RT-P2P: A Scalable Real-Time Peer-to-Peer System with Probabilistic Timing Assurances

Fei Huang* Binoy Ravindran* E. D. Jensen‡

*ECE Dept., Virginia Tech
Blacksburg, VA 24061, USA
{huangf,binoy}@vt.edu

‡The MITRE Corporation
Bedford, MA 01730, USA
jensen@mitre.org

Abstract

We present RT-P2P, a real-time peer-to-peer (P2P) system that allows application-level end-to-end timing requirements to be satisfied in P2P systems. P2P systems are fundamentally characterized by: a large number of geographically distributed nodes that require little infrastructural support from the underlying network; an unbounded number of nodes (a permanently evolving set of nodes); and consequently no process/node with global knowledge of the system structure. Interesting features of such networks include the fact that they allow a rich set of nodes to act as relay points for other nodes, and a rich set of overlay paths (selected by peers) to be constructed. These features have traditionally made overlay routing – where end hosts have the flexibility of routing traffic to their destinations through the desired choice of intermediate overlay nodes (unlike in IP routing) – a very attractive approach for end-to-end performance optimizations in P2P networks. Key aspects of our RT-P2P infrastructure include a real-time P2P protocol, real-time communication algorithm, and analytical performance models. We analytically establish the timing properties of RT-P2P. Our simulation studies validate our analytical results.

1. Introduction

In this paper, we consider distributed real-time systems working over large-scale unreliable networks, e.g., those without a fixed network infrastructure or subject to interference, including mobile and wireless ad-hoc networks. Scalability and the ad-hoc nature of such system undoubtedly increase the complexity to tackle the real-time requirements under run-time uncertainties, including arbitrary message losses and node failures. For example, a tactical system in the battlefield may consist of sensor networks, satellite terminals and combat system [1], [2], which should be capable of identifying, tracking and attacking multiple objects with strict real-time constraints. There might be more than thousands of communication and computation nodes connected by ad-hoc and dynamic links under a harsh communication environment.

Traditional Client-Server model is hard to be deployed adaptively in such a large-scale ad-hoc environment with end-to-end timing assurance [3], [4]. As we know, Client-Server architecture typically involves a small cluster of servers to fulfill the service requests from connected clients [5]. Such architecture is server-dependent and can apply well above pre-configured static infrastructures [5]. However, (1) due to the ad-hoc property, the critical nodes in the system may leave from the server connections, which may provoke the failure of the entire real-time system; (2) moreover, the system consists of dynamic and hybrid networks, which makes it impractical to timely pre-configure the system to a Client-Server architecture without a prior knowledge of the underlying infrastructures; (3) scalability-wise, newly joined clients will demand more network resources, such as servers and bandwidth, yet they are limited in most situations.

In our study, we found *Peer-to-Peer* (P2P) overlay networks can solve the above constraints on distributed real-time systems without any hierarchical organization or centralized control. P2P system is a special distributed system overlayed above the underlying physical networks, where all responsibilities are uniformly divided among all participants, known as *peers* [6], [7]. Typically, a P2P system exhibits the following features [5], which make a natural fit to the aforementioned system requirements.

- **Resource aggregation.** In P2P systems, peers can serve both as clients and servers. In light of this, P2P systems can leverage the resources from all peers rather than conventional centralized resources where a very limited number of servers provide the core value to the services.
- **Dynamism and Ad-hoc communication.** P2P systems assume a highly dynamic working environment, where resources, such as computation nodes, will be entering and leaving the system frequently and arbitrarily. Also, system communications are built on ad-hoc links.
- **No hierarchical organization.** P2P systems can directly work atop hybrid existing networks without hierarchical organization.
- **Massive scalability.** P2P systems can accommodate millions of peers. For example, Skype, a VoIP software based on P2P technology, has been well-known to accommodate millions of users to talk online simultaneously [8].

In light of the above features, we bring up the concept of real-time P2P system (RT-P2P), which is capable of ensuring system end-to-end real-time constraints. Previous work in P2P system mainly focused on specific applications without timing constraints, such as file sharing and file storage in [7], [9], [10]. For example, in [9], Rowstron et al introduced the P2P protocol and APIs of Pastry, which provides the functionalities of global file sharing. Although the reliability of Pastry is proved to be high, they did not consider the real-time properties. On the other hand, although some of the existing P2P applications imply certain timing constraints, such as multimedia streaming [8], [10], they can only work on IP networks, and the timing constraints are egoistically defined without considering requirements from global tasks. For example, when Skype shares the bandwidth with TCP flows, Skype will dominate the bandwidth with no pity on the tasks built upon TCP flows [11]. Such selfish behavior undoubtedly undermines the system-wide timeliness behaviors. By no means, those P2P applications can be recognized as real-time P2P systems.

Our major contributions of this work include (1) proposing the novel concept of real-time peer-to-peer system; (2) designing the protocols and real-time communication algorithms for RT-P2P with the consideration of system uncertainties; (3) presenting an analytical model for RT-P2P with theoretical insights into the scalability and timeliness properties; (3) completing a series of simulations based on RT-P2P algorithms, which validates real-time performance of RT-P2P.

The rest of the paper is organized as follows. In Section 2, we discuss system models and protocols of RT-P2P. In Section 3, we build the theoretical model to analyze the real-time properties for RT-P2P. In Section 4, we present a real-time communication algorithm to enhance the real-time feature of RT-P2P. In Section 5, we report our simulation studies about scalability and timeliness performances. We conclude the paper in Section 6.

