Journal of Engineering Science and Technology Review 6 (5) (2013) 129 - 136

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Research Article

On Analysis of a Chord-based Traffic Model for Web Service Discovery in Distributed Environment

Cong Gao*, Jianfeng Ma and Xiaohong Jiang

Shaanxi Key Laboratory of Network and System Security, School of Computer, Xidian University, Xi'an, 710071- China

Received 6 October 2013; Accepted 28 December 2013

Abstract

Recent years have witnessed a thriving development of web service all over the world. Almost every imaginable service in our everyday life becomes available online. With the increasing number of web services, a lot of works has been done with regard to the issue of web service discovery. The prevailing service discovery architectures are centralized architecture and distributed architecture. Since the former is considered to be weak at robustness and scalability, a significant research attention is focused on the latter one. However, most of the existing works do not cover the modeling of traffic. To address this problem, we in this paper present a Chord-based traffic model for web service discovery in distributed environment. The proposed model analyzes the node behavior through the operation of five queues. The traffic of a node is regulated by a traffic management module. Different traffic management policies are analyzed for the purpose of improving the availability and latency of service discovery in a Chord network. In order to evaluate our model, we conduct extensive experiments and make an analysis of the simulation results.

Keywords: Discovery, Web Service, Service Information, Traffic Modeling, Peer-to-peer Networks

1. Introduction

With the increasing deployment of web service on the Internet, the issue of web service discovery becomes a widely-recognized topic in the research area [1], [2] and [3]. Generally, there are two architectures for web service discovery: centralized architecture and distributed architecture. The centralized architecture [4] stores the information of web service in a centralized manner. Two popular centralized mechanisms are registry and index. The web service information stored in a registry is authoritatively controlled by the registry owner. A famous example of registry is UDDI [5]. Due to several important technical drawbacks, the UDDI is considered to be imperfect; and a lot of improvements have been made [6], [7] and [8]. An index collects the web service information from the Internet. Unlike a registry, an index does not control what information is provided to the users. A well-known example of the index is the search results given by a search engine. Due to some intrinsic shortcomings of a centralized architecture, such as bottle neck and robustness, the researchers have done many works for the distributed architecture, such as [9], [10], [11] and [12]. All these works are based on a P2P network. The advantages of P2P network accord the above contributions scalability and reliability. However, the modeling and analysis of traffic is not involved.

In this article, we address the issue of web service discovery by introducing a Chord-based traffic model. In the proposed model, a node contains five queues: incoming queue, answer queue, forward queue, query queue and outgoing queue. For each node, there is a traffic management module which regulates the dequeue operations of the answer queue, the forward queue and the query queue. For the traffic management module, it is convenient to conduct a policy making based on the percentages of processing ability which are allocated to deal with the above three queues. In a word, our model provides a way to optimize the availability and latency of service discovery in a Chord network by modeling the node behavior.

The reminder of this article is organized as follows. Section 2 provides a brief review of the Chord protocol. Section 3 details our traffic model and the traffic management policy. In Section 4, we evaluate the proposed model by simulation and present an analysis of the simulation results. Finally, we make conclusions and highlight the future work in Section 5.

2. Review of Chord

In computer network, a distributed environment is often implemented as a peer-to-peer (P2P) network. According to the connection mode of the nodes and the way of resource allocation, P2P networks are categorized as unstructured and structured (or a hybrid type combing the two) [13]. In unstructured peer-to-peer networks, there is no specific structure for the nodes. All connections among the nodes are randomly formed [14]. There are many famous protocols for unstructured P2P networks, such as Gnutella [15], Gossip [16] and Kazaa [17]. Since there is no structure in

* E-mail address: glidergao@gmail.com

unstructured P2P networks, a high robustness could be achieved and the effect of single point failure is low. However, the flooding problem which could cause a large scale of congestion for the whole network is a major drawback of unstructured P2P networks. In a structured P2P network, nodes are regulated by a structured model. The most notable model is the distributed hash table (DHT) [18]. The nodes in a DHT-based P2P network could issue a query using a hash table. Literatures have proposed plentiful protocols for structured P2P networks, such as Pastry [19], Chord [20], Tapestry [21] and Kademlia [22].

In particular, our traffic model is based on the Chord protocol. The basic idea of Chord is as follows. The information of a resource is stored on a specific node in the form of key-value pair. A query is supposed to be answered by the node which is responsible for the key contained in the query; and a reply which contains the corresponding value of the key will be sent by the answering node. All nodes and resources are uniquely mapped to a 2^m identifier space using consistent hashing. Suppose there is a consistent hash function $cHash(str)$ whose return value is always an m-bit binary string irrespective of what the independent variable str is. Thus, the lengths of the id for a node and the key for a resource are both m-bit. Conventionally, the id of a node is obtained by $cHash(ip)$, where ip is the IP address of the node. The key of a resource is obtained by $cHash(value)$, where $value$ is the resource name or location. For example, the $value$ of a web service may be the corresponding URL of the web service. The values of a node id and a resource key are both in the range of $[0, 2^m - 1]$.

