RTQG: Real-Time Quorum-based Gossip Protocol for Unreliable Networks

Bo Zhang, Kai Han, Binoy Ravindran
ECE Dept., Virginia Tech
Blacksburg, VA 24061, USA
{alexzbzb,khan05,binoy}@vt.edu

E. D. Jensen
The MITRE Corporation
Bedford, MA 01730, USA
jensen@mitre.org

Abstract

We consider scheduling real-time tasks in the presence of message loss and Byzantine node failures in unreliable networks. We present scheduling algorithms called RTQG and RTQG-B. The algorithms use quorum-based gossip communication strategies for dynamically and dependably discovering eligible nodes. Compared with its predecessors,our protocol exhibits better performance. RTQG utilizes quorum systems to limit the range of each gossip round. Using the intersection property of quorum systems, RTQG has advantages in message propagation and robustness to Byzantine node failures. Our simulation studies verify our analytical results.

1. Introduction

We consider distributed threads in real-time systems as sequences of tasks. Each task is requested by its current head node, and executed on the next head node. After the successful execution the next head node becomes the current head node. For an individual task, the source node tries to discover a proper destination node in the network to execute the task it requests. The destination node needs to execute the task and send back its acknowledgement in the given deadline. During this process, some communication protocol is utilized to make the task executed successfully. Several algorithms have been presented for this problem. For example, RTG-L [5] provides assurances on thread time constraint satisfaction in large-scale unreliable networks. At its core, RTG-L contains a gossip protocol. Our work builds upon RTG-L. The most important difference is that our work introduces quorum system for message communication. Generally, the source node first gossips message in its closest quorum, and then some elements in this quorum continues gossip the message, until the destination node is discovered. Due to the intersection property of quorum system, the node can have the knowledge of the whole system by accessing any one quorum. Moreover, we design Byzantine quorum system to deal with Byzantine failures in networks, which were not considered in RTG-L.

Given a universe U of elements, a quorum system $\mathcal{Q} = \{Q_1, ..., Q_m\}$ on U is a family of subsets of U such that any two quorums Q_i and Q_j have a non-empty intersection. [1] Quorum systems are widely used in distributed systems for achieving mutual exclusion, consistent data replication, and dissemination of information. In typical quorum-based algorithms, each client accesses the system by accessing all the elements in some quorum Q_i belonging to $\mathcal{Q}$. The intersection property ensures that any Q_i would suffice to operate on behalf of the system.

Since the accesses of one quorum by clients (which are themselves nodes in the network) have to be implemented by messages sent along the network, the performance of quorum-based systems now crucially depends on the delays introduced by these accesses. [4] In fact, we would like the logical quorums $Q_i \in \mathcal{Q}$ to be mapped to closely clustered physical nodes in the network so that we do not incur large delays in trying to reach far-flung parts of the network. We call it Quorum Deployment Problem. [3] In this paper, we design a quorum placement algorithm to map the nodes in network in a quorum system.

We also consider the arbitrary (Byzantine) failures of nodes. We introduce Byzantine quorum systems to deal with this situation. We design communication protocol for this system and compare it with the original one.

The rest of the paper is organized as following: In Section 2, we discuss models and algorithm objectives, Section 3 illustrates our quorum-based gossip protocols. Algorithms are analyzed in Section 4. In Section 5 we report our simulation studies. We conclude the paper and identify future work in Section 6.

2. Models and Algorithm Objectives

2.1. Objectives

We design algorithms to meet the following goals:

1. The quorum placement algorithm, combined with the communication protocol, can schedule task with probabilistic termination-time satisfactions in the presence of message losses and node/link failures.

2. The designed quorum system and communication protocol can deal with Byzantine failures of nodes.

3. We seek to minimize communication delays during the message communication.

4. We try to reduce the message overhead during the scheduling process as much as possible.

3 Algorithms

3.1 Building Quorum System

We want to determine a map $f : U \rightarrow V$ (which we call a placement of the quorum Q on the nodes of G). Under such placement we design a communication protocol for a single task execution in the network.

We describe the protocol by first introducing the definition of delay.

