CHAPTER III

THE INTELLIGENT DOMAIN NAME SYSTEM

This chapter describes the proposed system named the "Intelligent Domain Name System" (IDNS) or Sharable Name System (SNS), referred in [46, 47] that eliminates the limitations and drawbacks of a variety of name services founded in Chapter 2.

3.1 Limitations and Drawbacks of Various Name Services

Referring to Table 2.7 from Chapter 2. using a single name to identify various objects with their unique information is not fully supported in the existing name services. Though. the DNS supports sharing unique name. this feature is intended only to help load balancing. For example, in the case of web browsing, the implemented architecture is replicated database over the network while DNS implements the rerouting mechanism using shortest path search method to provide the quick response to user.

We classify the limitations and drawbacks of a variety of name services into 3 groups: uniqueness, character suppport, 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 always occurs. For example, a name in the form of a URL is accessed a specific web page such as www.google.com (Google). When user accesses Google web page, the process to select the best representation for a given response from multiple representations available, called content negotiation [48], is activated. Though there are supplementing techniques such as using content negotiation, this does not truly solve the limitation of sharing name. Currently Google(www.google.com) relies on content negotiation policy to automatically display the local web site to the user based upon a language preference setting in user's browser. The illusion to the user is that only the name www.google.com exists, but the system maps a name to the local site. For example, a user in Thailand could get Google from www.google.co.th.

- *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 been increasing.

- *Anonymity*

  It is sometimes not necessary to identify the source of message [49]. Unfortunately, the name service structure does not support this type of requirement. Therefore, all sources must be identified.

Then, the new naming system called Intelligent Domain Name System (IDNS) is

proposed to support sharing name and non-ASCII characters. However, the anonymity issue will not be considered and supported in IDNS.

3.2 The New Naming System: IDNS

The main purpose of designing the IDNS architecture is to manage one name for various objects. The organizations with more than one office location are required to share the names which perform the same functions. Therefore, obtaining one accurate result from mapping one name to many objects is important. The example of Prince of Songkla University and its campuses is shown in Figure 3.1.

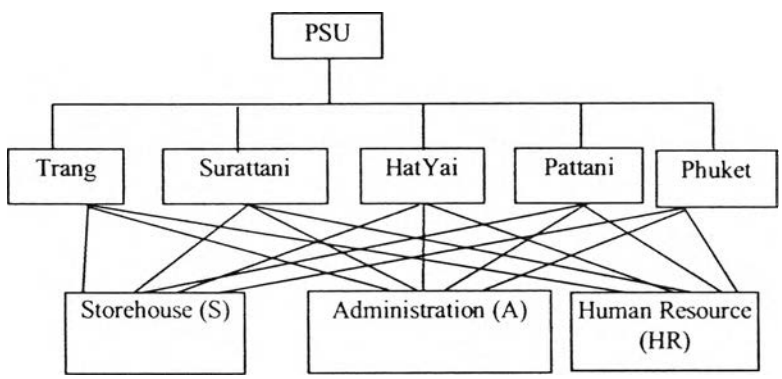

Figure 3.1: Prince of Songkla University and Campuses

Figure 3.1 shows that Prince of Songkla University (PSU) has 5 campuses located in different provinces: HatYai, Pattani, Phuket, Surattani, and Trang. Each location consists of 3 divisions: Storehouse (S), Administration (A), and Human Resource (HR) divisions. Each division of every location performs the same tasks. For example, the Administration division at HatYai performs the same functions as Administration divisions of other four campuses. The structure of PSU contrasts with hierarchical representation.

Using hierarchical name space, a parent-child relationship provides a unique name. This feature ensures that a name uniquely identifies a single leaf in the tree. Therefore, if we can refer to the name "Administration" for every campus, it is clearly understand and easy to remember for everyone. Furthermore, using the same name for the same task or the same object helps PSU maintains unity of its organizations.

3.2.1 Principle Design Concepts

The IDNS is designed under the following requirements:

1. Scalability: IDNS will remain effective when there is an increasing number of resources and users.

2. Decentralized administration via delegation: a domain can be divided into subdomains. Each subdomain, or delegation, is responsible for maintaining all the data.

3. Opacity: the use of a name conceals from the user's location of the objects. Location opacity enables resources to be accessed without requiring knowledge of their locations.

4. Efficiency: IDNS is a distributed database with hierarchical structure that allows different parts of the naming database to be maintained by different entities. Additionally, caching reduces the number of queries that cannot be answered locally and need to contact other servers.

5. Character support: IDNS is able to support both ASCII and non-ASCII character-based query.

3.2.2 The Structure of IDNS

The name space of IDNS is called a global name space. The global name space is hierarchical with a single root at the top. Each name is a path in the global name space.

