REFERENCES

1. Wieringa, R. and Jonge, W. (1995). Object identifier, keys, and surrogates – object identifiers revisited. In Theory and Practice of Object Systems. 1(2): 104-114.

2. Birrel, A.D., Levin, R., Schroeder, M.D., and Needham, R.M. (1982). Grapevine: An Exercise in Distributed Computing. Communications of the ACM, 25(4): 260-274.

3. Oppen, D.C., and Dalal, Y.K.(1983). The Clearinghouse: A Decentralized Agent for Lacating Named Objects in a Distributed Environment. ACM Transactions on Office Information Systems. 1(3): 230-253.

4. Mockapetris, P. (1987). Domain Names - Concepts and Facilities. RFC1034 [Online]. Available from: http://www.ietf.org/rfc/rfc1034.txt [2000. November 15]

5. Mockapetris, P. (1987). Domain Names - Implementation and specification. RFC1035 [Online]. Available from: http://www.ietf.org/rfc/rfc1034.txt [2000. November 15].

6. Mockapetris, P.(1988). Development of the Domain Name System. Proceedings of SIGCOMM'88, 88: 123-133.

7. Lampson, B. w. and DEC Systems Research Center (1986). Designing a Global Name Service. Proceedings of the 5th annual ACM Symposium on Principles of Distributed Computing, 5: 1-10.

8. Ramsey, R.(1994). All about administering NIS+, 2nd ed. CA: Sunsoft.

9. PCNA staff. (1997). Understanding NDS. PC Network Advisor. 81: 9-12.

10. Andrew. C., et al. (1999). Novell NDS Developer's Guide. CA: John Wiley & Sons Inc.

11. Sheresh. B., and Sheresh. D. (1999). Understanding Directory Services. USA: New Riders.

12. Sun. S.X. (1998). Internationalization of the Handle System - A Persistent Global Name Service. The 12th International Unicode Conference, 12: 1-11.

13. Sun. S., Lannom L., and Boesch B. (2003). Handle System Overview [Online]. Available from: http://www.ietf.org/rfc/rfc3650.txt [2003, November 21].

14. Siegel. J. (1988). OMG Overview: CORBA and the OMA in enterprise computing. Communications of the ACM. 41(10): 37-43.

15. Yang. Z., and Duddy. K. (1996). CORBA: A Platform for Distributed Object Computing. ACM SIGOPS Operating Systems Review. 30(2): 4-31.

16. Kwon. J.H., Jeong, M.S., and Park J-T. (2001). An Efficient Naming Service for CORBA-based Network Management. Proceedings of IFIP/IEEE International Sysmposium on Integrated Network Management, pp.765-778.

17. Stevens, W.R. (1994). TCP/IP Illustrated, The Protocols. vol. I. NJ: Addison-Wesley.

18. Hunt, C. (1998). TCP/IP Network Administration, 2nd ed. CA: O'Reilly & Associates Inc.

19. Socolofsky, T. and Kale, C. (1991). A TCP/IP Tutorial, RFC1180 [Online]. Available from: http://www.ietf.org/rfc/rfc1180.txt [2001. January 14].

20. Albitz, P. and Lui, C. (1998). DNS and BIND. 3rd ed. CA: O'Reilly & Associates Inc.

21. Postel, J.(1981). Internet Protocol, RFC791 [Online]. Available from: http://www.ietf.org/rfc/rfc791.txt [2000, September 6].

22. Hinden, R. (1996). IP Next Generation Overview. Communications of the ACM. 39(6): 61-71.

23. Deering, S. and Hinden, R. (1998). IP Version 6 Addressing Architecture, RFC2373 [Online]. Available from: http://www.ietf.org/rfc/rfc2373.txt [2001. July 4].

24. Deering, S. and Hinden. R. (1998). Internet Protocol. Version 6 (IPv6) Specification, RFC2460 [Online]. Available from: http://www.ietf.org/rfc/rfc2460.txt [2001. December 4].

25. Coulouris, G., Dollimore, J., and Kindberg. T. (2001). Distributed Systems Concepts and Design, 3rd ed. Essex: Addison-Wesley.

26. Klensin, J. (2003). Role of the Domain Name System (DNS) [Online]. Available from: http://www.ietf.org/rfc/rfc3467.txt [2003, February].

27. Faltstrom, P., Hoffman, P., and Costello, A. (2003). Internationalizing Domain Name in Applications (IDNA),RFC3490 [Online]. Available from: http://www.ietf.org/rfc/rfc3490.txt [2003, March 3].

28. Hoffman, P., and Blanchet, M. (2003). Nameprep: A Stringprep Profile for Internationalized Domain Name (IDN), RFC3491 [Online]. Available from: http://www.ietf.org/rfc/rfc3491.txt [2003, March 3].

29. Costello. A. (2003). Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA), RFC3492 [Online]. Available from: http://www.ietf.org/rfc/rfc3490.txt [2003, March 3].

30. Tanenbaum, A.S., and Steen, M.v. (2002). Distributed System: Principle and Paradigms. NJ: Prentice Hall.

31. Schneier, B. (1996). Applied Cryptography, 2nd ed. USA: John Wiley & Sons, Inc.

32. Kahate, A. (2003). Cryptography and Network Security. Singapore: McGraw-Hill Companies.

33. Diffie, W. and Hellman. M.E. (1976). Multiuser Cryptographic Techniques. Proceedings of AFIPS National Computer Conference. pp. 109-112.

34. Welder, C., Reynolds, J. and Heker, S. (1992). Technical Overview of Directory Services Using the X.500 Protocol, RFC1309 [Online]. Available from: http://www.ietf.org/rfc/rfc1309.txt [2002, March 16].

35. Nash, A. et al. (2000). PKI-Implementing and Managing E-Security. USA: Tata McGraw-Hill.

36. Rivest, R. Shamir, A., and Adleman, L. (1978). A Method for Obtaining Digital Signature and Public key Cryptography Systems. Communication of the ACM, 21(2): 120-126.

37. Arms, W.Y. (2001). Uniform Resource Names: Handles, PURLs, and Digital Object Identifiers. Communications of the ACM. 44(5): 68.

38. Object Management Group (OMG). (2000). Naming Service Specification. OMG Document formal/2000-09-19, Version 1.0.

39. Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., and Lilley, J. (1999). The Design and Implementation of an Intentional Naming System. Proceedings of the 17th ACM SOSP, pp.186-201.

40. Minami. M., Morikawa, H., and Aoyama, T. (2003). An Interface-based Naming System for Ubiquitous Internet Applications. Proceedings of 9th International Conference, RTCSA 2003, pp. 312-327.

41. Minami, M., Morikawa, H., and Aoyama, T. (2003). The Design of Naming-based Service Composition System for Ubiquitous Computing Applications. 2004 Symposium on Applications and the Internet workshops, pp. 304-312.

42. Martin, E. (1995). X/Open Federated Naming - specification for uniform naming interfaces between multiple naming systems - Technology Information. Hewlett-Packard Journal. 46(6).

43. Homburg. P.C. (2001). The Architecture of a Worldwide Distributed System. Ph.D. Thesis. Advanced School for Computing and Imaging, VRIJE Universiteit.

44. Tan. J.K., Leong, K.Y., and Tan, T.W. (1999). iDNS, an Experimental DNS System with Unicode Support. Proceedings of the 14th International Unicode Conference, pp.1-12.

45. Seng, J. and Yap, J. (2001). iDNS – The Next Big Step in the Internet Saga [Online]. Available from: www.i-dns.net/pdf/APIA-01-06-21-Seoul.pdf [2001, June 24].

46. Preechaveerakul, L. and Bhattarakosol P. (2004). Is It Possible to Use One Name for Many Sites? Proceedings of International Conference on Internet Computing 2004, pp. 360-365.

47. Bhattarakosol, P. and Preechaveerakul, L. (2004). How to Use One Name for Many Sites? Proceedings of the International Conference on Cybernetics and Information Technologies, Systems and Applications, pp. 57-62.

48. Fielding, R., et al. (1999). Hypertext Transfer Protocol – HTTP/1.1, RFC2616 [Online]. Available from: http://www.ietf.org/rfc/rfc2616.txt [2004, June 25].

49. Fonner, L. (2001). Fixing a flawed domain name system. Communications of the ACM, 44(1): 19-21.

50. Postel, J. (1980). User Datagram Protocol, RFC768 [Online]. Available from: http://www.ietf.org/rfc/rfc768.txt [1980, August].

51. Information Science Institute. (1981). Transmission Control Protocol, RFC793 [Online]. Available from: http://www.ietf.org/rfc/rfc793.txt [2000, September 14].

52. Yergeau, F. (2003). UTF-8, a transformation format of ISO 10646, RFC3629 [Online]. Available from: http://www.ietf.org/rfc/rfc3629.txt [2003, November 24].

53. The Documentation Team of Algorithma (2004). User Manual, Algorithma 2004 [Online]. Available from: http://www.csci.csusb.edu/public/class/cs455/cs455_2004/UserManual2004.pdf [2006, January 15].

54. Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1974). The Design and Analysis of Computer Algorithms. USA: Addison-Wesley.

55. Adams, G. (1993). Internationalization and Character Set Standards. ACM Press. 1(1): 31-39.

56. Baase, S. (1988). Computer Algorithms: Introduction to Design and Analysis, 2nd ed., USA: Addison-Wesley.

57. Berthodd, O., Federrath, H., and Kohntopp, M. (2000). Project "Anonymity and Unobservability in the Internet". Proceedings of the 10th conference of Computers, freedom and privacy: challenging the assumptions, pp. 57-65.

58. Davis. M. and Lee, C. (1990) Unicode. IEEE International Conference on Systems, Man and Cybernatics. pp. 499-504.

59. Deitel, H.M., and Deitel, P.J. (2003). Java: How to Program. 5th ed., NJ: Pearson Education Inc.

60. Farrel, C., Schulze, M., Pleitner, S., and Saldoni, D. (1994). DNS Encoding of Geographical Location, RFC1712 [Online]. Available from: http://www.ietf.org/rfc/rfc1712.txt [2002, November 30].

61. Kurose, J.F., and Ross, K.W. (2001). Computer Networking: A Top-Down Approach Featuring the Internet. USA: Addison-Wesley Longman, Inc.

62. Martin, J.C. (2003). Introduction to Languages and the Theory of Computation, 3rd ed., NY: McGraw-Hill.

63. Mullender, S. (1993). Distributed Systems, 2nd ed., USA: Addison-Wesley.

64. Phillips, A.P. (2002). Four ACEs: A Survey of ASCII Compatible Encodings. Proceedings of 22nd International Unicode Conference, pp. 1-15.

65. Preiss. B.R. (1999). Data Structures and Algorithms. Object-Oriented Design Patterns in Java. USA: Wiley Publishers.

66. Rivest, L.R. (1990). Cryptography. In J. v. Leeuwen (ed.). Handbook of Theorectical Computer Science, pp. 717-755. Elsevier.

67. Schuba. C.L., and Spafford. E.H. (1993). Addressing Weaknesses in the Domain Name System Protocol. M.S. Thesis, Coast Laboratory, Department of Computer Science. Purdue University, West Lafayette, IN.

68. Topping, S. (2000). The Multilingual domain name race: on your work... get set... WAIT! As the Web continues to expand globally, the English-oriented system needs updating - but how? [Online]. Available from: http://www106.ibm.-com/develperworks/library/u-domains.html [2000, June 24].

69. Rosen, K.H. (1999). Discrete Mathematics and Its Applications, 4th ed., Singapore: McGraw-Hill.

70. Piff, M. (1991). Discrete Mathematics: An Introductin for Software Engineers. Cambridge: Cambridge University press.

APPENDICES

APPENDIX A

This section presents a following journal paper generated from this dissertation. This paper is available online at http://pune.sancharnet.in/kpr_tmrf/ijcsa.html

- Preechaveerakul, L. and Bhattarakosol, P. Intelligent Naming System: An Alternative for Enterprise Naming Management. **International Journal of Computer Science and Applications**. 3(1): 92-103.

ISSN 0972 - 9038

International Journal of Computer Science & Applications

Volume 3 Issue 1 March 2006

Editor-in-Chief
Rajendra Akerkar

Technomathematics Research Foundation

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.

