CHAPTER IV

IMPLEMENTATION AND ANALYSIS

4.1 Implementation

The IDNS has been implemented as a client/server java program using UDP as the transmission protocol. Since UDP is unreliable service which does not guarantee that data sent by one process will arrive to the destination process in proper order. Then, our program have defined the size of UDP datagram to be 512 bytes. If the size of each query or response is larger than 512 bytes, the program will divide it. Moreover, to make the IDNSM transmits queries and responses between a client and a server using UDP more reliable, the application also checks the transmission of flow data control between a client and a server.

The implementation is supposed that the global name space of PSU shown in Figure 4.1 contains the following delegated name servers:

1. "psu" which is located at Hatyai.

2. "phuket.psu" which is located at Phuket.

3. "science.psu" which is located at Hatyai.

4. "chem.science.psu" which is located at Trang.

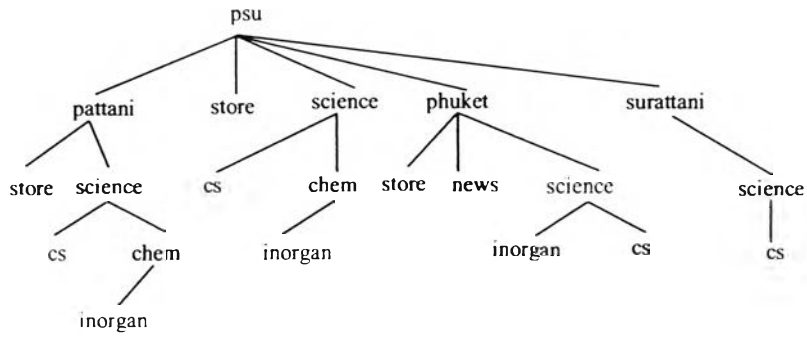

Figure 4.1: The PSU name space.

Figure 4.1 shows a hierarchical structure of PSU name space. The following examples show the result of each global name queried from the client (C) which is located nearby different local name servers.

Example 1. Suppose that a local name server is "psu".

1.1 A client requests a query name "store.psu".

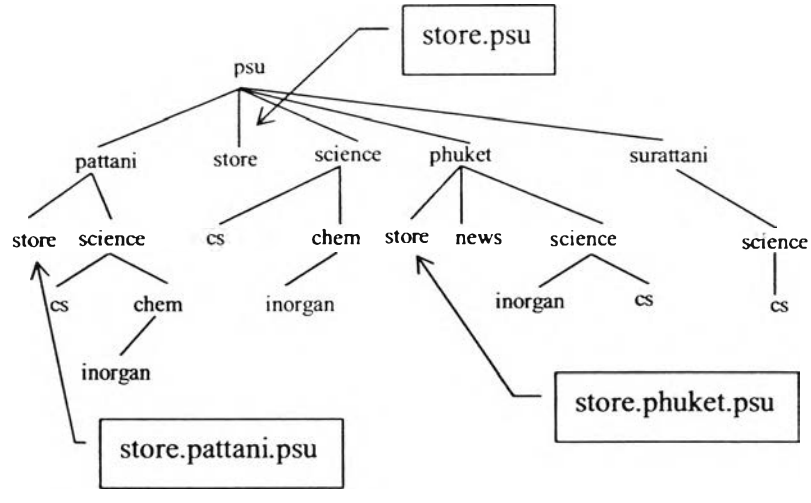

Figure 4.2: A possibility of retrieving a global name "store.psu" in PSU name space.

From Figure 4.1, when a client queries a global name"store.psu", the PSU name space shows a single unique name "store.psu". However, the IDNS supports shar-

ing name, the result from querying "store.psu" is not only single name as shown directly in Figure 4.1, but also other global names that require to use "store.psu" instead of their global full host name. In this case, there is a possibility to retrieve the following results: store.psu, store.pattani.psu, and store.phuket.psu as shown in Figure 4.2.

The result from the implemented system shows the output in Figure 4.3 and Figure 4.4 depending on a client chooses whether "Result" or "All Results" button. After user clicks one of the results, this experiment will invoke the web site related to this favorite result.

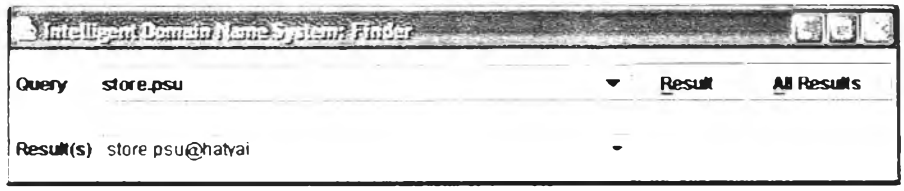

Figure 4.3: A single result queried from IDNS.

