RSM-based Gossip on P2P Network[1]

Hai Zhuge and Xiang Li

China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences, 100080, Beijing, China
zhuge@ict.ac.cn, xiangli@kg.ict.ac.cn

Abstract. Classification is a kind of basic semantics that people often use to manage versatile contents in daily life. Resource Space Model (RSM) is a semantic model for sharing and managing various resources using normalized classification semantics. Gossip-based peer-to-peer (P2P) techniques are reliable and scalable protocols for information dissemination. Incorporating RSM with gossip-based techniques forms a new decentralized resource sharing mechanism with the improved performance of unstructured P2P systems. Theoretical analysis and experiments validate the feasibility of the mechanism. Such incorporation is a way to synergy normalization and autonomy in managing decentralized large-scale complex resources.

1 Introduction

P2P systems aim at decentralization, scalability, ad-hoc connectivity, reduced cost of ownership and anonymity [1]. Unstructured P2P networks allow peers to self-organize and resources to be randomly placed. Such networks have low maintenance cost and are robust against accidental failures. Simulating the propagation of contagious diseases, gossip mechanisms have attractive scalability, reliability and degradation properties in realizing information dissemination in large networks [2]. Every node that receives a message randomly selects a certain number of nodes from its neighbors to multicast the message. They scale well since the load of nodes grows logarithmically compared with the number of nodes in the network. The performance of the gossip mechanisms can be improved in semantic space by designing appropriate mapping from the network into semantic space [12]. Ontology has been used to improve structured P2P systems [10]. Classification is a kind of basic semantics that people often use to effectively manage versatile contents in daily life.

---

[1] Keynote at the 7th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'07), Hangzhou, China, 2007. This work is supported by the National Basic Research Program of China (973 Semantic Grid Project, Grant No. 2003CB317001), the International Cooperation Program of Ministry of Science and Technology of China (Grant No. 2006DFA11970), and the EU 6th Framework Program GREDIA (Grant No. IST-FP6-034363).

*Technical Report of Knowledge Grid Research Center, KGRC-2007-01, March, 2007, www.knowledgegrid.net/TR.

A Resource Space Model RSM is a semantic model for effectively sharing and managing various Web resources (information, knowledge and services) based on normalized classification semantics [11]. Incorporating resource space with gossip mechanisms is a way to improve the performance of P2P network.

2. Incorporating RSM with P2P

An *n*-dimensional Resource Space represents *n* kinds of partition on a set of resources. A Resource Space can be mapped onto a partition tree (e.g., Fig. 1(a) can be mapped onto Fig. 1(b)). The classification semantics of a partition tree can be used to improve the performance of a P2P system because a peer could get the satisfied answer with high probability by interacting more frequently with the peers of the same community sharing common interests. Peers also need to communicate with peers of other communities.

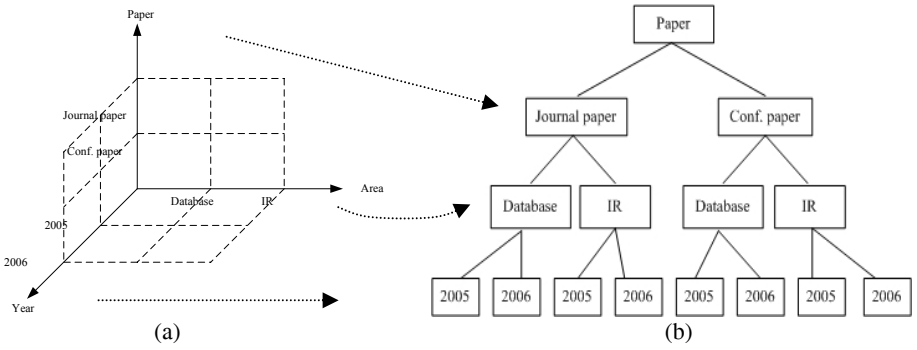


Fig. 1. (a) A 3-dimensional resource space. (b) The partition tree corresponding to Fig.1 (a).

Each leaf corresponds to peers in the same category. The tree is a rather stable commonsense, and the whole system could use only a part of it. The communities in leaves of the partition could change with joining and departing of peers.