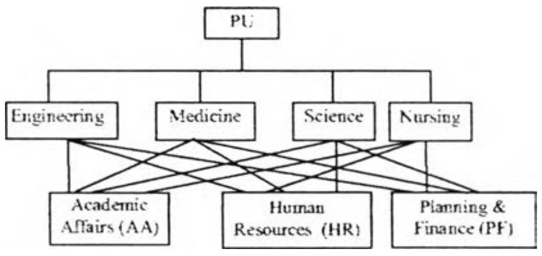


Figure 1: Prototype University and faculties.

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*.

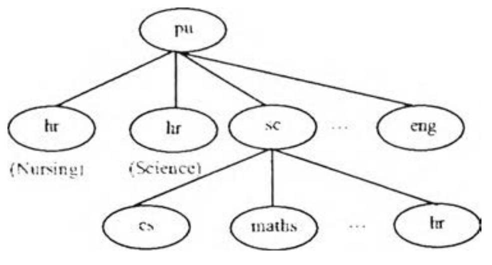


Figure 2: The organization of INS.

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_n) and for easily remembering, Obj_n 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.

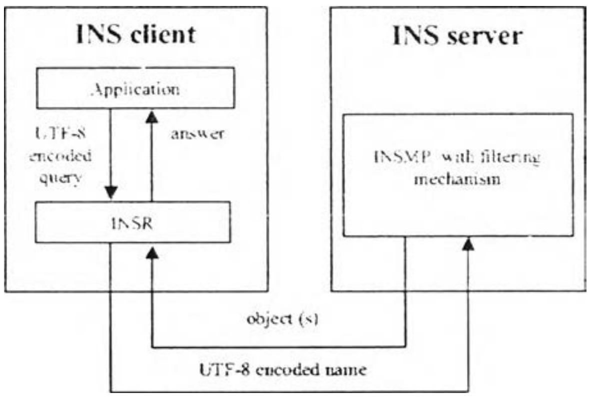


Figure 3: The INS protocol design.

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].

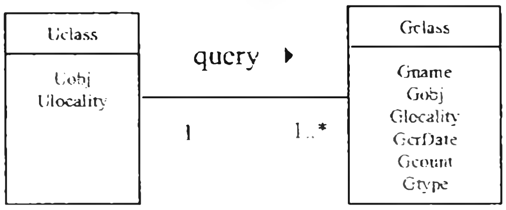


Figure 4: Class diagram of User and Global and their relationship.

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 ($\{g_1, g_2, \ldots, g_k\}$) 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.

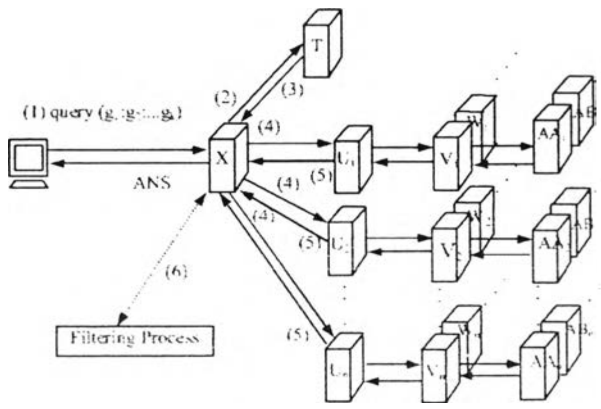


Figure 5: The INS mapping process (INSMP).

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.

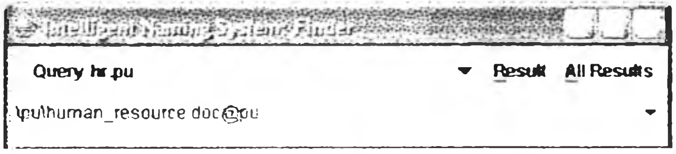
Example 2: Suppose a client at Faculty of Science queries "hr.pu" to its local name server "sc.pu", the ANS returns {*obj1*, *obj2*}. Therefore, the FilterProcess is invoked and the final result is ***obj2*** because *obj2* has the same location as the client.

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).

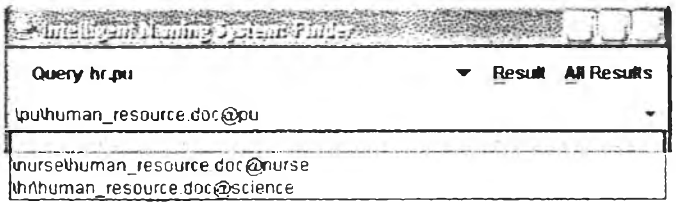
1) A client requests a query name that a local name server "pu" can resolve it.

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.



\pu\human_resource.doc@pu

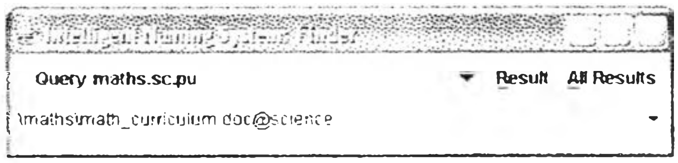
Figure 6: A single result of "hr.pu".



\pu\human_resource.doc@pu
\nurse\human_resource.doc@nurse
\hr\human_resource.doc@science

Figure 7: The multiple results of "hr.pu".

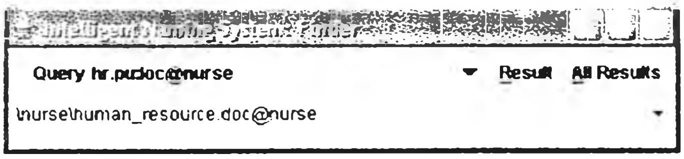
2) A client requests a query name that a local name server "pu" cannot resolve it. Then a local name server sends this query to other related name server that has been delegated. The result shows in Figure 8.



\maths\math_curriculum.doc@science

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

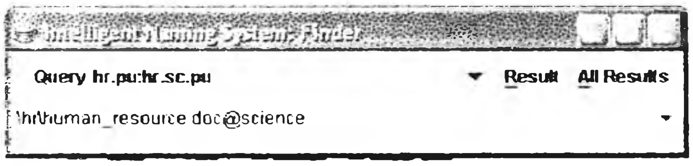
3) A client requests a query name with specific location and the output shows in Figure 9.



\nurse\human_resource.doc@nurse

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.



Query hr.pu:hr.sc.pu

\hr\human_resource.doc@science

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

$$\tau = \alpha - \{\text{“:”}, \text{“.”}\}.$$

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.

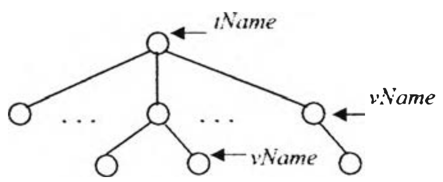


Figure 11: A Name space N.

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

$$g = \langle vName_n\rangle\langle dl\rangle\langle vName_i\rangle\langle dl\rangle \ldots \langle tName\rangle\langle dl\rangle$$

where *tName* is the top of the tree,
$vName_i$ is any node in N ($1 \le i \le n$, $n \in I$),
$vName_n$ 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 $(g_1; g_2 \ldots g_k)$ 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).

APPENDIX B

This section presents a following conference paper which is a part of this dissertation.

- Bhattarakosol. P. and Preechaveerakul. L., "How to Use One Name for Many Sites?", in *Proc. of the International Conference on Cybernetics and Information Technologies. Systems and Applications(CITSA 2004)*, Florida, Orlando, USA. July 21-25. pp. 57-62. 2004.

International Conference on Cybernetics and Information Technologies, Systems and Applications

ISAS

CITSA

2004

10th International Conference on Information Systems Analysis and Synthesis

July 21-25, 2004 Orlando, Florida ~ USA

PROCEEDINGS

Volume I

Communications, Information Technologies and Computing

Edited by

Hsing-Wei Chu

Jose Aguilar

Jose Ferrer

Organized by IIIS

International Institute of Informatics and Systemics

Member of the International Federation of Systems Research

How to Use One Name for Many Sites?

Pattarasinee. BHATTARAKOSOL
Department of Mathematics
Faculty of Science. Chulalongkorn University
Bangkok. Thailand 10330

and

Ladda PREECHAVEERAKUL
Department of Mathematics
Faculty of Science. Chulalongkorn University
Bangkok. Thailand 10330

ABSTRACT

"Name" is an important identifier that is used globally. Without names. people cannot indicate things easily. However. in the real world. people may want to use the same name to identify many objects such as a university uses the same university name for every campus. Unfortunately that this need does not fully support in the existing naming system in the computer world. If we consider seriously, we will find that there are three limitations and drawback of the current naming system: uniqueness. scalability. and anonymity has been occurred. Therefore. this paper proposed a new architecture of the naming system called Sharable Name System (SNS) that solves all limitations of the current naming system. supports and maintains the unity of the organizations using one name for various objects. Thus. using this architecture the limitation for setting names will be eliminated.

Keywords: Naming System and Unity

1. INTRODUCTION

"Name" is an important identifier that is used globally. Names are used to share resources. to uniquely identify entities. to refer to locations. and so on [17]. In the computer world. a vast interconnected collection of host computers called Internet. needs names to refer to the addresses. An open protocol standards. TCP/IP protocols are comforted machines from anywhere to communicate. Standard and non-standard application protocols have been developed for services especially. name service which is a fundamental service to all computer networks. We describe a variety of name service in section 2 and conclude limitations and drawback of them in section 3. We propose the new naming system called Sharable Name System (SNS) which address the limitations and drawback of current name services. This paper contributes for the name service that can be shared: one name to many objects. In section 5, we discuss our work. Finally. we give our conclusion.

2. RELATED WORK

2.1 TCP/IP Protocol

The TCP/IP protocols [6, 14] are the open protocol standards. freely available and developed independently from any specific computer hardware or operating system. It provides the technology that glues internet together. TCP/IP protocols are facilitated machines from anywhere communicated. Standard and non-standard application protocols have been developed. TCP/IP protocol suite contains the standard application protocol for services like file transfer. remote login, and electronic mail. For example. a file transfer client uses the standard FTP protocol to transfer files between 2 machines. a remote login client uses the standard TELNET protocol. and an electronic mail client uses the standard SMTP or POP protocol to transfer or access email. Most network managers choose standard application protocols when the existing protocols suffice.

2.2 A Variety of Name Services

Name Service. a fundamental service to all computer networks. is used to map physical resources to logical names. Several name services have been developed.

2.2.1 Global Name Service (GNS)

In 1986. Global Name Service (GNS) have been designed at the DEC system Research Center [7] to provides facilities for resource location. mail addressing and authentication. It was designed on the basis of Grapevine [3] and Xerox Clearinghouse [12] including these requirements such as large size. long life. high availability. fault isolation and tolerance of mistrust. The naming database in GNS composes of a tree of directories holding names and values. Each directory is assigned an integer value. which serves as a unique directory identifier (DI). GNS successfully addresses the need for scalability and re-configurability with the exception of the solution adopted for merging and moving directory trees [4, 16]. For example. two previously separated GNS services may be integrated with the introduction of a new root above the two existing roots. The well-known directory tables are used to store the previous directory identifiers and remap to the current real root directory of the naming database. Whenever the real root of the naming database was changed. all GNS servers are informed of the new location of the real root. In a large-scale network. reconfiguration may occur at any levels. This makes the table grow rapidly. conflicts with the scalability goal.

2.2.2 Handle System

Handle System [16]. a general-purpose global name service and developed by CNRI (http://www.cnri.reston.va.us), allows secured name resolution and administration over the Internet. It is a persistent name service such that the system allows a name to persist over changes of location, ownership, and other network state conditions. Additionally, it serves for very large number of entities and allow administration at the name level. Its structure is hierarchical. The disadvantage of this system [2] is the effective use requires users install the special browser software.

2.2.3 Network Information Service Plus (NIS+)

The Network Information Service Plus (NIS+) [13], a proprietary name service and targeted within organization, was developed by the Sunsoft engineering team. It stores information about workstation addresses, security information, mail information, Ethernet interface, and network services in central locations where all workstations on a network can access it. The name space is hierarchical and able to divide into multiple domains. NIS+ focuses on making network administration more manageable over a variety of network information.

