Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication

Vincent Wing-Hei Luk*, Albert Kai-Sun Wong, Chin-Tau Lea and Robin Wentao Ouyang

Abstract

Real-time group communication is an indispensable part of many interactive multimedia applications over the internet. In scenarios that involve large group sizes, sporadic sources, high user churns, and random network failures, gossip-based protocols can potentially provide advantages over structure-based group communication algorithms in ease of deployment, scalability, and resiliency against churns and failures. In this paper, we propose a novel protocol called Redundancy Reduced Gossip for real-time N-to-N group communication. We show that our proposed protocol can achieve a considerably lower traffic load than conventional push-based gossip protocols and conventional push-pull gossip protocols for the same probability of successful delivery, with higher performance gains in networks with smaller delays. We derive a mathematical model for estimating the frame non-delivery probability and the traffic load from overhead, and demonstrate the general correctness of the model by simulation. We implement a functioning prototype conferencing system using the proposed protocol, completed with functions including NTP synchronization, dynamic group size estimation, redundancy suppression, and other features needed for proper operation. We perform experiments over the campus network and PlanetLab, and the prototype system demonstrates the ability of our protocol to maintain robust performance in real-world network environments.

Keyword: Terms real-time distributed, Network protocols, Network communication, Multicast, Simulation, Verification, Reliability

I. Introduction

Real-time group communication is a fundamental part of many emerging interactive internet multimedia applications such as group chats [1,2], voice and video conferencing [1,3,4], telepresence [5,6], web-based classrooms [4,7], virtual reality [7], distributed collaborative environments [1,7-9], online multiplayer games [10,11], social networking applications [12,13] and social games [14,15], etc. Real-time group communication over the Internet presents the following requirements that must be considered:

1. The delay requirement of real-time communication is stringent - generally assumed to be comparable to what is required for conversational voice. The one-way delay should be kept below 400 msec [16]. Protocols for streaming are typically not designed with this stringent delay requirement in mind.
2. Communication among the group members is N-to-N in that a random number of active sources may generate voice, video, and control data information to be distributed to all other members at the same time. Protocols that consider individual sources in isolation may not be optimal in such a scenario.
3. The peers are sporadic meaning that each peer may switch between active and idle state rapidly.
4. There is a high degree of user churn meaning that users may join and leave the group dynamically at will.

There are three conventional approaches for real-time group communication. The first approach is network layer multicast [17], which means the use of IP multicast. The second approach is to use a centralized server (Figure 1a) for forwarding and mixing of multimedia streams. The third approach is to construct a fully

* Correspondence: vincentl@ece.ust.hk
Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, Hong Kong

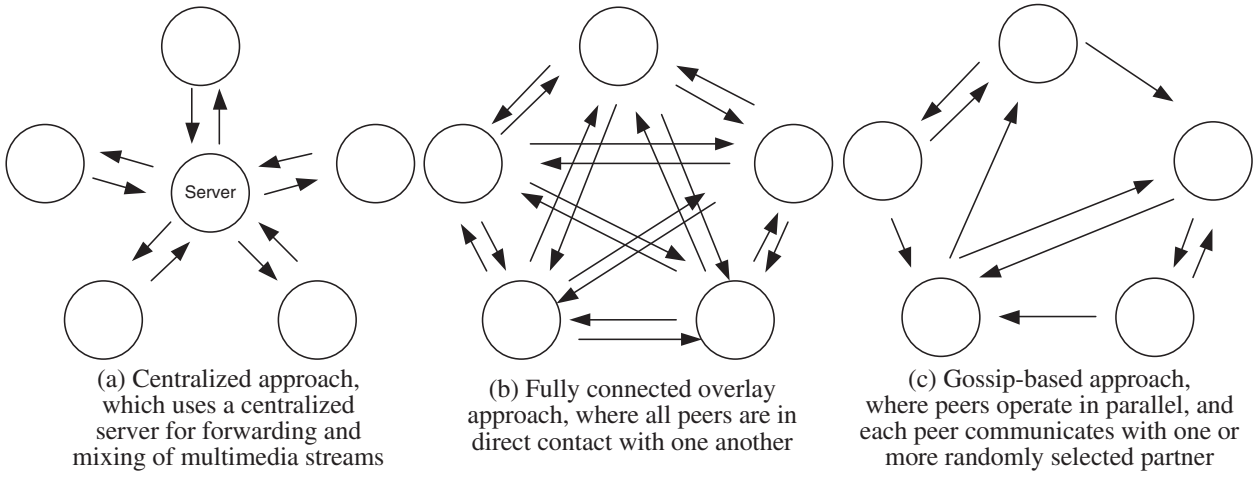


Figure 1 Illustration of three types of group communication topologies: (a) Centralized approach; (b) Fully connected overlay; and (c) Gossip-based approach.

connected peer-to-peer overlay network (Figure 1b) [18] that has all peers in direct contact with one another. The first approach enables server-free group communication, but currently IP multicast is not widely deployed, rendering this approach impractical over the general internet. The second approach requires a powerful central node with sufficient bandwidth, and faces problems of scalability and single point of failure. Skype, which can be considered a centralized-server approach [19,20] where the "centralized server" is also a peer node that is promoted by a peer election mechanism, for example, is not designed to support a large number of users or accommodate random user churns. The third approach requires that all users must have sufficient uplink bandwidth, which scales with the number of users, N, to broadcast their streams to other users. It also faces a serious problem with user churns because any potential source must quickly learn of all peers present.

In recent years, two leading approaches for supporting scalable real-time N-to-N communications have emerged: structure-based [21-42] and gossip-based [43,44].

Structure-based approaches require participating nodes to form a certain deterministic structure, often a tree constructed as a solution to a delay-constrained minimum Steiner tree problem by heuristics [25,26,29-34,36]. In such tree-based systems, bandwidth usage is very efficient as no duplicated messages are sent. The total bandwidth consumption can be further reduced by incorporating the mixing of audio streams within the structure [22-24,35], or by combining IP multicast in LAN [29]. In N-to-N group communication, multiple peers may generate information concurrently. Therefore, the authors in the papers [29,33,37] have argued that multiple source-specific multicast trees should be constructed instead of just one shared multicast tree. Other optimizations have also been proposed, such as resources sharing among trees of different sessions [39], and the 2-hop delayed-bounded tree [40,41]. Examples of structure-based approaches that are not tree-based include chained-based overlay using layered coding [42] and snow-ball chunk [38].

Previous studies have shown that if the user churn is low so that the structure is stable, and if the network loss-rate is also low, then structure-based systems can perform very well. In the presence of user churn and network degradations, however, structure-based systems may become unreliable because the overhead for tree maintenance and message recovery may increase with a snowball effect, as pointed out in the papers [45-47]. Based on experience learned from the evolution of live streaming protocols, Zhang et al. [48] have also concluded that structure-based multicast protocols are impractical on the Internet because of user churn and network degradation dynamics. Note that churn-coping strategies for structured approaches were discussed in the papers [21,37]. But due to the fundamental limitation of structured approaches, the tolerated churn rate is very low. For example, a churn rate of 4/minute with a group size of 50–200 requires 2 seconds of recovery time [37]. Chu et al. have also acknowledged the poor transient performance in larger group sizes in their work [21]. A scheme using multiple distribution trees with Multiple Description Coding (MDC) is proposed as a churn coping measure in the paper [49], but this scheme can only be used for traffic types where MDC is applicable, i.e., video. MDC is not applicable to gaming control data and it is questionable whether it is applicable for voice.

Gossip-based protocols have been considered by many researchers to be reliable in a probabilistic sense as their randomized nature helps to "route around" peer churn

and network degradation [50]. Gossip-based protocols have first been examined for information dissemination in what is known as randomized rumor spreading [51] or epidemic algorithm [52]. In a gossip-based protocol, each cycle of information spreading consists of multiple phases of gossip and in each phase, peers operate in parallel and each peer communicates with one or more randomly selected partners (Figure 1c). In synchronous gossip [51], a phase is launched simultaneously by all peers, and one phase is completed before the start of the next phase. Synchronous gossip assumes that the period of a phase is larger than the one-way delay between any pair of nodes, a condition that is unrealistic for real time communication. In asynchronous gossip [53], peers do not operate in synchronous phases but gossip asynchronously in response to messages received. In either synchronous or asynchronous gossip, the number of phases or the number of times a message is relayed in a cycle must be limited to a very small number independent of the population size for real-time communications because of the stringent delay requirement. This real-time requirement leads also to the use of push [54] rather than pull to reduce the amount of time needed for each phase. For example, Verma et al. [43] have proposed the use of an adaptive fanout to control an asynchronous infection pattern over a limited number of phases in a push manner, and Georgiou et al. [44] have derived the probability for successful rumor spreading in relation to the number of gossip targets under a given number of phases. To the best of our knowledge, all of the existing asynchronous gossip schemes for real time communication use one push or push-pull operation in one phase, and each gossip phase is independent of other phases. These push protocols usually produce a large number of duplicated messages and thus have a low bandwidth utilization efficiency.