As shown in Fig. 2, peers can be classified into communities corresponding to the leaves of the semantic partition tree. Each peer maintains neighbors in a hierarchical structure. The number of layers of the hierarchical structure a peer maintains depends on the depth the peer lies in the partition tree. Taking a peer *p* in the bottom-left community of the partition tree for example, it should maintain four layers of its neighbors, denoted as View(*i*) where $0 \le i \le 3$. View(*i*) is a set/list containing the neighbors' information (address etc.) that shares the nearest common ancestor at *i*th level with *p*. *p*'s View(3) maintains the information of some peers within the same community, while *p*'s View(2) maintains the information of its neighbors having the nearest common ancestor at level 2, and so on.

When a peer sends a query, it will make a decision on which level(s) in its view should be selected to forward the query (category of the level are relevant to the query). Then, neighbor(s) at that level will be selected to forward the query. When a query reaches a community, a gossip-based mechanism will be adopted to

disseminate the message. The peer that receives and could answer the query sends back the corresponding resources.

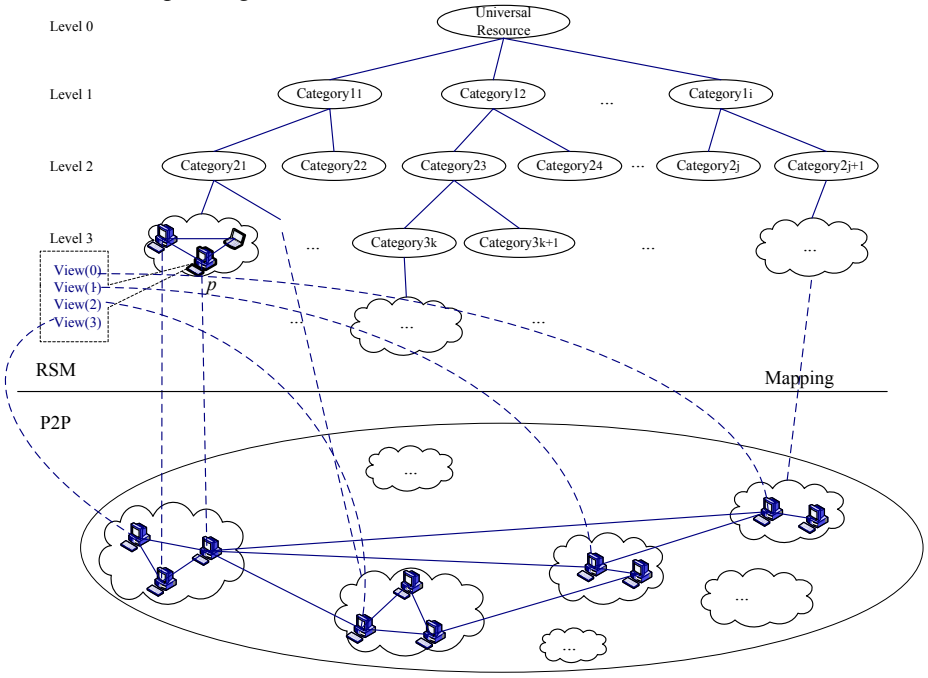


Fig. 2. Incorporating one resource space with a P2P network.

3. The Construction Mechanism

It is reasonable to assume that all peers share a consistent knowledge of partition. A partition hierarchy could be known by informing the newly joined peer of the partition ontology.

The similarity between the peer and the category is measured by $\cos\theta = A \bullet B / (||A|| \bullet ||B||)$, where A and B are term vectors of the new peer and the category respectively [3]. The category with the maximum similarity value is chosen as the category that the joined peer belongs to.

The static partition of the resource space can bring important advantage to P2P systems. When a new peer joins, it just needs to contact one peer which will feed back the partition information on the resource space. Using the partition information, the peer determines which category it belongs to. If some resource indices included in the peer belong to the other categories, the indices are reissued to other peers that are in charge of those resource categories. Using the information the first-contacted peer provided, the peer contacts the peers in the categories it belongs to and update its neighbors' information.

