國立交通大學

資訊科學與工程研究所

碩 士 論 文

多功能個人/團體通訊系統

Multi-Function Personal/Group Communication System

研 究 生：李後瑋

指導教授：張明峰　教授

中 華 民 國 九 十 六 年 六 月

多功能個人/團體通訊系統

Multi-Function Personal/Group Communication System

研　究　生：李後瑋　　　Student：Hou-Wei Lee

指導教授：張明峰　　　Advisor：Ming-Feng Chang

國　立　交　通　大　學

網　路　工　程　研　究　所

碩　士　論　文

A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

`

多功能個人/團體通訊系統

學生：李後瑋　　　　　　　　　　　　　　　指導教授：張明峰 博士

國立交通大學網路工程研究所

摘要

傳統的通訊系統以一組數字或一個帳號來代表一個通訊點，例如手機系統的識別元是手機號碼，然而這樣的識別元並無意義，無法和通訊點聯想。我們提出一個新的通訊系統，除了支援原本的單一、特定識別元也支援多屬性識別元，讓發訊者在尋找通訊目標時，即使不知道對方的特定識別元，也可透過輸入”姓名”，”地址”，”年紀”，”就讀學校”，等屬性，來找到對方。

依屬性設定不同，此系統可以提供多種通訊服務。例如，廣告商可以將廣告發送給在”喜好”屬性中設定符合商品特性的使用者，或是團體成員也可以約定設定相同屬性，就可以讓所有成員看到團體訊息。我們提供簡訊通訊功能和語音通訊功能供使用者使用，語音通訊部分目前只支援一對一談話。

在系統設計部分，我們提出以結構化點對點(P2P)網路為基礎的解決方案，我們也整合、修改現有研究，分別提出字串、數字、混種型態屬性值的散布和查詢方式。此外，為解決通訊點加入、離開系統造成連結錯誤或儲存的資料遺失的問題，我們也提出相關解決方案。

Multi-Function Personal/Group Communication System

Student: Hou-Wei Lee Advisor: Dr. Ming-Feng Chang

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

Traditional communication system uses a specific ID, such as a cell phone number or an SIP account, to specify a communication node. Such a specific ID contains little information that we can associate with the communication node. We propose a multi-function personal/group communication system (MFPGC system) supporting both traditional specific IDs and multiple unspecific attributes. A user can find a callee by unspecific, but meaningful attributes such as “Name”, “Address”, “Age”, and “University” in this system.

MFPGC system can support various communication scenarios by different attribute settings. For example, an advertiser can publish advertisement to users with suitable “interest”, “preference”, or “habits” attributes. A group of members can also define several attributes with the same values and thus a member can send messages to all other members without knowing others’ specific IDs. In our implementation, we provide one-to-one and one-to-many text message (like an instant message in cell phone system) communication and one-to-one voice communication functionality.

In our system design, we propose a structured P2P based solution. We integrate and enhance existing solutions for string type attributes, numerical type attributes, and hybrid type attributes publish/query. Furthermore, to solve problems caused by the dynamic user behavior of joining/leaving the system, we propose a data duplication solution.

`

誌謝

首先我要感謝我的指導老師，張明峰教授。教授溫和而耐心的指導，讓我在實作和閱讀中學會獨立思考與研究的方法；在構思本論文時，也多次指正我思考的盲點，才能順利完成這篇論文。我很感謝在求學過程中，能得到張教授的指導，研究所這兩年獲益良多。

接下來我要感謝實驗室的同仁，博今同學、名均同學、聖全同學，和我一起修課奮鬥，一起熬夜寫程式，一起打球，一起練馬拉松，同甘共苦，讓我兩年過得一點也不孤單。也感謝坤楊、忠育、冠璋、君飛等學弟，讓我這段時間的生活增添不少色彩，很高興能認識你們。

我要特別感謝 Vita，妳的鼓勵和規勸，不僅讓我在疲憊得到安慰，也激起我更多成長的動力。最後要感謝我親愛的家人，由於你們精神和經濟上的支持，讓我可以在求學的過程中一路順坦，沒有後顧之憂的完成學業。

`

Tables of Contents

`

List of Figures

`

List of Tables

Chapter 1 Introduction

1.1 Overview

In a traditional communication system, each endpoint has a specific identifier, such as telephone number, Skype [1] account, or SIP [2] URI. The specific identifier is unique so that a communication session can be built accurately without ambiguity. However, the specific ID is usually meaningless or contains insufficient information that it is difficult to know that who is the owner of the specific ID. On the other hand, a caller is unable to communicate with a callee if he does not know the specific ID even though he knows more details about the callee. There are more attributes that can specify an endpoint, such as the owner's personal information or the location of this endpoint. These attributes are unspecific, and they are unable to substitute the specific ID. However, applying the unspecific attributes in a communication system to specify an endpoint or a group of endpoints could make the communication more flexible.

The following examples show the usage of unspecific attributes.

John wants to make contact with his elementary school classmate, Kevin. John knows the name and nickname of his classmate, the name of the elementary school, and the years they studied. In a communication system with unspecific attributes, John can send messages or make voice communication with one or more callees matching Kevin's specification.

`

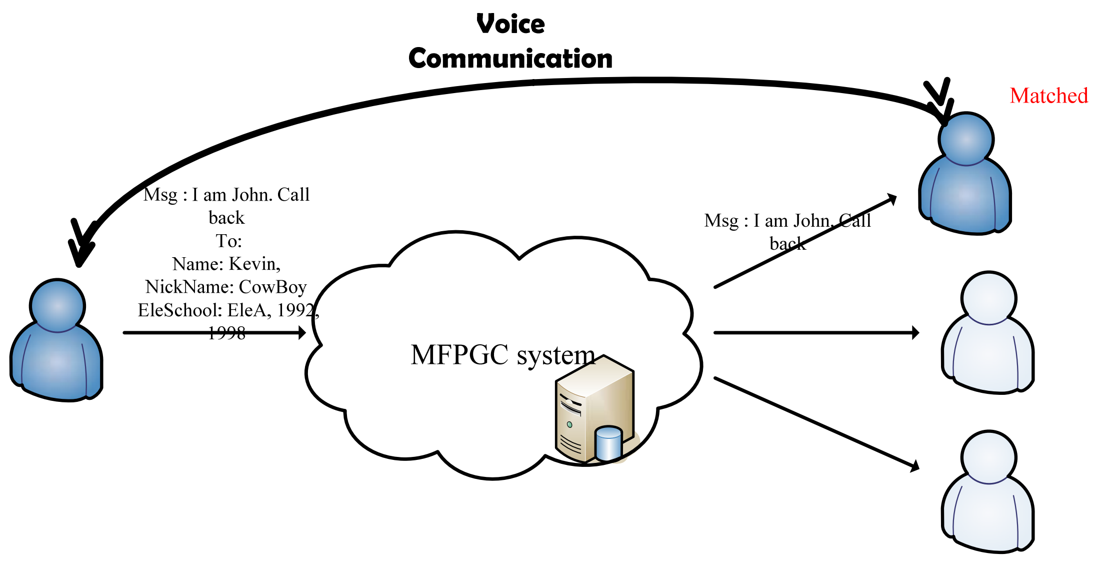

Voice Communication
Matched
Msg : I am John. Call back
To:
Name: Kevin,
NickName: CowBoy
EleSchool: EleA, 1992, 1998
MFPGC system
Msg : I am John. Call back

Fig. 1 An example of one-to-one voice communication

Linda and her orchestra members set the name of the orchestra as an attribute of their communication device, so that Linda can send message to all the members without knowing all the members' phone numbers.