2.2.4 Novell Directory Service (NDS)

Novell Directory Service (NDS) [1, 8] was introduced in 1993 as a method of managing Netware networks. It supports management of basic network resources including applications, services, and other information concerning the network. NDS is a full-function directory service based on the 1988 X.500 standard and provides a single, logical tree-structure view of all resources on the network. The protocols used in NDS are primarily proprietary.

2.2.5 Domain Name System (DNS)

Domain Name System (DNS) was designed in 1984 and specified in [9, 10, 11]. DNS is a distributed database based on the TCP/IP protocol. DNS maps human-readable host names to numerical IP-addresses. The database structure is strictly hierarchical. A name such as sc.chula.ac.th has a structure as shown in Figure 1.

As the variety of naming systems referred above, we summarize the features of each name service in Table 1 and found their limitations.

3. LIMITATIONS AND DRAWBACK OF VARIOUS NAME SERVICES

Since the structure of each name service referred in section 2 is hierarchical. Most of them have the limitations and drawback that we categorize into 3 groups: uniqueness, scalability and anonymity.

Uniqueness. The structure of each name service is hierarchical, therefore each name in a tree must be unique.

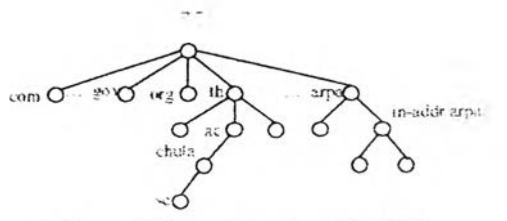


Figure 1 The organization of the DNS.

Scalability. Most of the existing naming system was designed for people from a single language (English), not other languages.

Anonymity. It is sometimes necessary to say things anonymously [5]. The name service structure is hierarchical so that it is easily traced back to its ancestors in the tree.

In order to solve these limitations and drawback, hence we propose the new naming system called Sharable Name System (SNS).

4. THE NEW NAMING SYSTEM

Our main purposes for this new naming system are targeted in and between organization and the prototype maps names to objects. Additionally, each name is able to be shared.

4.1 System Environment

Our system environment is considered based on the real world organizations with more than one office locations. For example, the company of Thai Airways International has offices around the world under the same name that is "Thai Airways International". The other example is an academic organization, Prince of Songkla University (PSU), this university has 5 campuses located in different provinces that are HatYai, Pattani, Phuket, Surattani, and Trang. Figure 2 presents the structure of PSU which shows that each location consists of 3 divisions: Storehouse (S), Administration (A), and Human Resource (HR) divisions. Although these divisions have the same name but their locations are different. Each division of every location performs the same tasks. For example, the Administration division at HatYai has the same functions as Administration divisions of other four campuses. Thus, everybody can clearly understand and easy to remember. Let us consider documents flow around every campus of PSU. If HatYai campus calls the student information system as Student File while Pattani campus calls the same system as Student Infor, the confusions can occur for outsiders of these organizations. Therefore, to avoid this confusion, every documents of the same type of information should use the same document name. Thus using

Table 1 Features of the existing name service.

	Sharing unique name	Scalability	Anonymity
Global Name Service (GNS)	No	Yes/No[1]	No
Handle System	Yes	Yes	No
Network Information Service Plus (NIS+)	No	No	No
Novell Directory Service (NDS)	No	No	No
Domain Name System (DNS)	No	Yes	No

[1] Not scalable in large-scale network when merging and moving directory tree

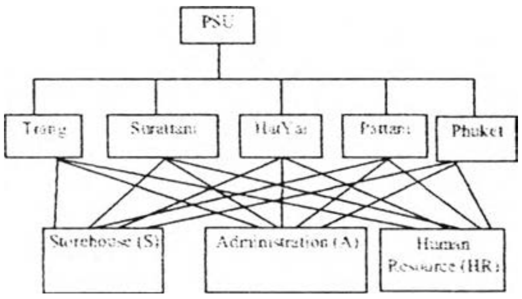


Figure 2 Prince of Songkla University and campuses.

the same name for the same task or the same object helps PSU maintains unity of its organizations.

Therefore, the system environment that will be taken into the consideration of this research is an organization that consists of many branches with the same name and every same type of document uses the same document name as well. This situation is called "one name – many objects – one result" (ONMOOR).

4.2 Problem Statement

Considering the situation of a person who wants to contact a HR division of PSU or looking for a student information of PSU, the problem that arises is "which location (or document) is the right place (or document) for that person?". Using a computer searching system like google.com, the user will obtain more than one answers at a search result, and the user may not be satisfied according to time consuming in searching for the right one. Therefore, solving the problem of "one name – many objects – one solution" to serve the user needs is a challenging problem.

Since there are many naming systems in the IT world, those systems normally implemented in the hierarchical structure and each node must be named differently. Therefore all existing naming systems cannot solve the problem stated above.

4.3 Principle Design concepts

Normally, each organization would have a standard which leads to the questions. We found that the existing name services are not able to solve them. Therefore, we propose a new name service based on a client-server model called Sharable Name Service (SNS). The SNS is designed on the basis of DNS and requires the server hold.

1. Scalability: it will remain effective when there is a significant increase in the number of resources and the number of users.
2. Decentralize administration via delegation: a domain can divide into subdomain. Each subdomain or delegation is responsible for maintaining all the data
3. Opacity: the use of a name conceals from the user location in objects. Location opacity enables resources to be accessed without requiring knowledge of their location.
4. Efficiency: it is a distributed database with hierarchical structure that allows different parts of the naming database to be maintained by different entities. Additionally, caching obviates the query that cannot answer locally and need to query other servers.

4.4 The Structure of SNS

The global name space is a hierarchical structure with a single root at the top. Each name is a path in the global name space. The objects are the global names that point to information about individual objects. A hierarchical structure of the SNS employs the concept of sets. Figure 3 illustrates the logic organization of the SNS in the sense of a global name able to be shared. A single root at the top in Figure 3 is named as "PSU" and other nodes in the tree are global names. A global name may logically identify not only a single node, but nodes in the tree. Each global name may point to information about individual object or a set of information about individual object. This means that the SNS maps each global name to a set of objects. For example, in Figure 4, the new structure provides a sharable name such a *store.psu* from the left branch would map storehouse at HatYai (obj1) and *store.psu* at the right branch maps storehouse at Pattani (obj2). However, if we call *store.psu*, the result provides a set of objects, {obj1, obj2}.

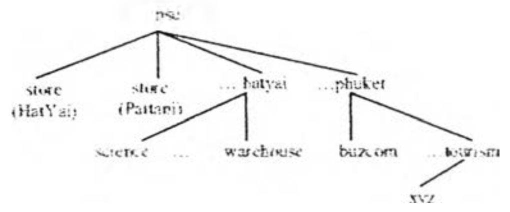


Figure 3 The organization of SNS.

4.5 The SNS Message Protocol

TCP/IP protocol suite contains standard application protocols for services. However, the existing standard protocols do not serve a need for sharing names. Therefore, we have designed a new protocol called SNS message protocol. The SNS queries and responses are often contained within TCP packets. Communication inside the SNS protocol is carried in a single format called an SNS message. All messages – both queries and responses – are divided into 3 sections: Header, Question and Answer section as shown in Figure 4.

The header section contains fields specified whether a message is a query or response, and also specified which of the remaining sections are presented. The ID field is an 8-bit identifier. The fg is a 1-bit field that specifies whether this message is a query (0) or a response (1). The Opcode is a 3-bit field that specifies a kind of query in this message.

The RCODE is a 4-bit field with the return code. The values have the following interpretation

0 No error condition
1 Format error – name server cannot interpret the query.
2 Server failure – name server cannot process the query.
3 Not implemented – name server does not support the requested query type.
4-15 Reserved for future use.

The next two 16-bit fields specify the number of questions and the number of answers respectively. For a query, the number of question is normally 1 and the other field is 0. For a reply, the number of answer is at least 1.

The format of the question section is a query name. The query name is a global name being look up. It is a sequence of one or more labels. The query name field (QName) is stored in Unicode. Each Unicode character is encoded in UTF-8 (UCS Transformation Format). Each label is a length and a number of

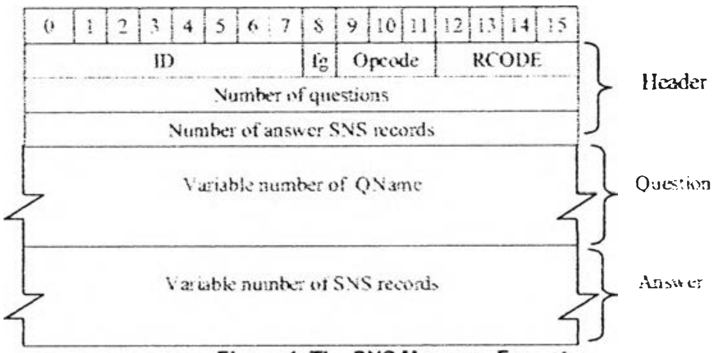


Figure 4 The SNS Message Format

byte per character followed by the number of characters. The name terminated with a byte of 0. The following examples of global names show how they are stored.

A global name "store.psu" is stored as follow.

5.1	s	t	o	r	e	3.1	p	S	u	0

The other global name is [illegible] stored as follow:

4.3	ส	น	.	ก	3.3	ค	ล	ง 0

The last example shows a Thai global name which uses non-ASCII characters.

The global name that is able to stored in Unicode expands to encompass various languages not only English. Therefore, the limitations of scalability as we mentioned previously has been solved.

The answer section may contains one of more records called the SNS records. The number of records is specified in the corresponding field in the header section. Each SNS record contains name, time-to-live, SNS data length and SNSDATA field. Figure 5 describes format of an SNS record.

The name field is a global name which SNSDATA corresponds. The name field has a same format as described in query name field.

The time-to-live field (TTL) is the 16-bit field that specifies the time interval (in seconds) that the SNS record may be cached before it should be discarded.

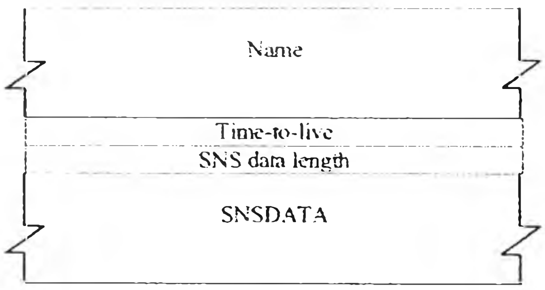


Figure 5 Format of SNS record.

The SNS data length specifies the amount of SNS data.

The SNSDATA is a variable length string that describe the global name varies according to a set of objects including the two important attributes: location and created Date.

4.5.1 Query and Response example

The following query and response illustrates name server behavior. When a client queries a global name"store.psu" to a name server, the question section is stored store.psu. The query message format may look like:

Header	Opcode = query
Question	QName = store.psu
Answer	<empty>

The response may be:

Header	QR = Response, Opcode = query
Question	QName = store.psu
Answer	store.psu 86400 17 obj1 HatYai 13012001 store.psu 86400 18 obj2 Pattani 05032002

4.6 Active Components

SNS is a client-server architecture; therefore the active components are name servers and resolvers. Resolvers are the clients that access name servers. Normally, a resolver is a part of an application. A resolver queries a name server to map names to objects. The concepts of sets and trees are provided for name to object translation. Each tree contains nodes which resolve to a set of objects or may refer to other servers. A path through the tree terminates at a node. The set of objects contained that in the node is returned as the result of the path. The querying type should be one of the following types: *full path* such as a global name: *store.psu* or a combination of global names: *store.psu:warehouse, hatyai.psu*, and some parts of the path such as *store.psu:loc@hatyai*. The path *store.psu:warehouse, hatyai.psu* is a set of paths.

Before we explain the resolution process, let us take a closer look at Global Name Properties.