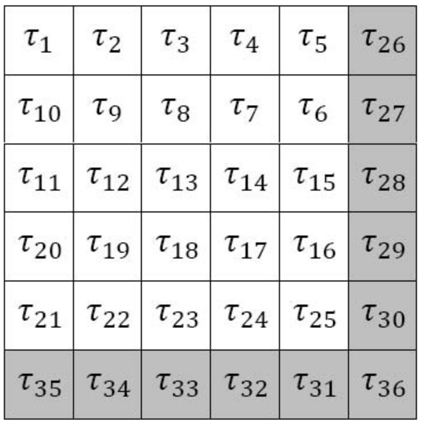
Definition 1. *Delay: We define the delay between nodes as the Round Trip Time (RTT) from one node to the other. Hence, the quorum accessing delay from a node $v = f(u)$ to a quorum Q belonging to $\mathcal{Q}$ is the maximum RTT from v to all elements of the quorum Q. Hence we model the delay as the RTT of v to the farthest-away element of Q:*

$$\delta_f(v, Q) = \max_{u \in Q}[\tau(v, f(u)] \qquad (1)$$

Then we introduce the definition of Grid quorum system. [7]

Definition 2. *Grid quorum system: On a universe U of k^2 elements, the k^2 elements are laid out on a k by k square grid M, and each quorum Q from $\mathcal{Q}$ is formed by taking all the elements from some row and some column of M. Hence each quorum has 2k-1 elements, and there are k^2 quorums in $\mathcal{Q}$.*

We use Algorithm 1 to map $N := k^2$ nodes into a $k \times k$ square grid M. Let $\tau_1 \geq \tau_2 \geq ... \geq \tau_{k^2}$ be RTTs which indicate delays from the source node v_m to these k^2 nodes in decreasing order. We use v_m to denote the source node in G. Thus we have $\tau_{k^2} = 0$.



τ_1	τ_2	τ_3	τ_4	τ_5	τ_{26}
τ_{10}	τ_9	τ_8	τ_7	τ_6	τ_{27}
τ_{11}	τ_{12}	τ_{13}	τ_{14}	τ_{15}	τ_{28}
τ_{20}	τ_{19}	τ_{18}	τ_{17}	τ_{16}	τ_{29}
τ_{21}	τ_{22}	τ_{23}	τ_{24}	τ_{25}	τ_{30}
τ_{35}	τ_{34}	τ_{33}	τ_{32}	τ_{31}	τ_{36}

Figure 1. Grid quorum system:the closest quorum of source node shaded

Algorithm 1: Quorum Placement Algorithm for Quorum-based Gossip Protocol

```
1  for each $i \in [1, k-1]$ do
2      if i is odd then
3          Place $ith$ $k-1$ RTTs on the $ith$ row of $M$, from $M_{(i,1)}$
           to $M_{(i,k-1)}$(left to right);
4      else
5          Place $ith$ $k-1$ RTTs on the $ith$ row of $M$, from
           $M_{(i,k-1)}$ to $M_{(i,1)}$(right to left);
6  for each $i \in [1, k-1]$ do
7      Place $\tau_{((k-1)^2+i)}$ on $M_{(i,k)}$;
8  for each $i \in [1, k-1]$ do
9      Place $\tau_{(k^2-i)}$ on $M_{(k,i)}$;
10 Place $\tau_{k^2}$ on $M_{(k,k)}$;
```

3.2 RTQG: Real Time Quorum-Based Gossip Protocol

In our gossip protocol, we make a slight modification to definitions in [6] to make them compatible with our quorum based gossip. We make following definitions:

Definition 3. *Gossip round r_i: denotes the rth gossip time interval in ith row-sharing quorum (defined later), at the beginning of which nodes send out messages.*

Definition 4. *I_r^i: Denotes the number of newly informed nodes in ith row-sharing quorum during gossip round r_i.*

Definition 5. *U_r^i: Denotes the number of uninformed nodes in ith row-sharing quorum at the end of gossip round r_i.*

Definition 6. *F_r^i: The number of messages a node sends out in ith row-sharing quorum at the beginning of gossip round r.*

Definition 7. M_r^i*: The number of messages issued in* ith *row-sharing quorum during gossip round* r_i.

Algorithm 2: Gossip Protocol

```
On gossiping a message $msg$;
$msg.r++$;
/*randomly selects msg.f targets*/
for each $i \in [1, ..., msg.f]$ do
    SEND ($target_i, msg$);
```

Using the same argument in [6], we show the relationship between M_r^i and F_r^i. Note the number of nodes is reduced to $\sqrt{N}$.

$$M_r^i = F_r^i * I_{r-1}^i = (\sqrt{N} - 1) * \ln(U_{r-1}^i / U_r^i) \qquad (2)$$

Here, F_r^i can be adjusted by application users.