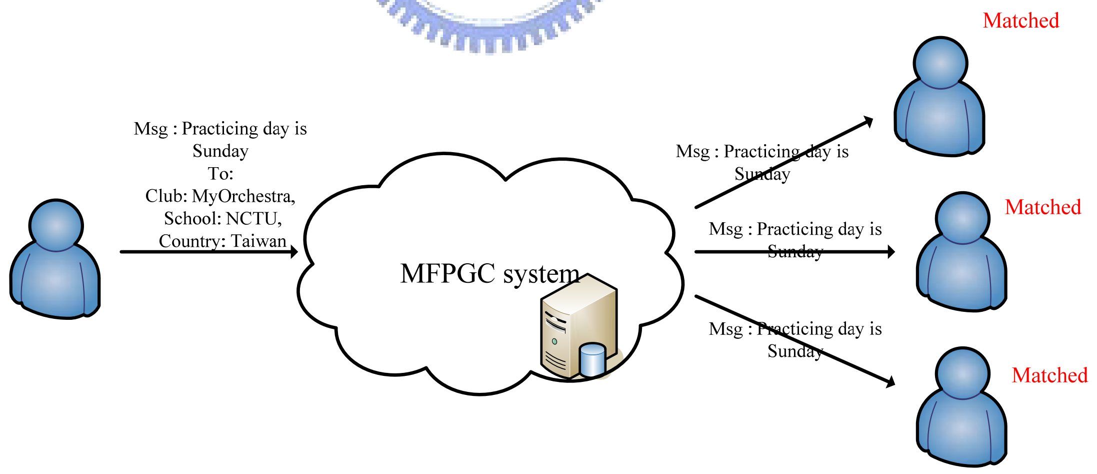

Matched
Msg : Practicing day is Sunday
To:
Club: MyOrchestra,
School: NCTU,
Country: Taiwan
MFPGC system
Msg : Practicing day is Sunday
Matched
Msg : Practicing day is Sunday
Msg : Practicing day is Sunday
Matched

Fig. 2 An example of one-to-many communication

Such a communication system utilizing unspecific attributes to help specifying an endpoint can support both one-to-one, and one-to-many communications. The members of a group have some common attributes that match the caller's (or the message sender's) contacting target settings. In this thesis, the caller and the message sender are denoted as content publishers (CP); the callee and the message receiver are denoted as content receivers (CR) for abbreviation. The size of a group depends on how general or specific the attributes are. If the CP uses only few general attributes to describe the CR, the CR group will be large; if the CP uses many special attributes to describe the CR, the CR group could be very small, even with just one or zero member.

To compose such a communication system, each user has to register his/her attributes in a database. One simple solution of such a system is a client/server architecture: users register their attributes to a central database server, and the CP queries this server to find the matched CRs. However, the heavy load of storage and bandwidth of the server could be a bottleneck of the system, and client/server architecture has single point failure problem. We choose peer-to-peer (P2P) network architecture to compose our communication system so that this system could be more robust and scalable.

In this thesis, we propose and implement a communication system that supports both specific ID and unspecific attributes communication. This system supports voice communication and message communication, and the communication target could be a person or a group. This system will be referred to a multi-function personal/group communication system (MFPGC system).

1.2 Related Work

P2P (peer to peer) is an architecture where each peer serves as a server and a client. P2P network distributes storage and network loads to every peers, so that a central server is eliminated. There are two categories of P2P network architectures: unstructured P2P and structured P2P. Gnutella [8] is a famous file sharing application that applies unstructured P2P. Each peer in Gnutella searches by flooding message and limits the flooding scope by time-to-live (TTL). However, flooding is still bandwidth costly and it doesn’t ensure that all the correct targets could be found.

Structured P2P makes use of distributed hashing table (DHT) to publish or query a data. DHT provides a function to map a key to a hash value. Current proposed DHT systems include Tapestry [9], Pastry [10], Chord [4], CAN [6] and Koorde [11]. We give more detail for Chord. Chord is a DHT based structured P2P. Each peer in Chord records at most *log(N)* other peers information in its finger table. Through looking up the finger table, Chord can provide O(*log(N)*) routing efficiency. The structure of Chord is a ring type structure. The dynamic joining/leaving behavior of peers may disrupt the complete ring. Chord uses notification and stabilization protocol to keep the ring complete so that it can answer queries even though the system is continuously changing. SHA-1 is adopted as hashing function of a string type keys to an m-bit hash value. SHA-1 is a uniform hashing function that can help balancing the load of storage and bandwidth. Chord can support single string type attribute lookup applications, but MFPGC system requires multiple string type and numerical type attributes lookup.

There have been many researches discussing P2P multiple attributes look-up [3] [5] [7]. Most of them use inverted indexing for information distribution. An indexing maps an object to a list of words, as inverted indexing maps a word to a list of objects. Each attribute (a word) of an object is hashed and then published to the P2P system together with the source node information. To find a target source, the query is propagated from peer to peer, and intersects the results. Several algorithms, such as

`

cache or bloom filter [14], were proposed to improve the query efficiency.

Query of numerical type attributes is also known as range query, and there have been several proposed solutions [3] [12]. A range query is a query that asks the peers with a numerical attribute falling in a range such as “18 < age < 20”. SHA-1 hash cannot support range query because it doesn’t keep numbers with near value in nearby peers. A locality preserving hashing function is required to take use of the numerical feature to improve search efficiency. MAAN [3] propose a locality preserving hash function that normalizes the numerical value and than multiply it with the maximum hash value so it can map to an m-bit system. However, this system still takes O(N) worst case range query.

However, there is no single research that could be directly applied to compose MFPGC system because the attributes set by users have privacy issue, and the attributes types are variant, “string type”, “numerical type”, and “hybrid type”. To the best of our knowledge a total solution for such a system with above features has not been presented.

1.3 Objective

This thesis proposes the concept of a new communication system and focus on implementing an efficient and reliable system that well supports voice and message communication and unspecific attributes lookup. There are four points that we concern when designing this system. 1. User information is private data, so the whole text can’t be directly published to or stored in the network. 2. Support multiple attributes query. 3. Support range query. 4. Optimize the storage and bandwidth efficiency of our system.

We apply Chord as base architecture of MFPGC system. However, Chord can not directly support multi-attributes query nor range query which MFPGC system require. We adopt inverted indexing to implement multi-attributes query functionality, and adopt range guard [13] for range query. We also propose necessary attributes hashing and local data storage strategy to improve system efficiency and to protect user privacy.

1.4 Summary

The remaining of thesis is organized as follows. Chapter 2 describes the essential knowledge background of P2P networking mechanism and desired functionalities of our system. Chapter 3 shows the details of our system design. Chapter 4 presents the implementation issues. Conclusion is given in Chapter 5.

Chapter 2 Background

MFPGC system is a P2P system based on Chord. In this chapter, we first introduce the functionalities of Chord that we used. Secondly, we give the details and examples for multiple attributes query, range query, and hybrid type attribute query. Then we give a summary for this chapter.

2.1 Chord

As mentioned in previous chapter, Chord is a DHT based structured P2P. Each peer has an *m*-bit *identifier* that is produced by hashing its peer address. The hash function adopted by Chord is SHA-1 and both node IDs and keys are hashed by SHA-1. A peer in Chord maintains a predecessor and a successor connection, and all the peers form a complete ring as shown in figure 1. A new peer must know at least one working peer in the system so it can send join message to join the ring. For convenience, one or more nodes are permanently alive in the system and known as "famous node" so that other peers can efficiently join the system without flooding the joining message.