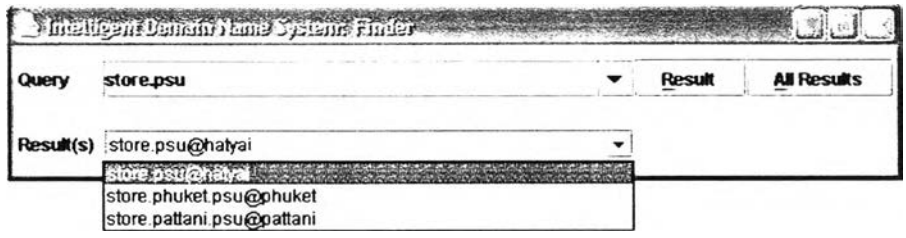

Figure 4.4: The Multiple results queried from IDNS.

1.2 A client requests a query name "news.phuket.psu".

From Figure 4.1, a client query a global name "news.phuket.psu" which is a path in an inverted tree. Since a local name server is "psu", and "psu" does not know

the answer. Then, "psu" sends this query to another related name server which is "phuket.psu". Then, "phuket.psu" returns the result to a client. The output is shown in Figure 4.5.

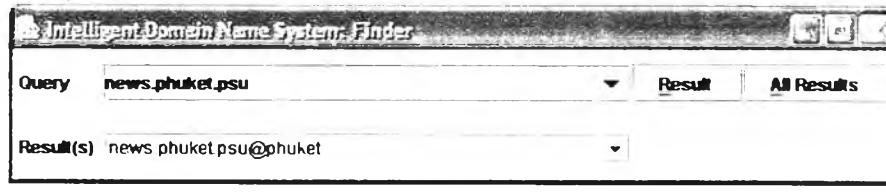

Figure 4.5: The result of a global name "news.phuket.psu" from another server.

1.3 A client requests a query name with specific location: "chem.science.psu:loc@pattani"

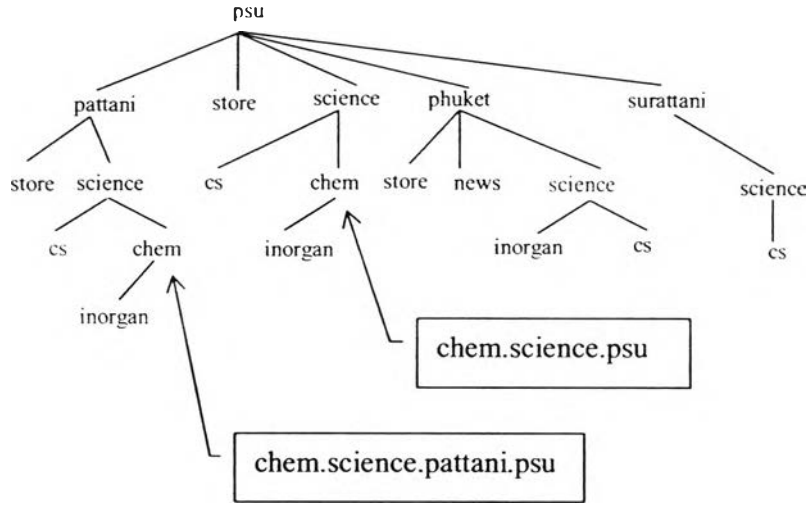

Figure 4.6: A possibility of retrieving a global name "chem.science.psu" in PSU name space.

Referring to Figure 4.1, a client queries a global name "chem.science.psu", there is a possibility to retrieve many results because of the property of sharing name as shown in Figure 4.6. However, the query is specified a location, then the output

related to the location is returned. The result is shown in Figure 4.7.

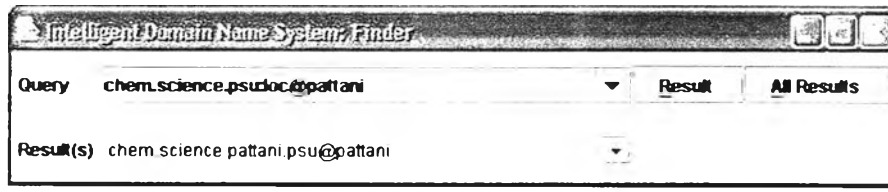

Figure 4.7: The result of a global name with specific location.