The structure of a Chord network could be logically interpreted by an abstract model which is called the Chord ring. For a Chord ring where the identifier space is 2^m, there are 2^m uniformly distributed positions on the Chord ring which are labeled from 0 to $2^m - 1$ in the clockwise direction. Nodes are located on the Chord ring according to their ids. Each node on the Chord ring has a successor and a predecessor. By the successor of a node we mean the next node on the Chord ring in the clockwise direction; while the predecessor of a node is the next node in the counter-clockwise direction. Keys are assigned to nodes according to the rules given below. Considering the existing nodes on the Chord ring, a key is assigned to the first node whose id is greater than or equal to the key. The lookup operation of a key is implemented based on a component which is called finger table. For a node n in the Chord ring where the identifier space is 2^m, the detailed definition of its finger table is given in [20]. On average, a query will be forwarded $\frac{1}{2}\log n$ times before reaching its destination [23]; and this is a remarkable performance.

Moreover, the Chord protocol is resilient to node failure and could handle node join smoothly. Since this paper is not focused on explaining the operation of Chord, we omit the details such as key lookup, stabilization of node join/failure and load balance. In order to facilitate the presentation of our model. We briefly make the following summary for the operation of key lookup. A query travels on the Chord ring in the clockwise direction. If an intermediate node receives a query which it is unable to answer, it will forward the query according to the specification of the Chord protocol. When the query arrives at a node which is able to answer it, the node will send a reply designated to the corresponding node which issued the query.

3. Node Model and Traffic Management Policy

3.1 Node Model

In this section, the node behavior is modeled in order to facilitate the analysis of the traffic in a Chord network. The fundamental concept we employed is that the node behavior is driven by discrete events. For simplicity, we only consider two types of messages in the Chord network: a query message and a reply message. We set up five message queues in a node: query queue, incoming queue, answer queue, forward queue and outgoing queue. All these queues possess the classic property of First-In-First-Out (FIFO). In our node model, there are two interfaces that node n interacts with other nodes in the Chord network: the incoming queue and the outgoing queue. Considering a node n in the Chord network, we give the definitions of the above five queues and draw a flow chart of messages in Fig. 1.

Definition 1. Incoming Queue IQ_n. All messages which are about to flowing into node n are enqueued in the incoming queue. It is the only interface through which other nodes in the Chord network could interact with node n. We denote the total number of incoming messages by mi_n. Suppose there are qi_n query messages and ri_n reply messages. Trivially, we have $mi_n = qi_n + ri_n$.

Definition 2. Outgoing Queue OQ_n. All messages which are about to flowing out node n are enqueued in the outgoing queue. It is the only interface through which node n could interact with other nodes in the Chord network. We denote the total number of outgoing messages by mo_n. Suppose there are qo_n query messages and ro_n reply messages. Trivially, we have $mo_n = qo_n + ro_n$.

Definition 3. Query Queue QQ_n. The query messages issued by node n are enqueued in the query queue. The number of query messages issued by node n is denoted by q_n.

Definition 4. Answer Queue AQ_n. Among the incoming query messages, node n might be able to answer some of them. We denote the number of these messages by qr_n, where $0 \le qr_n \le qi_n$. And these query messages are enqueued in the answer queue.

Definition 5. Forward Queue FQ_n. The portion of the incoming query messages which node n is unable to answer is enqueued in the forward queue. We denote the number of these messages by qf_n, where $0 \le qf_n \le qi_n$. Trivially, we have $qi_n = qr_n + qf_n$. In addition, the portion of the incoming reply messages which are not designated to node n is also enqueued in the forward queue. We denote the number of these reply messages by rf_n, where $0 \le rf_n \le ri_n$. Suppose the number of reply messages which are designated to node n is denoted by rs_n. Then, we have $ri_n = rf_n + rs_n$.

