Journal of Computer Science 1 (3):323-331, 2005
ISSN 1549-3636
© Science Publications, 2005

Modeling Load Balancing in Heterogeneous Unstructured P2P Systems

Zhi Jun Li and Ming Hong Liao
Computer School of Science and Technology, Harbin Institute of Technology, China

Abstract: Load balancing is a generally concerned problem in peer-to-peer (P2P) systems. Many researches on load balancing in the structured P2P systems have been launched currently, such as Chord or other DHTs. Although the researches on load balancing in unstructured P2P systems are emerged nowadays, the simple mechanisms achieved can only perform effectively in uniform environment. In this study, the influence on load balancing of the heterogeneity existed universally in unstructured P2P systems are analyzed, the unstructured P2P systems and their load balancing and the heterogeneity are modeled. Based on the formal model, the load balancing is analyzed quantitatively under static and dynamic environment and the typical load balancing algorithms are also analyzed. Some important conclusions are drawn which can be used in new models of load balancing in unstructured P2P systems.

Key words: Unstructured P2P Systems, Load Balancing, Heterogeneity, Data Flow Analysis

INTRODUCTION

The load balancing is a generally concerned problem in P2P systems [1-4]. In nowadays, a lot of researches have been launched on load balancing in the structured P2P systems, such as Chord [5] and other DHTs [6-8]. The load balancing in structured P2P systems mainly focuses on the heterogeneity of the DHT, such as the heterogeneity of the address space [1-4].

Although structured P2P systems have been researched in several past years because of their properties such as determinable, provable and formally specifiable [5-8], they are rarely used in practice for P2P's dynamic property and undeterminable property in nature. However, the unstructured P2P systems are the mostly commonly used today. And the unstructured P2P systems are more useful than structured P2P systems in some situations [9-12].

The researches of this study mainly focus on the load balancing in unstructured P2P systems. In spite of few, there are some researches in load balancing in unstructured P2P systems, such as [13-17]. However, [13] performs only in isomorphic environment with simple mechanisms by providing a method of building an evenly random graph to obtain good load balance. The algorithms provided in [14~16] are very scanty and assumed as a matter of course without formal reasoning.

FORMAL MODEL OF LOAD BALANCING

Before formally describing the load balancing in unstructured P2P systems, we define the data flow as the foundation of this formal model.

Definition 1: A data flow in P2P networks is a function *F*: *P*×*P*→*D*, where the *P* is the set of all peers in the system and *D* is the data.

F(*p*,*q*) refers to the data flowing from peer *p* to peer *q*. And, in practice, *F*(*p*,*q*) normally takes some properties of the flowing data, such as the property of the data occurrence. So the *F*(*p*,*q*) can exhibit as many different forms and we provide two simple forms here which will be used in this study.

The first data flow form is a boolean data flow defined as:

$$F(p,q)=\begin{cases}0 & F(p,q)= \\ 1 & F(p,q)\neq\end{cases} \tag{1}$$

The second data flow form takes the flux property of the data flow. It is defined as:

$$F(p,q)=len(d\mid d=F(p,q)) \tag{2}$$

where the *len*(*d*) is the length of the data *d* and *F*(*p*,*q*) represents the flux flowing from *p* to *q*.

The formal analyses in this study are all based on the definition presented in formula (1) for simplification and without loss of generality. In fact, the definition by formula (2) is a more accurate and the results derived from (1) are also usable for (2) while the same analysis carried through under (2) is absolutely more complicated and more accurate. The analyses based on the formula (2) or other more complex formulas such as introducing information entropy will be presented in future works.

Formal Load Description: The load imposed on the peers can be divided into three portions: computing load, storage load and communication load. For each load type, it is reasonable to assert that the amount of the load is proportional to the amount of data flowed into that peer, if all peers are good. For peer *p*, the amount of inflow $F_{in}(p)$ is $\sum_{q\in P}F(q,p)$. Accordingly, for the amount of outflow $F_{out}(p)$ is $\sum_{q\in P}F(p,q)$.

Therefore, the load at *p*(denoted as *L*(*p*)) is:

$$L(p)=C\cdot F_{in}(p) \tag{3}$$

In the following analyses, the *C* is always ignored for simplicity and the results are still tenable without *C*.

Formal Load Balancing: The load factor at peer *p* is used to describe the load's impact on *p*. It is the quotient of the load divided by the capacity of the peer (*C*(*p*)), i.e. $\delta(p)=L(p)/C(p)$.

Obviously, the complete load balancing must satisfy:

$$(\forall i \neq j)\delta(p_i) = \delta(p_j) \tag{4}$$

The rule (4) is a theoretical rule aiming at obtaining the absolutely complete load balancing. However, it is almost impossible to implement and in practice, the permission of partial imbalance is absolutely necessary. So the operable rule for load balancing is -balance. It means

$$(\exists \varepsilon \geq 0)U \leq \varepsilon \tag{5}$$

where U depicts the imbalance quantity of the P2P systems and U can be calculated as following.

Absolute Distance: U is calculated by formula (6).

$$U = (\forall i, j)\max\{|\delta(p_i) - \delta(p_j)|\} \tag{6}$$

Formula (6) only considers the maximum distance between the most loaded and most idle. So it examines the extreme condition of imbalance and is the upper bound for U and neglects a majority of imbalance details such as how many peers are overloaded or how many peers are idle.

The Standard Deviation of Load Distribution:

$$U = \sqrt{\sum_{p \in P} (\delta(p) - \mu)^2} \tag{7}$$

The   in formula (7) is the system load factor defined as *TL*/*TC*, in which the *TL* is the total load in the system and *TC* is the total capacity, i.e. $TL = \sum L(p)$ and $TC = \sum C(p)$. The formula (7) quantitatively depicts the imbalance distribution regarding to the mean point . Comparing with the formula (6), formula (7) is more accurate and more complicated. On the other hand, the formula (6), although naive, it gives the upper bound of the imbalance and is quite useful under some situations. Consequentially, the theoretical analyses in this study are mostly based on the formula (7) without specific illumination.