2. Real-time Peer-to-Peer System

2.1. Rationale

In RT-P2P system, the core problem beside scheduling is real-time communication, including query routing (i.e. algorithms about searching a wanted peer in P2P community). In this paper, we assume all the nodes in RT-P2P are underloaded, which means local tasks on each node can always be scheduled to satisfy the real-time requirements. This assumption allows us to focus on real-time communication in RT-P2P instead of scheduling issues.

In real-time applications, system will frequently query the P2P community for the objects that can provide required services. Query models which are widely used in state-of-the-art P2P paradigms, include *Centralized Directory Model*, *Request Flood Model* and *DHT Model* [5], [6]. In centralized directory model, peers publish the service and routing information on a central server, such as Napster [12]. This model can be easily deployed, but it has the scalability problem and may face a single point of failure. In request flood model, peers broadcast their requests among all the community. It can reach a scalability of hundreds of thousands of nodes, such as Gnutella [13]. However, the message overhead is heavy for this method. In DHT model, each peer is assigned a random ID upon registration, and each peer also knows a given number of other peers [5]. When a service is advertised on such a system, an ID is assigned to the service based on a hash of the service name. Peers will then forward the routing information of this service towards the peer whose ID is closest to the service ID in hash table. When a peer requests a service from the P2P system, the request will go to the peer with the ID most similar to the service ID. This process is repeated until a routing record is found. Although DHT model is very efficient for large-scale system, it has the problem that the service IDs must be known ahead of the query [5].

In many real-time applications, such as [14], service IDs can hardly been known before posting a request for a given service. In this paper, we present a super-peer-based backbone to facilitate real-time communication. Based on this backbone, a gossip-based query algorithm is properly designed to disseminate the query and detect the routing paths. Gossip scheme can exhibit highly scalable and robust behavior under dynamic and unreliable network conditions [15], which makes a well fit to the system requirement. In respect of message overhead, RT-P2P smartly utilizes the neighborhood information to avoid redundant queries. Once the gossip scheme returns feasible routing paths, network probabilistic theory will help estimate optimal paths for real-time communication in consideration of system uncertainties. Moreover, peers are allowed to register at multiple super-peers, which not only enhances the reliability of ad-hoc communications but also improves the scalability. From the theoretical and simulated results discussed later, RT-P2P is approved to outperform the P2P strategy without super-peer backbone and the traditional Client-Server paradigm with gossip-based query in both scalability (number of peers) and reliability (task success probability).

2.2. Distributed Task Model

We consider a distributed task abstraction that models a causally-dependent, multi-node sequential flow of execution. An example of such an execution flow is one that is caused by a series of nested, remote method invocations—e.g., invocation of remote method $\mathcal{A}$ causes the invocation of remote method $\mathcal{B}$; invocation of method $\mathcal{B}$, in turn, causes the invocation of remote method $\mathcal{C}$, and so on. Such an execution flow can also be caused by a series of chained, publication and subscription events, or due to topical data dependencies—e.g., publication of topic $\mathcal{A}$ depends on subscription of topic $\mathcal{B}$; publication of $\mathcal{B}$, in turn, depends on subscription of topic $\mathcal{C}$, and so on.

Thus, a distributed task can be viewed as being composed of a set of *subtasks*, where a subtask constitutes the portion of the task's execution on a node. If the task models a series of nested, remote method invocations, then a subtask constitutes a maximal length sequence of contiguous method executions on a node. If the task models a series of chained, publication and subscription events, then a subtask constitutes the execution of a subscription/publication service on a node.

We call the initial subtask of a task as the task's *root*. The node hosting the root is called the task's *root node*. A task's most recent subtask — i.e., the task's current execution locus — is called the task's *head*, and the node hosting this *head* is called the task's *head node*.

We assume that the number of subtasks of a task is known. Also, the execution time estimates of the subtasks are also assumed to be known. The application is thus comprised of a set of tasks, denoted $T = \{T_1, T_2, \ldots\}$.

2.3. System Model

The system consists of a set of nodes, denoted as $NS = \{n_1, n_2, n_3, \ldots\}$. Each node is represented by a peer in the RT-P2P. RT-P2P can be described by a graph $G(V, E)$ with peers denoted as V and links between peers denoted as E, where $N = |V|$ and $M = |E|$.

Nodes, i.e. peers, in the real-time system usually have different capabilities including available bandwidth, processing speed and storage space, etc. Thus, exploiting those capability difference can benefit the timing assurance in real-time task processing and communications, where "the more power, the more responsibility" is obligated [16]. In terms of that, RT-P2P employs a super-peer based backbone topology, where a subset of peers with powerful capability, called *super-peers*, are selected to coordinate peer aggregation, query routing, object replication and optimal (or heuristic) path selection.

Each super-peer aggregates and administrates a group of peers, similarly to a parent-children tree structure. It is worth mentioning that peers are only connected with super-peers instead of interconnected with each other in RT-P2P. As illustrated by Figure 1, super-peers construct a backbone frame for RT-P2P for intergroup communication and task processing. According to [17], such super-peer-based backbone topology will benefit system scalability and facilitate the real-time communications by higher query routing efficiency and lower traffic load. In addition, since the major communications are administrated by super-peers, the authenticated trustworthy super-peers will enhance the security of RT-P2P.

In RT-P2P, peers may dynamically join or leave the peer group or network. Super-peers are also dynamic; however, in the super-peer selection process, relatively stable peers are preferred. We also assume in the underlying physical networks, nodes may crash, links may fail transiently or permanently, and messages may be lost, all arbitrarily.