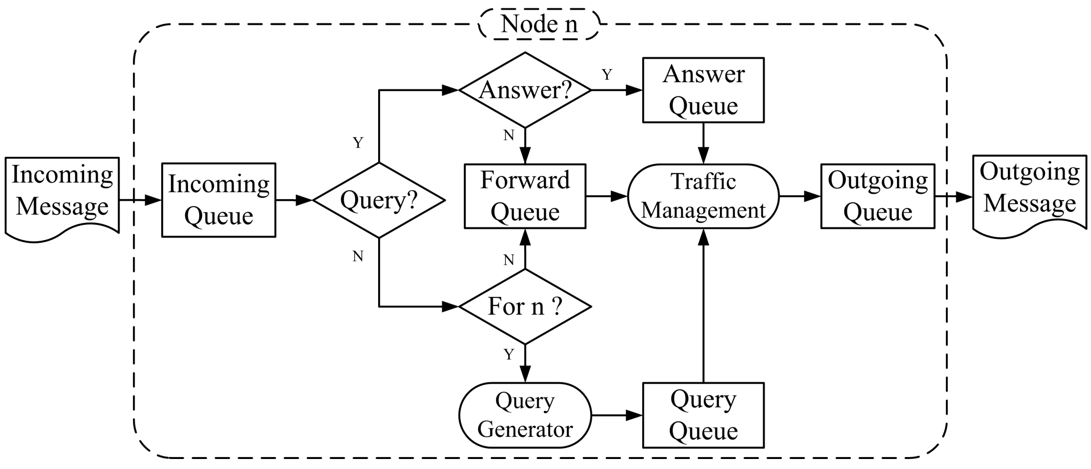

Fig. 1. Flow chart of messages

3.2 Traffic Management Policy

In order to facilitate the presentation of our model, we formulate the processing ability of node n as

$$pa_n = \frac{mi_n + mo_n}{time\ period}.$$

For an individual node n, the value of pa_n is constant. In case of a severe denial of service (DoS) attack, the processing ability of node n might be exhausted by incoming messages, then there are no effective outgoing messages, namely $mo_n = 0$. However, we in this paper assume that the values of mi_n and mo_n are both in normal range. Furthermore, we assume that node n always has sufficient processing ability to deal with the incoming messages. Thus, our traffic model is focused on the analysis of processing ability which deals with outgoing messages. We denote this part of processing ability by poa_n and suppose the value of poa_n is constant in each time step.

Considering three queues QQ_n, AQ_n and FQ_n, in each time step, the number of messages dequeued from the above three queues are denoted by qq_n, aq_n and fq_n, respectively. And the sum of the three numbers could not exceed the value of poa_n, namely

$$qq_n + aq_n + fq_n \le pao_n.$$

Since all the messages dequeued from QQ_n, AQ_n and FQ_n are supposed to enqueue in OQ_n, it is necessary that there is a traffic management module in node n. In order to maximize the utilization of the processing ability poa_n, the traffic management module should dynamically allocate the processing ability to meet different dequeue demands of the three queues. More importantly, the priority of dequeue operations of the three queues should also be considered for the purpose of improving the performance of the Chord network. In short, the essential function of the traffic management is to regulate the enqueue operation of OQ_n.

For an incoming query message q which is dequeued from IQ_n, node n checks its own database to determine whether it could be answered. If node n is able to answer it, the query message is enqueued in AQ_n. It is supposed that when the query message q is dequeued from AQ_n, the corresponding reply message r will be constructed and enqueued in OQ_n. If node n is unable to answer the query message q, node n should forward it to another node in the Chord network according to the specification of the Chord protocol. That is to say, the query message q is enqueued in FQ_n. It is supposed that when the query message q is dequeued from FQ_n, it will be enqueued in OQ_n. Similarly, for a query message q which is issued by node n, it is supposed that the query message q will be enqueued in OQ_n when it is dequeued from QQ_n.

For an incoming reply message r which is dequeued from IQ_n, node n checks whether it is designated to itself. If the reply message r is designated to node n, it is handed over to the query generator module for the purpose of resolving the corresponding query message generated previously. Otherwise, node n should forward it to another node in the Chord network according to the specification of the Chord protocol. That is to say, the reply message r is enqueued in FQ_n. It is supposed that when the reply message r is dequeued from FQ_n, it will be enqueued in OQ_n.

As mentioned above, all the query messages and reply messages dequeued from QQ_n, AQ_n and FQ_n are supposed to enqueue in OQ_n. We normalize the total processing ability poa_n which deals with OQ_n by one. The percentages of processing ability allocated for QQ_n, AQ_n and FQ_n are denoted by Q_n, A_n and F_n, respectively. Then, we have

$$Q_n + A_n + F_n = 1,$$

where $Q_n \ge 0$, $A_n \ge 0$ and $F_n \ge 0$. In order to analyze the performance of the Chord network, we choose two primary metrics availability and latency. Specifically, they are defined as follows.

Definition 6. By the availability of node n we mean the answer rate of the query messages issued by node n. We denote the availability of node n by av_n. Note that the term answer indicates that for a query message issued by node n, the corresponding reply message is received by node n. The availability of the network is given by the average availability of all nodes in the network; and it is denoted by AV_n.

Definition 7. By the latency of a reply message designated to node n we mean the time steps between the time when the corresponding query message is sent by node n and the time when the reply message is received by node n. We denote the latency of a reply message r by la_r. If node n dose not receive the corresponding reply message of a query message, the latency is considered to be infinite. Thus, we do not take it into account. The latency of the network is given by the average latency of all reply messages in the network; and it is denoted by LA_n.

