ATLANTIS PRESS
Advances in Intelligent Systems Research, volume 136
4th International Conference on Sensors, Mechatronics and Automation (ICSMA 2016)

A Robust Unstructured Mobile Peer-to-Peer Files Sharing System under Higher Peer Churn Rate

Xin Zhang[1, a], Yinghu Xia[2] and Chunqing Lin[3]

[1] Research Institute of Petroleum Exploration & Development, CNPC, Beijing, 100083, China

[2] Supervision Station of Downhole Operation Department of Shanshan Oilfield, CNPC, 839000

[3] Oil Production Engineering Research Institute of Huabei Oilfield, CNPC, 062550

[a]zhangxin92@petrochina.com.cn

Keywords: Mobil peer-to-peer; Churn; Files sharing; Machine learning

Abstract. In the unstructured Peer-to-Peer systems, a core operation is efficient location of resources. Conventional informative searching algorithms, however, always cannot perform well under peer churn rate network environments. In this paper, we designed a robust unstructured peer-to-peer files sharing system. When forwarding query, we usually choose the peer with the highest forward probability. But in this paper we didn't compute the forward probability according to the current neighbor node's individual search history but to a leaning model based on machine learning technologies. We proposed a more robust way for searching. The experimental results show that our methods are more effective and efficient under higher peer churn environment.

Introduction

Among all kinds of the files sharing systems, the unstructured Peer-to-Peer network is a very important one. This self-organizing and scalable network is tolerant to higher peer (node) churn, which means it is necessary to deal with peers' join and left with higher maintenance cost. However, because there is no relation between network topology and the placement of the files, the searching algorithms in such network are always more or less blind.

Flooding algorithm was applied in the Gnutella network in the early days [1]. It is a simple but very robust searching algorithm, because we can always find the objects successfully under higher peer churn environment. But the worst shortcoming of this blind searching algorithm is generating too much network overload. When the network becomes very large, it almost cannot work, because each peer has to deal with I/O request ceaselessly [2].

Though the famous Random Walk algorithm [2,3] can reduce the network overload greatly, it has lower query hit rate and longer query hit delay, because it chooses a neighbor peer randomly at each forward step without considering which one will be more probable to find the objects. Both the flooding and the Random Walk are totally blind algorithms.

Unfortunately, the Peer-to-Peer networks always have a very high peer churn rate, the record of the neighbor's forward history would become unreliable. For example, a neighbor has a high forward probability for a query object during a period of time, because there are relatively more paths during this time between this neighbor and the peers that have the query object. If the number of such paths became fewer after that time, the forward probability would not be accurate any more. Actually, the real-world Peer-to-Peer network topology changes very quickly, so that we even cannot have "a period of time". Furthermore, because peers always have very short lifetime as others' neighbors and there are limited messages through each peer about a specific query object, it is very hard to obtain enough forward information to distinguish which neighbor is more probable to find a specific object.

Related Works

To overcome the shortcoming of the simple flooding algorithm, researchers presented some improved algorithms. Expanding ring algorithm uses floods with increasing TTLs which can halves the per-node message overload [2]. In the same paper, Random Walk algorithm for searching was also presented, and the experimental results showed that the 32 walker random walk can reduce

message overload about two orders of magnitude compared to flooding. Reference [3] and [4] analyzed the Random Walk algorithm in theoretical, which proved that it deeply can get better searching performance than flooding.

Reference [5-11] proposed an adaptive probabilistic search algorithm. This method computes each peer's forward probability using the peer's index value for a specific query object. The performance of APS becomes worse when the objects stored in the network change rapidly.

Machine learning technologies are also applied for searching by some researchers, most of whom use the reinforcement learning to compute the forward probability [12~14]. Actually, there is no more difference between these methods and APS or PQR. They also need to update the gain value along the inverse query hit paths and only show good performance under relatively static network environments. Reference [15] also used the machine learning technologies, but it just focused on the neighbor selection instead of the searching problems.

Feature Definition and Gain Update

To build the routing table based on the machine learning technologies, we should define the feature of the peer and the query object. And the routing table is also designed according to the basic single neighbor forward result ($< neighbor, query \quad object, gain, \mathbf{L} >$), so we should provide the method about how to get and update the gain.