In this paper, we propose a new asynchronous way of gossiping with limited delay. In our scheme, a peer establishes connectivity with multiple peers and uses a limited number of push-pull operations in each information spreading cycle. This repeated push-pulls between two peers during each cycle (details in Sec. III-A) results in a much smaller number of duplicated messages compared to conventional push-based gossip protocols and conventional push-pull gossip protocols for real-time applications [43,44]. Hence, we name our protocol Redundancy Reduced Gossip (RRG).

It is worth noting that some gossip protocols proposed for ad hoc networks also use fixed connectivity [55-57]. However, their connectivity is confined to near-neighbor links. In our scheme, the connectivity is randomly established among all participants. Melamed et al. in [47] also proposed the use of a gossip push to fixed downstream neighbors, but none of the related works [47,55-57] uses multiple push-pulls in each cycle.

Results that are presented in this paper include:

1. A novel protocol, called Redundancy Reduced Gossip, for real-time N-to-N dynamic group communication is proposed. The protocol allows the distribution of information from an arbitrary number of random sources within a group, with low latency, minimal membership maintenance, and without assumption on the underlying network condition. The proposed protocol can achieve a given successful delivery probability with a considerably lower traffic load than conventional push gossip protocols and conventional push-pull gossip protocols for real time.
2. A mathematical model is developed and presented for analyzing the frame non-delivery probability and overhead of RRG. The model provides useful insights into the design of our protocol. It can also be used to evaluate the performance of other related protocols.
3. A Linux-based prototype system running the protocol is implemented and tested. Some details and challenges of the implementation are described. Experiment results of the system operating over a LAN as well as over the PlanetLab [58] are collected and analyzed. The prototype system demonstrates the ability of our protocol to maintain robust performance in real-world network environments.

The rest of the paper is organized as follows. Section II presents an overview of related works. Section III describes the RRG protocol. Section IV presents the performance evaluation results from a mathematical model and from the simulator. Section V presents the prototype design, challenges and network experiment results. Section VI concludes the paper.

II. Related works

Using gossip for real-time task execution systems has been proposed in the papers [59,60], but the research focuses and methods in these works are different from ours. Huang et al. [59] have proposed a gossip-based super-node architecture for query and routing in 1-to-1 information dissemination. Han et al. [60] have adopted the adaptive fanout gossip model proposed by [43] for peer discovery and applied the model to a real-time distributable thread scheduling problem.

Push-pull gossip has been studied in the papers [51,61,62]. For example, in the paper [61], one push-pull is used in one phase for the computation of aggregate information. Karp et al. [51] and Khambatti et al. [62] have proposed the use of push- followed by pull-gossip

in two separate stages. They try to combine the expediency of push-gossip with the lower redundancy of pull-gossip. However, these two phase solutions [51,62] are not applicable to real-time communications.

Three-phase pull or lazy push gossip [63] is studied in the streaming papers [48,64-66]. It is important to note that streaming applications have a less stringent delay requirement (buffer built-out delays of 10 – 30 seconds are quoted in these papers). Each execution of the three-phase cycle of advertise-request-delivery is targeted to deliver information to only a single layer of peers. In RRG, each execution of the greeting-response-closure cycle is targeted to deliver information from all sources to all peers.

Using gossip to establish a random graph for information dissemination has been proposed in the papers [28,67,68]. Liang et al. have proposed the use of an on-demand tree for short-lived interactions [67]. However, the proposed scheme does not maintain the spanning tree for a prolonged period of time; hence, no repair mechanism to cope with failures is possible. Chunkyspread [28] uses a simple controlled flooding mechanism over a random graph maintained by Swaplinks [69] for trees construction. It also uses multiple trees to react quickly to membership changes. However, the tree heights are not bounded in the protocol. In contrast, the information dissemination of RRG is strictly bounded to around 3 hops to support the real-time communication delay constraint of 400 ms [16]. Carvalho et al. have proposed to probabilistically combine lazy push gossip and pure push gossip to obtain an emergent structure [68]. The use of lazy push gossip, however, hinders its applicability to real-time communications as we discussed above.

Asynchronous gossip has been studied for other purposes as well [70,71]. Boyd et al. have used asynchronous gossip to address the "averaging problem" in sensor networks [70]. Ram et al. have studied asynchronous gossip for summing the component functions in a distributed multi-agent system [71]. These protocols are not targeted for real-time N-to-N group communications.

Deb, Médard and Chour have studied N-to-N gossip with and without network coding in [72]. Their primary contribution is to quantify the gain of network coding in a multiple-source scenario. They assume synchronous gossiping with only one gossip target per peer per phase. Their study is not applicable to real-time group communications.

Several studies [46,47,50,63] have proposed the combining of the gossiping and structure-based approaches. These hybrid approaches combine the advantage of bandwidth efficiency in structure-based approaches with the churn-coping capability of gossiping approaches. Gossip is employed in the recovery of loss packets after the initial delivery by a structure-based approach. Gupta et al. have used gossip in a sub-tree topology to reduce the traffic load [50]. Our RRG scheme can be extended to include these techniques.

Birman et al. [73], Gu et al. [24] and Lao et al. [32] have proposed the combining of the use of infrastructure and peer-to-peer approaches for real-time group communications. The objective of our paper is different from theirs. We focus on a pure peer-to-peer approach without any infrastructure support.

This paper is different from our original Globecom conference version of this paper [74] in three aspects. First, the protocol has been improved by the incorporation of a delayed response strategy. Second, additional performance evaluations are presented which include the traffic load performances under different scenarios and with user churn. Third, a prototype system running the protocol has been implemented and tested over the HKUST campus network and PlanetLab. In this paper, we identify the fact that the performance gain of our protocol is higher in networks with small delays.

III. Proposed N-to-N gossiping protocol

A. Protocol description

Our N-to-N gossiping protocol consists of n nodes, or peers, that operate in cycles. (The terms "peer" and "node" will be used interchangeably in this paper). Each cycle is initiated at fixed intervals and is identified by a global cycle ID. For simplicity, we assume that there is a global synchronization of the cycle ID and frame rate, and that this synchronization is achieved through the use of NTP. The use of a global cycle ID eliminates the need of a peer to manage the sequence numbering of sources individually and the need to transmit sequence numbers of individual chunks in a packet. Other mechanisms to achieve synchronization are possible but we assume that NTP is used so that we can focus on other aspects of our protocol. Each peer in a cycle can generate at most one information frame (e.g. a voice frame) to be distributed to the remaining $n-1$ peers through a multi-phase gossiping mechanism. The key to our protocol is the use of a synchronous global cycle ID and synchronous media generation. By "synchronous media generation" we mean that the packet generation rates are exactly the same for all active nodes. Most N-to-N real-time communication protocols in the literature have either assumed an asynchronous operation or have assumed a synchronous operation without addressing how this synchronicity is achieved. If using asynchronous operation, we would need to transmit and process individual sequence numbers as well as to perform frequency alignment across multiple streams. Also, the bundling of information from different sources into one transmitted packet cannot be done in as straightforward a manner -

in our protocol, we simply need to bundle information frames with the same cycle ID.

To meet the real-time requirement, we limit the number of phases to 3. In other words, in each cycle, each peer will be engaged in a 3-phase gossip with a random set of other peers, regardless of the number of frames to be distributed. Successive cycles can overlap each other in time. For ease of illustration, we first describe our protocol as if the launch of cycles and phases by different peers are synchronized. But in our protocol this is, in fact, not the case and more details will be added later on.

If a node is already in possession of an information frame to be spread in a specific cycle, the node is called an *infected node*.

Phase 1 (the greeting phase)

In this phase, each node randomly selects a small number of nodes and sends a **GREETING** message to each of them. If a node is already infected when it launches its greeting phase, its GREETING message will contain information frames that it has already received. The selected nodes are called the "***children***" of the selecting node, which is called the "***parent***" of its selected children.

During the greeting phase, connectivity is established for the entire network for the specific cycle. If some nodes are left out, then these nodes will surely not be able to receive the transmitted messages in that cycle. The degree of the established connectivity obviously depends on the number of peers that each node will select during the greeting phase. This number is called the ***fanout*** and is determined in our protocol using a dynamic group size estimation mechanism (discussed below). An illustration of our protocol operation in a group with 8 peers is shown in Figure 2a, where only one peer, *peer 1*, is a source in a cycle. The fanout is assumed to be 2 for all peers. Peers that are already infected at the beginning of each phase are colored black, and peers to be infected by the end of the phase are colored grey in the figure. The information of *peer 1* is transmitted to *peer 4* and *peer 8* via the GREETING message. These two nodes, colored in grey, are infected at the end of this phase.

Phase 2 (the response phase)

During this phase, a node will send a **RESPONSE** message to all of its parents. If the child node is already infected at the beginning of this phase, the RESPONSE message contains all its received original frames (from different sources); if un-infected, the RESPONSE message contains no real data. In the example of Figure 2b, *peer 5* and *peer 1* are the parents of *peer 8*. Since *peer 8* is infected by *peer 1* during the greeting phase, *peer 8* will send the information frame from *peer 1* to *peer 5* during the response phase.