Based on the definitions of availability and latency, we analyze how they are influenced by the three percentage parameters mentioned above. Given a time period $[t_a, t_b]$, where $t_b > t_a$, let us consider a query message q which is issued by node n. For all the queues involved, we assume that there are no enqueue/dequeue operations of other messages except for the query message q and its corresponding reply message r.

As illustrated in Fig. 2, the query message q is answered by node m, and node k is the only intermediate node between node n and node m. Briefly speaking, if $t_b \geq t_a + 11$, the corresponding reply message r of the query message q will eventually be received by node n. In this case, the current availability of node n is $av_n = 100\%$; and the latency of the reply message r is $la_r = 11$. When $t_b < t_a + 11$, node n will not receive the reply message r. In this case, the current availability of node n is $av_n = 0\%$. Since the reply message r is not received by node n, there is no value of the latency.

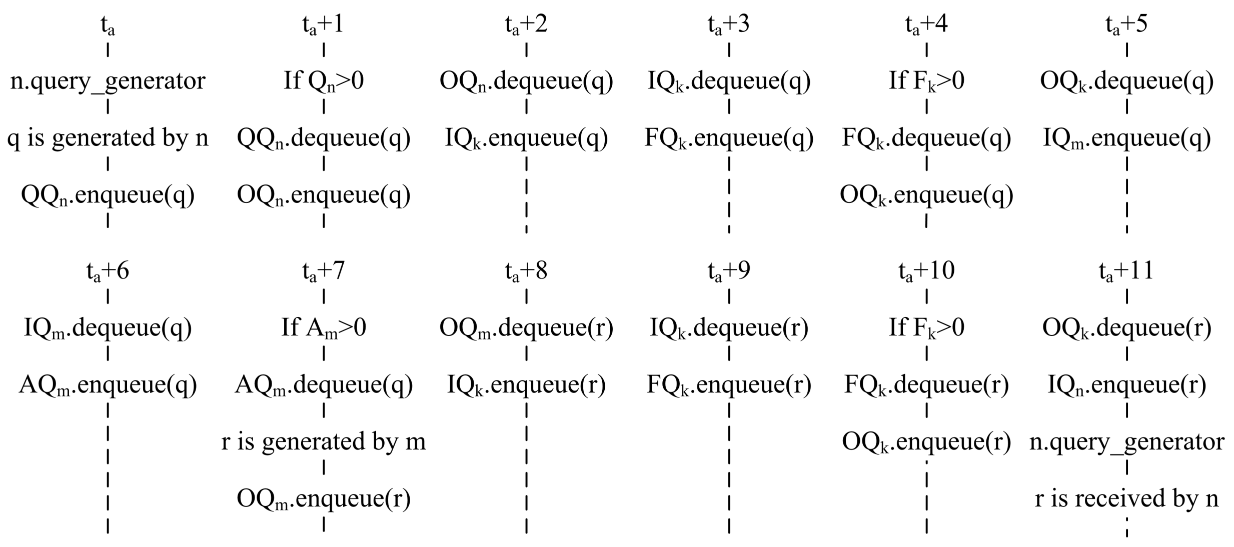

Fig. 2. Sequence chart of the query and answer process

There are three percentage parameters in Fig. 2: Q_n, F_k, and A_m. Suppose $Q_n = 0$ at $t_a + 1$, for the query message q, the dequeue operation of QQ_n and the subsequent enqueue operation of OQ_n become unable to execute. The two operations are postponed until $Q_n > 0$. This introduces an extra delay, and we denote it by t_{d1}. Similarly, when $F_k = 0$ at $t_a + 4$, $A_m = 0$ at $t_a + 7$ and $F_k = 0$ at $t_a + 10$, we denote the extra delays by t_{d2}, t_{d3} and t_{d4}, respectively. If $t_b \geq t_a + 11 + t_{d1} + t_{d2} + t_{d3} + t_{d4}$, the reply message r will be received by node n. Then, the latency of r is $la_r = 11 + t_{d1} + t_{d2} + t_{d3} + t_{d4}$.

Suppose there are h intermediate nodes between the querying node n and the answering node m, they are denoted by set $\{k_1, k_2, \mathrm{L}\ , k_h\}$. Before a query message q arrives at the answering node m, there exist $h+1$ possible points which could introduce an extra delay to the query message. We denote them by the node set $\{n, k_1, k_2, \mathrm{L}\ , k_h\}$. For the querying node n, the value of Q_n determines whether the sending of q is postponed. We denote the possible delay of q introduced by Q_n by t_{dn}, then the actual delay of q at the querying node n could be computed as

$$t_n = (Q_n > 0) ? 0 : t_{dn}.$$

For an intermediate node k_i which forwards q, the value of F_{k_i} determines whether the forwarding of q is postponed. We denote the possible delay of q introduced by F_{k_i} by t_{dk_i}, then the actual delay of q at the intermediate forwarding node k_i could be calculated as