In the partition tree, the universe resource is assigned *Level*0. Then, the universe space is first divided into a set of categories (in Fig. 1(b), it consists of *Journal paper* and *Conf. paper*), which constitute *Level*1. The categories in *Level*1 are further partitioned into categories constituting *Level*2, and so forth. Along with the peers joining the system, the leaf categories produced in the aforementioned mechanism can be further partitioned dynamically into different levels. The dynamic partition of leaves can be realized through a group size limit *gl*. When the size exceeds the limit, the group is partitioned into two parts with size $\lfloor gl/2 \rfloor$ and $\lceil gl/2 \rceil$ respectively.

The dynamic partition of the space works well with the skewed data distribution such as power-law distribution found in many circumstance. Meanwhile, the static characteristics of the resource space partition improve the scalability of P2P system that peers join and depart autonomously and frequently, and reduce the cost of update when peers join or depart.

The disseminated message contains the content and some assistant information: (1) the type of the message, which could be *join*, *leave*, *issuing*, or *query*; (2) time to live (*TTL*) of the message, which is used in *join*, *issuing* and *query* messages; and, (3) a list of identifiers of the peers that have received the message.

The following are definitions used for easier discussion:

(1) *fanout* — the number of neighbors one peer selects to disseminate when it receives a message;
(2) *TTL* (Time To Live) — the iterative rounds for a message to disseminate;
(3) *outView*(*i*) — the neighbors that peer *i* can send messages to; and,
(4) *inView*(*i*) — the neighbors that peer *i* can receive messages from.

3.1 Resource Issuing Process

When a resource index is issued by one peer, the peer *r* first decides the category the resource belongs to by utilizing the partition information of the resource space. A limit *l* restricting the whole steps of the process should be set. Along with each hop the message has transferred, *l* will be reduced by one. Then the peer forms an *issuing* message which includes the resource index, and sends the message to one of the peers in that category through its level views. When a peer receives the message, it first decides whether to add the index to its maintaining repository in consideration of its capacity. If the capacity will exceed its upper limit, then it randomly selects one neighbor from its proper level view and disseminates the *issuing* message. The issuing process will proceed until the resource index is accepted or *l* reaches zero. When *l* is zero, peer *r* also joins the community to manage the resources.

3.2 Peer Join Process

When one peer is going to join the system, it first connects to one of the contacted peers. With the information of the resource space fed back from the contacted peer, the newly joined peer decides its category with reference to the categories of its major resources. If there is more than one community in the category, the contacted peer randomly chooses a community. Then, the contacted peer forms a *join* message

including the joining peer's information, and forwards the message to one of the peers in that community utilizing its level views.

During the process, a limit *sl* restricting the whole steps of dissemination should be set. Along with each hop the message transferred, *sl* will be reduced by one. When the peer receives the message, it first decides whether to add the peer to its view with reference to its view size. If this causes the overflow of the view size, it will forward the *join* message to one randomly selected neighbor in the community until the peer is accepted or *sl* is zero. If the joining peer is still not accepted by one peer when *sl* reaches zero, the community is regarded as full and a new community should be created in the same resource space position as the full community. The joining peer forms its level views by exchanging information with peers in the same category.

The newly joined peer maintains the index information of its major resources and issues the resource indices not belonging to its resource space position to the system adopting the aforementioned resource issuing mechanism.

To further break the existing community into approximately equal size, a simple mechanism could be adopted: peers initiate a random interaction when they have not decided which community they belong to. If the contacted peer is undecided also, the two peers make choice for different communities. Otherwise, the peer chooses the different community from the contacted peer.

3.3 Peer Departure Process

When peer *r* wants to depart the system, the following method is used to keep the peers in *r*'s *inView* and those in its *outView* connected. For each peer (take peer *s* for example) in its *outView*, peer *r* selects one of the peer ID (*q* for example) from its *inView* randomly, then forms a *failure* message including *q* and forwards it to *s*. When *s* receives the message, it will substitute *r*'s ID(*r*) with *q* in its *inView*, then forwards a message with *r* and *q*. When *q* receives the message, it will update *r* with *q* in its *outView*. So the withdrawal behavior of a pivot peer will not lead to the partition of the whole network.

When a peer crashes without notifying other peers, the peers can detect this situation by interchanging their states periodically [8]. After a certain period elapsed, if no response is returned from one of its neighbors, the peer deems it as being crashed and removes it from corresponding view.