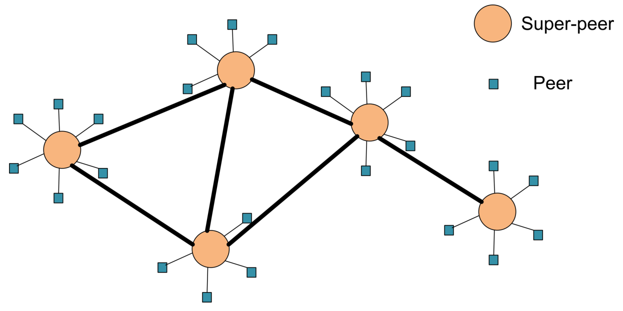


Figure 1. RT-P2P

2.4. Peer Registration in RT-P2P

Peer arrival. On registration, a peer will first broadcast its *Request Joining Message* (**RJM**) containing its *peerID* (the unique ID of peers in RT-P2P) and resource requirements (e.g., bandwidth) to the neighboring super-peers within certain proximity. To restrict the diameter of peer group size and ensure the peer locality, RT-P2P employs a time-to-live (TTL) mechanism or some network services like *traceroute* as the proximity metric, depending on the application requirement. Such property of low diameter can guarantee a short length of query path and a low number of message overhead [16].

Once a super-peer receives a **RJM** message, it will reply with *Joining Acceptance Message* (**JAM**) if join conditions have been satisfied, e.g., super-peer still has space to accommodate this peer, and security authorization of this peer has been passed. Otherwise, it will keep silent. It is worth mentioning that a peer can register at multiple super-peers. This feature will allow RT-P2P to overcome random attacks and tackle the smooth mobility of peers [16].

Once a registration has been completed, a super-peer will have the registration information about this newly arrived peer in its routing table, such as peerID and service meta-data which describes the services this peer can provide. In addition, backbone information (e.g., neighboring super-peer ID) is also contained in the routing table to enable the communications among super-peers.

As we know, the peers in RT-P2P are highly dynamic and unpredictable. So the routing table of super-peers should be updated periodically.

Peer departure. A peer in RT-P2P may fail or leave without notice. Accordingly, peer departure can be found when super-peer refreshes its routing table or other peer fails contacting this peer. If peer departure is known beforehand, the peer will advertise its departure to the related super-peers to enforce a refresh on their routing tables. Upon refreshing the routing table for peer departure, super-peer will remove the related index information from its routing table, and then release the resource assigned to that peer so new peer can join and use it.

2.5. Super-peer Administration in RT-P2P

Super-peers constitute the backbone of RT-P2P so as to enhance the real-time communication, including query routing and real-time processing. Each super-peer can be linked with a restricted amount (m_{sp}) of other super-peers, where m_{sp} is defined as the dimension of RT-P2P, denoted as D.

Adding new super-peer. With the increase of network size and the evolution of underlying networks, new super-peers should be added to adopt new peers or migrated peers. New super-peer should be elected from the candidate peers with certain qualification requirements, such as security authentication, available bandwidth, computation power and stability. The integration of new super-peer is supervised by one of the existing neighboring super-peers or qualified peers (in case of newly joined subnet which has no super-peers). In the process, index information of the peers, which will be managed by this new super-peer, should be copied or transferred from other super-peers or peers. Backbone index of neighboring super-peers should also be constructed for this new super-peer. At the end, it will advertise its existence to the peers in its group and neighboring super-peers in the backbone. Thus some peers in other group might immigrate to its group for better resources, such as bandwidth and proximate locality. Some of the similar algorithms are detailed in [16], [18].

Removing super-peer. Super-peers are dynamic in RT-P2P so they may leave without notice as well, though such notification is preferred. If a super-peer knows its departure in advance, it will broadcast this information to the peers in its group and require them to transfer to other super-peers. In addition, neighboring super-peers in the backbone will also be notified to update their backbone index. In case a super-peer fails without notice, one proximate neighboring peer will take over the coordination of the above transition.

If a super-peer still works in RT-P2P as an ordinary peer after leaving off the position, it should reconnect to the RT-P2P as a normal peer with the procedures described in Section 2.6.

Replacing super-peer. The replacement of super-peers involves the steps of adding new super-peer to take over the responsibility, transferring the index information in routing table and then removing the previous responsible super-peer. Detailed steps can be referred to the above two subsections similarly.

2.6. Query Routing in RT-P2P

When a peer completes a subtask, it will search for the next peer that can handle the consecutive subtask. In respect of the query, a gossip-based scheme is exploited. First, the required service will be described by metadata, which will be encapsuled in the query message and then sent through the super-peer backbone to identify the peer that can provide this service. During the gossip process, source super-peer will first check its routing table for the required service. If there exists such a peer in its group, it will direct subtask information to that peer for further process. Otherwise, the source super-peer will randomly query its super-peer neighbors (called first-round query). The number of queried super-peers in each round is defined as *fan out number*. If on the first round, the object is not identified in the queried super-peers, they, except those who previously forwarded the same message, will continue to forward this query to their neighbors (second round query). This process will be repeated until the object is found or timing restriction is met. To meet the real-time requirement, we employ a TTL mechanism to assure the timing constraints. During the gossip process, super-peer IDs on the way will be inserted into the message header for future routing.

We should keep in mind each super-peer will only forward the same query once according to the unique *query ID*. After the object is located, destination super-peer will reply the query to the source according to the routing information in the message header. Then source will activate the subtask on the object by sending detailed subtask information, such as runtime parameters. Because multiple routing paths may be found, we need to identify those that can meet the real-time requirements. For this purpose, we propose a real-time communication algorithm, which will be discussed in Section 4.

Reducing message overhead from query. Because excessive message overhead may lead to congestion problem and undermine the real-time properties of RT-P2P, we make efforts to reduce that overhead through the following method.

Super-peers will exchange their neighboring information with each other from time to time. Thus, before a super-peer A starts forwarding the query to one of its neighboring super-peers B, it will check local copy of B's neighboring information. If B has a neighboring super-peer C with lower super-peer ID number than A, A will not forward the query to B. This algorithm only allow one neighbor C to forward message to B. Therefore, it can significantly reduce the redundant queries.