$$t_{k_i} = (F_{k_i} > 0) ? 0 : t_{dk_i}.$$

Likewise, before the reply message r arrives at the querying node n, there also exist $h+1$ possible points which could introduce an extra delay to the reply message r. We denote them by the node set $\{m,k_1,k_2,\mathrm{L}\ ,k_h\}$. For the answering node m, the value of A_m determines whether the sending of r is postponed. We denote the possible delay of r introduced by A_m by t_{dm}, then the actual delay of r at the answering node m could be computed as

$$t_m = (A_m > 0)?0 : t_{dm}.$$

For an intermediate node k_i which forwards r, the value of F'_{k_i} determines whether the forwarding of r is postponed. We denote the possible delay of r introduced by F'_{k_i} by t'_{dk_i}, then the actual delay of r at the intermediate forwarding node k_i could be calculated as

$$t'_{k_i} = (F'_{k_i} > 0)?0 : t'_{dk_i}.$$

Hence, according to the definition of latency, the latency of a reply message r could be computed as

$$la_r = t_n + \sum_{i=1}^{h} (t_{k_i} + t'_{k_i}) + t_m.$$

Suppose there are g query messages sent by node n during the time period $[t_a,t_b]$. We denote them by the set $\{q_1,q_2,\mathrm{L}\ ,q_g\}$. For these g query messages, node n received g' corresponding reply messages during the time period $[t_a,t_b]$. We denote the received reply messages by the set $\{r_1,r_2,\mathrm{L}\ ,r_{g'}\}$, where $g' \le g$. Then, the availability of node n could be calculated as follows:

$$av_n = \frac{\sum_{i=1}^{g'} ((t_a + la_{r_i} \le t_b)?1:0)}{g}.$$

Since the processing ability which deals with outgoing messages is limited, the traffic management module should coordinates the values of Q_n, A_n and F_n for the purpose of improving availability and latency. In addition, the priority of dequeue operations of QQ_n, AQ_n and FQ_n should also be considered for the same reason. In general, for a Chord network, a significant amount of traffic volume is playing the relay function. Thus, we hold that F_n is the largest one. A small Q_n may cause a backlog in QQ_n, then the latency of the network increases and the availability of node n decreases consequently. A small A_n may lead to a backlog in AQ_n, then the availability of other nodes decreases and the latency of the network increases as well. In order to make a compromise, we hold that the values of Q_n and A_n are equal.

Moreover, different priority schemes are designed based on the above view. Let us take $F_n > A_n > Q_n$ for example, when FQ_n is not empty and the F_n processing ability for FQ_n is used up, the traffic management module of node n will try to expropriate the processing ability for QQ_n, regardless of whether QQ_n is empty. In case the Q_n processing ability for QQ_n is also used up, the traffic management module of node n will try to expropriate the processing ability for AQ_n, regardless of whether AQ_n is empty. Similarly, when AQ_n is not empty and the A_n processing ability for AQ_n is used up, the traffic management module of node n will try to expropriate the processing ability for QQ_n, regardless of whether QQ_n is empty. When both F_n and A_n are used up, since the priority of F_n is higher than that of A_n, as long as the traffic management module keeps expropriating the process ability for QQ_n to meet the demand of FQ_n, the demand of AQ_n is deferred. An extreme case is that all the processing ability for AQ_n and QQ_n is expropriated for the purpose of dealing with FQ_n. In this case, node n only forwards the query messages and reply messages it received; and does not issue query messages or send its own reply messages. Table I shows the total six combinations of priority for the three percentage parameters.

Table I. Priority Schemes

	Priority
P_1	$Q_n > A_n > F_n$
P_2	$Q_n > F_n > A_n$
P_3	$A_n > Q_n > F_n$
P_4	$A_n > F_n > Q_n$
P_5	$F_n > Q_n > A_n$
P_6	$F_n > A_n > Q_n$

4. Simulation and Analysis

4.1 Experimental Environment

In order to evaluate our traffic model, we develop a Chord-based simulation system to simulate the environment of web service discovery. In communication and computer network research, there are many prevailing network simulators. Examples of notable network simulator are NS-2 [24], NS-3 [25], OPNET [26], OMNeT++ [27] and NetSim [28]. To our best knowledge, OMNeT++ is the most appropriate modular architecture to simulate a P2P network. Hence, we choose OMNeT++ as the base platform to implement our system. All simulation results are obtained on HP with Inter Core2 Q9550 2.83GHz, 4GB RAM with Debian 2.6.32-48squeeze1 (Linux version is 2.6.32-5-686) and gcc 4.3.5 (Debian 4.3.5-4).

4.2 Simulation Parameters

In our system, the identifier space of the Chord network is $2^m = 2^{10} = 1024$. We use the consistent hash function SHA-1 [29] to calculate node ids and keys of web services.