Load Balancing and Availability: When $L(p)>C(p)$, the overloaded part will be dropped and the peer will be running fully loaded. Under the assumption that all peers perform well and the total amount of the overloads (or *TOL*) is $\sum_{\delta(p_i)>1} [L(p_i) - C(p_i)]$. Now, the system availability (denoted as *Av*) is

$$Av = 1 - \frac{TOL}{TL} \tag{8}$$

For the given *TL*, the only method to improve the availability is to reduce the number of overloaded peers. If *TL* *TC*, the 0-balance can ensure that the *Av* is always 1 even though the *TL* is very enough and close to *TC*. For the case of *TL*>*TC*, the 0-balance will maximize the *Av* and all peers run in a fully loaded status.
Consequently, the improvement of the availability has a broader sense than balancing the load.

Load Balancing in Unstructured P2P Systems: Because the focus of this study is the load balancing in unstructured P2P systems, it is necessary to describe its kernel properties formally:

Routing Scheme: The routing in unstructured P2P systems uses the per-node routing tables, whose entries maintain the references to other nodes which are denoted normally as *Neighbors*(*p*) (or *Ng*(*p*)). The next-hop selection for the query routing satisfies some probability distribution *Pr*. For the unstructured P2P systems, the situation that *Pr* is a uniform distribution is reasonable although some heuristic information can slant the *Pr*, such as the peers' trust value. In this study, we only consider the uniform distribution situation.
Moreover, suppose all peers are good, the status that some in-flow data will lead to some data out flowing is arisen, because the peer will retransmit the data or produces some data for response. i.e.

$$(\forall p \in P)F_{in}(p) = F_{out}(p).$$

Hence, we have

$$(\forall p \in P, q \in Ng(p))F(p,q) = \frac{F_{in}(p)}{|Ng(p)|} \tag{9}$$

Data Placement Scheme: The data (*d*) including its replica (*R*(*d*)) can be stored at will in unstructured P2P systems without restriction rules such as the ID-mapping property in DHT. Consequently, no one can determinately locate the placement where the data or its replica will be stored from theoretical aspect. This random data placement can be formally described as the probability that the peer sequence $p_1,p_2,\ldots,p_k$ passed by the query packages for data *d* is $r \cdot (1-r)^{k-1}$, where *r* is the probability of the *d*'s existence, i.e. $r=|R(d)|/|P|$.

Peer Discovery Scheme: In unstructured P2P systems, the peer discovery is blindfold by some random sniff, such as the ping in Gnutella [9]. The amount and update of the information discovered depends on the sniff width and sniff frequency. In realistic environment, only a subset (i.e. *Ng*) of peers are discovered comparing to all peers *P* and $|Ng| < |P|$.

STATIC LOAD BALANCING ANALYSES

In unstructured P2P systems, there are many environments where the heterogeneity exists such as the dissimilarity of peer capability, the difference of the data popularity and the variance of the connecting density in the overlay, etc [1, 2, 3, 4, 14, 15, 16, 17, 18, 19].
The heterogeneity mentioned above can raise the slope of the load in unstructured P2P systems and the formal analyses of their influence on load balancing based on the data flow analysis.

Firstly, we analyze the influence of each heterogeneous factor independently. Then, the correlations among multiple heterogeneous factors are discussed.

The Heterogeneity of Peer Capacities: Let the peers' capacities situate in interval [$c_{\min}$,$c_{\max}$] and suppose they satisfy the probability distribution *Pc*. So, the number of peers located in the capacity interval [*x*,*x*+*dx*] is |P|*$p_c(x)dx$, where p_c is the probability density function of *Pc*.

If all other heterogeneity is absent, $\delta(p) = \frac{TL}{|P|}\cdot\frac{1}{C(p)}$ and:

$$U = \mu\sqrt{|P|}\cdot\sqrt{\bar{c}^2\cdot E(\frac{1}{C^2(p)}) - 2\bar{c}\cdot E(\frac{1}{C(p)}) + 1} \qquad (10)$$

In formula (10), the *E* is the expectation of the corresponding distribution and $\bar{c}$ is the average of *C*(*p*), i.e. $\bar{c}$ =*TC*/|*P*|.

When *Pc* is a uniform distribution on [$c_{\min}$,$c_{\max}$] and let $c_{\max}$= $c_{\min}$, the *U* is:

$$U = \frac{TL}{\sqrt{|P|}\cdot c_{\min}}\cdot\sqrt{\frac{1}{\lambda}+\frac{4}{(\lambda+1)^2}-\frac{4\ln\lambda}{\lambda^2-1}} \qquad (11)$$

Obviously, for formula (11) the $U \le TL/\sqrt{|P|\cdot c_{\max}\cdot c_{\min}}$ and for sufficiently large , the *U* will approach this upper bound. Further, this upper bound can be rewritten as $\frac{\sqrt{\lambda}}{2}\cdot\mu\sqrt{|P|}$.

Therefore, if *Pc* satisfies the uniform distribution, the load imbalance is proportionate to the , to the square root of |*P*|.

The Heterogeneity of Data Popularity: The heterogeneity of data popularity will incur the load slope. The nodes stored with higher popular data will receive more load than nodes with lower popular data. With data popularity, it is reasonable to assert that the in-flow query will be distributed in accordance with the data popularity, i.e. $F_{in}(d) = Pop(d)\cdot\sum_{p\in P}F_{in}(p) = Pop(d)\cdot TL$, where *Pop*(*d*) is the measurement for *d*'s popularity and *Pop*(*d*) satisfies the distribution *Pp*. The $F_{in}(d)$ depicts the flow into the data *d*. We assume that the data *d* is stored on a set of peers *R*(*d*), Thus

$$F_{in}(d) = k\cdot\sum_{p\in R(d)}F_{in}(p) \qquad (12)$$

The *k* in formula (12) depends on the data stored on the peer *p* and is a complex factor changing with the synthesized popularity of data on *p*. We divide the analysis into two cases as:

Nonoverlapping *R*(*d*): $(\forall d_1, d_2)R(d_1)\cap R(d_2) = \phi$.

Firstly, we consider the simplest situation that |*R*(*d*)|= $\bar{r}$ and $|P| = \bar{r}|D|$.