3. Theoretical Analysis of RT-P2P

In this section, we will theoretically analyze RT-P2P from the perspectives of timeliness and message overhead. In light of that, we will analytically compare RT-P2P's theoretical performance with some other systems in later sections.

3.1. Probabilistic timeliness assurance

To understand the performance of RT-P2P practically, we consider the message loss ratio, which is the probability that a query message can fail during the transmission between two neighboring super-peers. It could arise from comprehensive reasons, such as node failure and message loss.

To reduce the complexity of analysis, we assume the message loss ratios in RT-P2P are equal to the same value mlr. Please keep in mind that in practical applications, message loss ratio may vary among different hops and even be non-stationary with the elapse of time. However, in the coordination of RT-P2P, we can define a restriction that two super-peers can be linked as neighbors only if the message loss ratio between them is less than or equal to mlr. In that way, we are pursuing a worst case analysis.

In addition, it is also assumed that each super-peer can accommodate n_p peers, and each peer are registered at m_p super-peers.

Timeliness property of RT-P2P. Let I_r be the number of newly queried super-peers after query round r, U_r be the number of uninformed super-peers at the end of query round r, and Q_r be the average query fan out number of each super-peer at the query round r with $Q_r \leqslant D$, where D is the dimension of RT-P2P. On the initial condition before query starts, $I_0 = 1$, $U_0 = N_{sp} - 1$ where N_{sp} is the number of super-peers in RT-P2P, and $Q_0 = Q$ where Q is the initial query fan out number. Iteratively, we can have

$$U_r = U_{r-1} \times [1 - \frac{Q_r \cdot (1 - mlr)}{N_{sp} - 1}]^{I_{r-1}}; \qquad (1)$$

$$I_{r-1} = U_{r-2} - U_{r-1}, \qquad (2)$$

$$Q_r = Q \times \left(\frac{U_r}{N_{sp} - 1}\right)^{(m_p - 1)/2}, \qquad (3)$$

where $N_{sp} = \lceil N \cdot m_p/(n_p+1) \rceil$ with N as the number of peers in RT-P2P.

Resulting from that, the success probability of subtask k can be carried out as

$$P_k = \begin{cases} 1 - \frac{\binom{N_{sp}-1-m_p}{N_{sp}-1-U_{fr}}}{\binom{N_{sp}-1}{N_{sp}-1-U_{fr}}} & m_p \leqslant U_{fr}; \\ 1 & m_p > U_{fr}, \end{cases} \tag{4}$$

where fr represents the final round restricted by the TTL mechanism.

Then, the success probability of one task is

$$P_T = P_k^{m-1}, \tag{5}$$

where m is the number of distributed subtasks belonging to the task.

If we have n_{task} tasks in the task set, the success probability of the task set can be obtained by

$$P_{TS} = P_T^{n_{task}}. \tag{6}$$

Message overhead of RT-P2P. To complete subtask k, the incurred message overhead is

$$N_k = \sum_{r=1}^{fr} [Q_{r-1} \times I_{r-1}]. \tag{7}$$

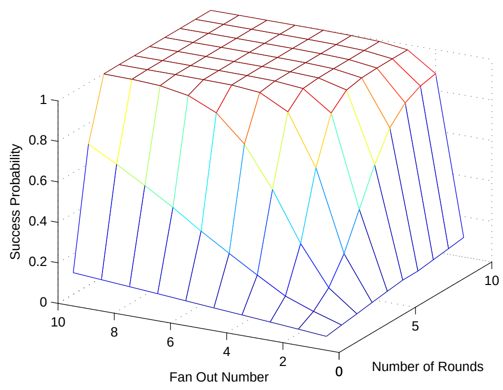
Furthermore, we can work out the message overhead to execute one task as

$$N_T = \begin{cases} N_k \times \sum_{i=0}^{m-2} P_k^i & m > 1; \\ 0 & m \leqslant 1. \end{cases} \tag{8}$$

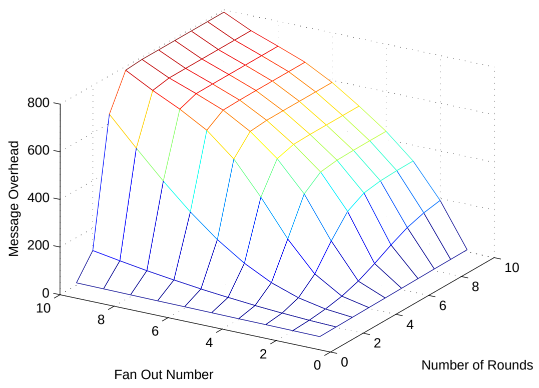
So for one task set, the message overhead can be mathematically expressed as

$$N_{TS} = N_T \times n_{task}. \tag{9}$$

Based on the analytical study about timeliness and message overhead, we can quantitatively illustrate the theoretical performance of RT-P2P in Figure 2(a) and Figure 2(b). For those two plots, we set the number of subtasks as 3. As shown in those two plots, when the number of query rounds increases and fan out number is high, both task success probability and message overhead will increase due to the incremental queries from gossip rounds. We can also utilize this model in future research to target the optimal settings for RT-P2P, where the trade-off between high task success probability and relatively low message overhead will be explored. For the convenience of illustration, we will discuss further theoretical results together with the experimental studies in Section 5.



(a) Task success probability



(b) Message overhead

Figure 2. RT-P2P

4. Improvements on Real-time Communication

Once a query returns a result according to the above discussions in Section 2, there might be a couple of routing paths in the outcome. Our purpose in this section is to discuss how to select a suitable path to the destination with probabilistic end-to-end timing assurance.