A node id is obtained through hashing its IP address. We generate 20 nodes which are randomly located on the Chord ring. The information of web services are extracted from the QWS Dataset 2.0 [30] which contains 2507 real web services on the Internet. For simplicity, we randomly select 600 web service records from the dataset. The key of a web service is obtained through hashing the URL of the web service. The 600 keys of the web services are assigned to the 20 nodes according to the specification of the Chord protocol; and this process is beyond the scope of this paper. The ids of the 20 nodes and the numbers of web service records for which each node takes responsibility are shown in Table II.

Table 2. Information of the 20 Nodes

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Id	17	32	85	123	214	496	566	594	595	630	641	645	677	747	788	865	884	913	951	1006
Record	20	10	28	18	42	170	42	14	0	30	4	2	14	40	18	44	26	22	24	32

For node n_i and its predecessor n_j on the Chord ring, if the ids of node n_i and node n_j are the smallest one and the largest one of the existing nodes respectively, the distance between n_i and n_j is computed as

$$dis_{ij} = 2^m - n_j.id + n_i.id\,.$$

Otherwise, we calculate the distance between them by

$$dis_{ij} = n_i.id - n_j.id\,.$$

As shown in Table II, the distance between node 6 and its predecessor node 5 on the Chord ring is $dis_{65} = 282$, and this value is much larger than that of other nodes between their corresponding predecessors. Consequently, the number of web service records for which node 6 takes responsibility is also larger than that of other nodes. In specific, we define the responsibility ratio RR_n as the ratio of the number of web service records for which node n_i takes responsibility and the distance between n_i and its predecessor n_j; and it is calculated as

$$RR_n = \frac{|n_i.records|}{dis_{ij}}\,.$$

As depicted in Fig. 3, the responsibility ratios of the 20 nodes are marked as blue crosses. The linear regression equation of them is drawn in red. Most of the nodes possess a responsibility ratio which is within the range of $[0.4, 0.8]$, except for node 9, 10, 11 and 17. The responsibility ratio of node 11 is close to 0.4. For node 8 and node 9 whose ids are 594 and 595 respectively, since node 9 is too close to its predecessor node 8, node 9 takes no responsibility for the web service records in our case.

4.3 Numerical Results and Analysis

For the evaluation of our traffic management model, we employ an empirical baseline setting of $F_n = 60\%$ and $Q_n = A_n = 20\%$. Each node randomly generates queries for the 1000 web services. The simulation is conducted under three priority schemes P_2, P_4 and P_6 within 200 time steps. When the simulation stops, it is likely that there are still query messages and reply messages flowing in the network. However, our evaluation does not take this part of messages into account. That is to say, we just leave them behind. Based on extensive experiments, the availability and latency of the Chord network is illustrated in Fig. 4 and Fig. 5, respectively.

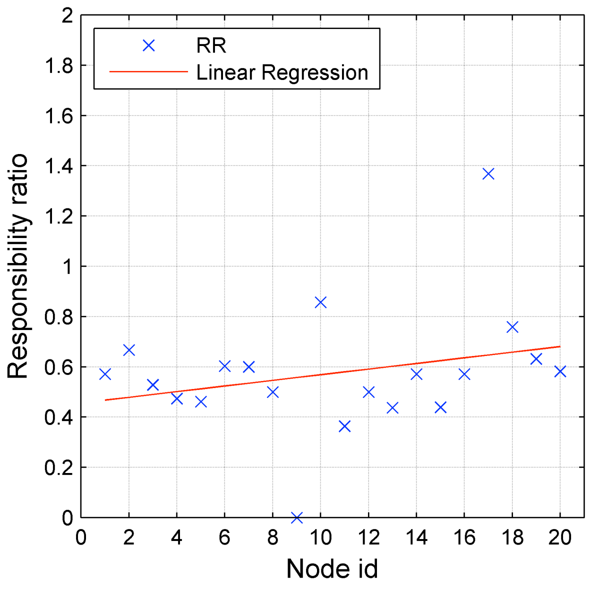

Fig. 3. Responsibility ratio of the 20 nodes

As shown in Fig. 4, the availability of the network under the above three priority schemes increases as time goes by. For availability of the network, the overall performances of the three priority schemes are ranked as $P_6 > P_4 > P_2$. In the early stage of the simulation, since query messages and reply messages are in transmitting phrase, the availability of the network keeps at zero. For P_6, P_4 and P_2, the first non-zero values arise around the time steps 50, 90 and 110, respectively. To some degree, the time of the first non-zero value also indicates a judgment of the performance of the three priority schemes in terms of latency. At the time step 200, the priority scheme P_6 shows the highest availability of the network which is $AV_n = 69\%$. The lowest availability of the network at the time step 200 is given by the priority scheme P_2 whose $AV_n = 30\%$.