Example 2. Suppose that a local name server is "science.psu"

2.1 A client requests a query name "cs.psu".

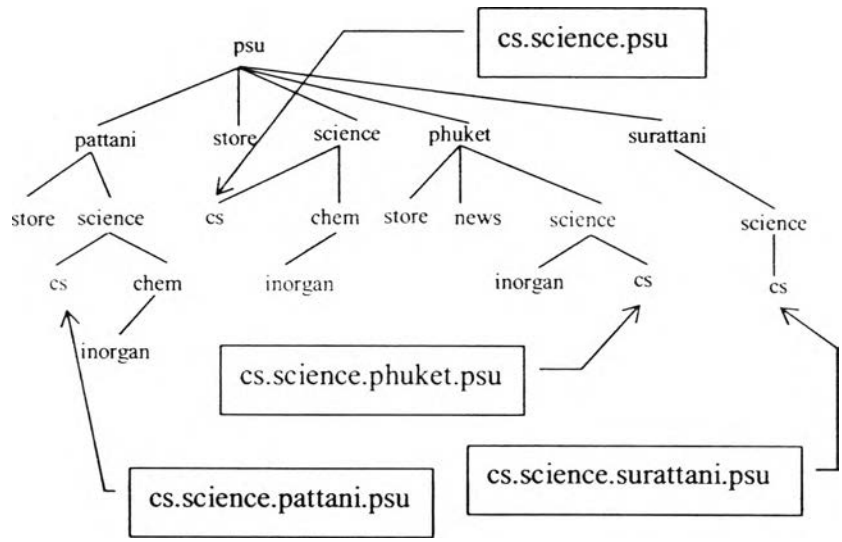

Figure 4.8: A possibility of retrieving a global name "cs.science.psu" in PSU name space.

Referring to Figure 4.1, a client queries "cs.psu". Although, there is no path in the PSU name space named as "cs.psu", the sharing name mechanism enables other global names to refer to "cs.psu". Figure 4.8 shows all possibilities of the required nodes, and the presented information can be either one or many nodes as shown in Figure 4.9 and 4.10 respectively.

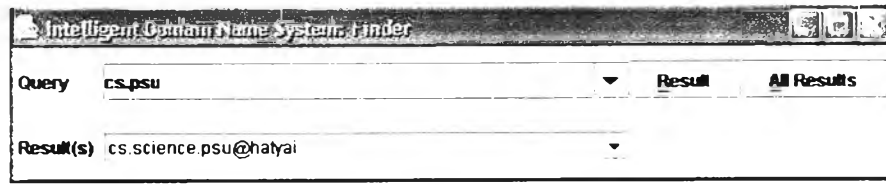

Figure 4.9: The result of a global name in another server.

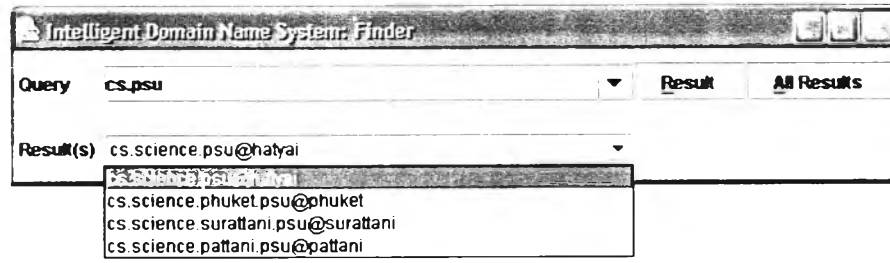

Figure 4.10: The multiple results of a global name in another server.

2.2 A client requests a query name "inorgan.science.psu" and knows that this name is related to chemistry department. Then, a client can request a query name with a context: "inorgan.science.psu:&chem".

Referring to Figure 4.1, a client requests a query name with a context "inorgan.science.psu:&chem" The sharing name mechanism enables multiple results as shown in Figure 4.11. However, the query specified a context "chem" and a client chooses "Result" button. Then, instead of the results are "inorgan.chem.science.psu" and "inorgan.chem.science.pattani.psu", the output shows a single result "inorgan.chem.science.psu" since the IDNSRP algorithm checks the client location which is nearby a local name server "science.psu". The result is shown in Figure 4.12.