Finger table is a critical concept of Chord that accelerates the lookup process to O(logN) (in a system with N nodes). A finger table is a table that records several other nodes' IP addresses. Each peer has a finger table storing at most logN nodes information. For a peer in a system applying m-bit identifiers, the peer's finger table record peers information with peer identifier $(k + 2^{x-1}) \% 2^m$, where *x* is an integer ranged from 1 to m. Table 2 shows an example of finger table maintained by peer with 6-bit identifier 8. The utilization of a finger table will be given later in this section.

`

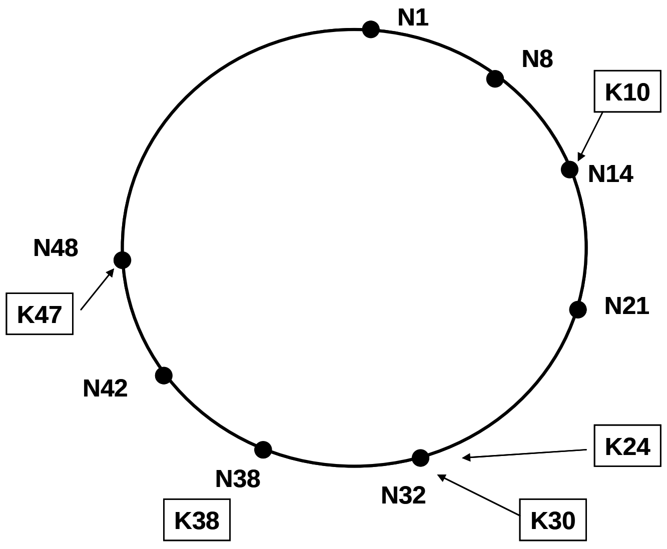


Fig. 3 A Chord ring with 8 nodes storing 5 keys

Table 1 The finger table of Node 8

Finger table index	Peer IP
$(8 + 2^0) \% 2^6 = 9$	IP(lookup(9))
$(8 + 2^1) \% 2^6 = 10$	IP(lookup(10))
$(8 + 2^2) \% 2^6 = 12$	IP(lookup(12))
$(8 + 2^3) \% 2^6 = 16$	IP(lookup(16))
$(8 + 2^4) \% 2^6 = 24$	IP(lookup(24))
$(8 + 2^5) \% 2^6 = 40$	IP(lookup(40))

Chord defines the following processes. Those basic processes are directly applied in MFPGC system.

A. Join

As mentioned before, the joining node hashes its IP address to a node identifier *k* and sends a “join request” containing the identifier and its IP address to one of the nodes on the Chord ring. The node receiving the “join request” forwards the request to the SUCCESSOR(k) by Chord looking up algorithm. SUCCESSOR(k) is the node on the Chord ring with a smallest but not smaller than *k* identifier. SUCCESSOR(k) returns an “ack message” with its IP address to the new peer to be the new peer’s successor. SUCCESSOR(k) also records node(k) as its predecessor. The original predecessor of SUCCESSOR(k) will update its successor to node(k) in further *stabilize* process.

B. Stabilize

Stabilize function is periodically processed by each peer to update the predecessor and successor information. In stabilize function, a peer ask its successor to return predecessor identifier. If the predecessor identifier of successor is not the peer's identifier, it should be ranged between the peer's and the successor's identifier which means it's a new joined peer. In such case, the peer sets the new peer as successor and notifies the new peer so that the new peer can update its predecessor. Stabilize function keeps the Chord ring complete in dynamic peer joining environments.

C. Fix finger

Fix finger is also a periodically called function. The functionality of fix finger is to update the entries in finger table. It is also the process for a newly joined node to set up the finger table. In this function, the peer calculates the peer ID of an entry and sends a fix_finger message to the respective peer through Chord lookup process. The peer sending message will receive a response with respective peer's IP address thus it could update the entry.

D. Key lookup process

The basic key lookup process is to forward the key to the successor till the key reach the peer with smallest identifier but not smaller than the key. After fix finger process, the key lookup process utilizes the finger table to reduce the lookup time.

Finger 2 gives an example showing that how the finger table works in key lookup. In this example, node 8 queries a data with key 54. By looking up the finger table, N8 forward the query to the peer with the identifier that is closest but not larger than key 54, thus it forward the query to N42. By the name rule, N42 forward the query to N51, and N51 forward to N56 which is the destination. With finger table, proven by Chord inventor, the lookup time complexity is O(log (N)).

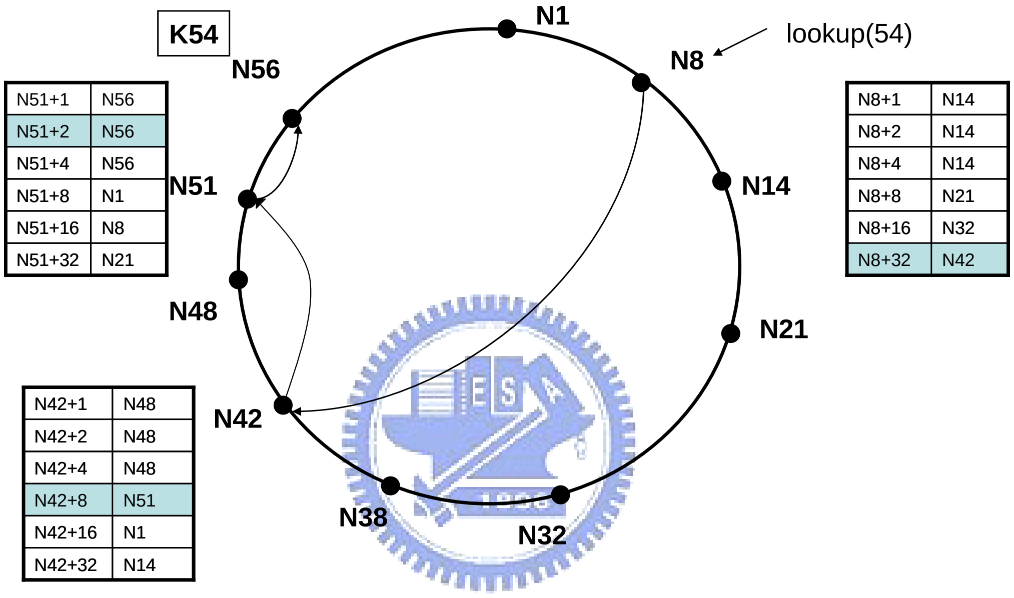


Fig. 4 An example of lookup by finger table

2.2 Multiple attributes query

Both the peer information and target specification are composed by multiple attributes, thus our system must support multiple attributes query. Each attribute in a peer’s information is hashed and published to a responsible peer. To make a multiple attributes query, at first, each of the attributes is hashed to make a list responsible peers, and then forward the query and the list to one of the responsible peers. The responsible peer searches its local database to collect the matched peer IDs of the respective attribute and then forward the query, the list, and the matched IDs to next responsible peer on the list. The next peer also searches its local database and

`

intersects the local search results with received IDs as matched IDs and then forward next peer. This procedure ends when all the peers on the list is traced. The matched IDs produced by last peer are query results. Figure 3 and figure 4 shows examples of peer information publishing and multi-attribute query.