If one existing community is small, it is necessary to merge communities in the same parent category: If a peer *p* wants to hand off its index, it should find its siblings first. If the siblings of the peer *p* are also leaves of the partition tree and the number of its siblings is one (*q*, for example), then simply coalesce *p* and *q*, make their direct parent a leaf, and assign peer *q* to that leaf. In this way, the indices of *p* and *q* merge into a single index that is assigned to peer *q*. If the number of its siblings is larger than one, then pick up one leaf that has the least load and hand off the index to the picked peer.

If the siblings of the peer *p* are not the leaves of the partition tree, perform the depth-first search in the sub-tree of the partition tree rooted at one of its siblings such as *q* until the leaves of the sub-tree are reached. Hand off the index of peer *r* in the leave to one of its siblings. Peer *r* takes over the index of peer *p*.

3.4 Query Processing Process

When sending a query, the peer first compares the query with its index on resources, and then adopts different mechanisms to gossip queries making use of neighbor lists at different levels. In most applications, the resources a peer possesses reflect its interests, and the queries from the peer would be similar to its interests with high probability. In this situation, the query could be answered in the community the query-initiator belongs to, and only the neighbor list at the lowest level is needed for the query processing. While the query fits perfectly with other level, it will be routed to that appropriate category, and a gossip process initiates there. When the query corresponds to more than one level, then a certain number of gossip processes take place in parallel in the corresponding categories. The top-k correlative categories can be selected to make trade-off between the whole network cost and the acceptable results.

Haming distance is a suitable distance measure for multidimensional non-ordered discrete data space [7], which can be regarded as the correlative metric between query and the categories. Hamming distance $dist(a_1, a_2)$ between vectors a_1 and a_2 in an discrete data space is the number of dimensions on which the corresponding components of a_1 and a_2 are different. The distance between a vector $a=(a_1, a_2, \ldots, a_d)$ and a discrete rectangle $S=S_1*S_2*\ldots*S_d$ can be defined as:

$$dist(a, S) = \sum_{i=1}^{d} f(a_i, S_i), \text{ where } f(a_i, S_i) = \begin{cases} 0 & a_i \in S_i \\ 1 & otherwise \end{cases}.$$

4. Performance Analysis

4.1 Reliability

Suppose the number of peers in the system is n, and the resource space partitions the peers into m categories. For the purpose of simplicity, group members are assumed to be evenly distributed, that is, the sizes of categories are equal to n/m approximately. And, the assumption will be relaxed in the experiments. We use the following notations for further discussion.

(1) s—the source peer of one message;
(2) ε—the probability of message loss during the gossip process.
(3) τ—the probability of a peer crash during the gossip process.
(4) A—the event that there is a directed path from s to all peers in the category s belongs to.
(5) B—the event that there is at least one link directed from s to other categories.
(6) $P(C)$—the probability that the event C happens.

Gossip style protocols are reliable in a probabilistic sense. By adopting the analysis in [4], the probability of a given peer receiving the disseminated message will be $1-(1/n^{fanout})(1 + o(1))$. And if the message loss is considered, the probability will be $1-(1/n^{(1-\varepsilon)\,fanout})(1+o(1))$.

We can disseminate the message in three different mechanisms:

(1) Using the partition tree, the original source peer could make a decision and pick up one level in its views. Thereafter one peer in the view at this level is randomly selected, and a gossip process is initiated with the selected peer being the source. In this precondition: P(every peer in the selected category will receive the message)= $P(A)\cdot P(B)= (1-(1/(n/m)^{(1-\varepsilon)\,fanout})(1+o(1)))\cdot(1-\varepsilon)\cdot(1-\tau)$. Applications could make the messages communicate reliably through protocols like TCP, and in this way, ε would approach zero. In addition, n, being a large number in P2P systems, leads the protocol to be reliable.