Phase 3 (the closure phase)

In this phase, only an infected parent node will send a **CLOSURE** message, containing all of its received original frames (i.e. from different sources) to its children. Un-infected nodes will not send out anything. In the example shown in Figure 2c, *peer 3* is a child of *peer 2* and *peer 8*. Thus both nodes will send a CLOSURE message to *peer 3*. *Peer 6* remains un-connected after all phases.

B. More details

1) Cycle Launches through NTP

In the analysis, we assume that cycles and phases are launched simultaneously. In practice, timing information is acquired through NTP (Network Time Protocol). NTP can only correct the clock of a host to within a few tens of milliseconds [75]. Due to this inherent timing inaccuracy in NTP, cycles will be launched asynchronously among nodes. The interaction between the timing inaccuracy in NTP and the message network delay has

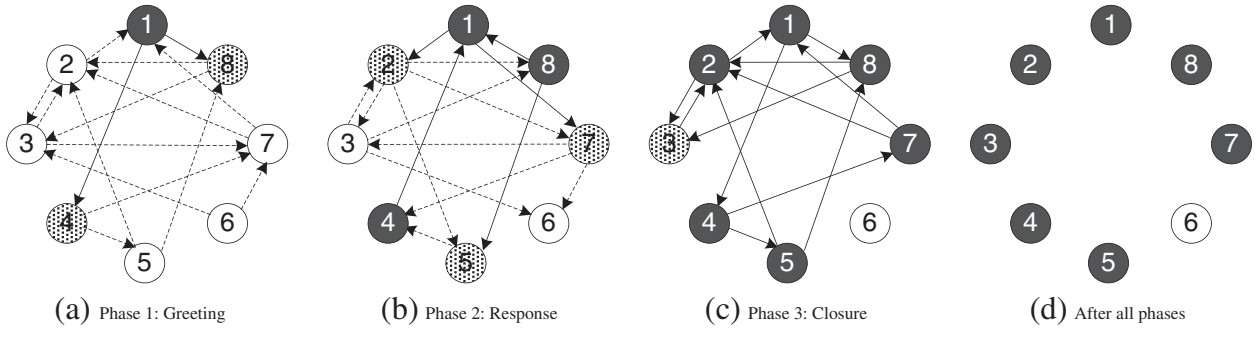


Figure 2 Information frame from peer 1 are diffused via gossip that are 3-phase message exchanges in one cycle: (a) Phase 1: Greeting, (b) Phase 2: Response, (c) Phase 3: Closure, and (d) After all phases; Fanout=2; Peer(s) shaded black is infected at the beginning of the phase; Peer(s) shaded grey is infected at the end of the phase; Peers shaded white remain uninfected at the end of the phase; A solid line refers to a message containing a frame; A dotted line refers to an empty message.

some important implications for the real-world performance of the protocol. Consider *peer 1* and *peer 2*, where *peer 1* is the parent of *peer 2*. Before *peer 2* sends back the RESPONSE message to *Peer 1*, ideally all GREETING messages should have arrived in *peer 2* so that the RESPONSE message will not be empty if any of its parents has sent a real message to *peer 2* during the GREETING phase. But this idealized situation may not be achievable because of network delays and the inaccuracy in NTP. Hence, to increase the probability that a RESPONSE or CLOSURE message will contain useful information, we introduce what is called a delayed response mechanism where a peer will add a delay of d_s msec before sending out the RESPONSE and CLOSURE messages after receiving the GREETING and RESPONSE. This is illustrated in Figure 3. We have used a d_s of 50 msec based on the fact that the NTP inaccuracy and network message delays are typically in the range of a few tens of milliseconds and on the consideration that this delay should not be too large because of the delay requirement for real-time communications.

2) Redundancy suppression and active peer list

Each gossip message that a peer sends out contains an Active Peer List (APL) that lists the source of each information frame that it has already received for that cycle with the actual information frame attached if appropriate. From the APL, the receiving peer extracts the information frames that it needs, and also avoids sending back to the sender information frames that the sender already has - this mechanism is referred to as *redundancy suppression* and its purpose is to reduce the total amount of traffic. Thus, in the APL of a message, some listed entries will have an information frame attached and some entries will not.

APL contains the complete contact information of a peer in 6 bytes - 4 bytes for the IP address and 2 bytes for the port number. The APL is included in every message that a peer transmits. The length of the APL is variable as the number of peers may vary. Therefore, APL is encoded in the Type-Length-Value (TLV) format, with 8-bit type and 16-bit length fields.

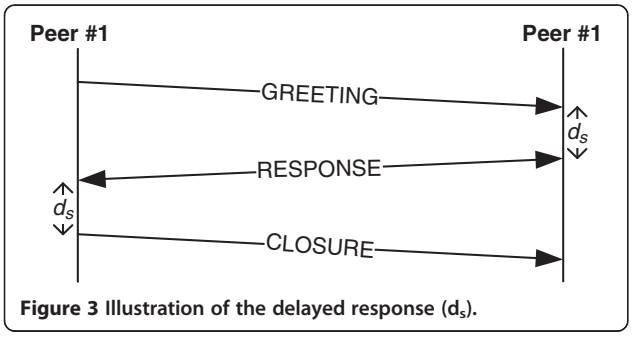


Figure 3 Illustration of the delayed response (d_s).

3) Bootstrapping

Any newly arrived peer is required to contact the bootstrapping point to acquire a list of peer contacts in the group, the current estimated community size (see Sec. III-B5 below) and the current cycle ID. The list of peer contacts does not need to be 100% correct because the new peer can learn the membership information from subsequent gossip, but it will impact the protocol performance. More details will be provided in Sec. IV-D. The newly arrived peer uses the current estimated size to determine its fanout value. Any peer in the group can be the bootstrapping point.

4) Dynamic group membership

A new peer joins the group through the bootstrapping point. Afterwards, its contact information is learned by other peers through the APL contained in the exchanged messages. Peers independently detect and remove a departed peer when that peer does not respond to a GREETING within a timeout.

5) Fanout estimation

In our protocol, each peer will independently decide how many peers to initiate gossip with based on its estimate of the current group size, a target information non-delivery probability, and the estimated non-delivery probability from Eq. 6 we derive in Sec. VI. We adopt and extend the gossip-based size estimation algorithm proposed by M Jelasity et al. [76] to support asynchronous operation. The details of the algorithm are beyond the scope of this paper. Hence, we omit the details of this algorithm in this paper.

IV. Performance evaluation

In this section, we present the analytical model for a key performance measure in the proposed protocol: the frame non-delivery probability as a function of fanout. We also compare, through simulation, the performance of the proposed RRG with that of the conventional push gossip approach in [43,44] and the conventional push-pull gossip protocols in [54,61]. The evaluation metric is the ratio of the non-delivery rate versus the traffic load. The effectiveness of the redundancy suppression is presented. Lastly, the impact of churn is studied.

A. Analytical model for frame non-delivery probability in redundancy reduced gossip

In this section, an analytical model to study the non-delivery rate of information frames in the proposed N-to-N gossiping protocol is developed. The analysis is

based on a perfectly-synchronized Redundancy Reduced Gossip protocol. Although a real implementation, as discussed in Sec. III above, has to be asynchronous, the synchronous assumption allows us to obtain a closed-form formula that provides some useful insights into the design tradeoffs of the protocol.

For source node s, its family tree (Figure 4) consists of the following members: parents, children, co-parents of parents of s, step children, grandchildren, and siblings. "*Parent*" and "*children*" were defined in Sec. III-A. The definitions of the other members of the family tree are given below.

"***Grandchild***": A child's child.
"***Sibling***": Two nodes having the same parent are called *siblings* of each other.
"***Co-parent***": A node and s are called *co-parents* of each other if they share any node as a common child.
"***Step Children***": If node A and B are co-parents of a node, A's children are called the "*step children*" of B and vice versa.

For a node to receive the broadcast message from source s in a cycle, the node must belong to the family tree of s. If not, the node will fail to receive the message. In the following we analyze the non-delivery probability p_l for a given node.

We first define the fanout as b. b is also the number of children of s.

Let's define the following random variables:

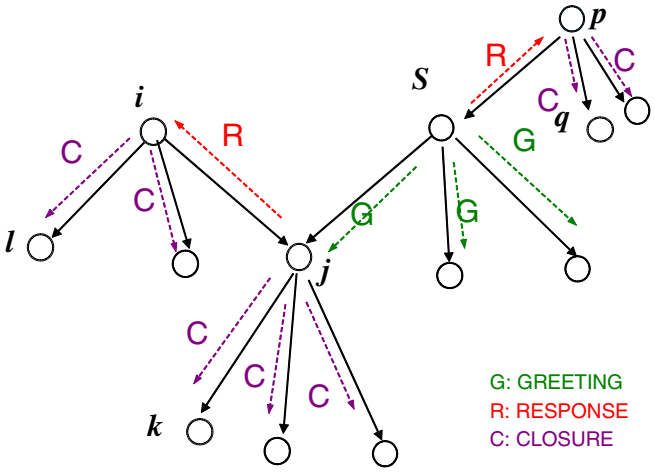
m_p the number of parents of s.
m_g the number of grandchildren of s.
m_{sb} the number of sibling of s.
m_c the number of co-parents with s.
m_s the number of step children of s.
We also define the following probabilities:
p_p the probability that a given node is a parent of s.
p_c the probability that a given node is a co-parent to s.
p_l the probability that the broadcast message of a cycle cannot be delivered to particular node.