4.7 Global Name Properties

All objects usually use names to indicate what they are. The objects may refer as documented files, images, addresses, and so on. Each name has its properties and certainly has an identity. Our work proposes a sharable name for objects, so that identifying the right object is very important issue. Therefore, we define a global class (Gclass) to store the information of each global object. Additionally, user class (Uclass) is defined to store some information such as localation. We also show a relationship between the global class and the user class in Figure 7.

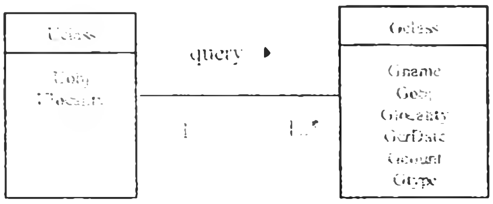


Figure 7 Class diagram of User and Global and their relationship.

Uclass stores one of the important attributes called Ulocality which is a value of client location. Gclass stores Gname, Gobj, Glocality, GetDate, and Gtype. Gname is a global name. Gobj is an object that related to Gname. Glocality is a value of a global object location. Every time a global name is queried, Gcount is increased by 1. Gcount will be set to 0 at the first time Gname is created or when time-to-live (TTL) is 0 or 1. Gtype attribute specifies whether a global name is a leaf node (Lf) or a node (Nd) in the global name space. Normally, a Ulocality attribute in Uclass comes from a location of a local name server that a client queries a global name. Each name server will store database of Gclass. Every time a new object is created, its information will be stored after registration process.

The example below refer to Figure 3.

Example a local name server is psu and stores Gclass

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

Gname	Gobj	Glocality	GetDate	Gcount	Gtype
store.psu	obj1	HatYai	15-01-2001	5	Lf
store.psu	obj2	Pattani	05-03-2002	2	Lf
hatyai.psu	obj3	HatYai	12-11-1997	8	Nd
phuket.psu	objn	Phuket	30-09-2000	6	Nd

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

Uobj	Ulocality
Uobj1	HatYai

4.8 Resolution Process

The resolution process in Figure 8 gives the ANS returned to a resolver either a unique object or a set of objects. Therefore, a local name server invokes a filtering mechanism to guarantee a client to receive the right unique object on the basis of location or created date. The resolution process explains as follow:

Let Name server X be a local name server

1. A client requests a query $(q_1q_2q_3...q_n)$ to name server X. While name server X gets a message from a client, name server X also gets and stores a client information such a location.
2. Name server X decompose each subquery $q_1, q_2, ..., q_n$ to name server B
3. Name server B returns other name server of each subquery to X
4. Name server X queries $q_1, q_2, q_3, ..., q_n$ to name servers $C_1, C_2, C_3, ..., C_n$
5. Each name server $(C_1, C_2, C_3, ..., C_n)$ resolves its subquery by either returning data held on the name server or querying the set of name servers that has been delegated. Each server returns the sets of objects found (ans_i) to name server X.
6. ANS is the intersection of all set of objects $(ans_1, ans_2, ..., ans_n)$.

```
If ANS is a set of objects {Obj1, Obj2, ..., Objn} then
        Call FilteringProcess
Else if ANS is a unique object
        Name server X returns ANS to a client
Else
        Name server X returns "no result" to a client
```

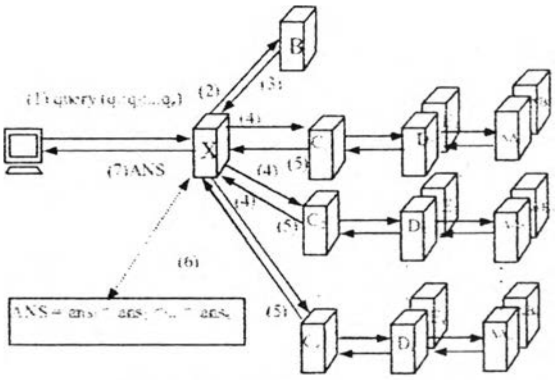


Figure 8 The new resolution process.

The algorithm of FilteringProcess describes below:

```
1. Use Quicksort algorithm to sort ANS ({Obj1, Obj2, ..., Objn}) based on counting factors: location and created date respectively.
2. Name server X takes a client location comparing to the location of each object in sorted ANS
3. If one object (Obji) in sorted ANS has a same location
       as a client
   then
         return Obji to a client
   Else if there are a set of objects ({Obj1, Obj2, ...,
          Objn}) or no object in sorted ANS has a same
          location as a client
       then
           For each object in sorted ANS
             Find obji with a latest created date
             Return obji to a client
```

This algorithm requires the expected time in O(n + n log n). So that, the complexity is O(n log n). The reason for choosing the latest created date object is based on the human nature that loves to get the new and up-to-date object. However, if the answer after filtering does not satisfy the client, the user can ask for the whole answer list on a basis of hit. In this case, our hit comes from the Gcount attribute in Gclass. The more value in Gcount is, the more that object is queried.

Using the example of Gclass and Uclass described in section 4.7, the above algorithm gains the following results:

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

Example 2: Suppose a client at HatYai queries store.psu to its local name server "hatyai.psu", the ANS returns {obj1, obj2}. Therefore, the FilteringProcess is invoked and the final result is **obj1** because obj1 has the same location as the client.

5. DISCUSSION

The SNS is based on TCP/IP protocol and provides facility for mapping one name to many objects. This feature fulfils a need of many organizations with many branches to maintain the unity of its organization and ease anyone to remember one name regardless of what the objects are. As a result, we expect that the SNS can be implied to use not only in any organizations but also in Internet such a referring to any web pages that may use the same name but different locations and FTP applications.

The structure of SNS employs both sets and trees. The property of tree helps the organization remains scalability. This means the increasing of the number of objects remains effective. Furthermore, the property of set provides one name can be referred to many objects.

The filtering mechanism guarantees users to obtain the right objects. The time complexity of the filtering algorithm is O(n log n).

The SNS message protocol that uses Unicode characters is designed to serve a need of naming objects both ASCII characters and non-ASCII characters. This property expands to encompass various languages. The SNS queries and responses are contained within TCP packets to guarantee that the data delivered between connections does not be lost, especially the SNS responsed data.

6. CONCLUSION

Names are used globally and name services are fundamental services to all computer networks. Various naming systems have been designed but most of their structures are hierarchical where each name maps to one object. This paper proposes SNS architecture to facilitate one name mapping many objects and obtains one result.

With the assumption that an organization may have many branches and use the same name for the same task in every branch maintains its unity. The current name services are implemented in the hierarchical structure. This structure contains a simple parent-child relationship which provides a unique name. This restriction guarantees 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. So, names at the leaf nodes may not represent individual objects but a set of objects. The structure satisfies that one name can be shared. However, an exact object returns to a client is considered. A resolution process including a filtering process based on user location or created date and hits, has been proposed to find a required object.

We also consider to extensiveness of the resolution process which will increases traffic. So, caches maintained in the name servers will be used to reduce the traffic. Finally, using the proposed SNS message protocol, all limitations and drawback that mentioned in section 3 has been completely solved.

References

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

[2] W. Y. Arms. "Uniform Resource Names, Handles, PURLs, and Digital Object Identifiers", **Communications of the ACM**, Vol. 44, No.5, May 2001, p. 68.

[3] A.D. Birrell, R. Levin, M.D. Schroeder, and R. M. Needham, "Grapevine", **Communications of the ACM**, Vol. 25, No. 4, April 1982, pp. 260-274.

[4] Coulouris, G., J. Dollimore, and T. Kindberg, "**Distributed Systems Concepts and Design**", Addison Wesley Publishers Limited, 3rd Edition, 2001.

[5] L. Fenner, "Fixing a flawed domain name system", **Communications of the ACM**, Vol. 44, No. 1, January 2001, pp. 19-21.

[6] C. Hunt, **TCP/IP Network Administration**. Sebastopol, O'Reilly & Associates, Inc., 2nd Edition, 1998.

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

[8] PCNA staff. "Understanding NDS", **PC Network Advisor**, Issue 81, March 1997, pp. 9-12.

[9] P. Mockapetris. "Domain Names - Concepts and Facilities, RFC 1034", **http://www.ietf.org/rfc/rfc1034.txt**, November, 1987.

[10] P. Mockapetris, "Domain Names - Implementation and specification, RFC1035", **http://www.ietf.org/rfc/rfc1035.txt**, November 1987.

[11] P. Mockapetris. "Development of the Domain Name System", **SIGCOMM'88, Symposium Communications Architectures and Protocols**, August 1988, pp. 16-19.

[12] D. C. Oppen and Y. K. Dalal. "The Clearinghouse : A Decentralized Agent for Locating Named Objects in a Distributed Environment", **ACM Transactions on Office Information Systems**, Vol. 1, No.3, July 1983, pp. 230-253.

[13] R. Ramsey. **All about administering NIS+**, Mountain View, California: Sunsoft, 2nd Edition, 1994.

[14] W.R. Stevens. **TCP/IP Illustrated Volume 1: The Protocols**, Addison-Wesley, 2000.

[15] S.X. Sun and L. Lannom, "Handle System Overview", *http://hdl.handle.net/4263537/4069*, August 2001.

[16] A.S. Tanenbaum and M.v. Steen. **Distributed Systems: Principles and Paradigms**, Prentice Hall, Inc., 2002.

[17] R. Wieringa and W. de Jonge. "Object identifier, keys, and surrogates - object identifiers revisited", **In Theory and Practice of Object Systems**, Vol. 1, No.2, 1995, pp. 104-114.

APPENDIX C

This section presents a following conference paper which is a part of this dissertation.

- Preechaveerakul. L. and Bhattarakosol. P., "Is It Possible to Use One Name for Many Sites?", in *Proc. of International Conference on Internet Computing (IC'04)*. Las Vegas. Nevada. USA. June 21-24. pp. 360-365. 2004.

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON INTERNET COMPUTING

IC'04

Volume I

Editors:
Hamid R. Arabnia
Olaf Droegehorn

Associate Editors:
Zhixiang Chen, Ping-Tsai Chung,
Hoda El-Sayed, Mao Lin Huang, Peter Langendoerfer,
Joan Lu, Xiannong Meng, Youngsong Mun, Anthony Scime

Las Vegas, Nevada, USA
June 21-24, 2004
©CSREA Press

Volume I,	ISBN: 1-932415-44-0
Volume II,	ISBN: 1-932415-45-9
Set,	ISBN: 1-932415-46-7

Printed in the U.S.A.

Is It Possible to Use One Name for Many Sites?

L. Preechaveerakul
Department of Mathematics, Faculty of Science
Chulalongkorn University, Thailand 10330

P. Bhattarakosol
Department of Mathematics, Faculty of Science
Chulalongkorn University, Thailand 10330

Abstract - *Names play an important role in all computer systems to resolve the objects. A naming system is necessary to implement. Although the existing naming system are implemented both in distributed and non-distributed systems, most of their structure are hierarchical. The structure obtains 3 limitations and drawback: uniqueness, scalability, and anonymity. In real world, the organizations with many branches sometimes require the situation called one name – many objects – one result for its unity. However, most of the current naming systems do not serve it. Therefore, this paper proposes a new naming system called Sharable Name System that can be solved the limitations and drawback and maintains unity of the organizations. With this proposed solution, the extension of network systems will not be limited.*

Keywords Distributed system, naming system

1. Introduction

Names have several purposes in any systems. An important purpose is to facilitate referring the objects. Names are used to share resources, to uniquely identify entities, to refer to locations, and so on [12]. In the computer world, a vast interconnected collection of host computers called Internet, needs names to refer to the addresses. An open protocol standards, TCP/IP protocols [3, 10] are comforted machines from anywhere to communicate. Standard and non-standard application protocols have been developed for services especially, name service which is a fundamental service to all computer networks. A variety of name service has been developed to map physical resources to logical names. Global Name Service (GNS) [4], Handle System [11], Network Information Service Plus (NIS+) [9], Novell Directory Service(NDS) [1, 5] and Domain Name System (DNS) [6, 7, 8] have hierarchical name space. We summarize the features of each name service in Table 1 and found their limitations.

We conclude limitations and drawback of them in section 2. We propose the new naming system called Sharable Name System (SNS) which address limitations and drawback of current name services in section 3. This paper contributes for the name service that can be shared: one name to many objects. In section 4, we discuss our work. Finally, we give our conclusion.