(2) The original source peer randomly selects one peer from each of its views, and disseminates the message to them, (the number of levels will be m at most). The selected peers launch the gossip process in its group in parallel. And under the condition: P(every peer in the system will receive the message)= $P^m(A)\cdot P(B)= (1-(1/(n/m)^{(1-\varepsilon)\,fanout})(1+o(1)))^m\cdot(1-\varepsilon)\cdot(1-\tau)$.
In the selecting process, if sending message to the selected peer is failed, another peer could be selected randomly in the view at the same level. In this way, the bad influence of m in the previous equation will be further reduced to guarantee the reliability of the mechanism.

(3) The original source peer selects peers in different level view with different probabilities. And then the selected peers will receive the message and disseminate it in their communities. Therefore, P(every peer in the selected communities will receive the message) = $P^l(A)\cdot P(B)$, where l is the number of communities being selected and $1 \le l \le m$. Consequently, $P^m(A)\cdot P(B) \le P^l(A)\cdot P(B) \le P(A)\cdot P(B)$, and the gossip process adopting the mechanism will be reliable.

4.2 Hop Count Expectation

With reference to [6], the total rounds $TTL(n, fanout)$ in the gossip-style system, necessary to infect an entire group of size n obeys: $TTL(n, fanout) = \log n\cdot(1/fanout+1/\log(fanout)) + c + o(1)$, where c is a constant. There exists a tradeoff between *fanout* and *TTL* in the network of n peers. Therefore in our systems, all peers are partitioned into different categories by one resource space. Assume the sizes of categories are equal approximately, i.e., n / m, the round of message dissemination in the sub-partitions will be: $TTL(n / m, fanout)$. Considering the category selecting process, the hop count of message dissemination in the whole system will be: $TTL_1(n, fanout) = 1 + TTL(n / m, fanout)= \log(n/m)\cdot(1/fanout+1/\log(fanout)) + c_1 + o(1)$, where c_1 is a constant.

5. Experimental Evaluation

Experiments are carried out on the topologies using flat gossip mechanism and using our semantic partitioning mechanisms. The experiments are carried out on two kinds of directed networks of 1000 nodes: random networks and random power-law

networks. Each experiment with different parameters (*fanout* and *TTL*) is repeated 100 times on each network we generated, and the initial node is randomly selected for each time. The average value of these 100 results is used to illustrate the feasibility.

Considering the graph of n nodes where the edge between each pair of nodes is present with probability $[\log(n) + c + o(1)] / n$. In the prerequisite, the probability that the graph is connected goes to $exp(- exp(- c))$, where c is a constant. And the target could be reached by defining the appropriate *View* sizes of nodes. For the networks constructed this way, there is a sharp threshold in the required *fanout* at $\log(n)$ [4]. Therefore, the gossip systems with size n will have promising effect when the *fanout* value is set to be around $\log(n)$. Two performance metrics are: (1) comparisons between average network load, and (2) the number of nodes that do not receive the message using different mechanisms.

5.1 Random Networks

Experiments are done on the directed graphs with 1000 nodes. An epidemic algorithm must make a tradeoff between scalability and reliability: larger views reduce the probability that nodes are isolated or that the network is partitioned, while smaller views help the network obtain better scalability. For the random networks, the number of neighbors of each node at lowest level is 10 in our experiments on average. And the view sizes of other levels are rather smaller, in the simulation, it is 2. The community size is 100 in the experiment.

5.2 Random Power-law Networks

Many large networks like the hyperlink network follow the power law distribution of node degrees [5]. The degree distribution is $p_k = Ak^{-\tau}$, where $A^{-1} = \sum_{k=2}^{k_{\max}} k^{-\tau}$, k is the degree, k_{max} is the maximum degree, and $\tau > 0$ is the exponent of the distribution [9]. Researches have shown that only when the virus accumulates to certain critical threshold, it will be prevalent. And, when the virus is working on the networks that follow the power-law distributions, the critical threshold does not exist. The virus does not need to accumulate to certain limitation, and it could propagate quickly through hubs of the network.