First, consider m_p. p_p is the same as the probability that the given node selects s as a child. Hence $p_p=b/n$-1 and the expected number of parents of s is

$$E[m_p] = (n-1)p_p = b \tag{1}$$

Since each parent has b children, E(m_g), the expected total number of grandchildren m_g, has order O(b^2). Likewise, E(m_{sb}), the expected total number of siblings m_{sb}, also has order O(b^2).

Next, consider m_c. First, we observe that p_c is one minus the probability that a given node X selects all b children from the set of n-1-b (s has not selected X as child) or n-b (s has selected X as child) nodes not selected by s out of n-1 nodes:



j: a child of ***s***,
k: a child of ***j*** and thus a grandchild of ***s***,
i: another parent of ***j*** and thus a co-parent with ***s***,
l: a child of ***i*** and thus a step-child of ***s***,
p: a parent of ***s***,
q: a child of ***p*** and thus a sibling of ***s***.

Figure 4 A family tree of the node s as source.

$$p_c = 1 - \left[\left(\frac{b}{n-1} \right) \frac{\binom{n-b}{b}}{\binom{n-1}{b}} + \left(1 - \frac{b}{n-1} \right) \frac{\binom{n-1-b}{b}}{\binom{n-1}{b}} \right] \tag{2}$$

An upper and lower bound can be written for p_c:

$$1 - \left(1 - \frac{b}{n-1} \right)^b < p_c < 1 - \left(1 - \frac{b}{n-b} \right)^b. \tag{3}$$

Since m_c follows a binomial distribution $B(n-1, p_c)$ with expected value $E[m_c]=(n-1)\ p_c$, the bound of Eq. 3 establishes that

$$E[m_c] \approx b^2 \tag{4}$$

for a large n. Note that the ratio of the standard deviation to mean, $(np_c(1-p_c))^{0.5}/(n-1)\ p_c$, is very small and hence the probability mass will center on the mean. Since each co-parent can have b children, the expected total number of step children has order $O(b^3)$. Among the six members—children, grandchildren, parent, co-parent, step children, and sibling—of the family tree, the number of step children is one order higher (in b) than the rest. This means that the size of the family tree of s is determined by the number of step children, which has the order $O(b^3)$. Note that $O(b^3)$ must assume the same order as n (i.e. $O(n)$). If the order is higher, many received frames will be duplicates and wasted; if lower, many nodes will not be part of the family tree and will not be able to receive the information frame from s. The above leads us to formulate b as $b=c\ n^{1/3}$, where c is a constant, and to focus on c in our analysis below.

The size of the family tree, as pointed out above, is determined by the number of step children when n is large and the probability mass for m_c is concentrated near its mean. Therefore, we can approximate the non-delivery probability p_l for a given source node as

$$\begin{aligned} p_l &= \sum_{k=0}^{n-1} P[a\ \ node\ \ is\ \ not\ \ a\ \ step\ \ child | m_c = k] P[m_c = k] \\ &\approx \left(1 - \frac{b}{n-1} \right)^{b^2} \end{aligned} \tag{5}$$

Substitute $b = c\ n^{1/3}$ into the equation above, we obtain

$$p_l \approx \left(1 - \frac{b}{n-1} \right)^{b^2} = \left(1 - \frac{b}{n-1} \right)^{\frac{c^3 n}{b}} \approx e^{-c^3} \tag{6}$$

Figure 5 below plots the analytical and simulation results for the frame non-delivery probability. The simulation is done by repeatedly generating random graphs and collecting statistics of each graph. As shown in the figure below, the difference between (6) and the simulation result narrows as the value of n increases. If the value of c becomes too large, there are a large number of duplicated messages. A large number of peers may

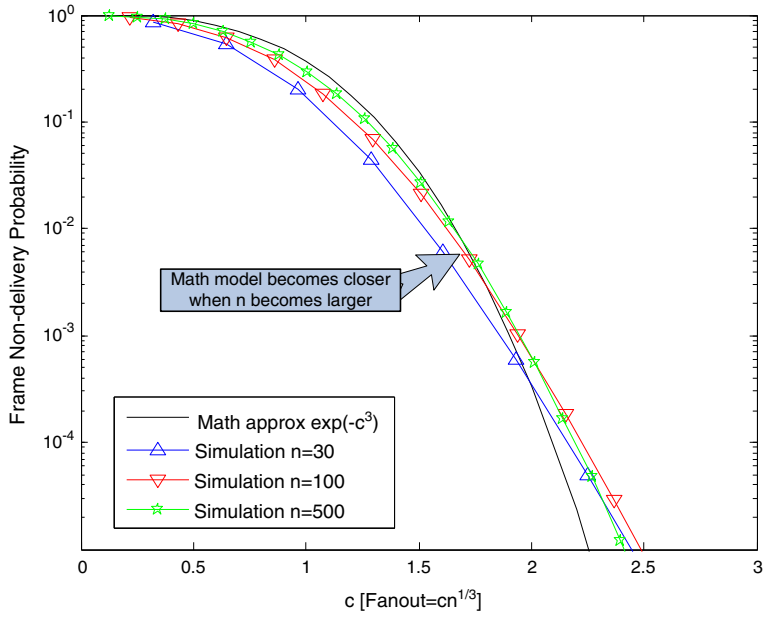


Figure 5 Frame non-delivery probabilities from analytic model and simulation.

also assume multiple roles in relation to s–they can be parents, children, etc., of s at the same time. This renders some of the approximations made in the approximate analysis inaccurate.

B. Frame non-delivery probability versus traffic load

Traffic load is another key performance metric in gossiping protocols. We define traffic load D as the actually measured average number of the same information frame that each peer receives. In other words, D is a measure of the redundancy or the bandwidth usage efficiency of the protocol. Without redundancy suppression, D should be in the order of c^3. With redundancy suppression, D is found to be smaller. By first focusing on the measured D, we are disregarding the protocol overhead, which we will come back to examine later.

In the following, we compare the frame non-delivery probability p_l as a function of traffic load D for the proposed gossip protocol and for the conventional push approach (used by most existing gossiping approaches [43,44]) and the conventional push-pull approach [54,61].

The model for the conventional push approach is as follows: a source node sends the information frame to b randomly selected nodes (phase 1), and each selected node will then push the frame to b other randomly selected nodes (phase 2), etc., with the selection of receiving nodes in each phase done independently. In addition, we include a buffer-map in the conventional approach to reduce redundancy [45-47]–a sending peer will avoid pushing to peers that are already marked in the buffer-map in messages that it has received, and will mark the buffer-map of peers that will receive the information frame in the outgoing message. Note that the buffer-map is more efficient than a list in this case because a list may contain almost all peers in the last phase of gossiping. We ignore the complexity involved in maintaining the mapping of buffer-map to peers in the presence of user churn.

The model for the conventional push-pull approach is as follows: each node randomly selects b nodes and initiates a two way push-pull with each selected node (phase 1 and 2). After that, unlike RRG, each node pushes all the possessed information frames to another b randomly selected nodes (phase 3). Phase 1 and 2 is independent of phase 3. The same buffer-map scheme is included as described in the conventional push approach above [45-47].

While closed-form formulae for the loss rate exist for many of the cases, a meaningful comparison of the loss rates requires that the comparison is done for the same traffic load. However, there is no closed-form formula to estimate the traffic load for protocols under various redundancy suppression schemes. Therefore, an event-driven simulation was developed for the comparison. The information frame is encoded at 8 kbps with a 20 ms sampling interval, i.e. 20 bytes per cycle per peer. We simulate the message propagation, node and link failure, network topology and link delay. As in common practice in simulating peer-to-peer algorithms in existing works [38,46,53], we do not simulate the network-level packet details (such as specific queuing delays) in order to make the simulation scalable. The link delay x_{ij} from node i to node j is assumed to be a random variable, which is determined by a Weibull distribution *Weibull(a,b)* (a is the scale parameter and $b=1.5$ is the shape parameter of a long tail delay). The average number of active peers is less than 3.

Another important simulation issue is related to the clock accuracy. The proposed protocol uses NTP to acquire time information. Due to the inherent timing inaccuracy in NTP, the cycle launch time at every node is not perfectly synchronized. As stated in RFC1305 [75], the timing accuracy of NTP is in the range of a few tens of milliseconds. The cycle launch time of peers is modeled to be uniformly distributed within 50 ms. As discussed earlier, d_s (the delay artificially added before sending out RESPONSE & CLOSURE) is set to 50 ms.