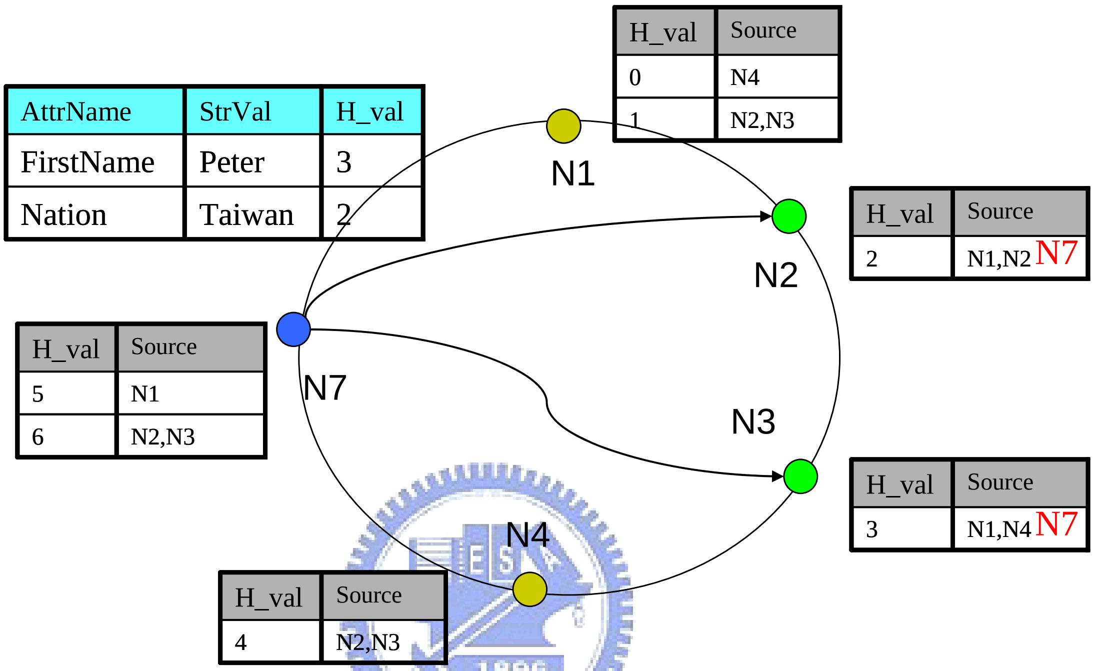


Fig. 5 An example of multiple attributes publishing

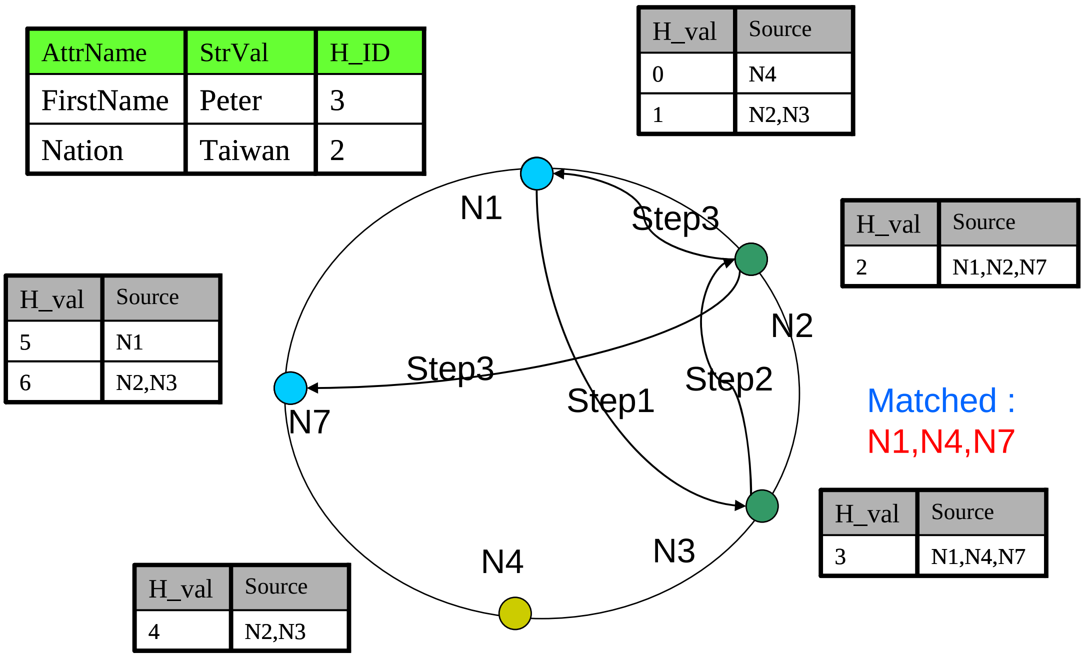


Fig. 6 An example of multiple attributes query

To make the multiple attributes query more efficient, the following strategies could be applied.

Re-order the responsible peers list: The storage size of the first peer to be traced on the responsible peers list affects the query efficiency a lot. If the first peer is responsible for a common attribute, then it would contain a lot of matched peers' ID, and the data to be transferred to the next peer will be massive. The strategy is to ask each peer on the responsible peers list the size of local database storage and then sort the list by progressive order, and thus reduce the data transferring and intersecting cost. Figure 5 shows an example of re-order the query sequence. The bandwidth loading is reduced if we choose a peer with minimum storage as the first query peer.

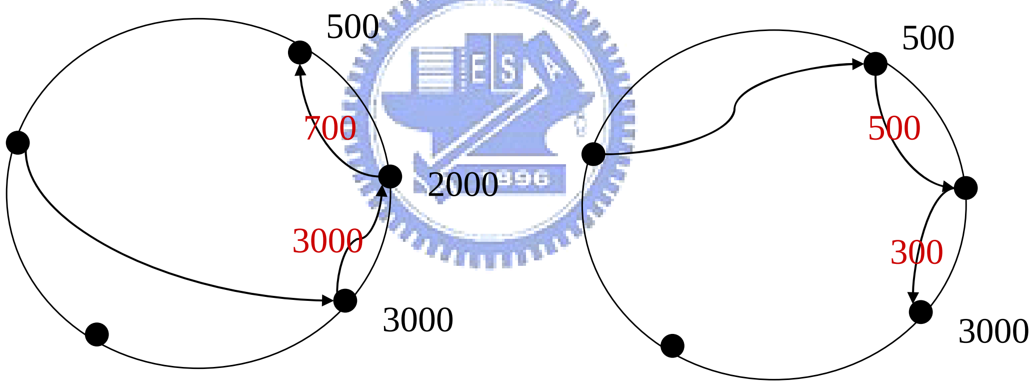


Fig. 7 Effects of re-order the query sequence

Single attribute dominated query: The previous described query scheme takes iterative sub-queries and do intersections to find the result. If there are M attributes in a query and N peers in the system, the routing time complexity of iterative sub-queries is O(*M* * *log(N)*). In single attribute dominated query resolution, proposed and applied in MAAN [3], a peer registers its peer information for each attribute domain, thus one responsible peer could do all the sub-queries to reduce the routing time complexity to O(*log(N)*). However, the peer information in MFFPGC system may be private data thus the complete peer information should not be

`

published to a single peer. Another shortcoming of single attribute dominated query is that there are multiple copies of peer information that the storage load of the system is multiplied. We refer to and adjust this scheme to be applied in our system design, and more detail will be given in chapter 3.