Similarly, Round Message Density in row-sharing quorum i at round r_i (RMD_r^i) is computed as following:

$$RMD_r^i = \frac{M_r^i}{\sqrt{N}} \qquad (3)$$

Randomly selecting gossip targets in one quorum makes messages uniformly distributed in the quorums and finally in the network. Thus, the likelihood of network congestion is reduced.

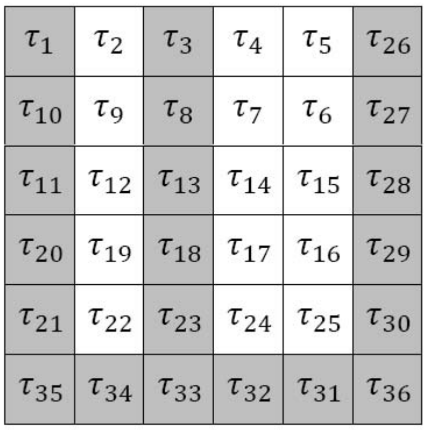
Definition 8. *Closest Quorum: Given a quorum system Q over U and a client i, the closest quorum for this node is defined as the quorum* $Q_c^i \in \mathcal{Q}$ *to which the client has the minimal access delay.*

Definition 9. *Row-sharing Coterie: Given above assumptions and a specific quorum* Q_i*, the row-sharing coterie* $\mathcal{Q}_r^i$ *is composed of quorums which share the same row of* Q_i*, but have different columns. We call quorums in this coterie row-sharing quorums of* Q_i*. So* $|\mathcal{Q}_r^i| = k - 1$*. Specifically, for each row element j in* Q_i*, the corresponding row-sharing quorum is denoted as* Q_r^{ij}*, and* $|Q_r^{ij}| = k - 1$ *; for each column element j' (except the intersect one) in* Q_i*,* $|Q_r^{ij'}| = \emptyset$*.*

Algorithm 3: RTQG on Source Node v_m

```
Build $msg$ ;
GOSSIP($msg$) in $Q_c^m$;
/*the source node gossips message in its closest quorum*/
WAIT(d);
/* wait till a designated deadline*/
if $msg.accept$ == TRUE then
    ABORT(holding section);
```

When the message loss ratio is high, the source node utilizes quorum based gossip protocol (RTQG) to determine



τ_1 τ_2 τ_3 τ_4 τ_5 τ_{26}
τ_{10} τ_9 τ_8 τ_7 τ_6 τ_{27}
τ_{11} τ_{12} τ_{13} τ_{14} τ_{15} τ_{28}
τ_{20} τ_{19} τ_{18} τ_{17} τ_{16} τ_{29}
τ_{21} τ_{22} τ_{23} τ_{24} τ_{25} τ_{30}
τ_{35} τ_{34} τ_{33} τ_{32} τ_{31} τ_{36}

Figure 2. Grid quorum system: the closest quorum and two row-sharing quorums of source node shaded

the destination node. The informed destination node also uses RTQG to send back its acknowledgement. Description of the protocol on source node, destination node and intermediate nodes are shown in Algorithm 3, 4 and 5, respectively.

Algorithm 4: RTQG on Destination Node v_n

```
Upon receiving a message $msg$;
$msg.accept$ = TRUE;
$msg.r$ = 1;
GOSSIP($msg$) in $Q_r^{mn}$;
/*the destination node sends back its acknowledgement by gossiping
in its row-sharing quorum of source node's closest quorum*/
EXECUTE();
```

We use quorum based gossip protocol to discover the destination node. The protocol can be divided into two phases:

1. Closest Quorum Gossip Phase: the source node v_m selects its closest quorum Q_c^m as the gossip range to send message msg. After this step each node in Q_c^m has the message msg.

2. Row-sharing Coterie Gossip Phase: each node i mapped to the row elements in quorum Q_c^m gossips the message msg in its row-sharing quorum Q_r^{mi}.

After sending out a query message to determine the destination node, the source node waits for a reply till a certain deadline d. If it does not receive any reply after this deadline, it will regard that the task cannot be finished, abort the

Algorithm 5: RTQG on Intermediate Node v_j

```
Upon receiving a message $msg$;
if $msg$ is a query message then
    if no reply yet then
        GOSSIP($msg$) in $Q_r^{mj}$;
else
    GOSSIP($msg$) in $Q_r^{mj}$;
```

task section. For intermediate nodes, if a query message has been replied, they will not gossip it any more.