The reason of considering power-law networks is that some unstructured P2P networks are characterized by random power-law and heavy tailed degree distributions. To keep the nodes connected, we adjust degree from 15 to 100 following the aforementioned distribution with $\tau = 2.0$ in constructing random power law networks. For each link, the start node and the end node are selected randomly, and as a result, the random power-law graph is constructed with average 14 neighbors in the lowest level. The view sizes of other levels are rather smaller, and its size is 2. The community size is 100 in the experiment

The simulation results from gossip networks without considering semantic partitions are denoted as *FlatGossip*, while the results making use of semantic partitions are denoted as *RSMGossip*. In the partition-based gossip mechanisms,

different number of gossip levels could be chosen according to the comparison between the query and the category the query initiating peer is in. If the query strictly belongs to one category, then routing the query to other categories will not bring any benefit. And in this situation, the results are denoted by *RSMGossip1*. When the query corresponds to several categories, it should be routed to all the categories to obtain the complete results. For example, the query is going to be answered in 3 or 5 categories, and the results are denoted by *RSMGossip3* and *RSMGossip5*.

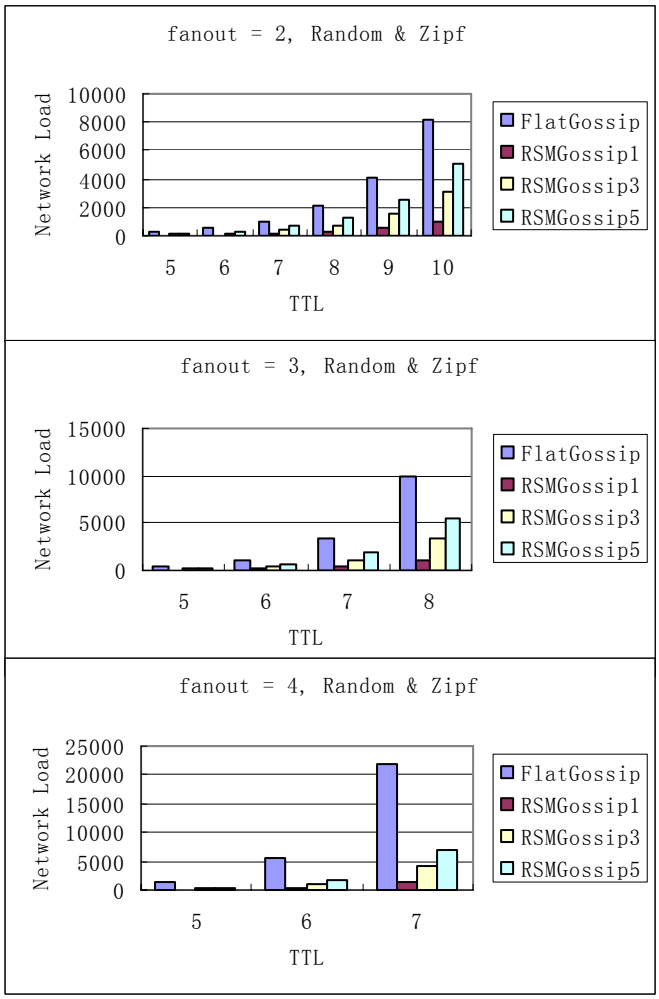


Fig. 3. Comparisons of the network load among different mechanisms in the 1000 node networks. (a). *fanout* = 2. (b). *fanout* = 3. (c). *fanout* = 4.

Both on the random networks and the random power-law networks, if the networks have the equal size and the gossip mechanisms have the same parameters (*fanout* and *TTL*), the network load will be the same. Fig. 3 shows average network load

according to different parameters. The horizontal axis denotes the parameter *TTL*, and the vertical axis denotes the average network load during 100 times operation. As Fig. 3(a) presents, we set the *fanout* value as 2 uniformly and range *TTL* from 5 to 10. Fig. 3(b) and Fig. 3(c) are obtained in the similar way by using different parameters' values. We can see from the figures that the network loads are reduced sharply when adopting the partition-based gossip mechanisms. Taking *fanout* = 3 for example, when *TTL* approaches 8, about 88.9%, 66.7% and 44.5% network loads are reduced by *RSMGossip1*, *RSMGossip3* and *RSMGossip5* respectively comparing to the flat gossip mechanism, which justifies our approaches.