Object Feature. Our methods compute the forward probability of the class the query object belonging to rather than the query object own self. The simplest way to classify the objects is building a class tree according to the application background of the Peer-to-Peer system. In this paper, we focus on the files sharing applications, so we can build the class tree according to the type of the file. For example, if there are three basic kinds of files in the system-“film”, “music”, “software”, and for each kind, there are detailed subclasses too, and then the class tree can described as shown as Fig. 1.

The feature of the files can be represented by a binary vector, the value of which means whether the file belongs to a peer class or not. Take the class tree in Fig. 1 for an example, we have three files: f_1 belongs to action and comedy, and f_2 belongs to R&B, and f_3 belongs to drivers software. The feature values of the three files now are: $< 1,0,1,0,0,0,0,0,0 >$, $< 0,0,0,0,0,1,0,0,0 >$, $< 0,0,0,0,0,0,0,0,1 >$. Obviously, the feature value represents the class that the file belongs to. Different files may have the same feature value. In this paper, we assume that we know each file’s feature value in advance.

Peer Feature. In the Peer-to-Peer files sharing systems, each peer stores some files, which can be used to represent the peer’s profiles.

Denote $F_{\text{file}}(Node)$ as the set of files that $Node$ owns, the feature of $Node$ can be obtained by the following formula:

$$F_{\text{ft}}(Node) = \sum_{f \in F_{\text{file}}(Node)} F_{\text{ft}}(f) . \tag{1}$$

The feature value of $Node$ is:

$$F_{\text{ft}}(Node_i) = F_{\text{ft}}(f_1) + F_{\text{ft}}(f_2) + F_{\text{ft}}(f_3) =< 1,0,1,0,0,1,0,0,1 > .$$

Figure 1. Example of a class tree

Gain Update. In out method, Gain Table (GT) is an important data structure, which records the neighbor's forward history. It includes not only the current neighbors but also the old neighbors. The item of Gain Table is a 6-tuple:

$$< F_{\mathrm{ID}}(N_{\mathrm{nh}}), F_{\mathrm{ID}}(f), F_{\mathrm{ft}}(N_{\mathrm{nh}}), F_{\mathrm{ft}}(f), \\ UpdateTime, Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} > \tag{2}$$

, where $F_{\mathrm{ID}}(N_{\mathrm{nh}})$ is the ID of the neighbor peer N_{nh} in the whole network.

There are also two auxiliary data structures: Current Neighbors Table (CNT) and History Neighbors Table (HNT). The item of CNT is a 2-tuple:

$$< F_{\mathrm{ID}}(N_{\mathrm{nh}}), JoinTime > \tag{3}$$

$$< F_{\mathrm{ID}}(N_{\mathrm{nh}}), JoinTime, LeaveTime > \tag{4}$$

When a query initiated by $Node_s$ finally finds the object f at $Node_t$, we update each peer's Gain Table along the searching path between $Node_s$ and $Node_t$:

$$Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} \leftarrow Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} + \Delta Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} \tag{5}$$

We assume that $\Delta Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)}$ is transferred iteratively between adjacent peers along the inverse path from $Node_s$ to $Node_t$ with exponential decay of distance:

$$\Delta Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} = \frac{G}{(dist(N_{\mathrm{nh}}, Node_t)+1)^l} \tag{6}$$

, where $dist(N_{\mathrm{nh}}, Node_t)$ is the hops of between N_{nh} and $Node_t$. Fig. 2 shows an example of gain update.

Forward and Routing Table Design

In this section, we introduce the query routing and several routing table design methods based on the machine learning technologies.