4.1. Inaccurate Network Information

In distributed real-time systems, end-to-end delay bound is one of the major criteria when we explore a feasible path. Towards that end, identifying a path capable of meeting the end-to-end delay requirement implies the knowledge of network state, such as link and peer metrics. However, such information of the network state may be inaccurate due to the following features of RT-P2P.

1) RT-P2P is a dynamic system. Network state including underlying network status may inherently change with the time.
2) RT-P2P is a large-scale system. Aggregating underlying network information may bring imprecision.
3) Due to the resource limitation, it is always not possible to frequently update the network information.

To explore the real-time communication for RT-P2P, we employ probabilistic models to deal with such network uncertainties. In that context, we can select the paths which are most likely to meet the end-to-end delay bound. Accordingly, the source peer should associate the stochastic properties of network components, i.e., probability distribution functions (pdf's), to a performance metric [19]. Such pdf's can be obtained not only from the advertised information from the peers, but also from the local construction based on parameters and characteristics of the network components. In our study, we focus on exploiting the probabilistic model. Other aspects, such as detailed discussions on how to keep and access the state information, can be found in [20].

4.2. Probabilistic Model on Paths with End-to-End Delay Requirements

Before the discussion of our probabilistic model, we should note it is more heuristic to first sort and select a few candidate paths by certain network criteria, such as response sequence of query routing, source-to-destination distance (e.g., IP network hops), and available bandwidth. Although such heuristic procedure may be inaccurate and not optimal, it does normally provide a good candidate set for us and saves much computational cost in the selection algorithm from probabilistic model. Otherwise, we will be trapped in a NP-hard problem, which has been proved in [19].

Suppose we have a global end-to-end communication delay requirement, denoted as D_{max}, and the pdf $f_{D_{l,p}}(d_{l,p})$ of the delay $D_{l,p}$ on link l, where l is a link in the candidate path p.

To meet the end-to-end communication delay requirement, we should have

$$\sum_{l,p} D_{l,p} \leqslant D_{max}. \tag{10}$$

Given a path p, we assume $D_{1,p}$, $D_{2,p}$, $\cdots$, $D_{L,p}$, are independent, where L denotes the number of links on path p. Furthermore, we denote the end-to-end communication delay on path p as $D_p = D_{1,p} + D_{2,p} + \cdots + D_{L,p}$. In terms of that, we can carry out the pdf of D_p as

Algorithm 1: Path Selection Algorithm [Path_SELECTION( )]

```
1  Create an empty metric set $\theta$;
2  Generate a set of candidate paths according to certain
   network criteria, such as bandwidth, etc.;
3  for each candidate path $p$ do
4      if $\sum_{l,p}(\alpha_{l,p} + \beta_{l,p}) < D_{max}$ then
5          Select this path;
6          Stop.
7      else if $\sum_{l,p}\alpha_{l,p} > D_{max}$ then
8          Continue to next path;
9      else
10         Calculate (12) for this path;
11         Record the result to $\theta$.
12 Select the path with highest probability from $\theta$ as the
   routing path;
```

$$f_{D_p}(d_p) = f_{D_{1,p}}(d_{1,p}) * f_{D_{2,p}}(d_{2,p}) * \cdots * f_{D_{L,p}}(d_{L,p}). \tag{11}$$

Consequently, we can calculate the probability of (10) as

$$P(D_p \leqslant D_{max}) = \int_{-\infty}^{D_{max}} f_{D_p}(t)dt. \tag{12}$$

It is worth mentioning we normally do a quantization on the time scale so that the above integration should be easily converted to a sum operation for the discrete variables. In addition, the convolution in (11) can be completed in a faster way, such as FFT.

Furthermore, we assume that the delay $D_{l,p}$ on link l of path p is distributed over the range of $[\alpha_{l,p}, \alpha_{l,p} + \beta_{l,p}]$, inclusively.

Theorem 1. *There exists a non-zero probability for a path to satisfy the end-to-end delay requirement iff* $\sum_{l,p} \alpha_{l,p} < D_{max}$.

Proof: If all the links on the path are using their minimal time and still cannot meet the delay requirement D_{max}, then there is no solution for this path to meet such end-to-end delay requirement. Accordingly, for a path to satisfy the delay constraint, we should hold this theorem [19]. □

Theorem 2. *If* $\sum_{l,p}(\alpha_{l,p} + \beta_{l,p}) < D_{max}$, *path p has a timing assurance probability of 1.*

Proof: It is intuitive that if all the links on the path cost their maximal time and still can meet the delay requirement D_{max}, then this path will hold a probability of 1 to meet such end-to-end delay requirement. □

Based on the above discussions, we can generate a path selection algorithm as Algorithm 1. To improve the system reliability, normally we can employ the scheme of multi-path transmission. If that, our algorithm should be revised to select multiple paths rather than one in the final step.

4.3. Special Cases

Identical density functions. In this special scenario, we consider that all the link delays of all the paths are following the same pdf, i.e., $f_{D_{l,p}}(d_{l,p}) \equiv f_Y(y)$ for all l, p. It is easy to see in this situation the optimal path is also the one with minimal hops [19]. In that way, we can simplify the path selection algorithm quite a bit.

However, we should notice that whenever the pdf's are not the same, we cannot claim the optimal path is also the one with minimal hops. Actually, it may be the worst selection, e.g., its $\sum_{l,p} \alpha_{l,p} > D_{max}$.