(a)

(b)

(c)

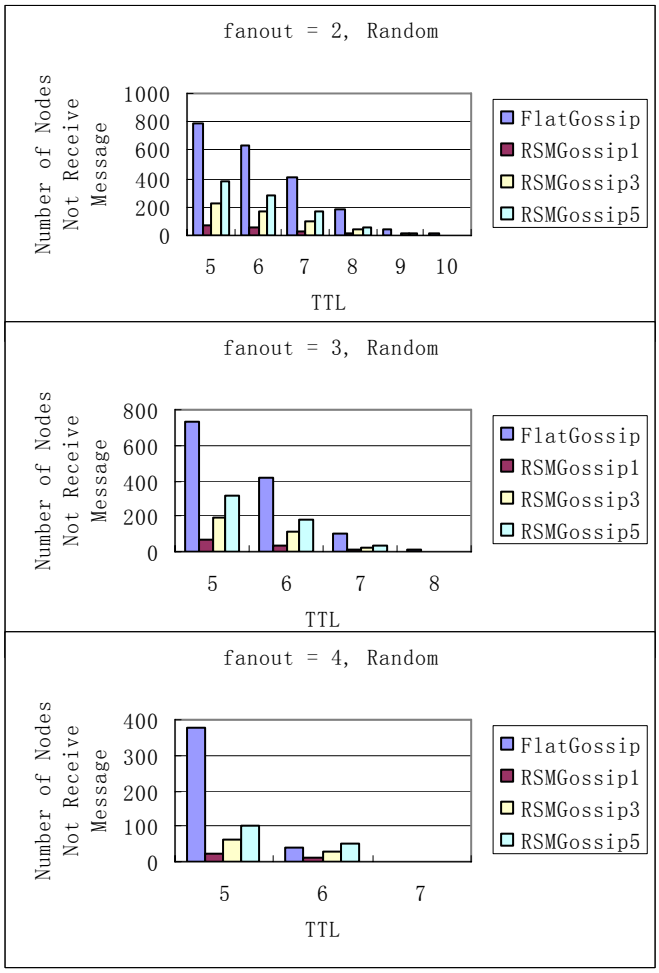
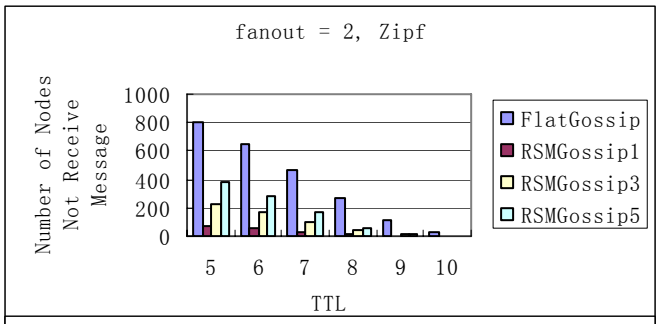


Fig. 4. Comparisons of the number of nodes that do not receive messages in the 1000-node randomly connected networks. (a). *fanout* = 2. (b). *fanout* = 3. (c). *fanout* = 4.

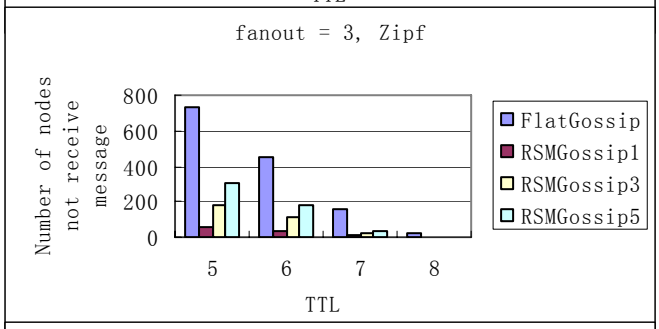
During message dissemination, the number of nodes that do not receive the disseminated messages is an important issue we concern. Compared with the previous algorithms, the number of nodes that do not receive messages decreases evidently

after adopting the proposed mechanisms as presented in Fig. 4. Taking *fanout* = 3 for example, when *TTL* approaches 6, about 36.44, 109.32 and 182.2 number of nodes, which should receive the disseminated message, has not received it when making use of *RSMGossip1*, *RSMGossip3* and *RSMGossip5* mechanisms separately. Meanwhile it is 416.57 for the flat gossip mechanism. Though it is partly because the range is decreased for the partition-based mechanism, the performance is improved considerably, which justifies the rationale of the semantic partitioning mechanism.