Then, $L(p) = F_{in}(p) = Pop(d)\cdot TL/\bar{r}$. Now,

$$\hat{U} = \mu\sqrt{|P|}\cdot\sqrt{|D|^2\cdot E(Pop^2)-1} \qquad (13)$$

where *E*(*Pop*^{2}) depicts the expectation of the square of *Pop* on distribution *Pp*.

If *Pp* is a uniform distribution, then the *U* is:

$$U \approx \frac{1}{\sqrt{3}}\cdot\mu\sqrt{|P|} \qquad (14)$$

for large system.

Besides, the data popularity normally satisfies the Zipf's law [23], now $\hat{U} \approx \frac{1}{\sqrt{\alpha-2}}\cdot\mu\sqrt{|P|}$ and α is the power of Zipf.

In conclusion, for any *Pp*, the load imbalance *U* is proportionate to the   and to the square root of |*P*|. The result is very similar with the capacities' heterogeneity.

Secondly, we examine the situation that the number of the replica for different data is different, i.e. the |*R*(*d*)| is different for different data. Now, the formula (13) is transformed into:

$$U \approx \mu\sqrt{|P|}\cdot\sqrt{|P|E((\frac{Pop}{\sqrt{|R|}})^2)-1} \qquad (15)$$

Obviously, when the *Pop* is proportional to the square root of |*R*(*d*)|, *U* is minimum. It is interesting to notice that this result for balance is quite similar with the replication strategy result in [19] for improving the expected search size(ESS) and whose precondition is also same with ours.

***R(d)* can Overlap:** Suppose the function *Sto*:*P*→2^D defines the data served by peers.

Now, the load imposed on the peer *p* is the summation of the share part imposed on the data stored on *p*. Thus:

$$\hat{U} \approx \mu\sqrt{|P|}\cdot\sqrt{|P|\cdot\sum_{p\in P}(\sum_{d\in Sto(p)}\frac{Pop(d)}{|R(d)|})^2-1} \qquad (16)$$

It is obvious that when $\sum_{d\in Sto(p)}\frac{Pop(d)}{|R(d)|}$ is a constant for all peers, the formula (16) is minimum.

Therefore, for the case that |*R*(*d*)| is same for all data, the load balancing need to ensure that the $\sum_{d\in Sto(p)}Pop(d)$ is equal for all peers as most as possible. Note that this expression exactly represents the popularity of the peer *p*.

The Heterogeneity on Overlay Density: With the links movement without constraint, the overlay in unstructured P2P systems normally uneven, such as the famous power-law overlay [19]. The overlay density at a peer *p* can be defined as the amount of links connecting with the *p*. In power-lay overlay, the probability of the peers with *k* links is *k* .

Here, we examine the situation that the overlay density satisfies the distribution P_N on links' number *N*. From the formal description of the routing schema on unstructured P2P systems above, we assume the load in all links is approximately equal for simplicity. Now, the load on each link is *TL*/Σ*N* and the *U* is:

$$U = \mu\sqrt{|P|}\cdot\sqrt{\frac{EN^2}{E^2N}-1} \tag{17}$$

For the power-law overlay, $U = \mu\sqrt{|P|}\cdot\sqrt{1/(\tau-1)(\tau-3)}$. It is obvious that if   is less than 3(including 3), *U* is maximum and extremely large, while with the increase of , the *U* will decrease and when   approaches the positive infinite, *U* is minimum, i.e. 0, now the links' number is same for all peers.

When P_N satisfies the uniform distribution in interval [0, |P|], we have $U \approx \mu\sqrt{|P|/3}$. It is interesting to notice that this result is completely same with (14).

Correlation Analysis: In practice environments, the heterogeneity mentioned above always emerges simultaneous and whose influence on load imbalance is correlated. In the definition of *U*, the load and capacity are all capable of addition, i.e. the imbalance incurred by some heterogeneity can be added with the imbalance incurred by others as follows.

Assume some heterogeneity will produce a load distribution $L_1(p)$, while the other heterogeneity will incur $L_2(p)$. The similar situations exist on peers' capacity and $C_1(p)$ and $C_2(p)$ describe the distribution caused by two different heterogeneity. Now, the   for *p* is $(L_1(p)+L_2(p))/(C_1(p)+C_2(p))$ and the analysis of *U* is much more complicated comparing with above.

Theorem 1: Suppose two correlated heterogeneity lead to the $L(p)=L_1(p)+L_2(p)$ and $C(p)=C_1(p)+C_2(p)$, then the imbalance satisfies the following inequality

$$0 \le U \le U_1 + U_2 \tag{18}$$

where U_1 and U_2 depict the imbalance incurred by two heterogeneity respectively.

Proof: Firstly, we stabilize peers' capacity as a constant. Now $U = \mu\sqrt{|P|}\cdot\sqrt{\frac{\sum[L_1(p)+L_2(p)]^2}{(\sum[L_1(p)+L_2(p)])^2}-1}$, in which the $L_1(p)$, $L_2(p)$ are all distributions on [0,1] after the simplification via deleting the common factors and $\Sigma L_1(p)=\Sigma L_1(p)=1$. And define a function as:

$$g(f(x)) = \sqrt{\frac{\int f^2(x)dx}{(\int f(x)dx)^2}-1}$$

It is obviously that $g(f(x))$ is a norm for f(x) when $\int f(x)dx = 1$. From the triangle inequality property of norm, $g(f_1(x)+f_2(x))$ $g(f_1(x))+ g(f_2(x))$, thus the same property for *U*.

The proof for other cases is similar and omitted here.

Corollary 1: It is feasible to balance the load in unstructured P2P systems via self-organization and self-optimization.

Corollary 1 presents a profound reason for the importance of introducing the self-organization in unstructured P2P systems. The load balance can be achieved via appropriately adjusting the load distributions and capacity distributions. The detailed adjustment mechanisms are a future work.

