $(g_1: g: \ldots: g_k)$ and $D_c = (g_1, g_2, \ldots, g_k)$ is a set of global names, $D_c \subseteq G$.

DEFINITION 4.8 Let r be a required object of the user if and only if $r = f_l(g, l)$ or $r = f_d(g, d)$.

THEOREM 4.1 Let r be the required object of the user. Then $r \in Z$.
Proof Let r be the required object of the user,
then $r = f_l(g, l)$ or $r = f_d(g, d)$,
$\exists (g, l) \in T$ or $\exists (g, d) \in V$.
Assume that $r \notin Z$ then
case 1. it does not exist $(g, l) \in T$ such that $r = f_l(g, l)$.
Hence, r is not the required object.
case 2. it does not exist $(g, d) \in V$ such that $r = f_d(g, d)$.
Hence, r is not the required object.
Thus $r \in Z$. □

THEOREM 4.2 Let R be a set of the required objects. Let $D_c = \{g_i \mid 1 \leq i \leq k\}$, $D_c \subseteq G$. If there exists $g_i \in S$, then $R \neq \varnothing$.

Proof Let R be a set of required objects,
Let $D_c = \{g_i \mid 1 \leq i \leq k\}$, $D_c \subseteq G$,
Given $g_i \in S$, then $\exists z \in Z$
such that $z = f_l(g, l)$ or $z = f_d(g, d)$,
then z is a required object.
By theorem 4.1, we have $z \in R$. □

Theorem 4.1 shows that a required object must be in a set function (Z). This means that if we request a global name (g) with a location (l) or creation date (d) in a naming system (S), we will obtain the object.

5. Discussion

The INS provides a facility for mapping a single name to many objects. This feature fulfills a need of many organizations with any divisions to maintain the unity of their organization and ease anyone to remember one name regardless of what the objects are.

The INS structure employs both sets and trees. The property of tree helps the organization to remain scalability. This means that even as the number of objects

increases, the protocol remains effective. Additionally, the property of the set provides one name is able to refer to many objects.

INS uses a client/server model. Generally, messages sent and received between a client and a server can be TCP or UDP. UDP does not guarantee that messages reach to the final destination in proper order. TCP, on the other hand, is a stream communication since the unit of data transferred is a sequence of bytes. When each client connects to a server and the connection is established, a new thread is created. Using separate thread for each client, the server can block without delaying other clients. The INSM needs the data sent and received for applications to be reliable. The unpredictable length of data is important. Without using TCP, a module to provide reliability must be provided. Using TCP which is a standard transport layer protocol will take less overhead than adding a new reliability module. Therefore, the INSM with TCP packets guarantees that the data delivered between client and server will not be lost, especially the INS response data.

The INSMP with filtering mechanism allows users to obtain the correct objects. The time complexity of the filtering algorithm is *O(n log n)*. The algorithm allowed showing the whole answer list on the basis of HIT for making the INS more flexible. The implementation of the INS prototype shows that INS can be worked properly. Finally, the definitions and theorem ensure that INS can function as designed.

6. Conclusion

Names are used to refer objects and name services are fundamental services of all computer networks. Various naming systems have been designed. Most of their structures are hierarchical where each name maps to one object. This paper proposes the new naming system called Intelligent Naming System (INS). The INS architecture facilitates a single name mapping many objects while still obtaining one result.

The assumption is that an organization with many divisions and each division uses the same name for the same task to maintain unity. The current name services are implemented in a hierarchical structure. This structure contains a simple parent-child relationship which provides a unique name. The restriction assures that a name uniquely identifies a single leaf in the tree. Names at the leaf nodes of the tree represent individual objects. Thus, it limits names to be shared.

The new structure, on the other hand, employs both the concepts of sets and trees. Hence, names at the leaf nodes may not represent individual objects but a set of objects. The structure allows for one name to be shared. However, mechanisms exist so the correct object is returned to the client. The INSMP with a filtering mechanism based on user location or creation date and HITs, has been proposed to find the required object. The implementation ensures that the new naming system can be worked. Furthermore, the theoretical proof of the INS' function has been presented.

7. Acknowledgement

The authors are grateful to Prof. Fergus O'Brien, School of Information Technology, Faculty of Informatics and Communication, Central Queensland University, who sparkled the idea of the research and gave his valuable time and advice. We would like to thank Mr. Robert Elz, Centre of Network Research, Faculty of Engineering, Prince of Sonkla University, for reading and correcting the manuscript, including valuable suggestion.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, , USA: Addison-Wesley, 1974.

[2] C. Andrew, E. Shropshire et al. *Novell NDS Developer's Guide*. CA: Novell Press, 1999.

[3] P. Bhattarakosol and L. Preechaveerakul. "How to Use One Name for Many Sites?," *in Proceedings CITSA 2004, Vol. 1 of Communications, Information Technologies and Computing*, pp. 57-62, 2004.

[4] B. w. Lampson and DEC Systems Research Center. "Designing a Global Name Service," *in Proceedings 4th ACM Symposioum on Principles of Distributed Computing*, pp. 1-10, 1986.

[5] L. Preechaveerakul and P. Bhattarakosol. "Is It Possible to Use One Name for Many Sites?" *in Proceedings International Conference on Internet Computing 2004*, pp. 360-365, 2004.

[6] PCNA staff. "Understanding NDS," *PC Network Advisor*, Issue. 8, pp. 9-12, March 1997.

[7] P. Mockapetris. "Domain Names – Concepts and Facilities, in RFC1034," http://www.ietf.org/rfc/rfc1034.txt (November 1987).

[8] P. Mockapetris. "Domain Names – Implementation and specification, in RFC1035," http://www.ietf.org/rfc/rfc1035.txt (November 1987).

[9] P. Mockapetris. "Development of the Domain Name System," in *SIGCOMM '88, Symposium Communications Architectures and Protocols*, pp. 123-133, 1988.

[10] R. Ramsey. *All about administering NIS+ 2nd Ed.*, CA: Sunsoft, 1994.

[11] S. X. Sun Lannom and B. Boesch. "Handle System Overview, in RFC3650," http://www.ietf.org/rfc/rfc3650.txt (November 2003).