(a)



(b)



(c)

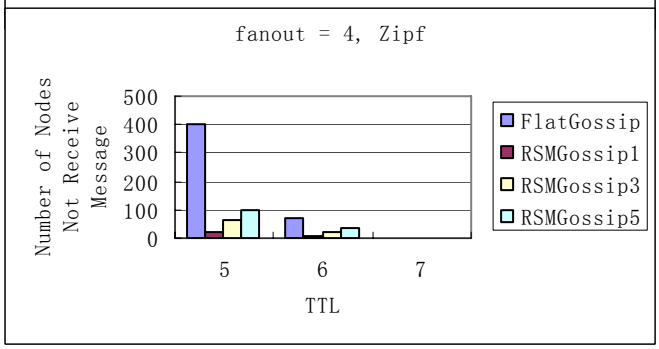


Fig. 5. Comparisons of the number of nodes that do not receive messages in the 1000-node random power-law networks. (a). *fanout* = 2. (b). *fanout* = 3. (c). *fanout* = 4.

For the random power-law networks, the results about the number of nodes that do not receive messages are presented in Fig. 5. We can see the similar phenomena as those done on the random networks: the number of nodes that do not receive messages is also reduced sharply comparing with the partition-based gossip

mechanisms to the flat gossip. Taking *fanout* = 3 for example, when *TTL* approaches 6, averagely about 35.77, 107.31 and 178.85 number of nodes do not receive the disseminated message when making use of *RSMGossip1*, *RSMGossip3* and *RSMGossip5* mechanisms respectively. Meanwhile for the flat gossip mechanism it is 455.13. The results are better than those on the random networks for the partition-based mechanisms, while the flat gossip mechanism performs worse on the random power-law networks than that on the random networks, which also declares the necessary of our approaches.

6. Conclusion

Incorporating the classification semantics with the gossip mechanisms can improve the performance of P2P network. The RSM-based gossip on P2P network owns the advantages of both RSM and P2P, and can synergy the normalization and autonomy in decentralized resource management. RSM's normalization theory, integrity theory and operation language can support semantic-rich applications over P2P networks.

References

1. Androutsellis-Theotokis, S. and Spinellis, D.: A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36 (4) (2004) 335-371.
2. Bailey, N.T.J.: The Mathematical Theory of Infectious Diseases and Its Applications. *Hafner Press*, 1975.
3. Berry, M.W. et al.: Matrices, Vector Spaces, and Information Retrieval. *Society for Industrial and Applied Mathematics Review*, 41 (2) (1999) 335-362.
4. Kermarrec, A.M., Massoulié, L. and Ganesh, A.J.: Probabilistic Reliable Dissemination in Large-scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14 (3) (2003) 248-258.
5. Leland, W.E. et al.: On the Self-similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking*, (1994) 1-15.
6. Pittel, B.: On Spreading a Rumor. *SIAM Journal of Applied Mathematics*, 47 (1) (1987) 213-223.
7. Qian, G. et al.: Dynamic Indexing for Multidimensional Non-ordered Discrete Data Spaces Using a Data-partitioning Approach. *ACM Transactions on Database Systems* (*TODS*), 31 (2) (2006) 439-484.
8. Renesse, R.V., Minsky, Y. and Hayden, M.: A Gossip-style Failure Detection Service. *Middleware98: IFIP international Conference, Distributed Systems and Platforms and Open Distributed Processing*, *Springer*, (1998), 55-70.
9. Sarshar, N. et al.: Percolation Search in Power Law Networks: Making Unstructured Peer-to-Peer Networks Scalable. In: *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, (2004) 2-9.
10. Schlosser, M. et al.: A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In: *Proceedings of the 2nd Int'l Conf. Peer-to-Peer Computing*, (2002).
11. Zhuge, H.: The Knowledge Grid. *World Scientific Publishing Co.*, 2004.
12. Zhuge, H. and Li, X.: Peer-to-Peer in Metric Space and Semantic Space. *IEEE Transactions on Knowledge and Data Engineering*, 19 (6) (2007).