DYNAMIC LOAD BALANCE ANALYSES

Time Related Data Flow

Definition 2: A data flow related with time in P2P networks is a function *F*: *P*×*P*×*T*→*D*. Now, *F*(*p*,*q*,*t*) refers to the data flowing from peer *p* to peer *q* at time *t* and now the $L(p,t)=\sum_{q\in P} F(q,p,t)/C(p,t)$.

Usually, the capacity of a peer is almost invariable and we assume $C(p,t)=C(p)$ in the following analyses. In unstructured P2P systems, $F(q,p,t)$ can be assumed as a constant $W(t)=TL/\Sigma_{p\in P}Ng(p,t)$ thusL(p,t)=$W(t)|Ng(p,t)|$. Based on this formula, it is obvious that *U* is invariable with time going ahead unless the movement of the data stored and the movement of the overlay.

Now, we will examine the influence of following dynamic evolutions on load balancing on by one.

* Data insertion and deletion
* Replica insertion and deletion
* Node join and departure

The Data Insertion and Deletion: Firstly, we take a complete balance state as the initial state, i.e. U_o=0 and the analysis with other arbitrary initial states can be established in the results obtained through this analysis.

When the current state is balancing and a new data *d* is inserted into the system, the new *U* must be larger than U_o (here U_o =0). Suppose *d* is served by peer *p* and the popularity of *d* is $\lambda\overline{p}$, where $\overline{p}$ is the average of data popularity. The flow received by other data will decrease by *m* times.

Here, $\overline{p} = 1/(|D|+1)$ and $m = 1-\lambda/(|D|+1)$. Now U_n is:

$$U_n = \frac{\lambda\mu|P|}{\eta(|D|+1)}\cdot\sqrt{1+\frac{2\eta+\eta^2}{|P|}} \tag{19}$$

Where   is defined as $C(p)=\eta\overline{c}$. For a large system, U_n is a constant $c\lambda\mu/\eta$. In the following, the U_n with U_o=0 is Φ .

Theorem 2: For arbitrary U_o, we have

$$\Delta U = |U_n - U_o| \le \frac{(\Phi+\sigma_o(p))^2}{\Phi} \tag{20}$$

where the $\sigma_o(p) = \delta_o(p) - \mu$ depicts the load deviation of p.

Proof:

$$\Delta U = |U_n - U_o| = \frac{|U_n^2 - U_o^2|}{U_n + U_o}$$

$$= \frac{\Phi^2 + 2(m\delta_o(p)-\mu)\Phi - \sum_{q\in P}(1-m)((1+m)\delta_o(q)^2 - 2\mu\delta_o(q))}{U_n + U_o}$$

The Replica Insertion and Deletion: We analyze the replica insertion firstly. Suppose a replica for data d with $|R(d)|\geq 1$is inserted into the peer p.
When U_o=0, we have

$$U_n = \sqrt{\sum_{q\in R(d)} (\frac{ml}{C(q)})^2 + (\frac{l}{C(p)})^2} \tag{21}$$

where $m=1/|R(d)|$ and $l=TL\ Pop(d)/(|R(d)|+1)$.
If $|R(d)|$ is large, $\Phi \approx \mu Pop(d)/[\eta\overline{c}(|R(d)|+1)]$ and $\Phi \approx \mu Pop(d)/(\eta\overline{c}|R(d)|)$ for small $|R(d)|$. Similar with theorem 2, formula (20) is also satisfied in this situation.
Further, we analyze the replica deletion. Suppose a replica served by p for data d also with $|R(d)|>1$ is deleted. Now, $m=1/(|R(d)|-1)$ and $l=TL\ Pop(d)/|R(d)|$. The result of analysis is same and $\Phi \approx \mu Pop(d)/[\eta\overline{c}(|R(d)|-1)]$.

The Node Join and Departure: The imbalance movement incurred by joining a peer p mainly contains three independent parts.
* The increase of the TL
The load introduced by p should be approximately equal with average load. Thus, the TL_n after joining the p is $TL_o\,(|P|+1)/|P|$.
* The alteration of the links' distribution
After p building up its links, the links' alteration of p and the link's alteration of peers in $Ng(p)$ will be influenced. Now, we examine the alteration of the load imposed on the link. Let the $|Ng(p)|=\rho\overline{n}$ where $\overline{n}$ is the average of peers' links and using $\overline{\varpi}$ to describe the load on each link, obviously, $\overline{\varpi}_n = TL_n/(|P|\overline{n}+\rho\overline{n})$. For large systems, we have $\overline{\varpi}_n \approx \overline{\varpi}_o$.
Therefore, the influence on imbalance movement by adding links are only existing for p and peers in $Ng(p)$. So, the U_n

$$U_n \approx \sqrt{U_o^2 + \sum_{q\in Ng(p)} (\frac{\overline{\varpi}}{C(q)})^2 + (\frac{\overline{\varpi}\rho\overline{n}}{C(p)} - \mu)^2}\,.$$

It is obvious that $\Phi = \mu\sqrt{\frac{\rho}{\overline{n}} + (\frac{\rho}{\eta} - 1)^2}$.
For arbitrary U_o, the property similar with theorem 2 still exist and $\Delta U \leq \Phi + \sum_{Ng(p)} (\frac{\overline{\varpi}}{C(q)} + \sigma_o(q))^2 / \Phi$.
* The insertion of p's data
The analysis of node's departure is similar with node's join and $\Phi = \mu\sqrt{\rho/\overline{n}}$.

MODELING LOAD BALANCING ALGORITHMS

Routing Strategy: One of the most frequently adopted balancing algorithms is data migration (or data replication). The load balance can be achieved by transfer hot data from heavily loaded peers to lightly loaded peers [15]. Before migration (or replication), an important preparative task is the hotspot detection. Now, the mechanism widely adopted is to monitor the access frequency to data within recent time interval periodically [15]. Based on this detection, there are mainly two approaches towards decision-making: centralized and distributed.
Centralized methods such as the *L_assign* in [15] make decisions at the centralized dispatcher, where the load balance information (or LBIs) are sorted and matched. Distributed methods select an appropriate distributed server to make decision. Such as the myopic greedy strategy in [16, 22], they select a permissible server with the smallest current latency.