Figure 6 shows that RRG requires less traffic load than the conventional push gossip and the conventional push-pull to achieve the same non-delivery probability. For comparison, we also plot in Figure 6 the curve e^{-D}, which is the probability of zero arrival given that the arrival is Poisson with mean D. In a gossip algorithm that is completely random, the Poisson model could be a reasonable first order model for the arrival of information frames at a particular peer. Figure 6 shows that both RRG and the conventional push gossip perform better than e^{-D} and the conventional pull-push gossip perform slightly worse than e^{-D}. Finally, Figure 6 shows that the performance gain of RRG is higher in networks with smaller delays, such as metro area networks, for the reason illustrated in Figure 7. That is, in situations when the network delay, denoted by $T_{1,2}$, between *peer 1* and *peer 2* is smaller than the offset in *peer 2*'s cycle launch time compared to *peer 1*'s (denoted $O_{1,2}$), *peer 1*'s GREETING message, which contains real data, arrives before *peer 2* sends out its GREETING. As a result, *peer 2*'s GREETING can also contain real data even though *peer 2* is not a source. We call this situation a *bonus relay*.

Traffic load in Figure 6 is measured in terms of average number of copies of an information frame received by each peer. But each message contains protocol headers, and the resulting overheads for the two compared protocols are different. When n=100 and the average number of active peers is less than 3, with c=2, the overhead in RRG is around 20% of total traffic. Most of the overhead is contributed by the APL, where membership information is carried and requires 6 bytes per peer. In the same settings, the overhead of the conventional push

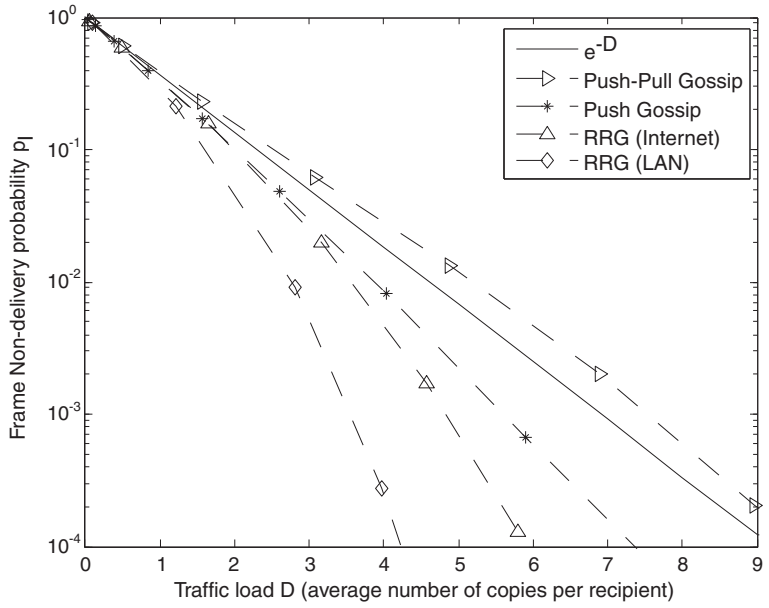


Figure 6 Performance comparison of redundancy reduced gossip (RRG), the conventional push gossip and the conventional push-pull gossip.

gossip and the conventional push-pull gossip are around 40% of total traffic. Most of the overhead is contributed by the buffer-map, which is at least 12 bytes per gossip message.

It is important to note that our protocol generates a smaller number of messages than the fully connected peer-to-peer overlay approach (Figure 1c) in N-to-N communication. The total number of messages in the fully connected overlay approach is *n(n-1)* while that of our protocol is *3bn*. If *n=100* with a target p_l of 10^{-2}, *b* is only *8* from Eq. 6. The number of messages of our protocol is only around 24% that of the fully connected overlay approach.

C. Effectiveness of redundancy suppression

Figure 8 shows the effectiveness of redundancy suppression by APL in RRG. We observe that this mechanism has reduced the traffic load by 35% at the non-delivery probability of 10^{-3}. As the connectivity between peers remains unchanged in each cycle, the average number of copies of an information frame that a peer receives is bounded by the fanout, as validated in our result.

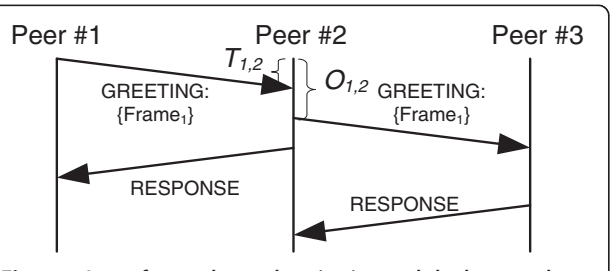


Figure 7 Imperfect cycle synchronization and the bonus relay.

D. Impact of churn of the proposed protocol

We construct a dynamic scenario with sudden changes in group size over a simulation length of 6500 cycles (130 sec). The timeout detection threshold is set to 500 ms. The mean of the link latency is 50 ms. As shown in Figure 9, the changes of group size are in the range of +/−10%, +/−25%, +/−50% of the original group size. We observe that the non-delivery probability converges in less than 50 cycles (1 sec) in the event of a −50% sudden change. This means that it takes about 50 cycles for peers to detect and remove departed peers from their gossiping candidate set. In the event of new peers joining, their memberships are quickly recognized throughout the group through GREETINGS sent by the new peers and by the APL in the gossiping messages.

RRG has one advantage over hybrid protocols, which combine gossiping with a structure-based approach. [46,47,50,63], under user failures. Figure 10 shows a timing diagram comparing RRG and hybrid protocols under a user failure. We assume that the failure detection timeout is *500 ms* in both protocols and the cycle launch interval is *20 ms* in RRG. Note that any data recovery after the timeout *(500 ms)* is too late to be useful for real-time communication. Once RRG detects which peers have failed after the timeout, RRG will remove the failed peers from the gossip target set and recover from the failure immediately. However, hybrid protocols still need some time to heal the structure (e.g. tree) to resume data

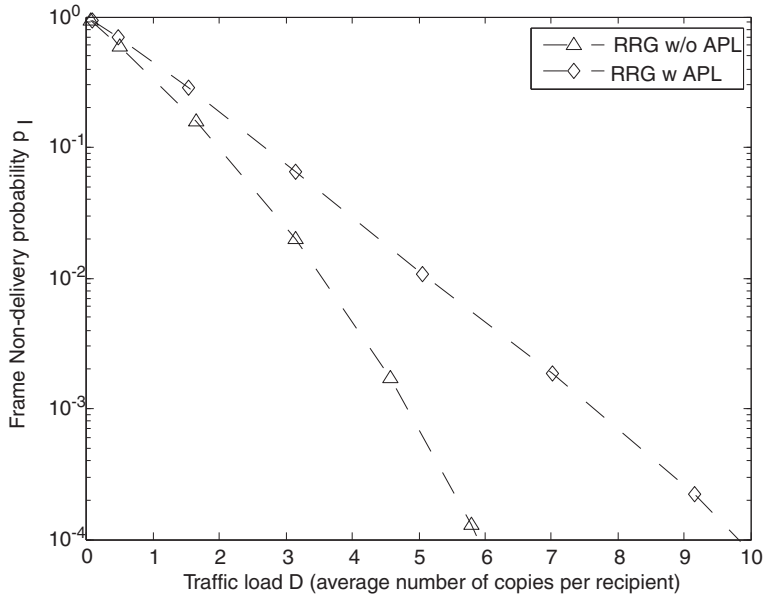


Figure 8 Effectiveness of the redundancy suppression by APL.

dissemination. RRG has a shorter convergence time under user failures. The cost for this shorter convergence time of RRG is the traffic load. RRG may have higher traffic load than hybrid protocols.

V. Prototype implementation and experimentation

The proposed protocol has been implemented in C on the Linux platform. An automated testing and measurement framework is also developed to support sound I/O from audio script to enable large-scale unattended testing for statistical measurement and, in the future, sound quality measurements. The prototype has been deployed on the HKUST campus network and on the PlanetLab testbed. In the following, we discuss several important aspects of the implementation.