Normal distribution. Suppose that all the link delays follow normal distribution $N(\mu_{l,p}, \sigma^2_{l,p})$,where $\mu_{l,p}$ and $\sigma^2_{l,p}$ represent the mean and the variance of the link delay for link l in path p, respectively. According to the property of normal distribution, the sum of independent normal distributed variables should also follow a normal distribution. In our case, path delay D_p thus follow a normal distribution. Mathematically, it can be expressed as

$$D_p \sim N(\mu_p, \sigma_p^2), \tag{13}$$

where $\mu_p = \sum_{l,p} \mu_{l,p}, \sigma_p^2 = \sum_{l,p} \sigma^2_{l,p}$.

Consequently, we have

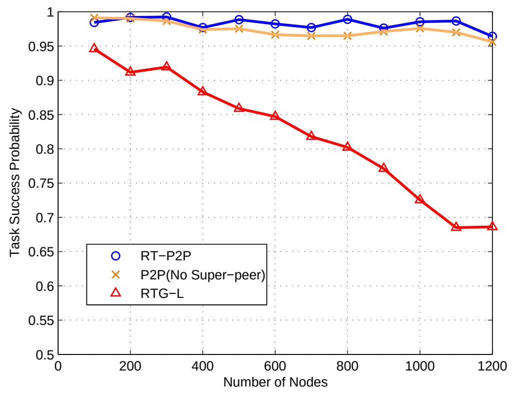
$$P(D_p \leqslant D_{max}) = \Phi(\frac{D_{max} - \mu_p}{\sigma_p}), \tag{14}$$

where $\Phi(\cdot)$ is the cumulative distribution function of standard normal distribution $N(0,1)$. Since all nodes are underloaded in task scheduling, we can simply select the optimal path that minimizes the metric

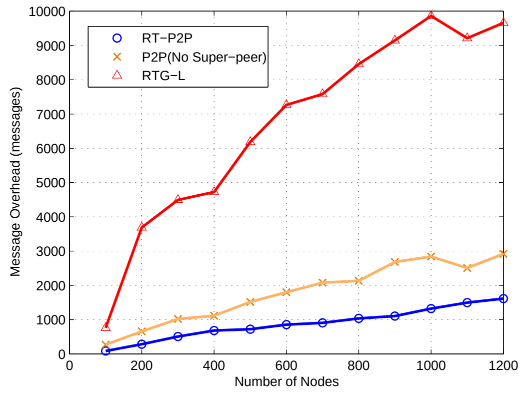
$$\frac{D_{max} - \mu_p}{\sigma_p}. \tag{15}$$

Intuitively, we can see the optimal path should be among those having a small mean and small standard deviation.

In this section, we specify two special cases where we can simplify the algorithm with less computational cost. Studies could be further extended to other special distributions, such as uniform distribution.



(a) Task success probability



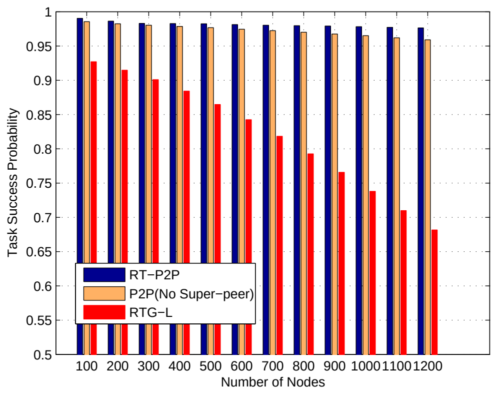
(b) Message overhead

Figure 3. Simulated Results: Real-time Properties $vs.$ Scalability (number of nodes)

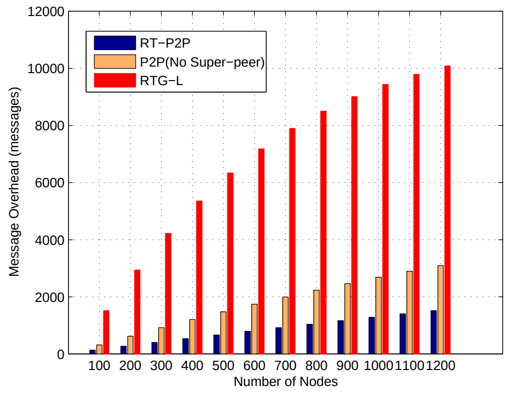
5. Simulation Results

In this section, we present our simulation studies on RT-P2P in terms of scalability, timeliness, and message overhead under different task and network conditions, such as number of subtasks, network size and message loss ratio. In addition, some comparisons between theoretical and simulated results are demonstrated to validate our analytical models for RT-P2P.

Because the query routing scheme of RT-P2P originates from gossip, we refer Han et al.'s algorithm – RTG-L [14], a typical gossip-based real-time algorithm built upon Client-Server model, as the baseline. Also, we validate our design of super-peer backbone by comparing RT-P2P with a modified P2P algorithm without such backbone, simply called P2P. (According to our literature search, till now there is no existing real-time P2P system to make further comparison.)



(a) Task success probability



(b) Message overhead

Figure 4. Theoretical Results: Real-time Properties ***vs.*** **Scalability (number of nodes)**

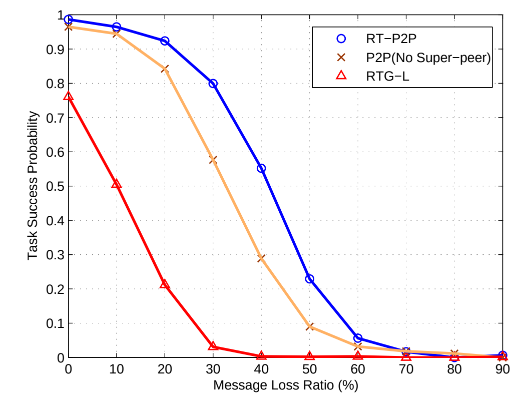
We will compare RT-P2P with RTG-L and P2P (without super-peer backbone) in the perspectives of scalability (number of nodes), reliability (influence from message loss) and task load (number of subtasks). The number of simulations is 200 in total. We consider such settings: the message loss ratio is 0.01, query fan out number is 3, and TTL $= 6$ time units (i.e., 6 query rounds). Each task contains 3 subtasks, where task succeeds **iif** all its 3 subtasks are successfully completed. Specifically, we consider a RT-P2P with the dimension of 3, where each super-peer accommodates 6 peers.