Centralized Decision-Making: Obviously, the correct centralized decision-making must depend on the correct LBIs, but for larger P, it is almost impossible to collect all updated LPIs. Therefore, it is mainly adopted in local load balancing. Suppose a centralized dispatcher(C) monitors a cluster whose members form into a set g. Assume after a set of load $LS=\{l_1,l_2,\ldots,l_m\}$ is received by C, the dispatch starts up. A dispatch scheme is an especial partition on LS, in which the empty set can be an element of the partition and the candidacy of the partition including empty set is $|g|$.
* If the current state is balancing, the optimal dispatch scheme must satisfy the properties:

$$\max\{|L_i/C(p_i) - L_j/C(p_j)|\} \leq \varepsilon \tag{22}$$

Here L_i is the load dispatched to peer member p_i and we assume $C(p_i)$ is decrease with i increase. If $m\ k$, the optimal dispatch scheme is $\{\{l_1\},\{l_2\},\ldots,\{l_m\},\ \ ,\ldots\}$ where $l_i\geq l_j$. For $m>k$, how to build the optimal dispatch scheme satisfying the formula (22) is an *NP-hard* problem.
However, in P2P systems, the load mainly focuses on the query. Therefore, it is feasible to assume that l_i is a constant. Under this assumption, the optimal dispatch scheme can be achieved via following two steps:
If current state is unbalancing, the examining of this situation is similar with above, except that the distribution partition should be optimized to satisfy the formula (22) with current (p) are also considered.
Secondly, we analyze the load balance at the cluster leader C. The overhead loads imposed on the cluster leader are numerous, including: 1) collecting LBIs; 2) receiving all loads flow into the cluster members; 3) performing dispatch algorithm. Because this study mainly focuses on the communication load, only the part 1) and 2) are need to consider. The loads incurred by 1) are non-determinate and are related with LBI's update frequency and the cluster's dynamic properties. Despite we only examine the loads incurred by 2), the header's capacity must be larger than total capacity of the members to achieve the load balance from above analysis. This condition severely constrains the cluster's form and its scale.

Distributed Decision-Making: In distributed decision-making, the next-hop nodes are selected with some heuristic properties for load balancing when routing. Many heuristic mechanisms are adopted in practice and they are mainly classified as follows.
Load based decision-making [14~16]
The probability of choosing q as the next query hop is proportional to the inverse of q's load, i.e.

$$Pr(n = q) \infty 1/L(q) \qquad (23)$$

It is often called as Inv-Load method. Obviously, the Inv-Load decision-making is effective only in homogeneous environment, because of not consideration of the peers' capacity. For analysis, we assume peers' capacities are same.
The analysis need to calculate the load movement ($L(q)$) for all q in P. $L(q)$ contains two parts, the load flowed into q and the load flowed out of q. At a moment, it is reasonable to assume that every peer sends a flow and all flows are received by a peer. Therefore, the amount of load flowed out of q is 1 and amount of inflow is the sum of shares flowed out of the q's neighbors and

$$\Delta L(q) = \sum_{p \in Ng(q)} \gamma(p)/L_o(q) - 1 \qquad (24)$$

(p) is the coefficient in (23) and $\sum_{n \in Ng(p)} \gamma(p)/L(n) = 1$.
Without losing generality, we can specialize the (24) by take the average value for (x), $L(x)$ and $Ng(x)$. Now, $\Delta L(q) = \mu\overline{c}/L_o(q) - 1$.

Theorem 3: The Inv-Load algorithm can balance the load if without other heterogeneities.

Proof: We can construct a sequence L_i with $L_i = L_{i\text{-}1} - 1 + \mu\overline{c} / L_{i\text{-}1}(1 \quad i)$. If $L_0 \quad \mu\overline{c}$ (the average of load), then L_1 L_0 due to $\mu\overline{c} / L_0 \quad 1$ and $L_i \quad L_{i\text{-}1}$ by induction, on the other hand, $L_i \quad \mu\overline{c}$ because when $L_i = \mu\overline{c}$, L_{i+1} equals to L_i and the L_i doesn't increase and converge at $\mu\overline{c}$. The identical situation comes forth for $L_0 \quad \mu\overline{c}$. Figure 1 illustrates the load balancing effect of Inv-Load.
In Fig. 1, we examine the load distribution movement for 10 peers with linear distribution in [10,100] as the initial state and the R_num depicts the iterative number.

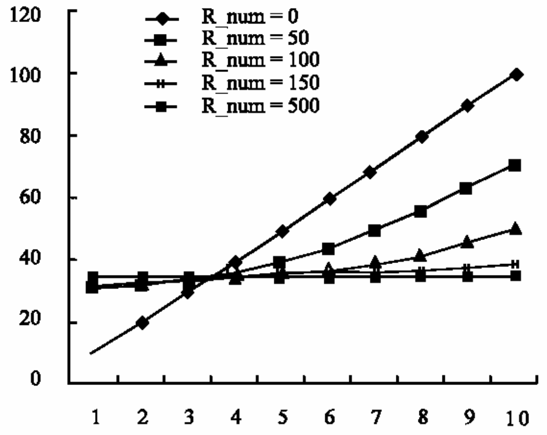

Fig. 1: The Load Balancing Effect of Inv-Load

Theorem 4: The Inv-Load algorithm converges with an approximately exponentially decreasing speed.

Proof: From equation $L_i = L_{i\text{-}1} - 1 + \mu\overline{c} / L_{i\text{-}1}$, we have $L(i) = (\mu\overline{c} / L(i) - 1) \quad i$. The corresponding continuous form is $L' = \mu\overline{c} / L - 1$, so $L(i) = \mu\overline{c} - e^{-(i+L(i)-G)/\mu\overline{c}}$, G is a constant for initial condition, $G = L_0 + \mu\overline{c}\ln(\mu\overline{c} - L_0)$. Because $L(i)$ is a value between L_0 and $\mu\overline{c}$, the $L(i)$ can be assumed stable comparing with varying i. Thus, $L(i)$ converge with a exponentially degressive speed. The result shown in Fig. 2 verifies the theorem 4.