A. Design and architecture

The prototype takes a modularized approach and contains nine key modules (Figure 11), as described below:

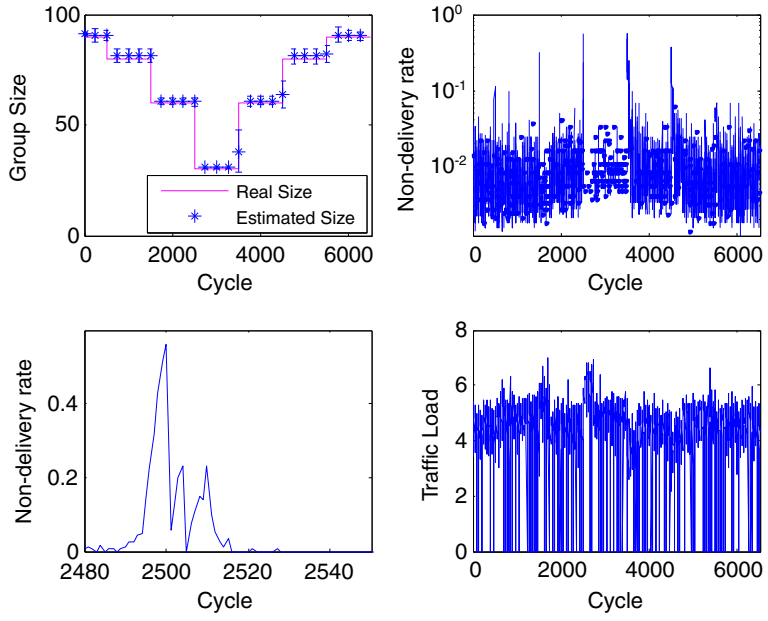


Figure 9 Performance under churn.

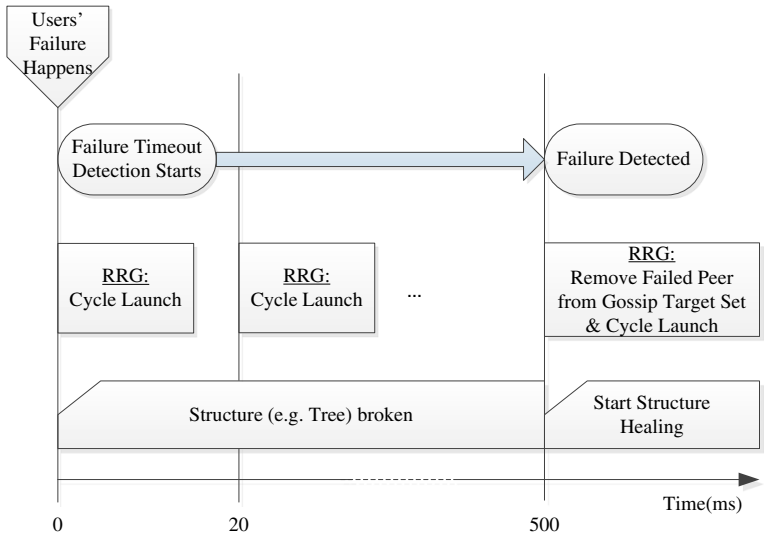


Figure 10 Timing diagram of RRG and hybrid protocols under user failures.

(1) ThreadNetIn: it receives packets from a network interface card (NIC);
(2) ThreadAudioInHW: it reads audio frames from a sound card buffer;
(3) ThreadAudioInFile: it reads audio frames from an audio script file;
(4) MainThread: it parses packets, implements voice activity detection and processes the gossiping logic;
(5) ThreadAudioOutJitterBuffer: it implements a jitter buffer and mixes audio frames from multiple remote parties;
(6) ThreadAudioOutHW: it writes frames to a sound card buffer;
(7) ThreadAudioOutFile: it writes frames to a file and records statistical information for performance evaluation;
(8) ThreadNetOut: it sends packets to the NIC;
(9) ThreadTimerService: it registers and invokes callback events after a specified elapsed time.

ThreadAudioInHW and ThreadAudioInFile are pluggable and interchangeable, and so are ThreadAudio OutHW and ThreadAudioOutFile. The design enables either human or machine based mouth-to-ear voice measurement or large scale remote unattended testing without actual sound I/O. All modules are connected by the producer-consumer design pattern [77]. Each consumer thread has a single work queue shared by possibly multiple producers.

The MainThread generates a GREETING, RESPONSE or CLOSURE according to the gossiping logic. The compiled GREETING, RESPONSE or CLOSURE encapsulated in UDP is sent to ThreadNetOut. Each frame is marked with a cycle ID, and each peer is synchronized using NTP [75].

B. Challenges

Our protocol is implemented on Linux platforms. As Linux is not a real-time operating system, this makes real-time scheduling difficult. To solve the problem, we implement a clock-driven event dispatching module to handle all time-related events, such as periodic sampling of sound cards, periodic cycle launch, de-jittering, playback, scheduling of next wake-up instances, etc. The clock-driven dispatching module invokes events according to a local clock in order to prevent the timing

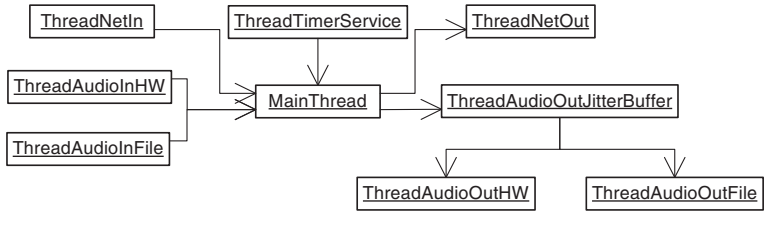


Figure 11 System diagram of the prototype.

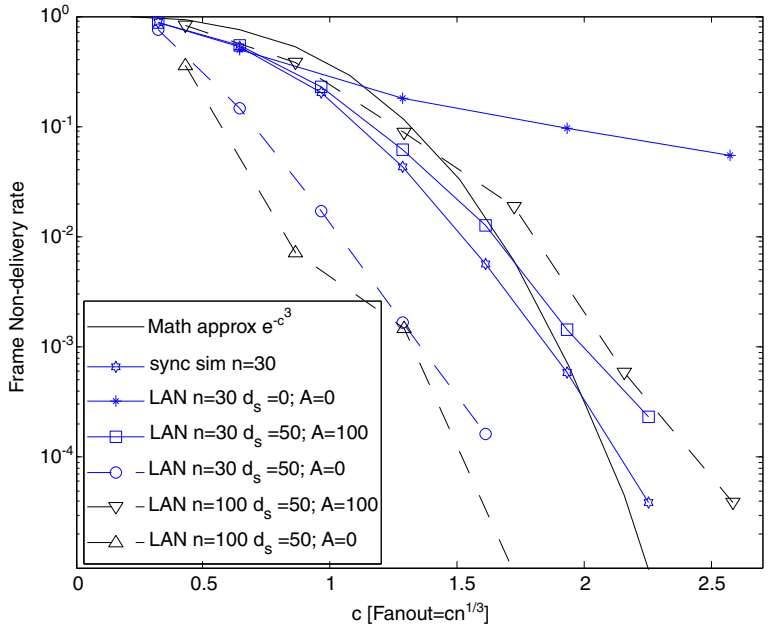


Figure 12 Measured frame non-delivery rate over campus network.

errors caused by factors such as system load, imperfect usleep() [78] and multi-threading switching, etc.

Another issue is that overloaded machines could cause inaccurate latency measurements, but finding idle machines as difficult in the PlanetLab. We observe that for a 30-party conferencing experiment, a maximum of 4 instances of our prototype peer may run on an idle Dual-core Pentium 4 3.2 GHz (Pentium D 940) concurrently. To tackle this problem, we use PlanetLab CoMon [79] to identify less loaded machines (which are quite rare) and then use Sirius Calendar Service [80] to reserve their CPU time [81].

C. Network experimentation results

We use our prototype system to measure the frame non-delivery probability p_l against c, where $c^3 = b^3/n$, to compare against the analytical result presented in Figure 5 in Sec. IV-A. We run two sets of experiments: one set over the HKUST campus LAN only and the second set over PlanetLab.

For experiments over the campus LAN, we deploy over 100 peers. Since all the machines are on the campus LAN, we add an artificial propagation delay of A in some of our experiments to better understand how the protocol would perform over a wide area. As shown in Figure 12, with $A=100$ and $d_s=50$ msec, the experimental results match the analytic model and the synchronous model simulation results in trend very well regardless of the number of peers n. Another observation is that in the LAN environment, with $A=0$ and $d_s=0$, the frame non-delivery probability remains high and becomes worse than the analytic result when c is increased. This can be understood from Sec. III-B1 where the reason for introducing the delay response d_s is explained. But once a delayed response $d_s = 50\ ms$ is applied when $A=0$, the performance is greatly improved and is much better than what the analytical model predicts. This can be explained by the bonus relay effect - when the propagation delay is small, there is a larger probability for the GREETING message from a source that contains information frame to have arrived at a peer before that peer launches its GREETING messages, leading to additional chances for spreading of the information frame.

For experiments over PlanetLab, because of the overloading conditions of PlanetLab as discussed above, we can only conduct measurements with 19 peers: 9 machines in the US, each supporting one peer instance, and 5 machines on the HKUST campus, each supporting two peer instances, as shown in Table 1.