3.3 RTQG for Byzantine Node Failures

In unreliable wireless networks, nodes can have arbitrary (Byzantine) failures. Specifically, if one node is fully controlled by a traitor or an adversary, it can perform destructive behavior to disrupt the system. The main idea of Byzantine Quorum System is to design a quorum system in which every two quorums have sufficient number of intersected elements to guarantee the majority vote when quorums are accessed by clients. For example, if B is the set of arbitrary faulty nodes, and $|B| = f$, we have to design every pair of quorums intersect in at least $2f + 1$ elements, and thus in $f + 1$ correct ones. If a read operation accepts only a value returned by at least f+1 servers, then any accepted value was returned by at least one correct server. More generally, the designed quorum system requirements enable a client to obtain the correct answer from the service despite the Byzantine failure of any fail-prone set.

We use the Grid Byzantine quorum system definition from [8].

Definition 10. *Grid Byzantine quorum system: Suppose that the universe of servers is of size $n = k^2$ for some integer k and that $B = \{B \subseteq U : |B| = f, 3f + 1 \leq \sqrt{N}\}$. Denote the rows and columns of the grid by R_i and C_i, respectively, where $1 \leq i \leq \sqrt{N}$. Then, the quorum system Is a masking quorum system for B.As shown in following:*

$$Q = \{C_j \cup \bigcup_{i \in I} R_i : I, j \subseteq 1, , \sqrt{N}, |I| = 2f + 1\} \tag{4}$$

Hence, under the placement of Algorithm 1, we can form a masking quorum system for arbitrary f node failures. In this system, the source node first sends message to its closest quorum by gossip. Then each row element of this quorum gossips the message to its row-sharing quorum. In our Byzantine quorum systems, there are $2f + 1$ elements in each row-sharing quorum to start first round gossip. We use Algorithm 3 for RTQG on sources node and make modifications to algorithms on destination node and intermediate nodes to deal with at most $2f + 1$ Byzantine nodes.

Algorithm 6: RTQG-B on Destination Node v_n

```
Upon receiving a message $msg$;
READ(query.msg) in $Q_c^n$;
/* Query its closest quorum to authenticate msg*/
WAIT(d);
/*wait until a designated deadline*/
if $query.accept$ == FALSE then
    STOP;
else
    $msg.accept$ = TRUE;
    $msg.r$ = 1;
    GOSSIP ($msg$) in $Q_r^{mn}$ ;
    /* the destination node sends back its acknowledgement by
    gossiping in its row-sharing quorum of source node's closest
    quorum*/
    EXECUTE();
```

In our revised algorithm, after receiving gossiping message from source node's closest quorum, the intermediate nodes first send queries to its closest quorum to ask whether this message is authenticated. Due to the masking property, the node can judge whether this message is authenticated or infected based on the data it read from its closest quorum. Actually at least $f + 1$ servers will return correct values. If the message is authenticated, the node continues gossiping this message; otherwise the message will be discarded. The descriptions of revised RTQG (RTQG-B: RTQG for Byzantine systems) on destination node and intermediate node are shown in Algorithm 6 and 7, respectively.

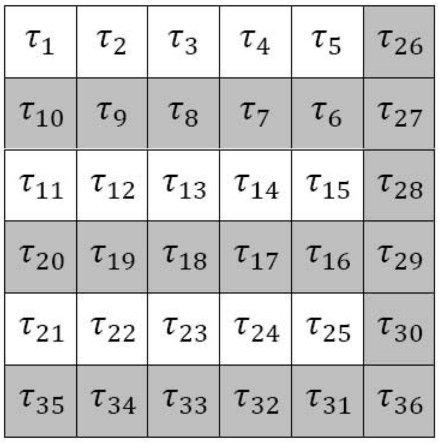

τ_1 τ_2 τ_3 τ_4 τ_5 τ_{26}
τ_{10} τ_9 τ_8 τ_7 τ_6 τ_{27}
τ_{11} τ_{12} τ_{13} τ_{14} τ_{15} τ_{28}
τ_{20} τ_{19} τ_{18} τ_{17} τ_{16} τ_{29}
τ_{21} τ_{22} τ_{23} τ_{24} τ_{25} τ_{30}
τ_{35} τ_{34} τ_{33} τ_{32} τ_{31} τ_{36}

Figure 3. Byzantine Grid quorum system: one quorum shaded

Intuitively, RTQG-B will incur more message overhead in the gossip round due to the cost of message authentication. We will discuss this issue detailed in our algorithm analysis.