Figure 2. Example of gain update

Design based on Template Match. When forwarding a query with query object f, firstly, each current neighbor peer N_{nh} in CNT finds top n most similar peers in HNT. We denote the set of these peers as $MS(N_{\mathrm{nh}})$. The Euclidean distance is used to measure the similarity between two peers:

$$d(N_{\mathrm{nh}}, \hat{N}_{\mathrm{nh}}) = \left[\sum_{h=1}^{n} (F_{\mathrm{ft}}(N_{\mathrm{nh}})_h - (F_{\mathrm{ft}}(\hat{N}_{\mathrm{nh}})_h)] \right]^{1/2} \tag{7}$$

The forward probability can be obtained according to the following formulas:

$$p(N_{\mathrm{nh}}, f) = p(\arg\min_{N_{\mathrm{hnh}}} d(N_{\mathrm{nh}}, N_{\mathrm{hnh}}), F_{\mathrm{ft}}(f)) \tag{8}$$

$$p(N_{\mathrm{hnh}}, F_{\mathrm{ft}}(f)) \approx \frac{N_{\mathrm{hnh}}}{\sum_{N_{\mathrm{hnh}} \in HNT} Gain_{N_{\mathrm{hnh}}, F_{\mathrm{ft}}(f)}} \tag{9}$$

In this method the Routing Table (RT) is actually the GT.

Design based on Clustering. The searching algorithm using the routing table based on clustering is called RISA-CL in this paper. In this method, we use clustering technologies, and considering the

context of Peer-to-Peer system, we adopt sequential algorithms for clustering. The routing table now is a 4-tuple:

$$\mathrm{RT} = < C_i, \mathbf{M}_{C_i}, F_{\mathrm{ft}}(f), UpdateTime, Gain_{C_i, F_{\mathrm{ft}}(f)} > \tag{10}$$

where

$$Gain_{C_i, F_{\mathrm{ft}}(f)} = \sum_{N_{\mathrm{nh}} \in C_i} Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)} \tag{11}$$

, and C_i is the set of peers (in HNT) that belong to this class, and $\mathbf{M}_{C_i}$ is the class center where:

$$\mathbf{M}_{C_i} = \frac{\sum_{N_{\mathrm{nh}} \in C_i} F_{\mathrm{ft}}(N_{\mathrm{nh}})}{|C_i|} \tag{12}$$

The current neighbor peer's forward probability for the file f is computed according to the following procedure:

For each current neighbor peer N_{nh}, we select the most similar class in routing table, and we measure the similarity between N_{nh} and C_i using the Euclidean distance between N_{nh} and $\mathbf{M}_{C_i}$:

$$d(N_{\mathrm{nh}}, C_i) = \left[\sum_{h=1}^{n} (F_{\mathrm{ft}}(N_{\mathrm{nh}})_h - (\mathbf{M}_{C_i})_h)] \right]^{1/2} \tag{13}$$

The forward probability of N_{nh} for f now is considered as the most similar class's forward probability for f :.

$$p(N_{\mathrm{nh}}, f) = p(\arg\min_{C_i} d(N_{\mathrm{nh}}, C_i), F_{\mathrm{ft}}(f)) \tag{14}$$

where

$$p(C_i, F_{\mathrm{ft}}(f)) \approx \frac{Gain_{C_i, F_{\mathrm{ft}}(f)}}{\sum_{i=1}^{m} Gain_{C_i, F_{\mathrm{ft}}(f)}} \tag{15}$$

Design based on linear regression. We can also model the peer's gain for each kind of files directly. In this method, we use the linear regression to estimate the gain function:

$$h_{F_{\mathrm{ft}}(f)}(N_{\mathrm{nh}}) = w_0 + \sum_{j=1}^{n} w_j f_j(F_{\mathrm{ft}}(N_{\mathrm{nh}})) = \mathbf{w}^T f(F_{\mathrm{ft}}(N_{\mathrm{nh}})) \tag{16}$$

, where $f_j(\cdot)$ are known as basis functions. We used the Gaussian basis function in this paper::

$$f_j(F_{\mathrm{ft}}(N_{\mathrm{nh}})) = \exp\left\{ -\frac{(F_{\mathrm{ft}}(N_{\mathrm{nh}}) - m_j)^2}{2s^2} \right\} \tag{17}$$

The cost function is:

$$Q_{F_{\mathrm{ft}}(f)}(\mathbf{w}) = \frac{1}{2} \sum_{N_{\mathrm{nh}} \in \mathrm{HNT}} (h_{F_{\mathrm{ft}}(f)}(N_{\mathrm{nh}}) - Gain_{N_{\mathrm{nh}}, F_{\mathrm{ft}}(f)})^2 \tag{18}$$