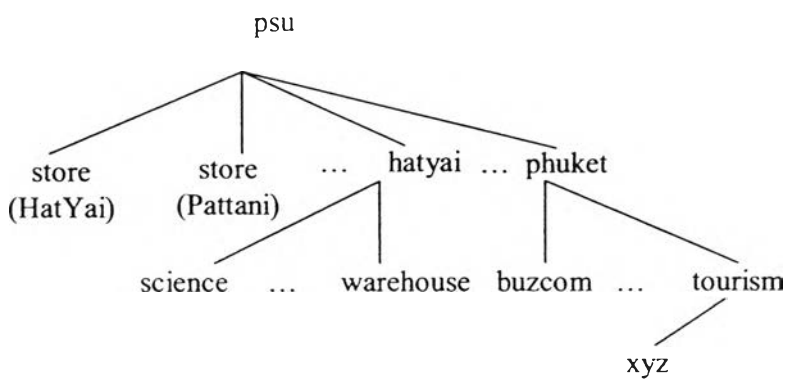

Figure 3.2: The logical organization of IDNS

The objects are the global names that point to information of individual objects. The hierarchical structure of the IDNS employs the concept of sets. Figure 3.2 illustrates the logical organization of the IDNS in the sense of a sharable name. A single root at the top 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 several nodes in the tree. Each global name may point to information of an individual object, or a set of information of individual objects. So, the IDNS maps each global name to a set of objects. For example, in Figure 3.2, the new structure provides a sharable name such as *store.psu* on the left branch would map storehouse at HatYai (obj1) and *store.psu* at the right branch

maps storehouse at Pattani (obj2). However, if *store.psu* is called, the result provides a set of objects: {obj1, obj2}.

The IDNS query (IDNSQ) could be one of the following types:

1) *A full name* such as "store.psu".

2) *A combination of global names* such as "store.psu:warehouse.hatyai.psu".

3) *A global name with location* such as "store.psu:loc@hatyai".

4) *A global name plus context* such as "xyz.psu:&tourism".

3.2.3 Active Components

IDNS is a client/server architecture. Figure 3.3 shows the IDNS protocol design. As shown in Figure 3.3, there are 3 significant components: IDNS client, IDNS server and IDNS communication.

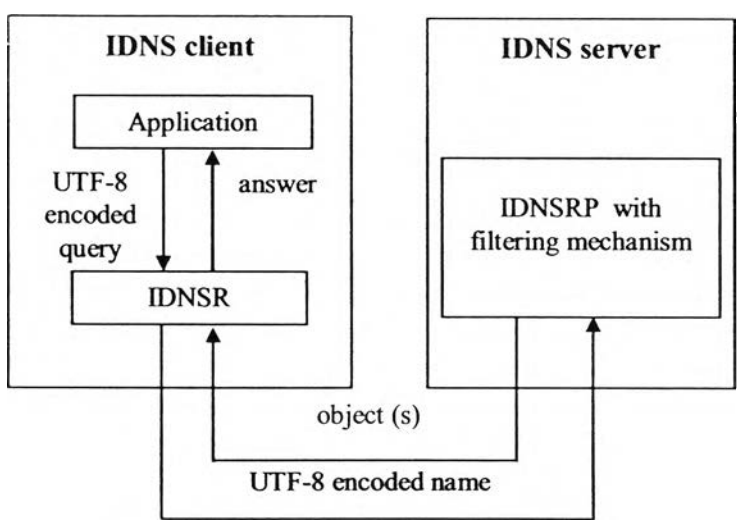

Figure 3.3: The IDNS protocol design.

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

2. The IDNS server contains an IDNS resolution process (IDNSRP). The IDNSRP is a process with a filtering mechanism to determine a mapping from a query name to an object. The IDNSRP 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).

3. the IDNS communication part is an IDNS message (IDNSM) that transmits queries and responses between a client and a server using UDP.
   Since, the communication between 2 sites can be TCP or UDP protocol. Then, we have tested the average transmission time using UDP [50] comparing with TCP [51] packets. The experiment is executed on the following assumptions:

   i. The network is uncongested and reliable channel.

   ii. UDP datagram size is 512 bytes.

   iii. The number of queries is 1000 queries.

   iv. The measurement method is a round trip time (RTT). The RTT is the different value between a starting time (ST) when a user requests a query and a time that user receive a result (ET).

$$RTT = ET - ST$$

ต้นฉบับ หน้าขาดหาย

ต้นฉบับ หน้าขาดหาย

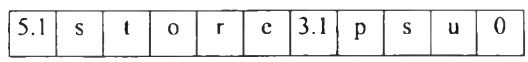

Figure 3.5: A global name "store.psu"

Figure 3.6 shows a Thai global name which uses non-ASCII characters.