Figure 3(a) illustrates when the network size increases from 100 to 1200 nodes, RT-P2P and P2P can hold a task success probability over 0.95 with few decrescence while RTG-L drops significantly from 0.94 to 0.63. As we know, RT-P2P utilizes a super-peer backbone topology with multiple registration scheme to adapt for the massive scalability and network dynamism. Thus, its query routing efficiency is much higher than RTG-L. Without super-peer backbone, P2P still functions better than RTG-L with a higher task probability. Although the task success probability of P2P (no super-peers) is just slightly worse than RT-P2P, its message overhead almost doubles that of RT-P2P as displayed by Figure 3(b).

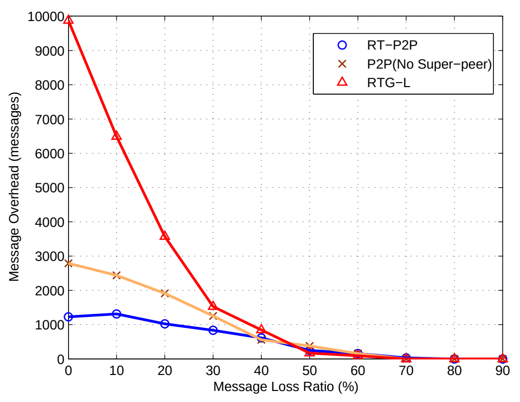
In terms of message overhead, Figure 3(b) further shows RT-P2P generates much less message overhead than RTG-L does. For example, when the number of nodes is 1200, RT-P2P's message overhead is 1652 while RTG-L's is 9663 which is 5.8 times greater than RT-P2P's. In addition, we can notice the message overhead of RT-P2P increases much slower than RTG-L and P2P when network size expands. The above observations not only indicate a better scalability of RT-P2P than RTG-L and P2P, but also approve that super-peer backbone and peer's multiple registration scheme of RT-P2P play the fundamental roles to enhance the task success probability and reduce message overhead.

Corresponding to the simulations shown in Figure 6, Figure 4 illustrates the theoretical results based on the analytical model discussed in previous sections. We can observe our simulated results conform closely to the theoretical models in both task success probability and message overhead, which in turn validates the correctness of our theoretical models. To save the space, we will not demonstrate the corresponding theoretical results for later simulations though they still match well.

Another interesting comparison between RT-P2P and RTG-L is in terms of message loss ratio. We repeat the simulation 200 times. For both algorithms, the number of subtasks is 3, and the network contains 1000 nodes. Other settings are all the same with previous simulations. From Figure 5(a), we can observe that when channel condition becomes worse, the task success probability will be undermined in all three algorithms. Comparatively, when message loss ratio increases, RT-P2P exhibits significantly better task success probability than RTG-L and P2P do. For instance, when $mlr = 30\%$, the success probability of RT-P2P is around 0.81 while RTG-L is unacceptably cut down to 0.03 and P2P is just 0.57 at the time. On the other hand, we may find, from Figure 5(b), the message overheads of all three algorithms commonly decrease when packet loss ratio increases. It is because that the higher transmission failure probability will discontinue more transmissions that should have been in consecutive nodes. As a result, less overhead will be issued. However, RT-P2P's message overhead is orders of magnitude lower than that of RTG-L and

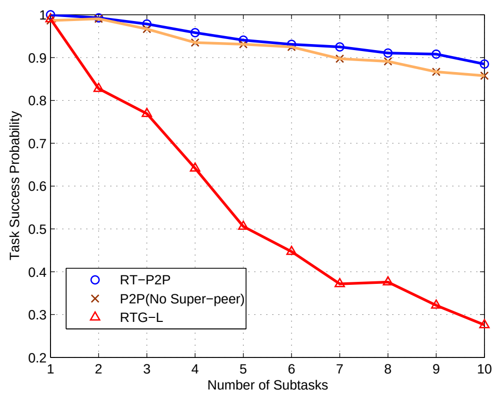


(a) Task success probability

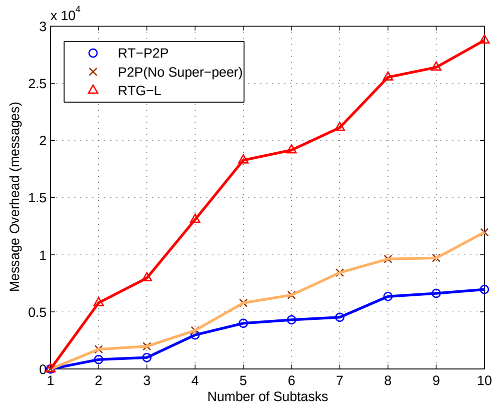


(b) Message overhead

Figure 5. Real-time Properties *vs.* Message Loss Ratio



(a) Task success probability



(b) Message overhead

Figure 6. Real-time Properties *vs.* Task Load

is about half of P2P's overhead, especially when message loss ratio is lower than 10%. When message loss ratio is extremely high, they all converge to 3 because message will only be issued to the 3 queried peers in the first-round gossip.