We use the batch gradient descent algorithm to estimate $\mathbf{w}$. The item of the routing table is a tuple of 4 elements:

$$\mathrm{RT} = < F_{\mathrm{ft}}(f), \mathbf{w}, f(\cdot), UpdateTime > \tag{19}$$

When receiving a new query message with object f, we can estimate the gain value of each current neighbor, and then the forward probability can be computed by:

$$p(N_{\text{nh}}, f) = \frac{h_{F_{\text{ft}}(f)}(N_{\text{nh}})}{\sum_{Node \in \text{CNT}} h_{F_{\text{ft}}(f)}(Node)} \tag{20}$$

The searching algorithm using the routing table based on linear regression is called RISA-LR in this paper.

Design based on single density estimation. In this method, we model the forward probability directly. For each kind of files, we assume that each $F_{\text{ft}}(Node)$ stems from a single probability distribution $p(\mathbf{x} \mid F_{\text{ft}}(f), \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$. We need to build training set using the GT at first, and then estimate the probability density. The item of the RT is a tuple of 3 elements:

$$\text{RT} =< p(\mathbf{x} \mid F_{\text{ft}}(f), \boldsymbol{\theta}), \boldsymbol{\theta}, UpdateTime > \tag{21}$$

The forward probability in the training set can be estimated by gain normalization:

$$\hat{p}(F_{\text{ft}}(N_{\text{nh}}) \mid \boldsymbol{\theta}, F_{\text{ft}}(f)) \approx \frac{Gain_{N_{\text{nh}}, F_{\text{ft}}(f)}}{\sum_{N_{\text{nh}} \in \text{HNT}} Gain_{N_{\text{nh}}, F_{\text{ft}}(f)}} \tag{22}$$

To estimate $\boldsymbol{\theta}$, we used ML (Maximum Likelihood) method:

$$\ln \hat{p}(F_{\text{ft}}(\text{HNT}) \mid \boldsymbol{\theta}, F_{\text{ft}}(f)) = \sum_{N_{\text{nh}} \in \text{HNT}} \ln \hat{p}(F_{\text{ft}}(N_{\text{nh}}) \mid \boldsymbol{\theta}, F_{\text{ft}}(f)) \tag{23}$$

The searching algorithm using the routing table based on single density estimation is called RISA-SDE in this paper.

Design based on mixture d ensity estimation. To estimate the forward probability more accurately, we can assume that for each kind of files, each $F_{\text{ft}}(Node)$ stems from mixture probability distributions:

$$p(F_{\text{ft}}(N_{\text{nh}}) \mid F_{\text{ft}}(f)) = \sum_{i=1}^{k} p(N_{\text{nh}} \mid \boldsymbol{\theta}_i, F_{\text{ft}}(f)) \tag{24}$$

. $\boldsymbol{\theta}_i$ can be estimated using the Expectation Maximization algorithm. The item of RT in this method is a tuple of $2k+1$ elements:

$$< p(\mathbf{x} \mid F_{\text{ft}}(f), \boldsymbol{\theta}_1), \boldsymbol{\theta}_1, \mathbf{L}, p(\mathbf{x} \mid F_{\text{ft}}(f), \boldsymbol{\theta}_k), \boldsymbol{\theta}_k, UpdateTime> \tag{25}$$

We retrain the learning model of RISA-LR, RISA-SDE and RISA-MDE at interval of a fixed time t for each peer.

Experiments

Experimental Setting. We use a Java based P2P simulation framework PeerSim [16] to implement our system. We compare RISA with three other traditional algorithms: Random Walk, APS and PQR both in static and dynamic network environments. We assume the peers' feature values stem from one Gaussian distribution in RISA-SDE and 3 Gaussian distribution in RISA-MDE.