2. Limitations and Drawback of Various Name Services

Since the structure of each name service referred in section 2 is hierarchical. Most of them have the limitations and drawback that we categorize into 3 groups: uniqueness, scalability and anonymity.

Uniqueness

The structure of each name service is hierarchical: therefore each name in a tree must be unique

Scalability

Most of the existing naming systems were designed for people from a single language (English), not other languages.

Anonymity

It is sometimes necessary to say things anonymously [2]. The name service structure is hierarchical so that it is easily traced back to its ancestors in the tree.

In order to solve these limitations and drawback, hence we propose the new naming system called Sharable Name System (SNS).

TABLE 1 FEATURES OF THE EXISTING NAME SERVICES

	Sharing unique name	Scalability	Anonymity
Global Name Service (GNS)	No	Yes/No[1]	No
Handle System	Yes	Yes	No
Network Information Service Plus (NIS+)	No	No	No
Novell Directory Service (NDS)	No	No	No
Domain Name System (DNS)	No	Yes	No

3. The New Naming System

3.1 System Environment

We consider the system environment based on most of organizations that have more than one office locations. For example, Prince of Songkla University (PSU) has 5 campuses located in different provinces that are HatYai, Pattani, Phuket, Surattani, and Trang. Figure 1 presents the structure of PSU and shows that each location consists of 3 divisions: Storehouse (S), Administration (A), and Human Resource (HR) divisions. Although these divisions have the same name, their locations are different. Each division of every location performs the same tasks. For example, the Administration division at HatYai has the same functions as Administration divisions of other 4 campuses. Using the same name for the same task helps everyone clearly understand and easily remembering, and also helps PSU maintains unity of its organizations. Therefore, the system environment that will be taken into the consideration of this research is an organization that consists of many branches with the same name and every same type of document uses the same document name as well. This situation is called "one name - many objects - one result" (ONMOOR).

3.2 Problem Statement

Solving the problem of "one name - many objects - one result" to serve the user needs is a challenging problem. Considering the situation of a person who need to contact a HR division of PSU or looking for a student information of PSU, the problem occurred is "which location

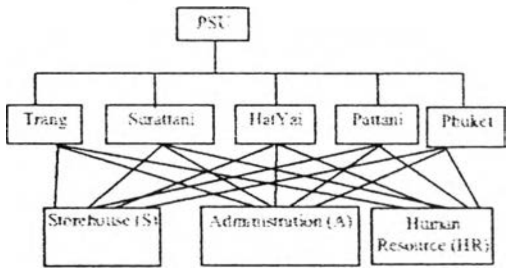


Figure 1 Prince of Songkla University and campuses.

(or document) is the right place (or document) for that person?". Using a computer searching system like google.com, the user will obtain more than one answers at a search result, and the user may not be satisfied according to time consuming in searching for the right one.

Because of many naming systems are hierarchical structure and each node must be named differently. Therefore all existing naming systems cannot solve the problem stated above.

3.3 The Structure of SNS

The global name space is a hierarchical structure with a single root at the top. Each name is a path in the global name space. The objects are the global names that point to information about individual objects. A hierarchical structure of the SNS employs the concept of sets. Figure 2 illustrates the logic organization of the SNS in the sense of a global name able to be shared. A single root at the top in Figure 2 is named as "PSU" and other nodes in the tree are global names. A global name may logically identify not only a single node, but nodes in the tree. Each global name may point to information about individual object or a set of

[1] Not scalable in large scale network when merging and moving directory tree

information about individual object. This means that the SNS maps each global name to a set of objects. For example, in Figure 2, the new structure provides a sharable name such a *store.psu* from the left branch would map storehouse at HatYai (obj1) and *store.psu* at the right branch maps storehouse at Pattani (obj2). However, if we call *store.psu*, the result provides a set of objects: {obj1, obj2}.

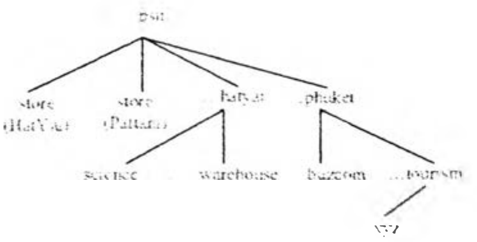


Figure 2. The organization of SNS.

3.4 The SNS Message Protocol

TCP/IP protocol suite contains standard application protocols for services. However, the existing standard protocols do not serve a need for sharing names. Therefore, we have designed a new protocol called SNS message protocol. The SNS queries and responses are often contained within TCP packets. Communication inside the SNS protocol is carried in a single format called an SNS message. All messages both queries and responses are divided into 3 sections: Header, Question, and Answer section as shown in Figure 3.

The header section contains fields specified whether a message is a query or response, and also specified which of the remaining sections are presented. The ID field is an 8-bit identifier. The fg is a 1-bit field that specifies whether this message is a query (0) or a response (1). The Opcode is a 3-bit field that specifies a kind of query in this message.

The RCODE is a 4-bit field with the return code. The values have the following interpretation:

0 No error condition

1 Format error name server cannot interpret the query.

2 Server failure name server cannot process the query.

3 Not implemented name server does not support the requested query type.

4-15 Reserved for future use.

The next two 16-bit fields specify the number of questions and the number of answers respectively. For a query, the number of question is normally 1 and the other field is 0. For a reply, the number of answer is at least 1.

The format of the question section is a query name. The query name is a global name being look up. It is a sequence of one or more labels. The query name field (QName) is stored in Unicode. Each Unicode character is encoded in UTF-8 (UCS Transformation Format). Each label is a length and a number of byte per character followed by the number of characters. The name terminated with a byte of 0. The following examples of global names show how they are stored.

A global name "store.psu" is stored as follow:

5.1	s	t	o	r	e	3.1	p	s	u	0

The other global name is สนุก.[illegible] stored as follow:

4.3	ส	น	ุ	ก	3.3	[illegible]	[illegible]	ะ	0

The last example shows a Thai global name which uses non-ASCII characters.

The global name that is able to stored in Unicode expands to encompass various languages not only English. Therefore, the limitations of scalability as we mentioned previously has been solved.

The answer section may contain one or more records called the SNS records. The number of records is specified in the corresponding field in the header section. Each SNS record contains name, time-to-live, SNS data length and SNSDATA field. Figure 4 describes format of an SNS record.

The name field is a global name which SNSDATA corresponds. The name field has a same format as described in query name field.

The time-to-live field (TTL) is a 16-bit field that specifies the time interval (in seconds) that the SNS record may be cached before it should

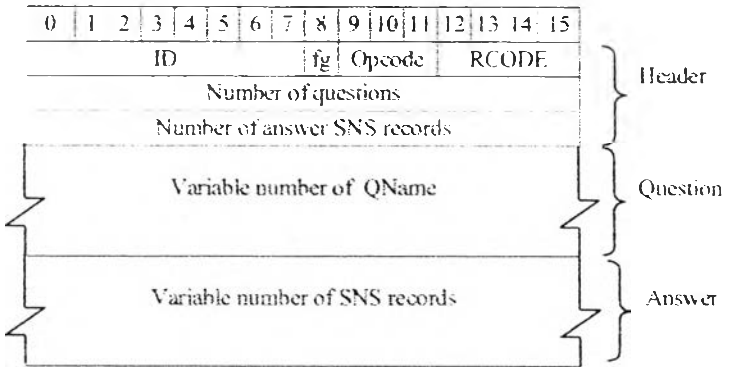


Figure 3 The SNS Message Format

be discarded. The SNS data length specifies the amount of SNS data.

The SNSDATA is a variable length string that describes the global name varies according to a set of objects including the two important attributes: location and created Date.

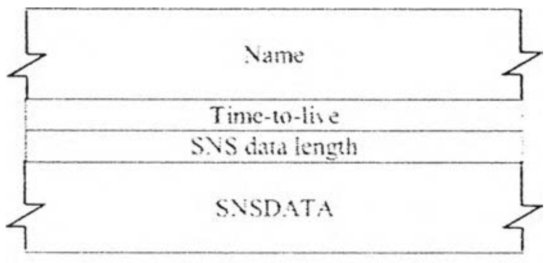


Figure 4 Format of SNS record.

3.5 Query and Response example

The following query and response illustrates name server behavior. When a client queries a global name "store.psu" to a name server, the question section is stored store.psu. The query message format may look like:

Header	Opcode = query
Question	QName = store.psu
Answer	<empty>

The response may be:

Header	QR = Response. Opcode = query
Question	QName = store.psu
Answer	store.psu 86400 18 obj1 HatYai 13012001 store.psu 86400 19 obj2 Pattani 05032002

3.6 Active Components

SNS is a client/server model. The active components are name servers and resolvers. Resolvers are the clients that access name servers. Normally, a resolver is a part of an application. A resolver queries a name server to map names to objects. The concepts of sets and trees are provided for name to object translation. Each tree contains nodes which resolve to a set of objects or may refer to other servers. A path through the tree terminates at a node. The set of objects contained that in the node is returned as the result of the path. The querying type should be one of the following types: *full path* such as a global name: *store.psu* or a combination of global names: *store.psu.warehouse.hatyai.psu*, and some parts of the path such as *store.psu: local.hatyai*. The path *store.psu:warehouse. hatyai.psu* is a set of paths.

Before we describe the mechanism of resolution process, let us take a closer look at the global name properties.

3.7 Global Name Properties

We use names to indicate the objects, and may refer as documented files, images, addresses, etc. Each name has its properties and certainly has an identity. Our work proposes a sharable name for objects, so that identifying the right object is very important issue. Therefore, we define a global class (Gclass) to store the information of each global object. Additionally, user class (Uclass) is defined to store some information such as location. We also show a relationship between the global class and the user class in Figure 5.

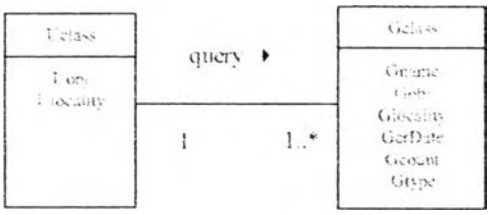


Figure 5 Class diagram of User and Global and their relationship.

Uclass stores one of the important attributes called Ulocality which is a value of client location. Gclass stores Gname, Gobj, Glocality, GcrDate, and Gtype. Gname is a global name. Gobj is an object that related to Gname. Glocality is a value of a global object location. Every time a global name is queried, Gcount is increased by 1. Gcount will be set to 0 at the first time Gname is created or when time-to-live (TTL) is 0 or 1. Gtype attribute specifies whether a global name is a leaf node (Lf) or a node (Nd) in the global name space. Normally, a Ulocality attribute in Uclass comes from a location of a local name server that a client queries a global name. Each name server will store database of Gclass. Every time a new object is created, its information will be stored after registration process.

The example below refer to Figure 5.

Example : a local name server is psu and stores Gclass.

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

Gname	Gobj	Glocality	GcrDate	Gcount	Gtype
store.psu	obj1	HatYai	13-01-2001	5	Lf
store.psu	obj2	Pattani	05-03-2002	2	Lf
hatyai.psu	obj3	HatYai	12-11-1999	8	Nd
⋮					
phuket.psu	objx	Phuket	30-09-2000	6	Nd

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

Uobj	Ulocality
Uobj1	HatYai

3.8 Resolution Process

To find the exact object, a resolution process has been constructed. When a client queries a name $(q_1 q_2 \ldots q_n)$ which may be one of the previously querying types, at a local name server, and the local name server does not know the answer, a process to decompose the query into subqueries $(q_1, q_2, \ldots, q_n)$ is invoked. Other 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) either a unique object or a set of objects. If ANS is a unique object, a local name server returns ANS to a client. If ANS is a set of objects, a local name server invokes a filtering process to guarantee a client to receive the right unique object based on a location or a latest created date.