Algorithm 7: RTQG-B on Intermediate Node v_j

```
Upon receiving a message $msg$;
READ(query.msg) in $Q_c^j$;
/* Query its closest quorum to authenticate $msg$ */
WAIT(d);
/*wait until a designated deadline*/
if $query.accept$ == FALSE then
    STOP;
else
    if $msg$ is a query message then
        if no reply yet then
            GOSSIP($msg$) in $Q_r^{mj}$;
        else
            GOSSIP($msg$) in $Q_r^{mj}$;
```

4 Algorithm Analysis

4.1 Delay Analysis

To successfully execute the task in a designated deadline, we need to design algorithms in which quorum access delays are minimized. [2] There are two kinds of quorum access delays, and both of them can occur on arbitrary nodes.In the global view, we focus on the average delay of all nodes in network.

4.1.1 Delay from nodes to their closest quorums

Lemma 1. *Given a network $G = (V, E)$, a quorum placement algorithm $f : U \rightarrow V$ and the corresponding quorum system $\mathcal{Q}$, we have $\delta_f(v_i, Q) \leq \delta_f(v_i, v_0) + \delta_f(v_0, Q)$. The equality occurs when v_0 is on the shortest path from v to Q.*

Proof. We can use intersection property of quorum system to prove this Lemma. Note we define the quorum access delay as the largest RTT from a node to the each element in the quorum. Hence if we need this node first visit v_0 before accessing the quorum, the delay will certainly increase unless v_0 is on the shortest path from it to the quorum. $\square$

Theorem 2. *Consider any placement $f : U \rightarrow V$ of a quorum system $\mathcal{Q}$ on the network G. The average access cost from nodes in G to its closest quorum in $\mathcal{Q}$ is bounded by delays from nodes to a given node v_0 and access delay from v_0 to its closest quorum.*

Proof. We derive the following relationship from Lemma 1:

$$\delta_f(v_i, Q_c^i) \leq \delta_f(v_i, v_0) + \delta_f(v_0, Q_c^0) \tag{5}$$

Note the closest quorum for two nodes can be different. Due to the intersection property of quorum system, we can derive the above equation.

Then we take the average:

$$\mathtt{Avg}_{v_i \in V}[\delta(v_i, Q_c^i)] \leq \mathtt{Avg}_{v_i \in V}[\delta_f(v_i, v_0) + \delta_f(v_0, Q_c^0)] \tag{6}$$

Then we have:

$$\mathtt{Avg}_{v_i \in V}[\delta(v_i, Q_c^i)] \leq \sum_{i=1}^{n}(\tau_i/n) + \delta_f(v_0, Q_c^0) \tag{7}$$

$\square$

We use $\tau_1 \geq \tau_2 \geq ... \geq \tau_n$ to denote RTTs from v_0 to other nodes in decreasing order. If one node in the network has the knowledge of its RTTs to all other nodes in the same network and its closest quorum access time, we can calculate the upper bounds of $\mathtt{Avg}_{v_i \in V}[\delta(v_i, Q_c^i)]$.

Corollary 3. *For any Grid quorum placement algorithm, we have $\delta(v_0, Q_c^0) \geq \tau_{(k-1)^2+1}$*

Proof. Note that the quorum size in Grid quorum system is $2k-1$. Thus the optimal situation for v_0 to access its closest quorum is that quorum contains all $2k-1$ closest nodes of v_0. In this way,

$$\min_{Q \in \mathcal{Q}} \delta(v_0, Q) = \tau_{(k-1)^2+1} \tag{8}$$

(Delay from v_0 to the farthest node in its $2k-1$ closest nodes). $\square$

Thus, we have

Corollary 4. *Algorithm 1 is optimal in minimizing $\mathtt{Avg}_{v_i \in V}[\delta(v_i, Q_c^i)]$*

Proof. Note that in Algorithm 1, the closest quorum of v_0 is the quorum containing the rightmost column and lowest row which are $2k-1$ closest nodes of v_0. $\square$

4.1.2 Delay from nodes to their row-sharing quorums

We design the quorum based gossip algorithm, in which the total delay in one quorum is the sum of delays of each gossip round. Thus, we use average access delay from one node to its row-sharing quorum (which defines its gossiping range) to evaluate the delay in one row-sharing quorum.