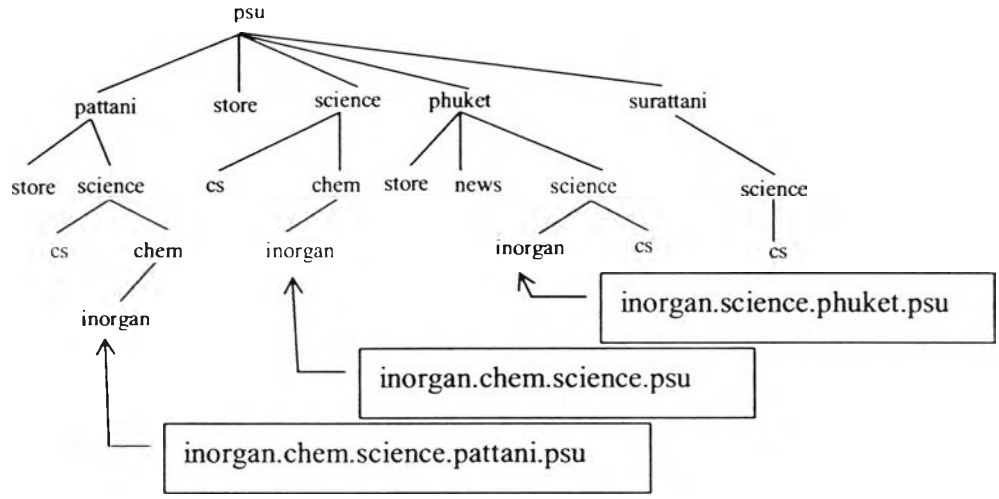

Figure 4.11: A possibility of retrieving a global name with a context.

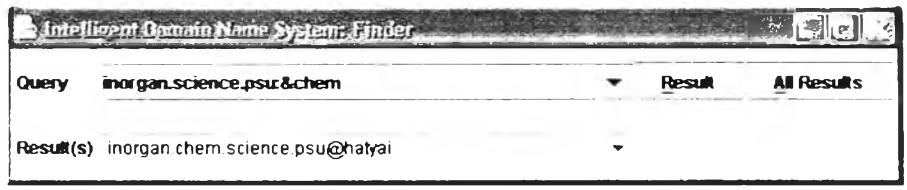

Figure 4.12: The result of a global name with a context.

4.2 IDNS Model and Its Analysis

IDNS is proposed as a method to allow sharable names. we now give a theoretical proof that it meets the design goal. The structure of IDNS is explained by the following definitions and theorem.

Definition 1. *Let* α *be a set of characters and* τ *is a string defined by*

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

Definition 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* $vName$, *shown in Fig. 4.13*

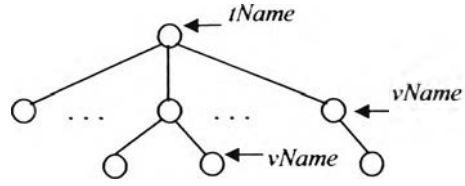

Figure 4.13: A Name space *N*.

Definition 3. *A global name g is a path in a name space N such that*

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

where $tName$ *is the top of the tree,*

$vName_i$ *is any node in* N $(1 \leq i \leq n,\ n \in I)$,

$vName_0$ *is a leaf node in* N, *and*

dl *is the delimiter character* ".".

Definition 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,*

$$f_l(T) = \{f_l(t) | t \in T\}$$

Definition 5. *Let* f_d *be a function of* $G \times D$ *into* O *and let* V *be a subset of* $G \times D$ *such that* $f_d(V)$, *called "*f_d *of* V*", is defined to be the set of image of elements in* V. *In other words,*

$$f_d(V) = \{f_d(v) | v \in V\}$$

Definition 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:* $\{g_1, g_2, ..., g_n\}$,

O *is a set of objects:* $\{Obj_1, Obj_2, ..., Obj_m\}$,

L *is a set of locations:* $\{l_1, l_2, ..., l_o\}$,

D *is a set of creation date of the objects:* $\{d_1, d_2, ..., d_p\}$, *and*

Z *is a set function of* T *into* O *or a set 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 7. *Let* Q *be a combination of global names* $(g_1 : g_2 : ... : g_k)$ *and* $D_r = \{g_1, g_2, ..., g_k\}$ *be a set of global names,* $D_r \subseteq G$.

Definition 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 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 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 1 we have $z \in R$. □

Theorem 1 shows that if there is at least one required object to the user, this required object maps a global name. Theorem 2 shows that if there exists a global name in a naming system, then a user will certainly get the required object.