The algorithm of FilteringProcess describes below:

```
1  Use Quicksort algorithm to sort ANS ({Obj1, Obj2, ..., Objn}) based on counting factors:
   location and created date, respectively.
2. A local name server takes a client location comparing to the location of each object in
   sorted ANS.
3  If one object (Obji) in sorted ANS has a same location as a client
   then
         Return Obji to a client
   else if there are a set of objects ({Obj1, Obj2,
              ..., Objn}) or no object in sorted ANS
         has a same location as a client
         then
              For each object in sorted ANS
                   Find obji with a latest created date
                   Return obji to a client.
```

This algorithm requires the expected time in O(n + n log n). So that, the complexity is O(n log n). The reason for choosing the latest created date object is based on the human nature that loves to get the new and up-to-date object. However, if the answer after filtering does not satisfy the client, the user can ask for the whole answer list on a basis of hit. In this case, our hit comes from the Gcount attribute in Gclass. The more value in Gcount is, the more that object is queried.

Using the example of Gclass and Uclass described in section 3.7, the above algorithm gains the following results:

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

4. Discussion

The SNS is based on TCP/IP protocol and provides facility for mapping one name to many objects. This feature fulfils a need of many organizations with many branches to maintain the unity of its organization and ease anyone to remember one name regardless of what the objects are. As a result, we expect that the SNS can be implied to use not only in any organizations, but also in Internet such a referring to any web pages that may use the same name but different locations and FTP applications.

The structure of SNS employs both sets and trees. The property of tree helps the organization remains scalability. This means the increasing of the number of objects remains effective. Furthermore, the property of set provides one name can be referred to many objects. The filtering mechanism guarantees users to obtain the right objects. The time complexity of the filtering algorithm is O(n log n). The SNS message protocol that uses Unicode characters is designed to serve a need of naming objects both ASCII characters and non-ASCII characters. This property expands to encompass various languages. The SNS queries and responses are contained within TCP packets to guarantee that the data delivered between connections does not be lost, especially the SNS responsed data.

5. Conclusion

The current name services are implemented in hierarchical structure. This structure contains a simple parent-child relationship which provides a unique name. Names at the leaf nodes of the tree represent individual objects. Thus, it limits names to be shared. This paper proposes SNS architecture to facilitate one name mapping many objects and obtains one result.

The new structure, on the other hand, employs both the concepts of sets and trees. So, names at the leaf nodes may not represent individual objects but a set of objects. The structure satisfies that one name can be shared. However, an exact object returns to a client is considered. A resolution process including a filtering process based on user location or created date and hits, has been proposed to find a required object.

We also consider to intensiveness of the resolution process which will increases traffic. So, caches maintained in the name servers will be used to reduce the traffic. Finally, using the proposed SNS message protocol, all limitations and drawback that mentioned in section 3 has been completely solved.

References

[1] Chris Andrew, Ed Shropshire et al "Novell NDS Developer's Guide", Novell Press, CA, 1999.

[2] Lenny Foner, "Fixing a flawed domain name system," *Communications of the ACM* 44 (1), 19-21, January 2001.

[3] Craig Hunt, "TCP/IP Network Administration," O'Reilly & Associates, Inc., CA, USA, 2nd Ed., 1998.

[4] Butler W. Lampson and DEC Systems Research Center, "Designing a Global Name Service," *In Proceedings 4th ACM Symposium on Principles of Distributed Computing*, Ontario 1986 pp. 1-10.

[5] PCNA staff, "Understanding NDS," PC Network Advisor, 81 (9-12), March 1997.

[6] Paul Mockapetris, "Domain Names - Concepts and Facilities, RFC 1034," *http://www.ietf.org/rfc/rfc1034.txt*, November 1987.

[7] Paul Mockapetris, "Domain Names - Implementation and specification, RFC 1035," *http://www.ietf.org/rfc/rfc1035.txt*, November 1987.

[8] Paul Mockapetris, "Development of the Domain Name System," *SIGCOMM '88, Symposium Communications Architectures and Protocols*, pp. 16-19, August 1988.

[9] Rick Ramsey, "All about administering NIS+," Mountain View, California : Sunsoft, 2nd Ed., 1994.

[10] W.Richard Stevens, "TCP/IP Illustrated Volume 1: The Protocols," Addison-Wesley, 2000.

[11] Sam X. Sun and Laurence Lannom, "Handle System Overview," *http://hdl.handle.net/4263537/4069*, August 2001.

[12] Roel Wieringa and Wiebren de Jonge, "Object identifier, keys, and surrogates - object identifiers revisited," *In Theory and Practice of Object Systems*, 1(2): 101 - 114, 1995.

APPENDIX D

This section presents a following conference paper which is a part of this dissertation.

- Preechaveerakul, L. and Bhattarakosol, P., "One Name - Many Objects - One Result: A Possible Solution for Naming System", in *Proc. of the 3rd International Symposium on Information and Communication Technologies*, Las Vegas, Nevada, USA, June 16-18, pp. 63-67, 2004

Proceedings of the International Symposium on

Information and Communication Technologies

Las Vegas, Nevada. June 16th-18th, 2004

Edited by:
Andreas Ahrens
Markus Aleksy
Ralf Gitzel
Thomas Kessler
Christoph Lange
Zakaria Maamar
Qusay H. Mahmoud
Hans-Dietrich Melzer
C. V. Ramamoorthy
Remzi Seker
A. J. Van der Merwe
Thomas Vergin
John Waldron
Song Y Yan

Published by the Computer Science Press
Trinity College Dublin, Ireland

ISBN: 0-9544145-5-1

A VOLUME IN THE ACM INTERNATIONAL CONFERENCE PROCEEDINGS SERIES

One Name – Many Objects – One Result: A Possible Solution for Naming System

L. Preechaveerakul
Department of Mathematics
Faculty of Science
Chulalongkorn University, Thailand
ladda.pr@student.chula.ac.th

P. Bhattarakosol
Department of Mathematics
Faculty of Science
Chulalongkorn University, Thailand
bpattara@sc.chula.ac.th

Abstract

Names are usually used to referring the objects. In the computer world, name services are fundamental services to all computer networks. Several name services have been developed to map physical resources to logical names. Most of current name services are hierarchical structure. Therefore, each name in a tree must be unique. In the real world, sometimes we want to share a name especially, in the organization with more than one office locations. This paper proposes a new naming system which supports the sharable name.

1. Introduction

Names have several purposes in any systems. An important purpose is to facilitate referring the objects. Name services are fundamental services that map physical resources to logical names. There are several name services have been developed. Global Name Service (GNS) [1], Handle System [6], Network Information Service Plus (NIS+) [5], Novell Directory Service (NDS) [4] and Domain Name System (DNS) [2, 3] have hierarchical name space. A name is a path in a tree and the nodes with a same parent must have different names. The hierarchical structure with a simple parent-child relationship provides a unique name. This structure enables a system to grow indefinitely. However, it restricts each name to be unique.

We describe limitation and drawback of the existing naming system in section 2. We propose the new naming system called "Sharable Name System" and discussion in section 3 and 4, respectively. Finally, we give future work in section 5 and conclusion in section 6.

2. Limitation and Drawback of Various Naming Systems

We categorize the limitation and drawback into uniqueness and anonymity.

- *Uniqueness*: the structure of each naming system is hierarchical; therefore each name in a tree must be unique.

- *Anonymity*: sometimes it is necessary to say things anonymously but the hierarchical structure of naming system is easily traced back to its ancestors in the tree.

Hence we propose the new naming system called Sharable Name System (SNS).

3. Sharable Name System (SNS)

3.1 System Environment

We consider the system environment based on most of organizations that have more than one office locations. For example, Prince of Songkla University (PSU) has 5 campuses located in different provinces that are HatYai, Pattani, Phuket, Surattani, and Trang. Figure 1 presents the structure of PSU and shows that each location consists of 3 divisions: Storehouse (S), Administration (A), and Human Resource (HR) divisions. Although these divisions have the same name, their locations are different. Each division of every location performs the same tasks. For example, the Administration division at HatYai has the same functions as Administration divisions of other 4 campuses. Using the same name for the same task or the same object helps everyone clearly understand and easily remembering, and also helps PSU maintains unity of its organizations. Therefore, the system environment that will be taken into the consideration of this research is an organization that consists of many branches with the same name and every same type of document uses the same document name as well. This situation is called "one name – many objects – one result " (ONMOOR).

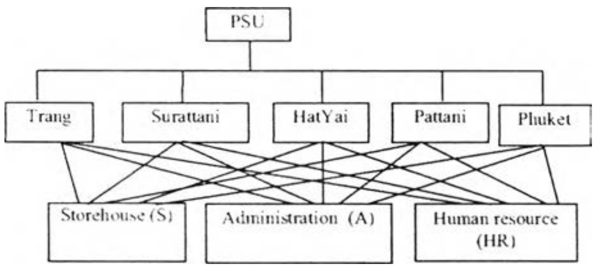


Figure 1 Prince of Songkla University and campuses.

3.2 Problem Statement

Solving the problem of "one name – many objects – one result" to serve the user needs is a challenging problem. Considering the situation of a person who needs to contact a HR division of PSU or looking for student information of PSU, the problem occurred is "which location (or document) is the right place (or document) for that person?". Using a computer searching system like google.com, the user will obtain more than one answers at a search result, and the user may not be satisfied according to time consuming in searching for the right one.

Because of many naming systems are hierarchical structure and each node must be named differently. Therefore all existing naming systems cannot solve the problem stated above.

3.3 Structure and Resolution Process of SNS

The SNS is a client/server architecture. The concepts of sets and trees are provided for name to object translation. A global name space is hierarchy. Each name is a path in the global name space. A path through the tree terminates at a node. Each node may resolve to a set of objects or refer to other servers. The querying type should be one of the following types: *full path* such as a global name; *store.psu* or a combination of global names; and some parts of the path such as *store.psu:loc@hatyai*. The path *store.psu:warehouse.hatyai.psu* is a set of paths. Figure 2 illustrates the logic organization of SNS in the sense of a global name able to be shared. The new structure provides a sharable name such a *store.psu* from the left branch would map storehouse at HatYai (obj1) and *store.psu* at the right branch maps storehouse at Pattani (obj2). However, if we call *store.psu*, the result provides a set of objects: {obj1, obj2}.

To find the exact object, a resolution process has been constructed. When a client queries a name (q_1:q_2: ...:q_n) which may be one of the previously querying types, at a local name server, and the local name server does not know the answer, a process to decompose the query into subqueries (q_1, q_2, ..., q_n) is invoked. Other name servers are queried to determine a mapping from each subquery name to a set of objects. When each subquery returns, a

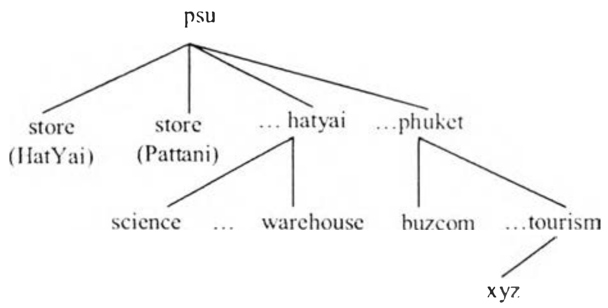


Figure 2 The organization of SNS.

process to intersect all subqueries is invoked and gives the answer (ANS) either a unique object or a set of objects. If ANS is a unique object, a local name server returns ANS to a client. If ANS is a set of objects, a local name server invokes a filtering process to guarantee a client to receive the right unique object based on a location or a latest created date.

Before we describe the algorithm of filtering process, we define a global class (Gclass) to store the information of each global object. Additionally, a user class (Uclass) is defined to store some information such as location. The relationship between Gclass and Uclass is shown in Figure 3.

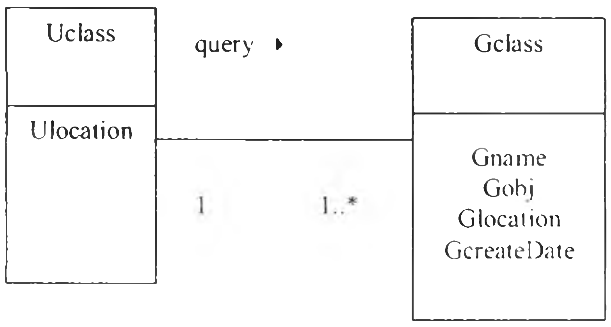


Figure 3 Class diagram of User and Global and their relationship.