In the initial state, we assume that the number of different files obeys Zipf distribution. We sort the 100 files' IDs in ascending order. The number of files in the whole network is denoted as R. For the ID with the index i in the sequence, the number of the files with this ID will be:

$$l = \frac{R \cdot i^a}{\sum_{j=1}^{100} j^a} \tag{26}$$

We choose a feature value for each of the l files randomly from the 20 different files feature values which we have presented. For each of the R files, we place it to a peer chosen randomly from the whole network. We denote the network size as N, and we assume $R < N$ in our experiments in the initial state. We deploy all the query algorithms to compare their performance in three representative

network topologies: Random graph networks, Small-world networks and Scale-free networks. The PeerSim framework has provided basic algorithms to build the three network topologies.

Performance Comparison in Static Network Environments. Here, we evaluate the performance of eight searching algorithms: RISA-TM, RISA-CL, RISA-LR, RISA-SDE, RISA-MDE, Random Walk, APS, and PQR in static network environments where no peer will join or left during the whole simulation time. The experimental results are shown in Fig.3 and Fig.4.

From the Fig.3, we can see that our methods have higher query hit rate especially the RISA-MDE. There is no more difference between APS, PQR and Random Walk in static network environments according to our experimental results. The machine learning based searching algorithms in this paper show better query performance because there are more nodes with high degree which can accumulate more information for training. Fig.4 shows the average number of messages per query. APR, PQR and Random Walk increase sharply with the TTL value. But our methods increase relatively slowly, and the message number decreases 50% in average compared to the traditional algorithms

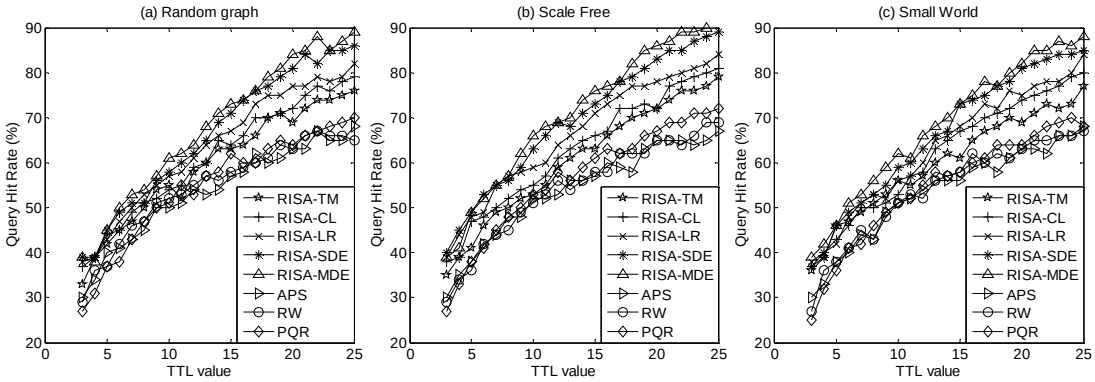

Figure 3. Query hit rate vs TTL value in static network environments

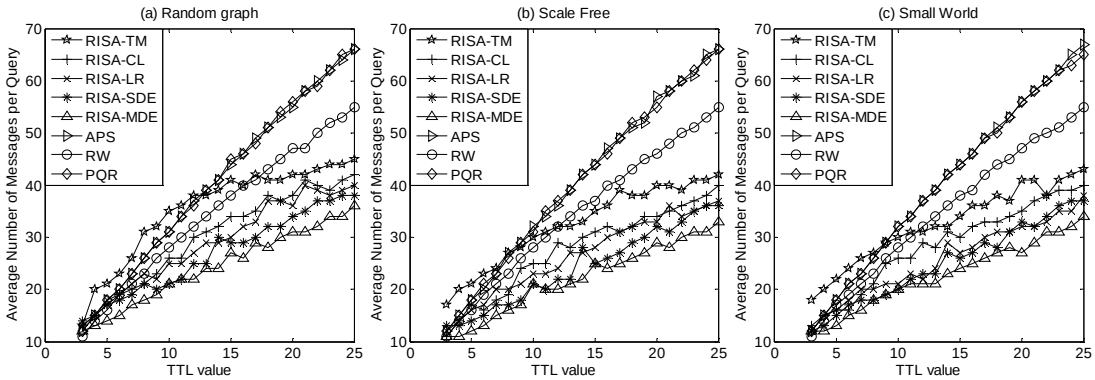

Figure 4. Averrage Number of Messages per Query vs TTL value in static network environment

Conclusion

In this paper, we proposed a robust unstructured Peer-to-Peer files sharing system. Unlike traditional informative searching algorithms which forward the query according to the current neighbors forward history about the query object, we compute the forward probability according to a special model about the class that the query object belonging to. First, we presented the object and peer feature definitions and the rule of updating the gain value. Then we presented five routing table design methods based on machine learning technologies. Finally we introduced the details of our robust informative searching algorithm. The simulation experimental results showed that our methods show better performance than traditional algorithms under both the static and dynamic network conditions.