As depicted in Fig. 5, the latency of the network under the three priority schemes is also increasing with the increase of the time step. Moreover, the rank of overall performances in terms of latency is $P_6 > P_4 > P_2$, which is the same with availability. Since the availability of the network is zero in the early stage of the simulation, the latency of the network cannot be calculated; then we show the value of latency by zero. For each priority scheme, the occurrence of the first non-zero value of latency coincides with that of availability. At the time step 200, the priority scheme P_6 shows the smallest latency of the network which is $LA_n = 145$. The largest latency of the network at the time step 200 is given by the priority scheme P_2 whose $LA_n = 231$.

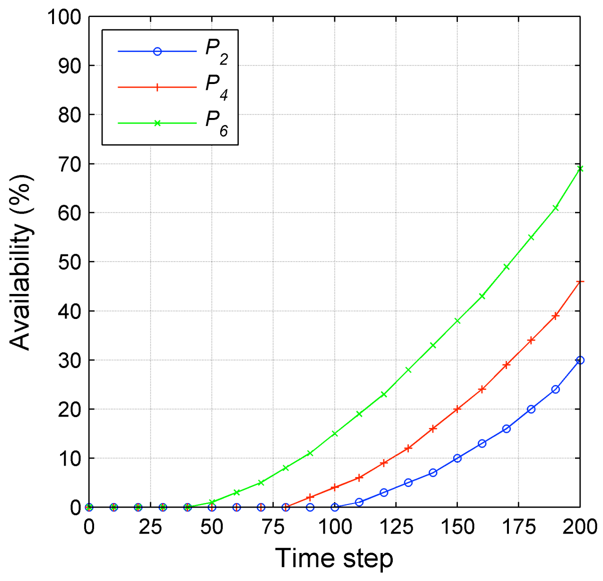

Fig. 4. AV$_n$ vs. time step

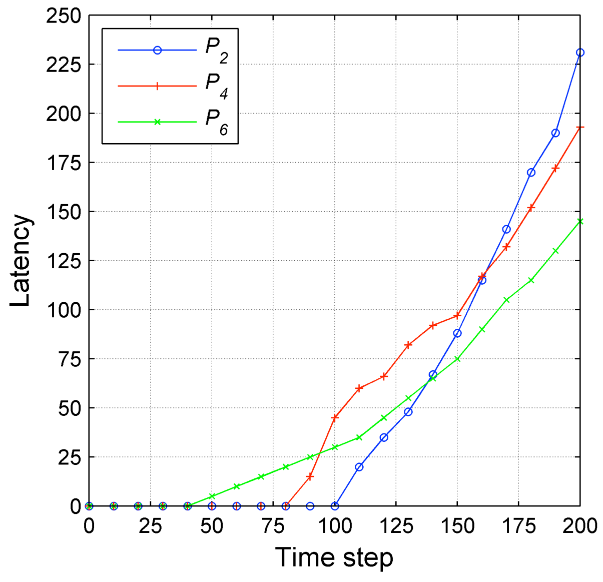

Fig. 5. LA$_n$ vs. time step

Based on the analysis of Fig. 4 and Fig. 5, we conclude that the comprehensive performances of the three priority schemes are ranked as $P_6 > P_4 > P_2$. In specific, the ranking result indicates that F_n is the dominate parameter; and it should be assigned with the highest priority. In the cases of P_4 and P_2, where F_n is assigned with the second priority, A_n is much more important than Q_n.

5. Conclusion and Future Work

This work presents a traffic model which is based on the Chord protocol. The traffic model focuses on the node behavior in a Chord network. Our model considers two major types of message during the process of web service discovery: query message and reply message. In order to facilitate the presentation of the traffic model, five message queues are designed in an individual node. The centerpiece of our model is the traffic management module in a node. The traffic management module is responsible for regulating the dequeue operations of the answer queue, the forward queue and the query queue. In other words, it controls the enqueue operation of the outgoing queue. We build a simulation system based on OMNeT++ and evaluate our model. The simulation results are analyzed in terms of availability and latency. Finally, we come to the conclusion that the percentage of processing ability which deals with the forward queue is of most importance; and it should be assigned with the highest priority. The second one and the third one are the percentages of processing ability which handle the answer queue and the query queue, respectively. However, there is room for improvement. In practice, both a query message and a reply message often possess different priorities which indicate different urgent degrees. The proposed model does not consider the priority of messages during the enqueue and dequeue operations. Consequently, the analysis of messages with different urgent degrees is absent. Moreover, in order to conduct a further in-depth study, the forward queue should be divided into two queues which deal with the query messages to be forwarded and the reply messages to be forwarded, respectively.