The following example refers to Figure 3.
Example: a local name server is psu and stores Gclass.

Suppose that Gclass contains 4 attributes: <Gname, Gobj, Glocality, GcreateDate>. The following table shows the data stored in Gclass.

Gname	Gobj	Glocation	GcreateDate
store.psu	obj1	HatYai	12-02-2000
store.psu	obj2	Phuket	23-09-2001
hatyai.psu	obj3	HatYai	01-10-2001
. . .			
phuket.psu	objx	Phuket	30-04-2000

Additionally, Uclass contains two attributes: <Uobj, Ulocation> and the example is shown below:

Uobj	Ulocation
Uobj1	HatYai

The algorithm of filtering process describes below:

1. Use Quicksort algorithm to sort ANS (*{Obj₁, Obj₂, ..., Obj_k}*) based on counting factors: location and created Date, respectively.
2. A local name server takes a client location comparing to the location of each object in sorted ANS.
3. If one object *(Obj_i)* in sorted ANS has a same location as a client

```
Then
    return Obj$_i$ to a client
Else if there are a set of objects ({Obj$_1$, Obj$_2$, .... Obj$_k$}) or no object in sorted ANS has a
      same location as a client
   Then
       For each object in sorted ANS
           Find obj$_j$ with a latest created date
           Return obj$_j$ to a client.
```

The reason for choosing the latest created date object is based on the human nature that loves to get the new and up-to-date object. However, if the answer after filtering does not satisfy the client, the user can ask for the whole answer list on a basis of recent created date.

Using the example of Gclass and Uclass described previously, the above algorithm gives the results:
Example 1: Suppose a client at psu queries store.psu to its local name server "psu", the ANS returns {obj1, obj2}. Thus, the FilteringProcess is invoked and the final result is **obj2** since no object in ANS has the same location as the client and the algorithm chooses the latest created date of the object in ANS and returns only one object back to the client.

Example 2: Suppose a client at HatYai queries store.psu to its local name server "hatyai.psu", the ANS returns {obj1, obj2}. Thus, the FilteringProcess is invoked and the final result is **obj1** the same location as the client.

4. Discussion

Based on TCP/IP protocol, the SNS provides facility for mapping one name to many objects. This property fulfils a need of each organization with many branches to maintain their unity and ease anyone to remember one name regardless of what the objects are. Furthermore, the structure of SNS that employs both sets and trees helps the organization remain scalability. This means that the increasing of the number of objects remains effective and the filtering process guarantees users to obtain the right objects.

5. Future Work

We are going to use mathematical theory to prove this proposed solution and simulate the SNS client/server program.

6. Conclusion

With the assumption that an organization has many branches and using the same name for the same task with every branch makes the organization maintain its unity. The current name services are implemented in the hierarchical structure. This structure contains a simple parent-child relationship which provides a unique name. This restriction guarantees that a name uniquely identifies a single leaf in the tree. Names at the leaves of the tree represent individual objects. Therefore, it limits names to be shared. Sharable Name System (SNS), on the other hand, provides each name enable to be shared. SNS employs both sets and trees. The organization with many branches and using the same name for the objects located in different locations retains the unity of its organization. As a result, we expect that SNS can be implied to use not only in any organizations, but also in Internet especially for referring to any web pages that may use the same name but different locations and FTP applications.

References

[1] Lampson B. w., DEC Systems Research Center (1986): "Designing a Global Name Service"; Proceedings 4th ACM Symposium on Principles of Distributed Computing, Ontario. pp. 1-10.

[2] Mockapetris P. (1987): "Domain Names - Concepts and Facilities. RFC 1034". *http://www.ietf.org/rfc/rfc1034.txt.*

[3] Mockapetris P. (1987): "Domain Names - Implementation and specification. RFC1035". *http://www.ietf.org/rfc/rfc1035.txt.*

[4] PCNA staff (1997): "Understanding NDS"; PC Network Advisor, Issue 81, pp. 9-12.

[5] Ramsey R. (1994): "All about administering NIS+". Mountain View, California : Sunsoft. Second Edition.

[6] Sun S.X. and Lannon L. (2001): "Handle System Overview", *http://hdl.handle.net/4263537/4069.*

APPENDIX E

This section presents a following conference paper which is a part of this dissertation.

- Preechaveerakul, L., O'Brien, F., Castro, M., and Bhattarakosol, P., "Intelligent DNS-based Naming System Mechanism", in *Proc. of the 1st Workshop on the Internet, Telecommunications and Signal Processing (WITSP' 02)*, Wollongong, Australia, December 9 -11, pp. 123-127, 2002.

PROCEEDINGS

OF THE FIRST

WORKSHOP ON THE INTERNET, TELECOMMUNICATIONS AND SIGNAL PROCESSING

WITSP'02

SUPPORTED BY

New South Wales Section of IEEE

South Australian Chapter of Signal Processing and Communications Societies

SPONSORED BY

Smart Internet Technology Cooperative Research Centre

School of Electrical, Computer and Telecommunications Engineering, University of Wollongong

Intelligent DNS-based Naming System Mechanism

Ladda Preechaveerakul
pladda@a serc rmit edu.au
Department of Mathematics, Faculty of Science, Chulalongkorn University, Thailand
Professor Fergus O'Brien, Dr Maurice Castro
Software Engineering Research Centre, RMIT, Australia
Assistant Professor Dr Pattarasinee Bhattarakosol
Department of Mathematics, Faculty of Science, Chulalongkorn University, Thailand

Abstract

The Internet currently connects millions of hosts around the world. The domain name system (DNS) translates nearly all human-readable names to IP addresses. Since its structure was designed to be hierarchical, host names must be unique. The size and nature of the Internet community has changed from a homogenous culture and common language to various linguistic, cultural and educational backgrounds. Supplements to the existing solutions do not solve cybersquatting and trade name issues. We address some of the failings of the DNS and outline a new Naming System that solves some of the problems of the existing system.

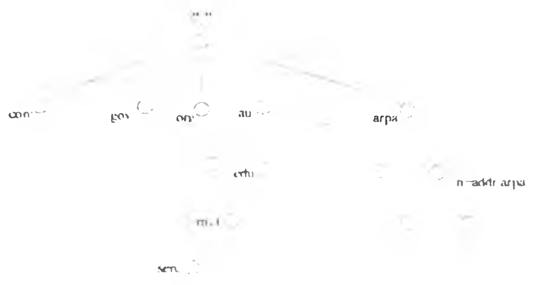


Figure 1: The organisation of the DNS

1 Introduction

The Internet connects millions of host computers around the world. The provision of a name service is a fundamental service in all computer networks. Since computers communicate with each other on the basis of host addresses, the wide variety of network applications such as electronic mail, www, and remote login need easily remembered names. Thus, there is a need to map names to addresses and the architecture called Domain Name System (DNS) for name translation has been constructed. This paper starts by taking an overview of the Internet Domain Name System to describe the background, the functionality and role of the Domain Name System in the Internet [1, 14]. The limitations and failings of Domain Name System [4] are discussed. Finally, we propose a name service which addresses some of the failings of the DNS.

2 An Overview of the Internet Domain Name System

2.1 Historical Development

In the late 1960s, the US Department of Defense's Advance Research Projects Agency, ARPA (later DARPA), began funding an experimental wide area computer network that connected important research organizations in the US, called the ARPAnet. Its goal was to allow government contractors to share expensive or scarce computing resources. From the beginning, users of the ARPAnet used the network for sharing files and software and exchanging electronic mail, then using shared remote computers. Through the 1970s, the *hosts.txt* file, mapped every name to address for every host connected to the ARPAnet and was maintained by SRI[1]'s Network Information Center (NIC). The size of *hosts.txt* grew in proportion to the growth in the number of ARPAnet hosts. When the population of the network exploded, the file based *hosts.txt* mechanism was unable to cope with rapid addition of hosts, name collisions, and consistency of the name space. These problems show that the *hosts.txt* mechanism didn't scale well.

In 1984, the Domain Name System was designed by Paul Mockapetris superseded *hosts.txt*.

2.2 Domain Name System

Domain Name System (DNS) is a distributed database based on the TCP/IP protocol. The DNS maps human-readable host names to numerical IP-addresses. The database structure is strictly hierarchical. A name such as serc.rmit.edu.au has a structure as shown in figure 1.

Each domain name is a path in a tree, called the domain name space. The tree has a single root at the top called "the root". Every node has a label of up to 63

[1]SRI: the Standford Research Institute in Menlo Park, California

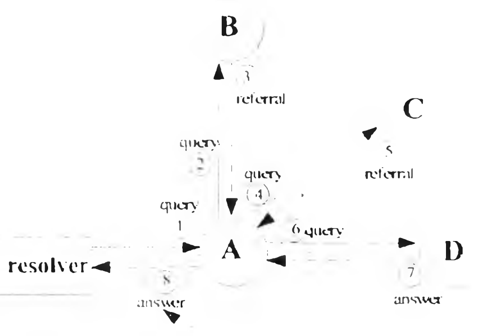


Figure 2: The Resolution Process

characters except the root and must have a unique domain name. A domain name that ends with a period is called an absolute domain name or a fully qualified domain name (FQDN) for example: serc.rmit.edu.au.

The existing domain name space is categorized into 2 types of top-level domain (TLD): generic (gTLD) and geographic or country-code (ccTLD). The information about the domain name space is stored in programs called name servers.

Resolvers are library routines called by programs to look up names. A resolver performs name resolution or asks a name server to do it by querying name servers to determine a mapping from a name to an address. Queries may be recursive, in which case the server must return the resolution response into a result, or non-recursive, in which case the server returns only the information it knows.

A generic resolution example is shown in figure 2 with a brief explanation

1. Name server A receives a query from the resolver.
2. Name server A queries Name server B.
3. Name server B refers name server A to other name servers, including C.
4. Name server A queries name server C.
5. Name server C refers name server A to other name servers, including D.
6. Name server A queries name server D.
7. Name server D answers.
8. Name server A returns answer to resolver

3 Limitations and Failings of the Domain Name System

Since the DNS structure is hierarchical, each host mapping to each IP address needs to be unique in the tree. There are 3 groups of limitations and failings in the DNS: uniqueness, scalability and anonymity.

Uniqueness

Legal and social problems have occurred. Intellectual property has been disputed. In this case, if the domain names chosen by a person seem related to the trade names or a company, the person may not be able to use them at all. The had faith registration of a domain name, called cybersquatting, also occurred. The popular address space has been exhausted in a very short time. There are many products and companies that legitimately use the same name. There is no systematic way to guess the required domain name and there is no practical way for them to share a name. Although search engines can sometimes assist, they cannot solve every query. The search engines do not cover the entire web. New, poorly referenced sites might as well not exist and not all search engines find all web pages.

Scalability

The size and nature of the Internet community has changed. Originally, it consisted of people from similar cultural backgrounds and a single language (English) and has expanded to encompass various linguistic, cultural and educational backgrounds [6].

Anonymity

It is sometimes necessary to say things anonymously. Domain names are easily traced back to someone higher in the tree. Thus, it is essentially impossible to protect a domain name from retribution while simultaneously advertising its existence to potential correspondents.

4 A Variety of Proposed Existing Solutions

We can divide the existing solutions into 2 groups. Those that supplement the existing DNS service and alternate name services.

4.1 Supplementing the Existing DNS Service

Several approaches have been made to supplementing DNS. First, the Internet Corporation for Assigned Names and Numbers (ICANN) had an agreement to introduce new gTLDs for supporting a wide variety of user groups [11]. Second, some companies such as Name Space or Image Online Design have tried to reconfigure either DNS servers or individual PCs to easily access more TLDs [2]. A new approach called RealNames has been proposed to help novice users avoid both the DNS and search engines altogether. However, none of these approaches resolve the underlying of uniqueness and postpone scalability issue.

Methods such as the single-proxy method, Onion Routing Protocol [8], Crowds [10] and Hordes has been considered to support the anonymity issue [12].