Use the same method as Theorem 2's Proof, we have

$$\mathtt{Avg}[\delta(v_i, Q_r^{0i} \setminus Q_c^0)] \leq \mathtt{Avg}[\delta(v_i, v_0)] + \tau_{(i-1)^2+1} \tag{9}$$

Theorem 5. *In Algorithm 1, given a node v_0, the access delays from row elements of Q_c^0 to their row-sharing quorums are optimal arranged so that they are as close as possible.*

Proof. From above equations, we discover that the time bounds for each gossip round of row-sharing quorum are different due to the different access delay for each row-sharing gossip. However, if we want to make gossip quorums propagation speeds as close as possible, we need to place nodes in the row-sharing quorum to meet this criterion. If we make closest $k-1$ nodes of v_0 the gossip node, delays from these nodes to its row-sharing quorum are approximate those from v_0 as much as possible.

Now we have $k-1$ gossiping quorums, in which the durations of gossip rounds depends on delays from v_0 to farthest-away nodes of those quorums. Hence, to make these durations as close as possible, we need to place farthest $k-1$ nodes separately in these quorums.

□

4.2 Overhead Analysis

We analyze message overhead for both RTQG and RTQG-B, and then we make comparison between them.

Lemma 6. *The number of messages issued during all gossip rounds in one quorum is* $\Theta(\sqrt{N}\log N)$

Proof. From equation (2), we have

$$\sum_{r=1}^{R} M_r^i = (\sqrt{N}-1) * \sum_{r=1}^{R} \ln(U_{r-1}^i/U_r^i) \tag{10}$$

The number of issued messages during all gossip rounds in one quorum is $\Theta(\sqrt{N}\log N)$. □

Corollary 7. *The number of messages issued during all gossip rounds in RTQG is* $\Theta(N\log N)$

Proof. In RTQG, there are $\sqrt{N}$ gossiping quorums. From Lemma 2 we have

$$\sum_{i=1}^{\sqrt{N}}\sum_{r=1}^{R} M_r^i = \sqrt{N} * \sum_{r=1}^{R} M_r^i = \Theta(N\log N) \tag{11}$$

□

However, in RTQG-B, which is designed for Byzantine quorum systems, more message overhead will be introduced into system. At each step the node gossiping message to its target nodes, it needs to access its closest quorum first to authenticate that message.

Corollary 8. *The number of messages issued during all gossip rounds in RTQG-B is* $\Theta(N^{3/2}\log N)$

Proof. In RTQG-B, the gossiping nodes need to query its closest quorum to decide whether this message is health or infected. Hence, we have

$$\sum_{r=1}^{R} M_r^i = (\sqrt{N}-1) * \sum_{r=1}^{R} \ln(U_{r-1}^i/U_r^i) * \sqrt{N} = \Theta(N\log N) \tag{12}$$

There are $\sqrt{N}$ gossiping quorums. So we have

$$\sum_{i=1}^{\sqrt{N}}\sum_{r=1}^{R} M_r^i = \sqrt{N} * \sum_{r=1}^{R} M_r^i = \Theta(N^{3/2}\log N) \tag{13}$$

□

Hence, in Byzantine quorum systems, more message overhead will be introduced due to the cost of message authentication. In our quorum based gossip algorithm, after the finish of gossiping in source node's closest quorum, the row elements of this informed quorum need to decide number of rounds it gossips in the row-sharing quorum. Hence, we arrange the closest node of source node gossiping to the row-sharing quorum which has the largest gossip round delay. And under this arrangement, the closest $\sqrt{N}-1$ nodes of source node are arranged corresponding to quorums which have farthest-away $\sqrt{N}-1$ nodes of source node. Then we have:

Theorem 9. *If all nodes in the system are underloaded, the probability for a task* d *to successfully complete its execution* Ps_d *is determined by giving the expected number of informed nodes during gossip periods.*

Proof. Let p_1^i and p_2^i be the number of rounds needed for a row element i to execute task d and receive a decision. Then we have:

$$p_{s_i} = \frac{I_{p_1^i}^T * I_{p_2^i}^T}{N} \tag{14}$$

Where $p_1^i \geq 0$ and $p_2^i \geq 0$, and $I_{p_1^i}^T$ and $I_{p_2^i}^T$ are total number of nodes during the current and next gossiping period, respectively.

Since there are $\sqrt{N}$ gossiping quorums, hence

$$Ps_d = \prod_{i=1}^{\sqrt{N}} p_{s_i} \tag{15}$$

□

5 Experimental Studies

In this section, we present simulation studies to evaluate RTQG and RTQG-B's performance under networks with or without Byzantine node failures.