Table 1 List of the planet lamb machines in use

1	PLANET11.CSC.NCSU.EDU
2	PLANETLAB1.CSUOHIO.EDU
3	PLANETLAB-2.CALPOLY-NETLAB.NET
4	PLANETLAB2.CS.COLORADO.EDU
5	PLANETLAB2.CS.UML.EDU
6	PLANETLAB2.CSUOHIO.EDU
7	PLANETLAB-3.ICS.UCI.EDU
8	PLANETLAB6.CSEE.USF.EDU
9	PLGMU5.ITE.GMU.EDU

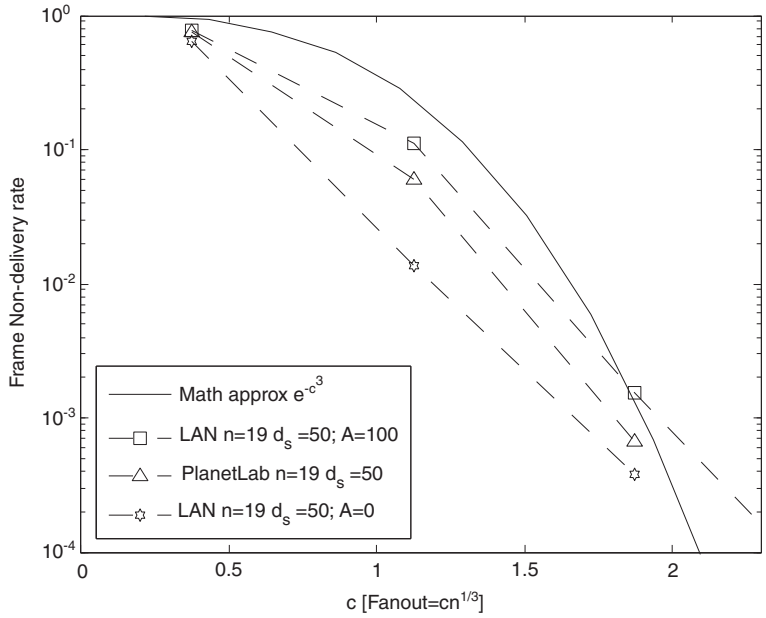


Figure 13 Measured frame non-delivery rate over PlanetLab.

For comparison, we also run experiments over the campus network with 19 peers. As shown in Figure 13, for d_s=50 and n=19, the performance of the PlanetLab falls between that of the campus network with A=0 and with A=100. This can be understood by the fact that some of the delays over the PlanetLab are larger than the offset in the cycle launch times of peers so that the chance of bonus relay is smaller over PlanetLab than over the campus network with no artificial delay. Therefore, the performance over PlanetLab is not as good as that over the campus network with no artificial delay.

Figure 14 plots p_l against the traffic load D, which is again the measured average number of the same frame received by each peer. As discussed before, D is an important measure that provides insight into the efficiency of the protocol. We observe in Figure 14 that when

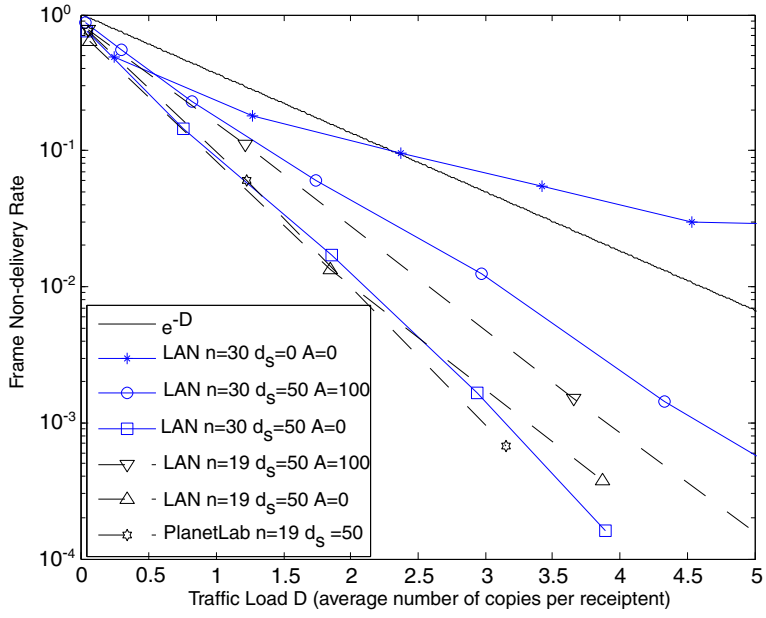


Figure 14 Measurement results of frame non-delivery against traffic load from implementation over campus network and PlanetLab.

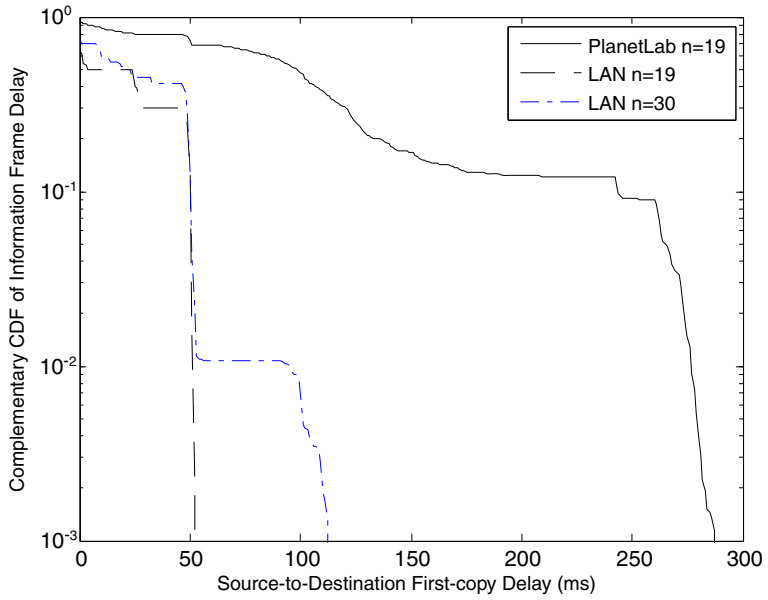


Figure 15 CDF of first information frame delay.

there is no delayed response (d_s = 0), the performance will not be as good as when d_s is set to 50 msec without a large artificial delay of *A=100* inserted. We have conducted a larger scale experiment on the campus network with 100 peers and observed similar performances to that in the 30-peer case. In general, the results show that the protocol is capable of achieving a non-delivery probability of 10^{-2} to 10^{-3} with a traffic load of 2 to 3.

Those experimental scenarios with larger delays (e.g. LAN *A=100* and PlanetLab) do not benefit much from the bonus relay, as explained in Sec. IV-B and Figure 7. This explains why the results shown in Figures 12, 13 and 14 indicate that experimental scenarios with larger delay produce what are much closer to the analytical results than those with smaller delays.

Another important performance measure is the information delay of the protocol as the protocol is targeted for real-time communication. We define information delay as the difference between the first copy arrival time of an information frame at a receiving peer and the

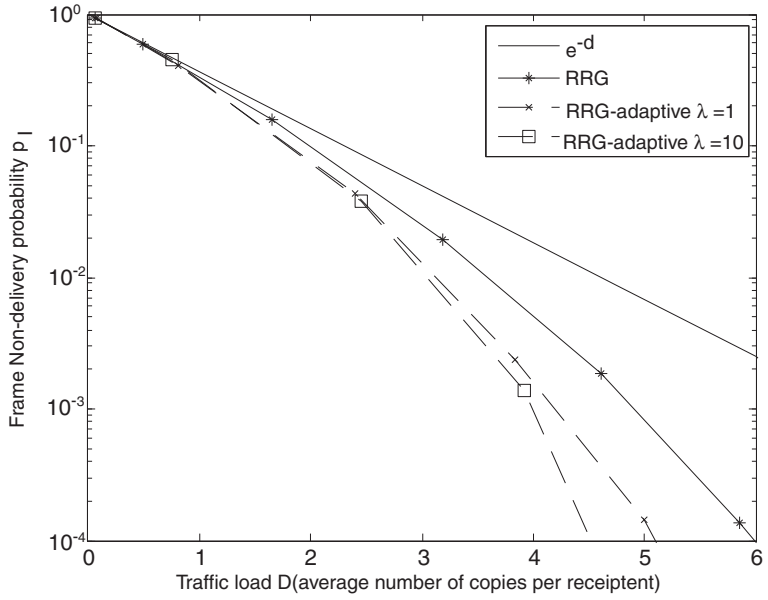


Figure 16 Performance comparison of RRG and RRG-adaptive.