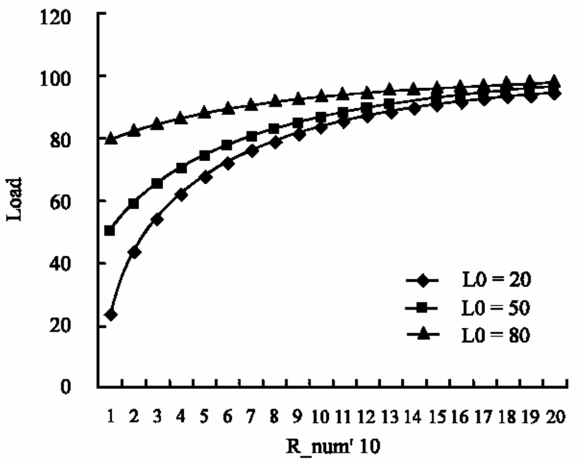

Fig. 2: The Convergence of Inv-Load with Average Load = 100

In spite of so fast the convergence speed is, the P2P's highly dynamic nature [24], the changing of workloads and the changing of peers, etc, will lead to there isn't sufficient time to achieve the convergence before the environment is changed. Figure 3 illustrates the load balancing oscillation incurred by the P2P's workload oscillation. Here, we introduce a coarse oscillation pattern for simplicity: periodically choosing a single point as a hot spot.

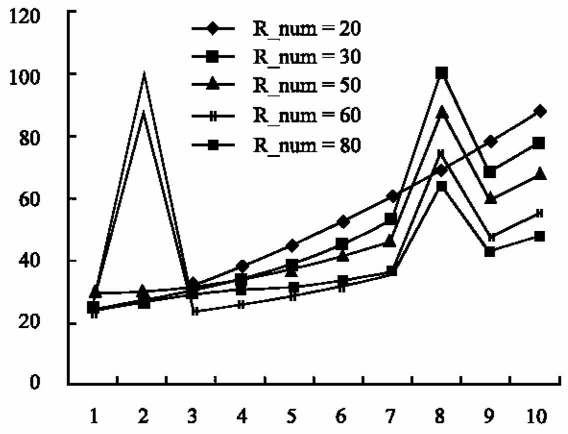

Fig. 3: The Oscillation in Inv-Load with Workload Oscillation Period = 30

* Available capacity based decision-making [14,15]

$$Pr(n = q) \infty C(q) - L(q) \qquad (25)$$

Comparing with Inv-Load method, this Avail-Cap method takes the heterogeneity in peers' capacity into

account. Moreover, the heterogeneity in load is also involved by a subtraction.
The analysis is identical with Inv-Load and we have

$$\Delta L(q) = \frac{C_o(q) - L_o(q)}{\bar{c}(1-\mu)} - 1 \qquad (26)$$

Like the $L(q)$ of Inv-Load, Avail-Cap can balancing the load and its convergence speed is $\partial \Delta L / \partial L = -1/\bar{c}(1-\mu)$, while the speed for Inv-Load is $\partial \Delta L / \partial L = -\mu\bar{c}/L^2$. Thus, the convergence speed of Avail-Cap is faster than Inv-Load when the workload is larger. Moreover, the oscillation is also existent for similar reason with the Inv-Load.
Maximum capacity based decision-making
The maximum capacity (Max-Cap) based allocation provided in [14] makes decision according to the peers' maximum capacity.

$$Pr(n = q) \propto C(q) \qquad (28)$$

And now $\Delta L(q) = C_o(q)/\bar{c} - 1$.
Comparing with Inv-Load and Avail-Cap, the Max-Cap can only smooth the skewness on peers' capacity, the slant on workload can't be flatted and the balancing effect is infertile. However, it possessed of an advantage that the Max-Cap algorithm is stable and robust and won't introduce the load oscillation by itself, which will be examined in next section.

The Influence of the Stale LBI: Nowadays, almost all balancing algorithms require the LBIs. Many practical reasons can incur the degraded LBIs and it is a very important property to minimize the influence of the degraded LBIs. We will analyze the influence of stale LBIs.
Firstly, we define *QN* as the query amount processed in the intervals between two latest LBIs reached the decision-maker. Because the LBIs are transferred only between the neighbors (with a hop distance), thus, we have *QN*=*UP*/*RP*, where *UP* is the update period and *RP* is request period.
The influence of the stale LBIs is maximum at the last query finished before the latest LBIs exchanged. At the point, L_n=L_0 + *QN* L = L_0 + L_1 + L_2 +… +(e_ L_1 +e_ L_2 +…), where L is the amount of query dispatched based on stale LBIs and e_ L_i = L– L_i = $\bar{N}$ ($dmf(C_o,L_o)$–$dmf(C_i,L_i)$), where *dmf* is the decision-making function. For the case that *UP*>>*RP*, we have L_o + L_1 + L_2 +…= $\mu\bar{c}$.
For the Inv-Load algorithm, $dmf(C_i,L_i) = \mu\bar{c}/L_i$. With the condition L_0 $\mu\bar{c}$, we have e_ L_i e_ L_{i-1} because of L_i L_{i-1}, which means that the distance deviating from balance is increasing with increscent speed. If *QN* is large enough, the speed will converge at a fixed value, i.e. the e_ L_i will converge into $\bar{n}$ ($\mu\bar{c}/L_0$–1). Theorem 4 indicates the speed of convergence for Inv-Load is very fast and we can presume e_ L_i $\bar{n}$ ($\mu\bar{c}$ /L_o–1). Therefore, if *QN* is larger than a threshold at which *QN* $\bar{n}$ ($\mu\bar{c}$ /L_o–1)= $(1-\mu)\bar{c}$, this peer will be overloaded and the Inv-Load algorithm will make this peer transforming into zero-loaded in the following round of decision-making based on new L_0. Consequently, the load will be oscillatory between zero-loaded state and overloaded state.
To improve the availability, the *UP* should be:

$$UP \le \frac{(1-\mu)\bar{c}L_0}{\bar{n}(u\bar{c} - L_0)} \cdot RP \qquad (29)$$

If L_0=$\bar{c}$, we have $UP \le \bar{c}/\bar{n} \cdot RP$ where the capacity is defined as the amount of query can be processed by a peer.
The similar analyses can be carried through for Avail-Load algorithm. The oscillation phenomenon is also exist and will be strengthen because the $Pr(n=q)$ equals 0 when $L(q)$=$C(q)$. The worst situation is that half peers are running overloaded with the other half idle.
Therefore, the Inv-Load and Avail-Load algorithms themselves will incur the load oscillation due to stale LBIs. Comparing with them, Max-Cap is more stable and robust because L doesn't correlative with L_o.