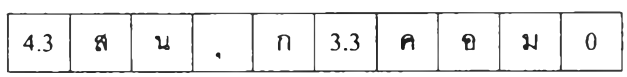

Figure 3.6: A Thai global name which uses non-ASCII characters.

The answer section may contains one or more records called the IDNS records. The number of records is specified in the corresponding field in the header section. Each IDNS record contains name, time-to-live(TTL), IDNS data length and IDNSDATA field. Figure 3.7 describes format of an IDNS record.

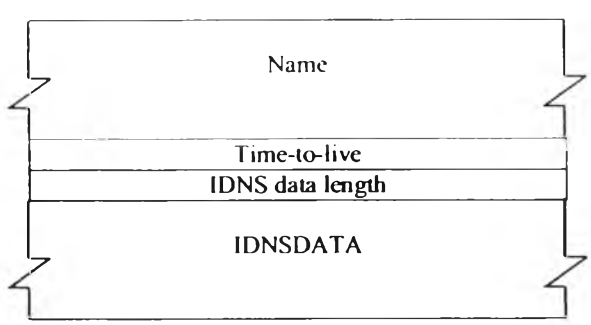

Figure 3.7: Format of IDNS Record

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

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

The IDNS data length specifies the amount of IDNS data.

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

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	fg = 0 Opcode = query
Question	QName = store.psu
Answer	<empty>

The response may be:

Header	fg = 1,Opcode = query
Question	QName = store.psu
Answer	store.psu 86400 18 obj1 HatYai 20011213 store.psu 86400 19 obj2 Pattani 20020305

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

The ONMOOR technique needs 2 important parts: The information part or Global name properties (GNP), and the processing part or IDNSRP.

1. Global Name Properties (GNP)

   The GNP is closely related to the filtering mechanism. Therefore, a global class

(*Gclass*) and user class (*Uclass*) are defined. Gclass stores the information of each global object. Each global object stores a global name (*Gname*), an object that related to a name (*Gobj*), the location of the object (*Glocality*), the creation date of the object (*GcrDate*), the type of the global name whether being a leaf node or an internal node in the global name (*Gtype*), and a global full host name (*Ghn*). *Uclass* stores some information such as location. Figure 3.8 shows the relationship between the global class and the user class.

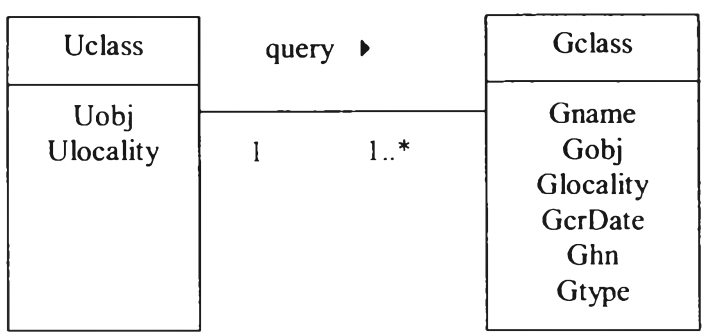

Figure 3.8: Class diagram of User and Global and their relationship.

The example below refers to Figure 3.2.

Example: A local name server is *psu* and stores *Gclass*. Suppose that *Gclass* contains six attributes: <*Gname, Gobj, Glocality, GcrDate, Ghn, Gtype*>. Additionally, *Uclass* contains two attributes: <*Uobj, Ulocality*> Table 3.2 and 3.3 show the data stored in *Gclass* and *Uclass*.

Gclass stores the information of each global object. Practically, the attribute of Gclass may be varied depending on the contents that use with the filtering mechanism.

2. The IDNS Name Resolution Process (IDNSRP)

Table 3.2: Data store in *Gclass*.

GName	*Gobj*	*Glocality*	*GcrDate*	*Ghn*	*GType*
store.psu	obj1	Phuket	2001-01-13	store.phuket.psu	Lf
store.psu	obj2	Pattani	2002-03-05	store.pattani.psu	Lf
hatyai.psu	obj3	HatYai	1999-11-12	hatyai.psu	Nd
⋮		⋮			⋮
phuket.psu	objx	Phuket	2000-09-30	phuket.psu	Nd

Table 3.3: Data store in *Uclass*.

Uobj	*Ulocality*
Uobj1	HatYai

The responsibility of the IDNSRP 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, and the local name server does not know the answer, a process to decompose the query into subqueries (g_1, g_2, ..., g_k) is invoked. Then, it asks these subqueries to a top name server. Other name servers are queried to determine a mapping from each subquery returns, and gives the answer (ANS). The algorithm to obtain the solution is elaborated as follows, and illustrated in Figure 3.9.