Acknowledgment

This work is supported by Program for the Key Program of NSFC-Guangdong Union Foundation (U1135002), Major national S&T program (2011ZX03005-002), National Natural Science Foundation of China (60872041, 61072066), the Fundamental Research Funds for the Central Universities (JY10000903001, JY10000901034, K5051203010) and the GAD Pre-Research Foundation (9140A15040210HK61).

References

1. Benatallah, Boualem, et al. "On automating Web services discovery." The VLDB Journal 14.1 (2005): 84-96.
2. Nayak, Richi. "Facilitating and improving the use of Web services with data mining." Research and trends in data mining technologies and applications (2007): 309-327.
3. Wang, Hongbing, et al. "Web services: problems and future directions." Web Semantics: Science, Services and Agents on the World Wide Web 1.3 (2004): 309-320.
4. Al-Masri, Eyhab, and O. H. Mahmoud. "Discovering web services in search engines." Internet Computing, IEEE 12.3 (2008): 74-77.
5. Org, U. D. D. I. "UDDI technical white paper." 2000O09O06)[2005O09O27]. http://www. uddi. org/pubs/Iru_UDDI_ Technical_White_Paper. pdf/20000906. html (2000).

6. Vu, Le-Hung, Manfred Hauswirth, and Karl Aberer. "QoS-based service selection and ranking with trust and reputation management." On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE. Springer Berlin Heidelberg, 2005. 466-483.
7. ShaikhAli, Ali, et al. "Uddie: An extended registry for web services." Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on. IEEE, 2003.
8. Ananthanarayana, V. S., and K. Vidyasankar. "Dynamic primary copy with piggy-backing mechanism for replicated UDDI registry." Distributed Computing and Internet Technology. Springer Berlin Heidelberg, 2006. 389-402.
9. Schmidt, Cristina, and Manish Parashar. "A peer-to-peer approach to web service discovery." World Wide Web 7.2 (2004): 211-229.
10. Emekci, Fatih, et al. "A peer-to-peer framework for web service discovery with ranking." Web Services, 2004. Proceedings. IEEE International Conference on. IEEE, 2004.
11. Papazoglou, Mike P., Bernd J. Krämer, and Jian Yang. "Leveraging web-services and peer-to-peer networks." Advanced Information Systems Engineering. Springer Berlin Heidelberg, 2003.
12. Gao, Cong, and Jianfeng Ma. "A Collaborative QoS-Aware Service Evaluation Method for Service Selection." Journal of Networks 8.6 (2013): 1370-1379.
13. Ahson, Syed A., and Mohammad Ilyas, eds. SIP handbook: services, technologies, and security of Session Initiation Protocol. CRC Press, 2008.
14. Filali, Imen, et al. "A survey of structured p2p systems for rdf data storage and retrieval." Transactions on large-scale data-and knowledge-centered systems III. Springer Berlin Heidelberg, 2011. 20-55.
15. Ripeanu, Matei. "Peer-to-peer architecture case study: Gnutella network." Peer-to-Peer Computing, 2001. Proceedings. First International Conference on. IEEE, 2001.
16. Haas, Zygmunt J., Joseph Y. Halpern, and Li Li. "Gossip-based ad hoc routing." IEEE/ACM Transactions on Networking (ToN) 14.3 (2006): 479-491.
17. Good, Nathaniel S., and Aaron Krekelberg. "Usability and privacy: a study of Kazaa P2P file-sharing." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2003.
18. Ranjan, Rajiv, Aaron Harwood, and Rajkumar Buyya. "Peer-to-peer-based resource discovery in global grids: a tutorial." Communications Surveys & Tutorials, IEEE 10.2 (2008): 6-33.
19. Rowstron, Antony, and Peter Druschel. "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems." Middleware 2001. Springer Berlin Heidelberg, 2001.
20. Stoica, Ion, et al. "Chord: A scalable peer-to-peer lookup service for internet applications." ACM SIGCOMM Computer Communication Review. Vol. 31. No. 4. ACM, 2001.
21. Zhao, Ben Y., et al. "Tapestry: A resilient global-scale overlay for service deployment." Selected Areas in Communications, IEEE Journal on 22.1 (2004): 41-53.
22. Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." Peer-to-Peer Systems. Springer Berlin Heidelberg, 2002. 53-65.
23. Morris, S. R., et al. "Chord: A Scalable Peer-to-peer lookup service for internet applications:[Technical Report. TR-819]." (2001).
24. The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns/index.html
25. The Network Simulator – ns-3, http://www.nsnam.org/releases/
26. OPNET Modeler Home Page: http://www.opnet.com/products/modeler/home.html
27. OMNeT++: Discrete Event Simulation System: http://www.omnetpp.org/
28. Network Simulator, http://tetcos.com/netsim_gen.html
29. FIPS, PUB. "180-1. Secure hash standard." National Institute of Standards and Technology 17 (1995).
30. Al-Masri, Eyhab, and Qusay H. Mahmoud. "Discovering the best web service." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.