The Cluster Based Balancing Algorithm: In cluster based P2P system, the query will firstly propagate to the cluster leader *C*, after the unsuccessful intra-cluster search, the query will be propagated into the other clusters by leaders' communication [15, 21]. Now, the load balancing is divided into two parts: inter-cluster balancing and intra-cluster balancing. And normally, most balancing algorithms for intra-cluster are centralized and distributed algorithms are adopted for inter-cluster load balancing. Here, we assume the cluster-based P2P system is a partition (G) on *P*.

Theorem 5: The intra-cluster balance can improve the load balance of the whole system.

Proof: without losing the generality, we assume the complete balance can be achieved in intra-cluster. Now, with intra-cluster load balance, the imbalance U_G (with intra-cluster load balance) is $\sqrt{\sum_{g\in G} |g| \cdot (\mu_g - \mu)^2}$, where $_g$ is the load factor imposed on the group *g*, i.e. $_g$=$L(g)/C(g)$ (the $L(g)$ denotes the total load imposed on all peers in *g* and $C(g)$ is the total capacity of *g*). The imbalance U_{NG} (without intra-cluster load balance) is

$$\sqrt{\sum_{g\in G} \sum_{p\in g} (\delta(p) - \mu)^2} \cdot$$

For every group *g*, we have:

$$\sum_{p\in g} (\delta(p) - \mu)^2 \ge \sum_{p\in g} (\mu_g - \mu)^2 + 2(\mu_g - \mu) \sum_{p\in g} (\delta(p) - \mu_g)$$

Obviously, $\sum_{p\in g} (\delta(p) - \mu_g) \approx 0$, so the lemma.

It is interesting to note that if peers in a cluster satisfy some properties, the whole system can achieve a good balance.

Corollary 2: If for each cluster *g*, $_g$= , we have U_G=0 without inter-cluster load balancing.
Data Migration and Replication (or Caching Schemes)
Data migration and replication (or caching schemes) are classical method to address flash crowds problem [17, 18]. The replication strategies adopted in P2P systems are diversiform. We will analyze a typical

replication strategy for load balancing in detail here and discuss other strategies curtly.

Path Replication: Replication along the response path routed by query is one of the most intuitional strategies, but majority of users believe this naïve strategy can manage the load balance problem in spite of no convincingly researches to prove.

In P2P systems, we can assume all peers launch the query with same probability. Now, the amount of replica is only related with current replica number and the data's popularity. The current replica number for *d* is defined as a server ratio for *d*, i.e., *Ser*(*d*)=|*R*(*d*)|/|P|. The *d*'s popularity is defined as the ratio of queries for *d* in all queries. The expectation of hops passed by a query for *d* is 1/*Ser*(*d*)-1 and after all peers completing a query, the amount of replica for *d* becomes $|R_n(d)| = |R_o(d)| + |P| \cdot Pop(d)(|P|/|R_o(d)| - 1)$. If the total capacity possessed by P2P systems is unlimited, the |*R*(*d*)| for every data *d* will increasing until |P|=|*R*(*d*)|. Now, the analysis is meaningless. Therefore, we should analyze after the equilibrium has been build under the capacity limitation. When the equilibrium is established under the capacity limitation, the increased amount of replica should equal with the deleted amount. Using *K*(*d*) to denote the |*P*|*Pop*(*d*)(|*P*|/*R*(*d*)-1), so the increased amount is *K*(*d*) and the deleted amount of replica at *d* is [*K*(*d*)]*R*$_n$(*d*)/ *R*$_n$(*d*). Under equilibrium, *R*$_n$(*d*)=*R*$_o$(*d*)= *R*(*d*), we have [*K*(*d*)]*R*(*d*)=[*R*(*d*)]*K*(*d*), where *R*(*d*) is the capacity limit of the system, a constant *S*; *K*(*d*) is the amount of replicas incurred after a query and it is also a constant(*F*) under equilibrium. So, *K*(*d*)=*GR*(*d*), where *G*=*S/F* and thus

$$R(d) = \frac{|P| \cdot (\sqrt{Pop(d)^2 + 4GPop(d)} - Pop(d))}{2G} \quad (30)$$

and $R(d) \approx |P|\sqrt{Pop(d)}/\sqrt{G}$. From formula (16), *R*(*d*) should be proportional to the square of *Pop*(*d*), not the square root of *Pop*(*d*). Therefore, path replication overmuch replicates the document with high popularity.

Other Replications: Derived from the analysis depicted above, the naïve path replication strategy is a practically feasible method, although it can't achieve optimal effect. The elaborate strategies supported by formal analysis are scarce within our knowledge. Although some replication strategies are described in [19], it results in a square root distribution for the purpose of decreasing ESS.

CONCLUSION

Load balancing is a crucial problem in P2P systems and we focus on the load balancing modeling in unstructured P2P systems for the lack of formal research.

The analyses mainly involve three parts: modeling the load balancing in static environment; modeling the load balancing in dynamic environment and modeling the balancing performance of several current typical balancing strategies. The analyzing results are listed in following tables.