2.3 Range query

Range query is to search for the peers with a numerical attribute falls in a range. For example, a user might want to find a friend with age ranged from 25 to 27. The numerical value could be an integer or a real value, and the comparing operators such as ‘< ’, ‘=’, ‘>’, ‘<=’, and ‘>=’ should be support for a range query.

Chord assigns each data an m-bit identifier via applying a consistent hashing function such as SHA-1. This approach can uniformly distribute the storage and bandwidth load to each peer because SHA-1 is a uniform hashing function regardless of the actual data distribution. However, SHA-1 also destroys the locality of a numerical attribute. Thus for a numerical attribute, we do not apply SHA-1 but use a locality preserving hashing function to produce the m-bits identifier. The locality preserving hashing function is defined as following description.

Definition 1: Hash function H is a locality preserving hashing function if it has the following property: H(vi) < H(vj) iff vi < vj , and if an interval [vi, vj] is split into [vi, vk] and [vk, vj], the corresponding interval [H(vi),H(vj)] must be split into [H(vi),H(vk)] and [H(vk),H(vj)].

A simple locality preserving hashing function is to project the numerical data to an m-bits identifier. The hashing function is

$$H(v) = (v - v_{min}) \times (2^m - 1)/(v_{max} - v_{min}), \text{ where } v \in [v_{min}, v_{max}]$$

`

The hashing function preserve the locality of data, thus a range query could be applied through forwarding the query to the peers responsible for identifiers ranged from H(v_{start}) till H(v_{till}).

2.4 Summary

In this chapter, we describe the functionalities of Chord that we use in our system. We also describe the definitions of multi-attribute query and range query and give preliminary solutions to the queries. In next chapter, we give the details of our system design.

Chapter 3 Design of our system

The main purpose of MFPGC system is to support a flexible communication system that can provide communications to called parties with non-specific attributes. A node in the system has to register the user's information which is a list of attributes. In this chapter, we give details of the design of MFPGC system, including the system architecture, the specification of a user's and a callee's information, the strategy to handle the dynamic user behavior joining and leaving the system, and how a call request is forwarded to set up voice communication setup.

3.1 System Architecture

The network architecture of MFPGC system is a structured DHT based P2P system. We adopt Chord as our architecture foundation, and thus the nodes in MFPGC system form a Chord ring as shown in Figure 5. A node in the system could be a PC, PDA, smart phone or any device that is capable of internet connection. A node in the MFPGC system plays both the roles of a server and a client thus there is no dedicated server in the system. The whole system is composed by users' devices.

Each node is required to install an application named as MFPGC system user agent. This UA contains a database, a voice communication module (a SIP UA), and a Chord engine. It also provides convenient user interface for user to register attributes or to set callee's attributes.

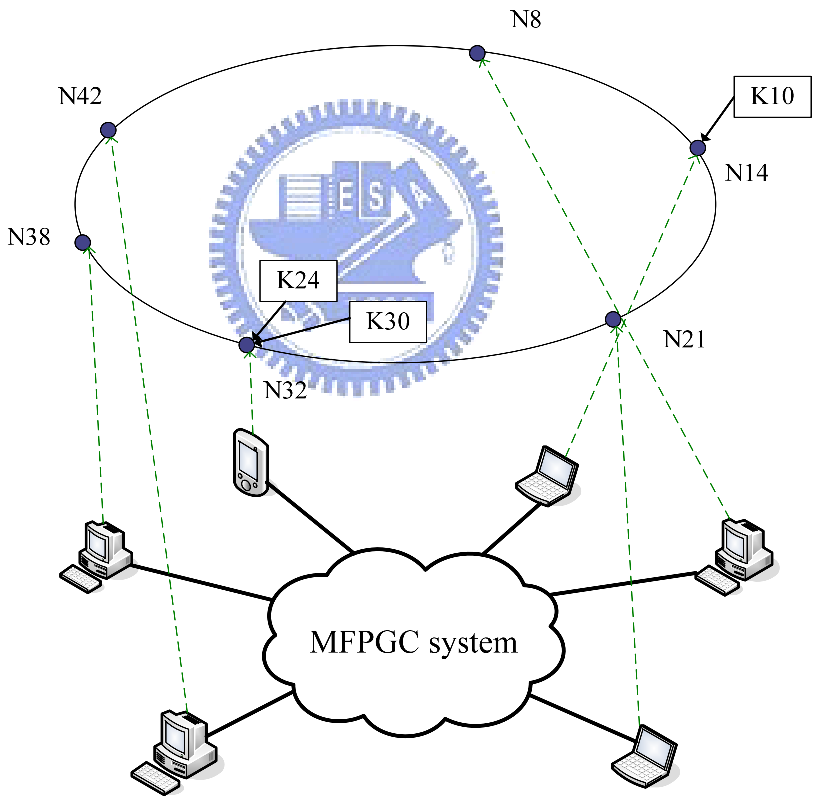


Fig. 8 The communication nodes in MFPGC system form a Chord ring

`

3.2 Specification of the information of a user/callee

Information of a user/callee is a list of attributes that describes the user/calee. First, we give definition of attributes.

3.2.1 Specification of an attribute

A simple attribute contains an attribute name and an attribute value. However, the type of an attribute value affects the hashing strategy; thus the type of the attribute should be indicated. A numerical type attribute contains one or two numerical type values; a hybrid type attribute contains one string type value, and one or two numerical values. We use three elements to describe an attribute's value. All the components of an attribute are shown in Table 2 which contains "Attribute name", "Attribute type", "String type value", "Numerical type value 1", "Numerical type value 2", "Necessary", and "Primary".

For a user to filter out undesired messages, an attribute can be set as a necessary attribute so that only senders that know the necessary attribute value could send message to this user. The "necessary" element is specified in information of a user but not in information of a callee. A "necessary" element is used for filtering requests and details of the filtering function are described in Section 3.3.3.

Table 2 Elements of an attribute

Element	Description
Attribute name	The name of the attribute
Attribute type	String, Numerical, or Hybrid

`

String type value	A string type value of this attribute
Numerical type value 1	A numerical type value of this attribute
Numerical type value 2	A numerical type value of this attribute
Necessary	Yes/ No, for a node to restrict receiving messages

Table 3 shows examples of attributes. “Name” is a string type attribute with “Peter” as attribute value. The user set “Name” as necessary attribute so the CP (content publisher) must set correct “Name” so he can publish information to the user. The next attribute “Age” is a numerical type attribute with one value “22”. The third attribute is a hybrid type attribute, “University”, which has string type value “NCTU” and numerical type values “1999” and “2003”. The two numerical values semantically mean the first year studying in NCTU can the last year studying in NCTU.

Table 3 Examples of 3 attributes with different types

Attribute Name	Necessary	Type	String value	Numerical Value 1	Numerical Value 2
Name	Yes	String	Peter		
Age	No	Numerical		22	
University	Yes	Hybrid	NCTU	1999	2003

3.2.2 Information of a user/callee

When a user joins the system, the user should register his or her information so that he or she could be reached by other users. A user’s information consists of

`

multiple attributes. Table 4 and 5 shows examples of nodes' information. Table 4 describes the attributes for a node belongs to a user named "Peter", and the communication node in Table 5 belongs to a pizza store in HsinChu.