4.2 Alternate Name Services

Several other name servers have been developed. In 1986, Lampson and his colleaques at the DEC Systems Research Center have designed a global name service (GNS) [7] for use in an internetwork which supports a naming database that may extend to include millions of host computers and eventually e-mail addressing for billions of users. A naming database in GNS compose of a tree of directories holding names and values. Each directory is assigned an integer, which serves as a unique directory identifier(DI). GNS successfully addresses the need for scalability and reconfigurability with the exception of the solution adopted for merging and moving directory trees [5]. For example, 2 previously seperate GNS services may be integrated with the introduction of a new root above the two existing roots. The well-known directory tables are used to store the previous directory identifiers and remap to the current real root directory of the naming database. Whenever the real root of the naming database changes, all GNS servers are informed of the new location of the real root. In a large-scale network, reconfiguration may occur at any level. This makes this table grow rapidly, conflicts with the scalability goal.

The Handle System was developed to be a persistent name service and allow secured name resolution and administration over the Internet [13]. Persistent name service means that the system allows name to persist over changes of location, ownership, and other network state conditions.

The Network Information Service Plus (NIS+) from the Sunsoft engineering team [9] and Novell Directory Service (NDS) from Novell. Inc. [3] are being developed their service not only to support in local area network but also supplement and compatability on the Internet. These systems are actively persuing the scalability, but are typically targeted within organisations.

Table 1 summarizes various naming system features.

Furthermore, we conclude each name service and its role and functionality in Table 2.

With these proposed solutions, we found that it couldn't solve the failings of the existing DNS. Table 3 summarize the features of the existing solutions

5 The New Proposal

We are proposing a name service which addresses some of the failing of DNS. We required that the server has:

1. Sharing names: the ability to share unique name equitably among multiple valid claimants.

2. Scalability: it will remain effective when there is a significant increase in the number of resources and the number of users.

3. Decentralize administration via delegation: a domain can divide into subdomain. Each subdomain or delegation is responsible for maintaining all the data.

4. Efficiency: it is a distributed database with hierarchical structure that allows different parts of the naming database to be maintain by different entities. Additionally, caching obviates the query which can't answer locally and need to query other servers.

5. Opacity: the use of a name conceals from the user location in IP-address components in a distributed system. Location opacity enables resources to be accessed without requiring knowledge of their location.

6. Local customisation: partially specified names can use sensible defaults based on user preferences and network setup.

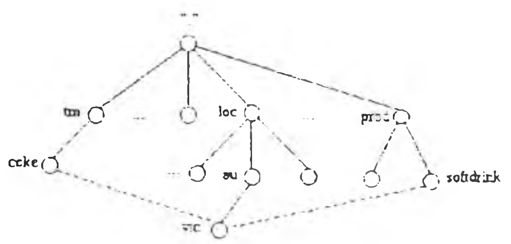


Figure 3: The organization of proposed DNS

The new service employs both the concepts of sets and trees to provide name to address translation. Each tree contains nodes which may refer to other servers or resolve to a set of host IP addresses. A path through the tree terminates at a node. The set of host IP addresses contained in the node is returned as the result of the path. The intersection of the sets from multiple paths yields the name to address translation. For example, the path marketA.vic.au.loc has a structure similar to the original DNS structure. A resolvable domain name is a set of paths such as coke.tm:vic.au.loc:softdrink.prod. Its structure is shown in figure 3.

The dashed line represents the combination of domain names to form a translation. To find the exact IP address, a new resolution mechanism has been constructed. When a client queries name at a local name server, as shown in figure 4, and the local name server does not know the answer, a process to divide the query into subqueries is invoked. Other name servers are queried to determine a mapping from each subquery name to a set of IP addresses. When each subquery returns, the process to intersect all subqueries is invoked and an answer is provided to the client. Figure 4 illustrates the full resolution process with a brief explanation.

1. A client need to know the IP address of

Feature	DNS	GNS	Handle System	NIS+	NDS
Human-readable	Yes	Yes	Yes	Yes	Yes
Unique Object Identifier (OID)	No	Yes	Yes	No	No
Hierarchical name space	Yes	Yes	Yes	Yes	Yes
Location Transparent	Yes	Yes	Yes	Yes	Yes
Support to Internet	Yes	Yes	Yes	No	No
Support to small network	Yes	Yes	Yes	Yes	Yes
Support to large-scale network	Yes	Yes	Yes	Yes	Yes
Scalability	Yes	Yes/No[2]	Yes	Yes	Yes
Security	Originally - No Since 1987 - Yes	Yes	Yes	Yes	Yes
High Availability	Yes	Yes	Yes	Yes	Yes

Table 1: Features of various naming system

Name Service	Role and Functionality
Global Name Service (GNS)	Support a large naming database distributed in an internetwork.
Handle System	A persistent name service for large number of entities and allow each existing local name space join the global handle name space.
NIS+ and NDS	Services in each organization, stored information of any resources where all workstations on network can access it.

Table 2: Name services and their role and functionality

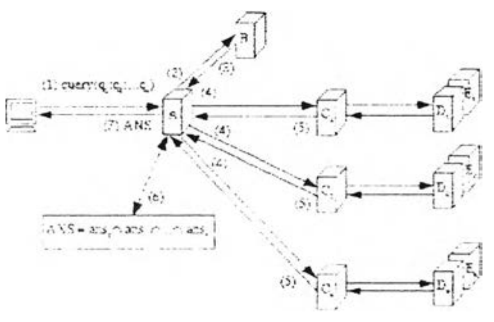


Figure 4: The new resolution process

q_1:q_2:q_3:...:q_n, then it contacts the local name server A

2. Name server A takes each subquery (q_1, q_2, q_3, ..., q_n) to name server B (root name server)

3. Name server B returns the root name server of each subquery to A

4. Name server A queries $q_{1..n}$ to name server $C_{1..n}$

5. Each name server($C_{1..n}$) resolves its subquery by either returning data held on the name server or querying the set of name servers a delegation has been made to. The server returns the union of the sets of IP-addresses found

6. Name server A intersects ans_1, ans_2, ..., ans_n

7. Name server A returns answer to resolver

6 Discussion

A hierarchical structure with a simple parent-child relationships of the existing DNS provides a unique domain name. This restriction guarantees that a domain name uniquely identifies a single leaf in the tree as shown in figure 1. Domain names at the leaves of the tree represent individual hosts. An interior node of the tree can name both a host and point to information about the domain. This structure limits the growth of the Internet community.

The new structure, on the other hand, employs both the concepts of sets and trees. Thus, domain names at the leaves of the tree may not represent individual hosts but a set of host IP addresses. The flexibility of this structure reduces the problems of cybersquatting and trade names issues and still scales. The mechanism reduces cybersquatting and trademark problems by allowing many organisations to share a name - as is the case with trademarks - which are distinguished by context. Cybersquatting is impractical when the monopoly enjoyed by a domain owner under the existing DNS is replaced by a system where names can be shared.

[2] not scalable in large-scale network when merging and moving directory tree

	Sharing unique name	Scalability	Anonymity
Add new gTLD	No	No	No
Reconfigure DNS server or PCs	No	No	No
RealNames	Yes	No	No
Proxy server or anonymous protocol	No	No	Yes
Global Name Service (GNS)	No	Yes/No[3]	No
Handle System	Yes	No	No
Network Information Service plus (NIS+)	No	No	No
Novell Directory Service (NDS)	No	No	No

Table 3: Features of the Existing Solutions

However, the intensiveness of the resolution process increases traffic. Caches maintained in the name servers will be used to reduce the traffic.

7 Future Work

We are currently working on simulating the new system. Following that we will examine some problems with the system such as merging the huge sets associated with geographical names and the handling of languages other than English.

Conclusion

The increasing demand for an Internet is affecting the present DNS used to map host names to IP addresses. The popular names are being exhausted by trademark holders and cybersquatters. Different methods have been developed to supplement the existing DNS but they do not solve the problems of intellectual property, cybersquatting, and name collisions. We propose a new name service that addresses some of the failings of DNS by remaining the uniqueness of name to address mappings.

References

[1] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly & Associates, third edition, 1998.

[2] A. Doman. Can the internet move beyond dot-com? http://www.networkmagazine.com/article /nmg20010226s0008. March 2001.

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

[4] L. Fonner. Fixing a flawed domain name system. *Communications of the ACM*, Volume 44, Number 1, pages 19–21, January 2001.

[5] J. Dollimore G. Coulouris and T. Kindberg. *Distributed Systems Concepts and Design*. Addison-Wesley Publishers Ltd., third edition, 2001.

[6] J. Horvath. What's in a name? http://www.heise.de/tp/english/inhalt/te/5557.1.html. December 1999.

[7] B.W. Lampson and DEC Systems Research Center. Designing a global name service. In *Proceedings 5th ACM Symposium on Principles of Distributed Computing*, pages 1–10, Calgary, Alberta, Canada, November 1986.

[8] P. Syverson M. Reed and D. Goldschlag. Proxies for anonymous routing. *12th Annual Computer Security Applications Conference, IEEE*, pages 95–104, December 1995.

[9] R. Ramsey. *All about administering NIS+*. Mountain View, California : Sunsoft, second edition, 1994.

[10] M.K. Reiter and A.D. Rubin. Crowds. *ACM transactions on Information and System Security(TISSEC)*. Volume 1, Number 1, pages 66–92, November 1998.

[11] Search Engine Report. Goodbye domain names, hello realnames? http://searchenginewatch.com/ sereport/00/05-realnames.html. May 2000.

[12] C. Shields and B.N. Levine. A protocol for anonymous communication over the internet. In *Proceeding 7th ACM conference on Computer and Communication Security*, pages 33–42, November 2000.

[13] S.X. Sun and L. Lannon. Handle system overview http://www.handle.net/overview-current.html. August 2001.

[14] W.R.Stevens. *TCP/IP Illustrated, Volume 1 : The Protocols*. Addison-Wesley, 1994.

[3] not scalable in large-scale network when merging and moving directory tree

Vita

Name: Miss Ladda Preechaveerakul

Date of Birth: 28th June 1967

Education:

- Ph.D. Program in Computer Science, Department of Mathematics, Chulalongkorn University, Thailand (May 2000 - April 2006).
- Visiting Ph.D. researcher at Software Engineering Research Centre (SERC), RMIT University, Melbourne, Australia (October 2001 - September 2002).
- M.S. (Computer Science), School of Applied Statistics, National Institute Development Administration (NIDA), Bangkok, Thailand (May 1991 - January 1994).
- B.Sc. (Mathematics), Department of Mathematics, Faculty of Science, Prince of Songkla University, Hat Yai, Thailand (June 1986 - March 1990).

Working Experience:

- Lecturer at Department of Mathematics, Faculty of Science, Prince of Songkla University, Hat Yai, Songkla, Thailand. (1995-2000)
- Lecturer at Department of Computer Science, Faculty of Science, Prince of Songkla University. (Since 2001)

Publications:

- Preechaveerakul, L. and Bhattarakosol, P., Intelligent Naming System: An Alternative for Enterprise Naming Management. International Journal of Computer Science and Applications. 3(1): 92-103, 2006
- Bhattarakosol, P. and Preechaveerakul L., "How to Use One Name for Many Sites?", in *Proc. of the International Conference on Cybernetics and Information Technologies, Systems and Applications(CITSA 2004)*, Florida, Orlando, USA, July 21-25, pp. 57-62, 2004.
- Preechaveerakul, L. and Bhattarakosol, P., "Is It Possible to Use One Name for Many Sites?", in *Proc. of International Conference on Internet Computing (IC'04)*, Las Vegas, Nevada, USA, June 21-24, pp. 360-365, 2004.
- Preechaveerakul, L. and Bhattarakosol, P., "One Name - Many Objects - One Result: A Possible Solution for Naming System", in *Proc. of the 3rd International Symposium on Information and Communication Technologies*, Las Vegas, Nevada, USA, June 16-18, pp. 63-67, 2004
- Preechaveerakul L., O'Brien, F., Castro, M., and Bhattarakosol, P., "Intelligent DNS-based Naming System Mechanism", in *Proc. of the 1st Workshop on the Internet, Telecommunications and Signal Processing (WITSP' 02)*, Wollongong, Australia. December 9 -11, pp. 123-127, 2002.

Scholarship:

- Project on Faculty Development in Shortage Area Fund through Ph.D. Program Scholarship.