Algorithm of IDNSRP

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.

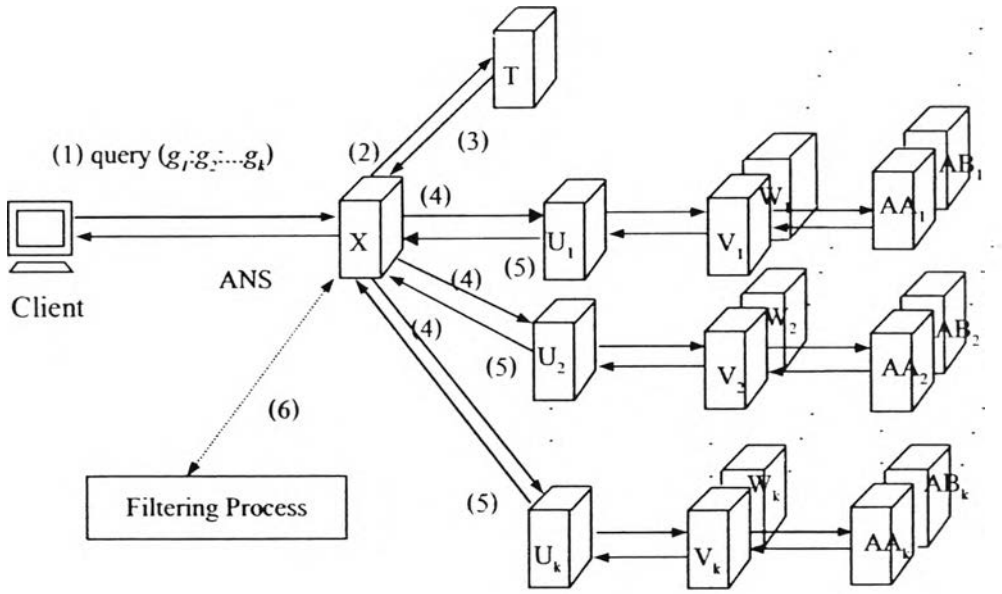

Figure 3.9: The new resolution process.

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 a top name server, 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 **FilteringProcess**.

be satisfied.

Assume that

1. The IDNS contains m nodes with n internal nodes.
2. The search tree T is a multiway (m–way) search tree [53] if
   2.1 T is an empty tree, or
   2.2 T is not an empty tree with the following properties:
      i) T is a node of type:

      $T_0,\ K_1,\ T_1,\ K_2,\ \ldots,\ K_n,\ T_n$

      where

      T_i, $0 \leq i \leq n$ are the subtrees of T

      K_i, $0 \leq i \leq n$ are key values,

      $1 \leq n < m$

      ii) $K_i < K_{i+1}$, $1 \leq i < n$

      iii) For every key value V in subtree T_i, $K_i < V < K_{i+1}$

      iv) Subtrees T_i, $0 \leq i \leq n$ are also m–way search trees.

Searching Method

A searching method is explained with the following psuedo-code.

To search the IDNS tree for a key value g, let m be a number of nodes in IDNS name space and let n be a number of internal nodes.

Finding an i for which $K_i \leq g < K_{i+1}$, $1 \leq i < n$ and $1 \leq n < m$

if key g is equal to K_i

then Found g

else

if subtree T_i is empty

then Not found

else Recursively search T_i for key g

The time complexity of the IDNS search tree can be considered as follows.

Let J be the number of keys on the tree, $J = (m\text{-}1)n$, where n is the number of internal nodes. All nodes have $m\text{-}1$ keys.

1. If applying the linear search on a node, it takes $O(m \log_m J)$ to search an m-way search tree.
2. If performing the binary search on the node, it takes $O(\log_2 m \log_m J)$ to search an m-way search tree.

The expected time of sorting n elements in FilteringProcess is $O(n \log_2 n)$ [54], and the comparison between the location or the latest creation date and ANS is $O(n)$. Thus, this FilteringProcess algorithm requires the expected time in $O(n + n \log_2 n)$. So, the complexity is $O(n \log_2 n)$. The reason for choosing the object with the latest creation date is based on the human nature that loves to obtain 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.

The following examples are based on the ONMOOR technique described in Section

3.2.5.

Example 1: Suppose a client at *psu* queries store.psu to its local name server *"psu"* which is located in HatYai, 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 creation date of the object in *ANS* and returns only one object to the client.

Example 2: Suppose a client at Phuket queries store.psu to its local name server *"phuket.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.