Table 4 Information of a node that belongs to a user

Attribute Name	Necessary	Type	String value	Numerical Value 1	Numerical Value 2
Name	Yes	String	Peter		
Age	No	Numerical		22	
University	Yes	Hybrid	NCTU	1999	2003
Work	No	Hybrid	Asus	2005	2005
Club	No	Hybrid	Chess	2000	2001

Table 5 Information of a node that belongs to a Pizza store

Attribute Name	Necessary	Type	String value	Numerical Value 1	Numerical Value 2
Name	Yes	String	PizzaHot		
Location	No	String	HSIN CHU		

A callee's information should be specified when a user is going to send text message or call request. A callee's information also consists of multiple attributes and its format is similar to a node's information. The difference is that an attribute in a callee's information does not contain a "necessary" component.

`

The attributes composing information of a user/callee could be either system defined attributes or user defined attributes. We provide some usual attributes as system defined attributes for users' convenience and better look-up efficiency. A user could also define new attributes as their wish thus the information is more flexible.

3.2.3 Filtering

One problem of the un-specific attributes based communication is that a receiver may receive advertisement or wrong messages. To diminish this problem, we provide filtering functionality for a user so that a user could receive messages closer to his/her demand.

To filter out unwanted message, a user could set some of a node's attributes as necessary attributes. A message will be routed to the node only if the description of targets matches all necessary attributes. For example, if a node set three attributes: "Name", "Age", and "Habit" as necessary attributes, all the messages that could not match those three attributes will be filtered out. Advertisement messages with target specification with only few common attributes will be blocked if a user sets necessary attributes.

There are other filtering functionalities we could provide, such as a blacklist that blocks all messages from some nodes or only receiving information from some nodes. On the other hand, our system itself would drop messages that sent to too many nodes. If the number of targets of a message exceeds a threshold, the message could be an advertisement that disturb users and the bandwidth loading to process this message is heavy, thus we drops those messages.

`

3.3 Dynamic user behavior handling

Each peer in the system might join or leave at any moment. The dynamic user behavior could break the complete Chord ring structure or cause registered data vanishing. We propose a data duplication scheme to solve this problem.

3.3.1 Join

When a peer joins the system, it shares some data from its successor. The data in the successor's database with smaller identifier than the new peer's identifier are sent to the new peer. The other joining actions follow the joining steps defined by Chord such as building finger table and connection stabling.

3.3.2 Graceful departure

When a peer is leaving the system, it informs its predecessor and successor. Then the leaving peer sends all the data in its database to its successor. To keep the Chord ring complete and help the neighbor peers find new neighbors, the leaving peer sends its successor information to its predecessor and its predecessor information to its successor. Then peer un-registers all its attributes and leaves the system.

3.3.3 Ungraceful departure

A peer may leave the system without processing the formal departure steps, which we call it ungraceful departure. There are two problems caused by ungraceful departure. The first problem is that the neighbors will lose their predecessor or successor connection. The second problem is that we would lose the data stored in the peer.

To make the system with ungraceful departure peers robust, we propose the following strategy. When a peer gets and stores a record in its local database, it also duplicates this record to *k* succeeding peers. And each peer repeatedly sends keep-alive message to its predecessor. If a peer *R* does not accept the keep-alive message from its successor, it will infer that the connection of the successor *S* is lost and requests the next peer *T* of the successor to be its new successor. If the peer *T* is also dropped so that *T* does not reply to be a new successor, *R* shall request the next peer of *T* to be a new successor. This process is repeated till finding a living new successor. Because the next *k* peers of *S* has a copy of the *S*'s data, any one of those peers can replace *S* without losing any information.

There is a case that more or equal to *k* successive peers leave in a period of time-to-live time thus the leaving peer replacement mechanism fails. However, the probability of such a case is proportional to P^k, where *P* is the probability of a peer dropped in a time-to-live period. The number of succeeding peers *k* affects the loading and the robustness of the system. The larger *k* is, the more the loading of the system, but the data robustness is also higher. We set 3 as the default value of *k*.

3.4 Send messages/ Make voice communication

When a CP is trying to send messages to or make voice communication with CR (content receiver), at first, CP should set the attributes of the CR as target specification. The attributes could either be string type, numerical type, or hybrid type. In addition to the attributes provided by the system, sender could use user-defined attributes to describe the target.

After setting the target's specification, the sender could send text messages or request to make voice communication. Of course, the sender could both sends text messages and request to call to the target(s).

The procedure to route the message to the CR is similar to the procedure to find the multiple attributes query result described in chapter 2. We adopt two strategies mentioned in chapter 2 to improve the query efficiency: reorder the list of responsible peers and single attribute dominated query resolution. However, the single attribute dominated query resolution is adjusted to fit the privacy issue and to reduce the storage cost. The range query mechanism is adopted for numerical type attributes publish/query. We also adopt range guard to improve range query efficiency.

3.4.1 Adjustment of single attribute dominating query resolution

The spirit of single attribute dominating query resolution is that it stores all attributes in each responsible peer so a single attribute responsible peer could do complete query without forwarding query to other peers. The shortcoming is that the storage loading is heavy and it violates the privacy-first principle.

We adjust the single attribute dominating query resolution and just the text of all attributes of a node in that node's database and only store the hashing result of a responsible attribute in a responsible peer. When a query is forwarded to the first responsible peer, it takes one more hop to forward the request to matched peers, and the complete query is processed in those matched peers.

Compared to single attribute dominating query resolution, the new resolution just takes one more hop to do complete query. The storage cost of new resolution is reduced because there is only one copy of the text of a node's information. And this copy is stored in node's local database without publishing to the network thus the privacy is kept.

However, to forward the request to matched peers by the first responsible peer might take a lot of bandwidth cost because the number of matched peers might be large. Thus we make a threshold that we only forward the request directly to the matched peers when the number of matched peers is below the threshold, otherwise, forward the request to next responsible peer to check the second attribute.

3.4.2 Range guard

The range query algorithm proposed in section 2.3 takes O(N) to do worst case range query. The worst case happens when the range is almost from MIN to MAX of the numerical attribute value.

Range guard [13] can reduce the worst case query complexity to O($\log N$). This strategy is simple that we choose k peers in the system as range guards, and the range guards collect numerical attributes from nearby peers. Each range query is responsible for a range of numerical value, and each range guard has connection to its predecessor range guard and its successor range guard. When a range query is processing, the query is only forwarded to those range guard thus the worst case query complexity is reduced to O(k). We set k as log N thus the complexity is reduced to O($\log N$).

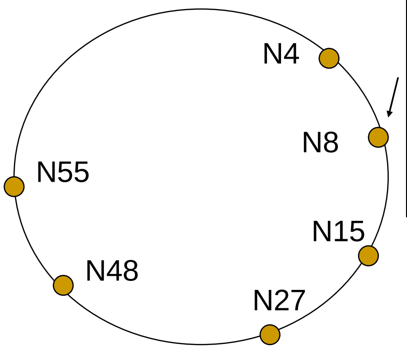
3.4.3 Hybrid type attribute query

The hybrid type attribute contains one string type attribute and two numerical attributes thus we have to consider both the feature of a string type attribute and a numerical type attribute.