Furthermore, we look into the influence from task load (i.e. number of subtasks), where simulation is executed with a network of 1000 nodes. Figure 6(a) illustrates when more subtasks are required to be executed, the success probability will decrease for all RT-P2P, P2P and RTG-L, which is because even if one subtask fails the whole task will claim a failure. However, RT-P2P and P2P can sustain a success probability above 85%, while RTG-L is very sensitive to the change in the number of subtasks. For example, when the number of subtasks becomes 6, the success probabilities of RT-P2P and P2P are both close to 0.93 while RTG-L's is unacceptably cut down to 0.46 in the simulation. Also, we should mention RT-P2P's task success probability is slightly higher than that of P2P because of its super-peer backbone. It is worth noticing when there is only one subtask in the task which means current node can do the work without invoking the next node, the task success probability is anyway around 1 in all three algorithms.

Under the same network condition, we now look at the message overhead in Figure 6(b). It is evident all three algorithms will incur more message overhead with the augment of subtask amount. Specifically, the message overhead of RTG-L is drastically influenced by the number of subtasks. For example, when the number of subtasks increases from 1 to 10, over 2.8×10^4 additional packets are sent out by RTG-L. In contrast, RT-P2P only requires 7000 more messages, and P2P requires 12000 more messages. Moreover, we should see when the number of subtasks is 1, the message overhead is zero because there is no communication required to locate next node since local node can complete the task itself.

From the experimental study, we notice RT-P2P exhibits better scalability and real-time properties than RTG-L and P2P in that RT-P2P utilizes the P2P model plus the super-peer backbone and multiple registration scheme while RTG-L employs only server resources (C/S model) and P2P doesn't adopt the super-peer backbone.

6. Conclusions and Future Work

In this paper, we present RT-P2P, a real-time peer-to-peer architecture that allows application-level end-to-end timing requirements to be satisfied in P2P systems. We discussed the protocols, such as peer arrival, peer departure, and super-peer administration, which are oriented towards the real-time requirements. In addition, we also bring forward a real-time path selection algorithm with probabilistic end-to-end timing assurance under run-time system uncertainties.

An analytical model is built for RT-P2P in this paper, which provides us an insight to the fundamental understanding of RT-P2P. Our simulation results reveal a better scalability and timeliness behavior of RT-P2P in comparison with Client-Server model based real-time gossip algorithm. The simulation study also validates the correctness of our theoretical model.

Many research directions can be pursued in future work. For example, as RT-P2P involves large amount of peers in real-time applications, security issues should be investigated to enhance the fault-resilience of RT-P2P. Future study can be also oriented to optimize RT-P2P with proper parameter settings, such as network dimension and query fan out number.

References

[1] Adam Blair, Thomas Brown, Keith M. Chugg, and Mark Johnson. Tactical mobile mesh network system design. In *IEEE MILCOM 2007*, October 2007.

[2] D. Chatterjee. Numerical modeling of conformal phased arrays on tactical systems. In *IEEE MILCOM 2006*, October 2006.

[3] K. G. Srinivasa, A. Bhatt, and S. Dhar et al. Selective querying and other proposals for a less congested gnutella network. In *Distributed Computing and Internet Technology*, pages 203–208. Springer Berlin/Heidelberg, 2007.

[4] Linda Kallstrom, Simone Leggio, and Jukka Manner et al. A framework for seamless service interworking in ad-hoc networks. *Computer Communications*, 29(16):3277–3294, October 2006.

[5] Dejan S. Milojicic, Vana Kalogeraki, and Rajan Lukose. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, 2003.

[6] C. Huang and C. Lei. Bounding peer-to-peer upload traffic in client networks. In *Proc. of Conference on Dependable Systems and Networks*, June 2007.

[7] J. Reuning and P. Jones. Osprey: peer-to-peer enabled content distribution. In *Proc. of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, June 2005.

[8] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proc. of INFOCOM 2006*, April 2006.

[9] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location and routing for lage-scale peer-to-peer system. In *Proc. of International Conference on Distributed Systems Platforms*, 2001.

[10] M. Wiberg. FolkMusic: a mobile peer-to-peer entertainment system. In *Proc. of International Conference on System Sciences*, January 2004.

[11] L. De Cicco, S. Mascolo, and V. Palmisano. An experimental investigation of the congestion control used by Skype VoIP. In *Wired/Wireless Internet Communications*, pages 153–164. Springer Berlin/Heidelberg, 2007.

[12] Napster website, online: *http://www.napster.com.* last visit: April 2008.

[13] Gnutella website, online: *http://www.gnu.org.* last visit: April 2008.

[14] K. Han, B. Ravindran, and E.D. Jensen. RTG-L: Dependably scheduling real-time distributable threads in large-scale, unreliable networks. In *PacRim Symposium on Dependable Computing*, December 2007.

[15] R. Friedman, D. Gavidia, L. Rodrigues, A. C. Viana, and S. Voulgaris. Gossiping on manets: the beauty and the beast. *SIGOPS Oper. Syst. Rev.*, 41(5):67–74, 2007.

[16] W. Nejdl and M. Wolpers et al. Super-peer-based routing strategies for rdf-based peer-to-peer networks. *Web Semantics: Science, Services and Agents on the World Wide Web 1*, pages 72–93, November 2003.

[17] P. Samar, M.R. Pearlman, and Z.j. Haas. Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 12(4):595–608, August 2004.

[18] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP-hypercubes, ontologies and efficient search on P2P networks. In *Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, April 2002.

[19] R.A. Guerin and A. Orda. QoS routing in networks with inaccurate information: theory and algorithms. *IEEE/ACM Transaction on Networking*, 7(3), June 1999.

[20] Y. Bejerano and Y. Breitbart et al. Algorithms for computing QoS paths with restoration. In *Proc. of INFOCOM 2003*, April 2003.