We evaluate the overall performance of the RTQG and RTQG-B algorithm in a 900-node unreliable system. Thus, a 30×30 Grid quorum system is built for quorum based gossip protocol. We use RTG-L as our baseline, which we introduced in this paper before. Tasks in the simulation environment are invoked and executed between arbitrary two nodes in a 900-node system. During the execution, if a source node cannot timely receive a reply from the destination node, it will abort the task section and announce the failure for the task. Thus, a task is successfully finished if its source node finds its destination node on time. The designated time interval for a remote invocation is represented by the number of rounds. Success Ratio (SR) is the probability of task successfully executed over the total number of rounds.

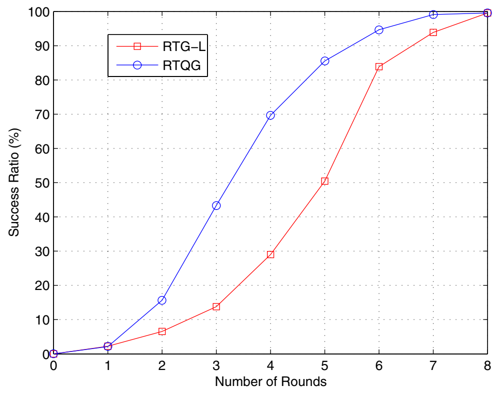


Figure 4. Condition of Message Losses

Figure 4 depicts the SR of RTG-L and RTQG under message losses. We set the message loss ratio as 30%. From this figure we observe that RTQG performs better than RTG-L. It is mainly because in RTQG, the gossiping area is restricted into one quorum. Thus, the gossip protocol will visit all $2\sqrt{N}-1$ nodes in one quorum in fewer rounds, and elements visited will start gossip process in its row-sharing quorum immediately. Because each gossip quorum is assigned to cover the network with most intersect with each other, the number of uninformed nodes in each quorum is minimized to $\sqrt{N}$. The probability of the informed nodes being visited repeatedly was also reduced. Thus, although RTG-L will achieve the same success ratio as RTQG finally, RTQG exhibits better performance when executed gossip rounds are not sufficient to cover all nodes in network.

Figure 5 depicts the SR of RTG-L and RTQG under node

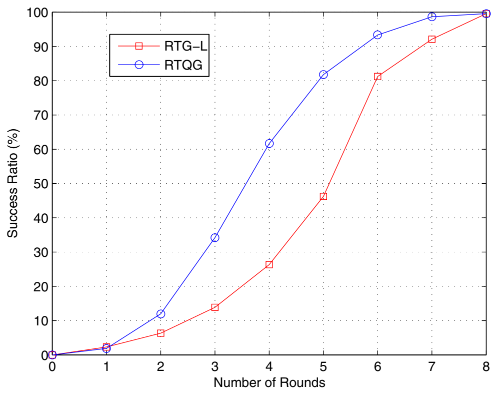


Figure 5. Condition of Node Failures

failures. Note here node failures mean benign failures, e.g. fail nodes do not exhibit Byzantine failures. We set the node failure ratio as 30%. From the figure we discover that both gossip protocols are robust to benign node failures. Again, RTQG performs better than RTG-L because of the same reasons in Figure 4. However, the difference between two protocols is smaller than Figure 1 shows. This observation suggests that under same node failure ratio, the performance of RTQG was more affected than that of RTG-L.

The main advantage of quorum based protocol is its performance under Byzantine node failures circumstance. We also do experiments to validate the theoretical result. In these experiments we compare RTQG-B with RTG-L. And we let one node exhibit Byzantine failure contacted in first round and 2nd round, respectively. After the node receives a message from a health node, it sends out a malicious message instead. If the health node receives the malicious message before it sending a true message, it will consider it as a true message and gossip this message to other nodes. We call this process "infect" and the corresponding nodes "infected nodes". Obviously, the earlier the Byzantine node is communicated, the more health nodes are infected.