Table 1: Analyzing Conclusions under Static Environment

Hetero-geneous Factor	Bias Distri-bution	Load Imbalance of System
Peer capacities	C	$\mu\sqrt{\lvert P\rvert}\sqrt{\overline{c}^2 E(\frac{1}{C^2}) - 2\overline{c}E(\frac{1}{C}) + 1}$
	uniform distribution	$\mu\sqrt{\lambda\lvert P\rvert}/2$ and $c_{max} = c_{min}$
Data popularity	Pop	$\mu\sqrt{\lvert P\rvert} \cdot \sqrt{\lvert D\rvert^2 \cdot E(Pop^2) - 1}$
	uniform distribution	$\mu\sqrt{\lvert P\rvert}/\sqrt{3}$
	zipf distribution	$\mu\sqrt{\lvert P\rvert}/\sqrt{\alpha - 2}$
Overlay density	N	$\mu\sqrt{\lvert P\rvert} \cdot \sqrt{\frac{EN^2}{E^2N} - 1}$
	power law overlay	$\mu\sqrt{\lvert P\rvert}/\sqrt{(\tau-1)(\tau-3)}$
	uniform distribution	$\mu\sqrt{\lvert P\rvert}/\sqrt{3}$
Combi-nation		$U_{1+2} \le U_1 + U_2$

Table 2: Analyzing Conclusions under Dynamic Environment

Dynamic Factor	Load Imbalance of System
Data enter/leave	$\Phi = \frac{\lambda\mu\lvert P\rvert}{\eta(\lvert D\rvert + 1)} \cdot \sqrt{1 + \frac{2\eta + \eta^2}{\lvert P\rvert}}$
Peer enter/leave	$\Phi = \mu\sqrt{\frac{\rho}{\overline{n}} + (\frac{\rho}{\eta} - 1)^2}$
	$\Delta U \le (\Phi + \sigma_o(p))^2/\Phi$

Table 3: Modeling Typical Balancing Algorithms

Stra-Tegies	Algo- ithms	Balancing Performance
Central	*L*-assign	A *NP* problem and can not directly used in P2P
Distri-buted	Inv-Load	Can balance the load the Stale LBI will incur the load oscillation
	Avail- Load	Can balance the load the Stale LBI will incur the load oscillation
	Max-Cap	Can only smooth the skewness on peers' capacity more stable and robust with stale LBIs
Cluster based		The intra-cluster balance must improve the load balance of the whole system

From above tables, we can achieve following guidable properties of load balancing in unstructured P2P systems.

* The load imbalance mainly depends on the |P| and μ .
* The self-adjust and self-organization are feasible methods.
* The imbalance allowance is preferable in dynamic status.
* The distributed decision making algorithms based on LBIs can load the balance with load oscillation.
* The cluster based load balance algorithm is a good trade off.

ACKNOWLEDGMENT

We thank Proc. Yihe Wang and Proc. Zheng Kuang for their helpful discussions and feedback.

REFERENCES

1. Karger, D.R. and M. Ruhl, 2004. Simple efficient load balancing algorithms for peer-to-peer systems. In SPAA, Barcelona.
2. Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard M. Karp and Ion Stoica, 2003. Load balancing in structured P2P systems. IPTPS, pp: 68-79.
3. Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp and Ion Stoica, 2004. Load balancing in dynamic structured P2P systems. Proc. of IEEE INFOCOM.
4. David, R. and Karger Matthias Ruhl, 2000. New algorithms for load balancing in peer-to-peer systems, In ASPLOS, pp: 190-201.
5. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan, 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Proceedings of the 2001 ACM SIGCOMM Conference.
6. Ratnasamy, S., M. Handley, R. Karp and S. Shenker, 2001. A scalable content-addressable network. In Proc. ACM SIGCOMM.
7. Rowstron, A. and P. Druschel, 2001. Pastry: Scalable, Distributed object location and routing for large-scale peer-to-peer systems. In Proceedings of Middleware.
8. Zhao, B.Y., J. Kubiatowicz and A.D. Joseph, 2001. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California, Berkeley.
9. The Gnutella protocol specification 0.6, http://rfc-gnutella.sourceforge.net
10. KaZaA file sharing network. KaZaA, 2002. In http://www.kazaa.com/
11. Morpheus file sharing system, 2002. http://www.musiccity.com/
12. FastTrack: P2P technology company, 2001. http://www.fasttrack.nu/
13. Pyun, Y. and D. Reeves, 2004. Constructing a Balanced, log(N)-Diameter Super-peer Topology. The Fourth IEEE Intl. Conference on Peer-to-Peer Computing.
14. Mema Roussopoulos and Mary Baker, 2003. Practical Load Balancing for Content Requests in Peer-to-Peer Networks. Technical Report.
15. Anirban Mondal, Kazuo Goda and Masaru Kitsuregawa, 2003. Effective Load-Balancing of Peer-to-peer systems. Proceedings of Data Engineering Workshop DBSJ Annual Conference, Kaga, Ishikawa, Japan.
16. Genova, Z. and K.J. Christensen, 2000. Challenges in URL Switching for Implementing Globally Distributed Web Sites. In Workshop on Scalable Web Services.
17. Stading, T., P. Maniatis and M. Baker, 2002. Peer-to-peer caching schemes to address flash crowds. Proc. IPTPS.
18. Angelos Stavrou, Dan Rubenstein and Sambit Sahu, 2004. A lightweight, Robust, P2P System to Handle Flash Crowds. In IEEE J. Selected Areas in Communications, Special Issue on Service Overlay Networks, Vol: 22.
19. Edith Cohen and Scott Shenker, 2002. Replication strategies in unstructured peer-to-peer networks. In ACM SIGCOMM'02, pp: 19-23.
20. Faloutsos, M., P. Faloutsos and C. Faloutsos, 1999. On power-law relationships of the Internet topology. ACM SIGCOMM Computer Communication Review, 29: 251-262.
21. Zhu, H., T. Yang, Q. Zheng, D. Watson, O.H. Ibarra and T. Smith, 1998. Adaptive load sharing for clustered digital library services. In HPDC.
22. Subhash Suri, Csaba D. Tóth and Younhong Zhou, 2004. Uncoordinated load balancing and congestion games in P2P systems, In Proc. 3rd Internat. Workshop on Peer-to-Peer Systems, LNCS, Springer-Verlag.
23. Li, W. Zipf's law. http://linkage.rockefeller.edu/wli/zipf/.
24. Markatos, E.P., 2002. Tracing a large-scale Peer-to-Peer System: an hour in the life of Gnutella. In Second IEEE/ACM Intl. Symposium on Cluster Computing and the Grid.