Publish

`

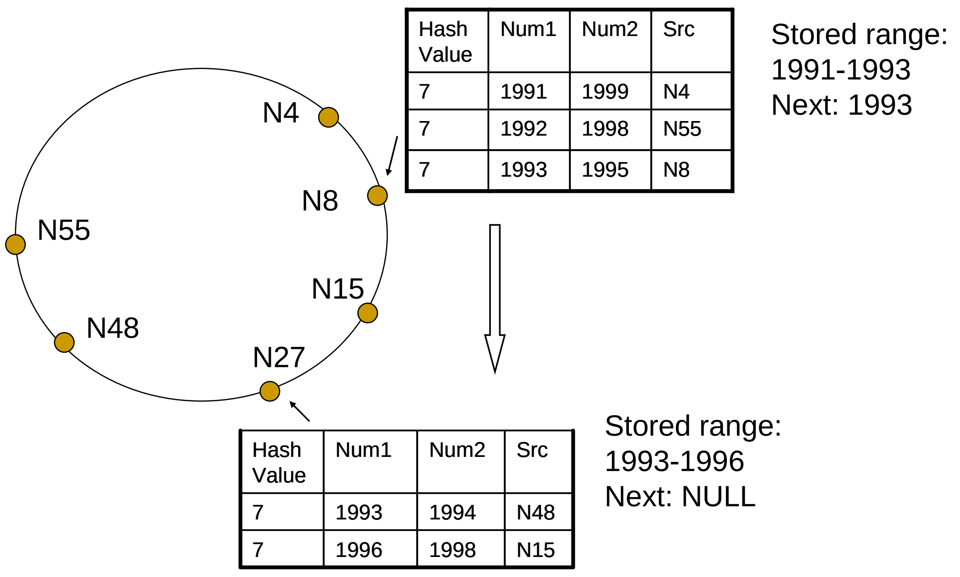
To publish a hybrid type attribute, at first, we hash the string type attribute value by SHA-1 to an m-bits identifier *k* and publish the attribute to a peer *s* with identifier *i* smallest but larger than *k*. If the hybrid data storage in peer *s* exceeds a threshold, peer *s* split half of the hybrid data to a peer with identifier *j*. If *MAX* is the maximum hashing value in the m-bits system and *n_split* be the number of times this peer splits its data, then we calculate *j* by the following function: $j = i + \frac{MAX}{2^{n\_split+1}}$. The hybrid data in a responsible peer is sorted by the first numerical value by progressive order thus the splitting just cut half larger part of the data to another peer. The peer should record the range it is responsible for thus when other data is larger than the range it is responsible for it could forward the data to next peer. In Figure 6, there are 5 records stored in *N8* which just reach the storage threshold and this is a system with 8-bits identifier (*MAX* = 63). Figure 7 shows the slitting of data in *N8*. By the function, the peer responsible for key 24 shares data.



Hash Value	Num1	Num2	Src
7	1991	1999	N4
7	1992	1998	N55
7	1993	1995	N8
7	1993	1994	N48
7	1996	1998	N15

Stored range:
1991-1996
Next: NULL

Fig. 9 An example of a node responsible for hybrid type attribute



Hash Value	Num1	Num2	Src
7	1991	1999	N4
7	1992	1998	N55
7	1993	1995	N8

Stored range: 1991-1993
Next: 1993

Hash Value	Num1	Num2	Src
7	1993	1994	N48
7	1996	1998	N15

Stored range: 1993-1996
Next: NULL

Fig. 10 An example of splitting

Query

The query of hybrid type data is related to the distribution method. At first, send the hybrid type query to a responsible peer by hashing the string type attribute value. The responsible peer checks the type of the attribute thus it knows this query should be handled by hybrid type method. The second step is to check if the numerical value 1 falls in the peer's responsible range. Do local query if the value falls in the responsible range then forward the query to a peer responsible for next range. The query of this attribute is done when the first numerical type value below the responsible range of the peer.

The local query is to check if an attribute A can bound the querying attribute B by its two numerical type values. If num1 of A <= num1 of B and num2 of A >= num2 of B then we say that A can bound B. For example, if a user studied in NCTU since 2002 till 2006, his/her data can match a query for a user having studied in NCTU since 2003 till 2004. To do local query for a hybrid type attribute is to find all the attributes that can bound the querying attribute. After forwarding the query to all peers that responsible for a range of value that could bound the querying attribute, we can get the query results.

3.4.4 Procedures to route messages to targets

The procedures to route messages to targets are similar to the procedures to do multiple attributes query. We do sub-query sequentially in responsible peers and the intersections of those sub-query results are matched targets. The difference is that a responsible peer does not return the query result to the request sender but directly forward the message to the matched targets.

The following steps are procedures to route messages:

Step 1: Hash each attributes to m-bits identifiers. Use SHA-1 hashing if the attribute is string type or hybrid type; use locality preserving function described in section 2.3 if the attribute is numerical type.

Step 2: Ask each responsible peer for the storage data size, reorder the responsible peers list progressively.

Step 3: Forward the message or the call request to responsible peers to the next peer in responsible peers list.

Step 4: In each responsible peer receiving the message or the call request do local database query and intersect the local result with previous query results find the matched peers. The manner of local database query depends on the type of the attribute.

Step 5: If the peer is the final peer on the responsible peer list or if the number of matched targets is less than or equal to a threshold, forward the message to those matched targets and go to step 6, otherwise, go to step 3.

Step 6: When a peer receive a message or call request, it do text check for the attributes to make sure it’s the correct target. Filter out the messages that do not match all the necessary attributes. Store the message or call request if it’s the correct receiver otherwise drop the message.

Chapter 4 Implementation

We have implemented MFPGC system user agent on laptop PCs to verify our design. The operating system of our implementation environment is Windows and each device the runs the MFPGC UA should be able to access the internet. To provide voice communication functionality, we use a SIP UA *cclsip1_6* accomplished by Industrial Technology Research Institute to be the voice communication module and establish a call by SIP.

4.1 Implementation of MFPGC system User Agent

4.1.1 The layers of a MFPGC system User Agent

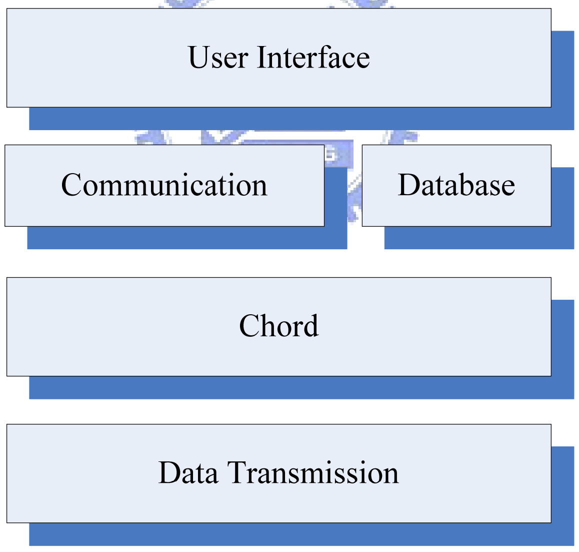


Fig. 11 Layers of a MFPGC system UA

As shown in figure 8, a MFPGC system UA could be partitioned into layers. The bottom layer is data transmitting layer. This layer is responsible for transmitting data between two nodes. Almost all the signaling messages in our system are transport by UDP. The second layer is Chord layer. The main functionality of Chord layer is to

`

route messages to responsible node by DHT algorithm. Chord layer also handles node joining and departure. The third layer from the bottom is communication layer. In this layer, a node can send text message to a group of nodes or make voice communication by SIP with one node. In Database layer, each node has a local database with two tables. One database table stores the attributes of the node and another stores data published by other nodes.