Figure 6 and Figure 7 depict the SR of RTQG-B and RTG-L under Byzantine node failures, when the Byzantine node is contacted in first round and second round, respectively. In Figure 6, we observe that RTQG-B performs dramatically better than RTG-L. However, in figure 7 the performance discrepancy is much smaller. We also find even RTQG-B cannot guarantee the same performance as RTQG in network without Byzantine failures. This is because in RTQG-B, the system is immunized to Byzantine failures based on the successful message deployment in the first quorum (the closest quorum of source node). If the Byzantine nodes are contacted in early rounds, there are several nodes in the closest quorum of source node infected. Hence, if these nodes are assigned as the gossip nodes to its

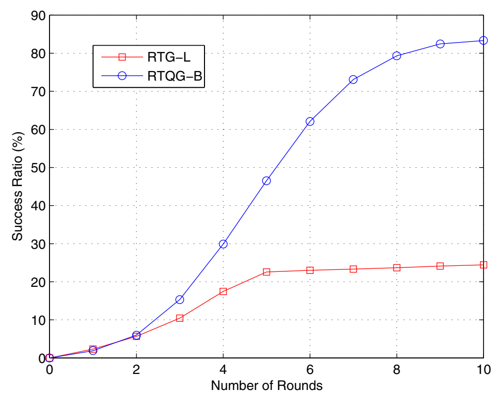


Figure 6. Condition of Byzantine Node Failures: one Byzantine node in 1st Round

row-sharing quorum, as it will not send true message, and elements won't get message from other elements, which make several quorums will not be informed. Under this scenario, the probability of task successfully executed is reduced. Fortunately, due to the limited size of one quorum, the number of infected nodes won't be large. Moreover, RTQG-B can deal with any Byzantine failures occurred in the row-sharing quorums by just letting informed nodes access its closest quorum to validate the authenticity of the message.

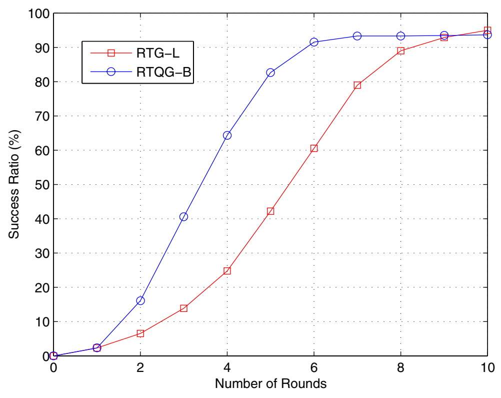


Figure 7. Condition of Byzantine Node Failures: one Byzantine node in 2nd Round

6 Conclusions and Future Work

In this paper, we first design a quorum placement algorithm to map nodes of networks to elements in quorum systems. Based on this placement algorithm, we propose a quorum based gossip protocol called RTQG for scheduling a single task in unreliable networks. Using the intersection property of quorum systems, we design RTQG-B algorithm to deal with Byzantine node failures in networks. Compared with RTG-L algorithm, our algorithm indicates improvements in success ratio for a single task execution. Moreover, RTQG-B algorithm can improve the performance dramatically under Byzantine node failures. Our simulation studies validate the algorithm's effectiveness.

Future work can focus on extending our work for scheduling distributable threads. The quorum placement algorithm needs to be executed only once before the communication process starting. For consecutive tasks in distributable threads, it will be more challenging to design a protocol to generate quorum system dynamically without introducing much more message overheads. Our work can be also extended to more flexible network environments, such as partitioned networks or wireless ad hoc networks. Utilizing quorum systems method in these networks is interesting and challenging.

References

[1] Y. Amir and A. Wool. Optimal availability quorum systems: theory and practice. *Inform. Process. Lett.*, 65(5):223–228, 1998.

[2] A. Fu. Delay-optimal quorum consensus for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(1):59–69.

[3] S. Gilbert and G. Malewicz. The quorum deployment problem. In *Proceedings of 8th International Conference on Principles of Distributed Systems (OPODIS)*, 2004.

[4] A. Gupta, B. M. Maggs, F. Oprea, and M. K. Reiter. Quorum placement in networks to minimize access delays. In *Proceedings of 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 87–96.

[5] K. Han, B. Ravindran, and E.D.Jensen. Rtg-l: Dependably scheduling real-time distributable threads in large-scale, unreliable networks. In *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2007.

[6] K. Han, B. Ravindran, and E.D.Jensen. Rtmg: Scheduling real-time dstributable threads in large-scale, unreliable networks with low message overhead. In *Proceeding of the 13th International Conference on Parallel and Distributed Systems (ICPADS)*, 2007.

[7] A. Kumar, M. Rabinovich, and R.K.Sinha. A performance study of general grid structures for replicated data. In *Proceeding of the 13th International Conference on Distributed Computing Systems*, pages 178–185, 1993.

[8] D. Makhi and M. Reiter. Byzantine quorum system. *Distributed Computing*, 11(4):203–213, Octobers 1998.