However, we just evaluated our methods in simulation network environments with too many limitations. The machine learning technologies used in this paper must need enough data to get a good estimation, but each peer cannot accumulate enough information for training. In our future works, we will carry out the experiments under more realistic environments, and design routing table using small-sample machine learning technologies.

References

[1] T. Klingberg and R. Manfredi. Rfc-gnutella, http ://rfcgnutella.sourceforge. net.

[2] Q. Lv,P. Cao,E. Cohen,K. Li,S. Shenker,Search and replication in unstructured peer-to-peer networks,in: Proc.of the ACM SIGMETRICS'02,ACM Press,2002:258–259.

[3] Gkantsidis C, MihailM, SaberiA. Random walks in peer-to-peer networks. In:Proceedings of IEEE conference on compute rcommunications,2004:7–11.

[4] Gkantsidis C,Mihail M,Saberi A.Hybrid search schemes for unstructured peer-to-peer networks. In:Proceedings of IEEE conference on computer communications, 2005.p.1526–37.

[5] D. Tsoumakos, N. Roussopoulos. Adaptive probabilistic search for peer-to-peer networks,in: Proc. of Third International Conference on Peer-to-Peer Computing (P2P'03),IEEE,September 2003:102–109.

[6] Qianbing Zheng, Xicheng Lu, Peidong Zhu, Wei Peng. An efficient random walks based approach to reducing file locating delay in unstructured P2P network[C]. Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE . 28 Nov.-2 Dec. 2005:980-984

[7] Ronasi,K.,Firooz,M.H.,Pakravan,M.R.,Avanaki,A.N.. An Enhanced Random-walk Method for Content Locating in P2P Networks [C].Distributed Computing Systems Workshops, 2007. ICDCSW '07. 27th International Conference ..22-29 June 2007.

[8] Filali Imen, Huet Fabrice. Dynamic TTL-Based Search in Unstructured Peer-to-Peer Networks [C].Cluster, Cloud and Grid Computing (CCGrid),2010 10th IEEE/ACM International Conference. 17-20 May 2010:438 – 447.

[9] V. Kalogeraki, D. Gunopulos, D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks,in: Proc. of Conference on Information and Knowledge Management(CIKM'02),2002:300–307.

[10] A. Kumar, J. Xu, and E. W. Zegura. Effcient and scalable query routing for unstructured peer-to-peer networks. In Proc. IEEE INFOCOM, pages 1162–1173, Miami, FL, United States, March 2005.

[11] Ming Xu, Shuigeng Zhou, Jihong Guang,et al. A path-traceable query routing mechanism for search in unstructured peer-to-peer networks. Journal of Network and Computer Applications, 2010:115-127.

[12] C. Y. Liau, A. N. Hidayanto, and S. Bressan. Adaptive peer-to-peer routing with proximity. In Proceedings of DEXA Conference, 2003.

[13] L. Gatani, G. L. Re, A. Urso, and S. Gaglio. Reinforcement learning for P2P searching," in Proc. of the International Workshop on Computer Architecture for Machine Perception (CAMP'05), 2005.

[14] Li, X. and J. Wu. Improve Searching by Reinforcement Learning in Unstructured P2Ps. Proceedings of the 26th IEEE International Conference Workshops on Distributed Computing Systems, 2006.

[15] R. Beverly and M. Afergan. Machine learning for efficient neighbor selection in unstructured p2p networks. In SYSML'07: Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[16] MONTRESOR, A., AND JELASITY, M. PeerSim: A Scalable P2P Simulator. In Proceedings of the 9th P2P Conference,2009.