The top layer is User Interface layer. In this layer, we provide multiple interfaces for a user to take use of the system including:

- Node joining and departure buttons
- Node attributes registration interface
- Interface to set target attributes, text messages to be sent, and Call back tag
- Interface that shows the messages and callback tag sent to the node

4.1.2 Event handling of a UA

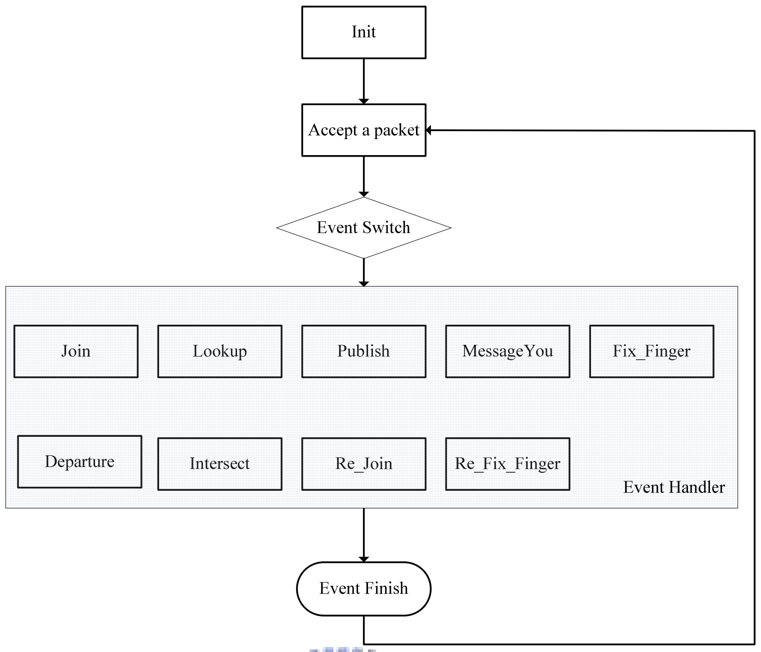


Fig. 12 State diagram of the events handling in a UA

We create a thread after the UA is aroused to receive packets from other nodes. Packets of different events could arrive any time without sequence. We can know the event of the packet by checking the header message attached in the packet. As shown in figure 9, when a packet is received, the application invokes related event handler to handle this packet. After handling the packet, this thread goes back to waiting other packets. We describe the job of each event handler as follows:

Join handler：When a node A accepts “Join” message from node B, it means B wants to join MFPGC system and be a new predecessor of A. Node A sets node B as predecessor and replies “Rep_Join” message to node B.

Rep_Join handler：When a node receives “Rep_Join” message, it means the joining procedure succeed and a suitable successor is found. The information of the successor is attached in the message.

`

Lookup handler：The packets with “Lookup” header take use of the node to forward to destination by Chord lookup algorithm. The peer accepting this message checks the finger table and forwards this message to next peer closer to the destination.

Fix_finger and Rep_Fix_Finger handler：A peer receiving “Fix_finger” message responses its ID and IP in “Rep_Fix_finger” message to be refresh the sender’s finger table.

Publish handler： “Publish” message brings published data to responsible node, thus in this handler we parse data from the packet and store it in a local database.

MessageYou handler：When a peer receives “MessageYou” packet, it means the peer is a candidate of message sender’s target. This packet contains the specification of the target, message sent to target, and maybe a call-back for the target to do voice communication. In this function, the peer checks the target’s specification to decide if it should keep the message or drop it. User could see the message or a call-back tag on the user interface, and he/she could further communicate with the message sender.

Departure handler：This message is sent when a peer leaves gracefully and it’s looking for a peer to succeed its responsible data. The peer received this message moves the duplicated data from leaving peer to its responsible data table and changes the predecessor connection to correct new predecessor.

Intersect handler： This type of packet contains the previous matched peers’ ID and the target specification composed by multiple attributes. In this function, the peer checks the specification to know the lookup key value and do local database query to find the matched IDs. The peer then intersects the matched IDs with previous matched

IDs. If the number of matched IDs is below the threshold, change the header of the packet to “MessageYou” and then multicast the packet to matched peers. Send the packet to next responsible peer, otherwise.

Chapter 5 Conclusions and Future Work

5.1 Conclusion

In this thesis we design a new communication system using unspecific attributes to specify the called party. We propose a structure P2P solution for this system and implement the system on PCs to verify our design. Our system support multiple attributes query and range query. We also developed an algorithm to support hybrid type attribute query. The dynamic behavior of nodes (joining and departure) is also considered to keep the data storing in the system more consistent. We provide a total solution for MFPGC system that is efficient and keeping user privacy.

Compared to traditional communication system, MFPGC system provides more flexible scenarios for one-to-one and one-to-many communication. A user can communicate with friends without knowing the specific IDs and a content receiver can receive interested information by setting attributes. The system design we proposed is scalable and robust even with users leave ungracefully.

5.2 Future Work

There are more features that could be combined with MFPGC system to improve the capability. An example is to support location information as attributes. The location information could be provided by orientating device, such as GPS. With locality information, a user could communicate with targets in a specific area. For example, a user can send message to all the users nearby or call someone who is in an interesting area to collect information. To support location information is not straight forward. A mobile device could change its location constantly thus the registered location should be updated frequently. It requires further investigation to efficiently maintain dynamic attributes in a P2P system.

Other future work could be to provide better security to avoid malicious attacking or billing mechanism. An AAA (authentication, authorization, accounting) mechanism can provide better protection for the users in the system and also critical for further billing system design.

Reference

[1] S. Baset and H. Schulzrinne. "An analysis of the skype peer-to-peer internet telephony protocol." Technical Report CUCS-039-04, Computer Science Department, Columbia University, New York, NY, Sep 2004.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. "Schooler, SIP: Session Initiation Protocol RFC 3261", June 2002.

[3] Min Cai , Martin Frank , Jinbo Chen , Pedro Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services", Proceedings of the Fourth International Workshop on Grid Computing, p.184, November 17-17, 2003

[4] Ion Stoica , Robert Morris , David Karger , M. Frans Kaashoek , Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, p.149-160, August 2001, San Diego, California, United States

[5] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. "Complex queries in DHT-based peer-to-peer networks". In Proceedings of the first International Workshop on Peer-to-Peer Systems, pages 242 – 250, 2002.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A scalable content-addressable network." In Proceedings of the 2001 ACM SIGCOMM, pages 161 – 172.

[7] A. Bharambe, M. Agrawal, and S. Seshan. "Mercury: Supporting scalable multi-attribute range queries." In Proc. SIGCOMM, 2004.

[8] M. Ripeanu. "Peer-to-peer architecture case study: Gnutella network." Technical report, University of Chicago, 2001.

[9] B. Zhao, J. Kubiatowicz and A. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report UCB/CSD-01-1141, 2001.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," Lecture Notes in Computer Science, Vol. 2218, 2001.

[11] F. Kaashoek and D.R. Karger, "Koorde: A Simple Degreeoptimal Hash Table," in The 2nd International Workshop on Peer-to-Peer Systems (IPTPS ' 03), 2003.

[12] O. D. Sahin , A. Gupta , D. Agrawal , A. El Abbadi, "A Peer-to-peer Framework for Caching Range Queries", Proceedings of the 20th International

`

Conference on Data Engineering, p.165, March 30-April 02, 2004

[13] Peter Triantafillou, Nikos Ntarmos, Theoni Pitoura, “The RangeGuard: Range Query Optimization in Peer-to-Peer Data Networks”, 3rd Hellenic Data Management Symposium, HDMS'04, June 2004.

[14] Burton H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM, v.13 n.7, p.422-426, July 1970

`