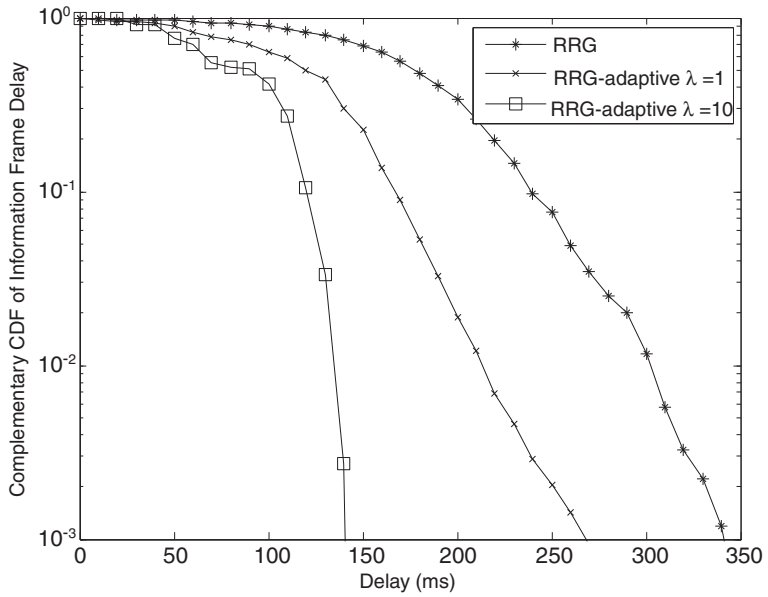


Figure 17 Complementary CDF of first information frame delay of RRG and RRG-adaptive.

generation time of the frame at the source. The clocks of all peers are synchronized using NTP throughout the experiment, and the clock drifts of the different machines are also examined to ensure that the end-to-end delay measurements are as accurate as possible. Figure 15 shows the CDF of the measured information delays with d_s = 50 ms. We observe that the 99.9-percentile of the first-copy delay is less than 120 ms over the campus LAN and less than 300 ms even over the cross-continent PlanetLab. Adding to this information delay a packetization and coding delay of, say, 50 ms the absolute one-way delay should be within the maximum limit of the 400ms for voice communications as specified in [16]; although, it will be quite a challenge to reduce the absolute delay to within 150 ms for transparent interactivity.

Finally, we use the prototype system to study the network behavior when the network experiences membership changes. In the case of a 30% decrease in group size, the non-delivery probability converges in less than 70 cycles (1.4 sec). In the case of a 30% sudden increase in group size, the convergence time is much shorter as new peers are quickly learned by other peers through their GREETINGS and subsequent gossip exchanges. Results are omitted here as they are similar to Figure 9 in Sec. IV-C.

VI. Future direction

There can be several directions for future improvements on the proposed RRG protocol. One is to make it adaptive to the traffic conditions. The RRG proposed so far changes its connectivity each cycle. But if some links have low latency, we can give higher priority to those links when the topology is generated in the next cycle. This has been explored in other gossip-based protocols [28,63,68].

To see if the same idea can improve RRG, we have simulated an extension of RRG, named as RRG-adaptive, with some capabilities in being adaptive to traffic conditions. The simulation setup for the improved protocol is the same as in Sec. V-B. Peers use the latency information learned by past cycles and sort other peers into a descending order of latency as $p = [p_1, p_2, \ldots, p_{n-1}]$. The selections of new peers in the next cycle will follow the following PDF $f(p_i, \lambda) = \lambda e^{-\lambda p_i}$, where λ is the parameter tuning the degree of preference for choosing lower latency peers. Of course, other distributions can also be used as long as they prefer lower latency peers.

Figure 16 compares the non-delivery probability between the RRG and RRG-adaptive. We can observe that RRG-adaptive requires slightly less traffic load than RRG to achieve the same non-delivery probability. We can also see that the performance gain of RRG-adaptive is higher in larger λ. This can be expected because, as

Table 2 Recovery time of RRG/RRG-adaptive under user churn

	Recovery time (sec)
RRG	1.12
RRG-adaptive λ =1	1.2
RRG-adaptive λ =10	6.34

shown in Figure 7, the phenomenon of bonus relay (see Sec. V-B) occurs more frequently when round trip delay is lower. Figure 17 compares the delay performance. It shows how the delay is reduced as a function of λ. We observe that the 99-percentile of the information delay of RRG-adaptive with $\lambda = 10$ is only half of that of RRG.

The adaptive peer selection results in a longer recovery time of user churn. We define recovery time as the convergence time of non-delivery probability in an event of user churn. Table 2 shows the recovery time of RRG and RRG-adaptive schemes under −80% sudden change of the original group size. We can observe that RRG-adaptive takes a longer time for recovery than RRG. We can also see the longer recovery time in larger λ. The adaptive scheme reduces the randomness of gossip and weakens its churn coping capability. We plan to study this tradeoff thoroughly in the future. Past knowledge learnt apart from latency, such as network topology or heterogeneous peer capabilities, can be considered in a randomized choice [28,50,68,82].

Another area for further work is to consider ways for suppressing redundant information delivery to further improve bandwidth efficiency. There is also an increasing interest in allowing the sources to adjust their coding rates to match the network conditions and peer capabilities (e.g. multi-rate and adaptive coded video sources) [42,83]. The idea is also applicable to RRG.

VII. Conclusions

In this paper, we present a novel protocol, called Redundancy Reduced Gossip, for real-time N-to-N dynamic group communication. The protocol allows multiple sources to distribute information across a group with low latency, minimal membership maintenance, and without an assumption on the underlying network condition. We have shown that a considerably lower traffic load than conventional push gossip protocols and conventional push-pull gossip protocols can be achieved with the same probability of successful delivery. We have also shown that better performance can be achieved in networks with smaller delays and when a delay response strategy is added to RRG, which is an asynchronous gossip protocol. We have derived a mathematical model for the frame non-delivery probability and overhead of the protocol. This model provides important insights into the design of our protocol and has been used to evaluate the performance of other related protocols. A functional prototype system has been implemented in C on the Linux platform. Its design is described, and it has been used to evaluate the performance of our protocol over our campus network as well as over a less organized global network (PlanetLab). Our experiments demonstrate that our protocol can maintain a robust performance in real-world network environments.

Competing interest
The authors declare that they have no competing interest.

Authors' contribution
VWHL, AKSW & CTL designed, implemented and evaluated the protocol and wrote the manuscript. RWO participated in the performance evaluation. All authors read and approved the final manuscript.

Acknowledgement
This work was supported by Hong Kong Research Grant Council project 620410.

Received: 18 October 2012 Accepted: 23 April 2013
Published: 17 May 2013

Reference

1. IBM (2010) IBM lotus sametime. [Online]. Available: http://www-01.ibm.com/software/lotus/sametime/
2. Google (2010) Google talk. [Online]. Available: http://www.google.com/talk/about.html
3. Skype Limited (2011) Skype video call. [Online]. Available: http://www.skype.com/intl/en-us/features/allfeatures/video-call/
4. Cisco (2010) WebEx solutions. [Online]. Available: http://www.webex.com/about-webex/index.html
5. Cisco (2010) Cisco TelePresense. [Online]. Available: www.cisco.com/web/go/telepresence
6. Polycom Inc (2010) Polycom telepresence solutions. [Online]. Available: http://www.polycom.com/products/telepresence_video/
7. Second Life (2010) Second life. [Online]. Available: http://secondlife.com/whatis/
8. Google (2012) Google hangouts. [Online]. Available: http://www.google.com/+/learnmore/hangouts/
9. EditGrid (2010) EditGrid. [Online]. Available: www.editgrid.com
10. Valve Corporation (2010) Counter-strike. [Online]. Available: http://www.valvesoftware.com/games/
11. Blizzard Entertainment (2010) World of warcraft. [Online]. Available: http://us.blizzard.com/en-us/games/wow/
12. Twitter (2010) Twitter. [Online]. Available: http://twitter.com/about
13. Facebook (2010) Facebook. [Online]. Available: http://www.facebook.com/
14. Five Minutes (2010) Happy farm. [Online]. Available: http://apps.facebook.com/happyfarmers
15. Zynga (2010) Café world. [Online]. Available: http://www.facebook.com/cafeworld?
16. ITU (1993) One-Way Transmission Time. In: ITU-T recommendation G.114. International Telecommunication Union, Geneva, Switzerland
17. Deering SE, Cheriton DR (1990) Multicast routing in datagram internetworks and extended LANs. ACM Trans Comput Syst 8:85–110
18. Lennox J, Schulzrinne H (2003) A protocol for reliable decentralized conferencing. In: NOSSDAV '03: Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video. , Monterey, CA, USA, pp 72–81
19. Kundan S, Gautam N, Henning S (2001) Centralized conferencing using SIP. In: Proc of the 2nd IP-Telephony Workshop. New York City, New York, USA
20. Baset SA, Schulzrinne H (2004) An analysis of the Skype peer-to-peer internet telephony protocol. In: Proceedings IEEE INFOCOM 2006 25TH IEEE International Conference on Computer Communications. IEEE Computer Society Press, Los Alamitos, CA, USA, pp 1–11
21. Yang-hua Chu SG, Rao S (2002) Seshan and Hui Zhang, A case for end system multicast, Selected Areas in Communications. IEEE Journal on 20:1456–1471
22. Li J, MutualCast (2005) MutualCast: A Serverless Peer-to-Peer Multiparty Real-Time Audio Conferencing System. Multimedia and Expo, 2005. ICME 2005. IEEE International Conference, pp 602–605
23. Irie M, Hyoudou K, Nakayama Y (2005) Tree-based Mixing: A New Communication Model for Voice-Over-IP Conferencing Systems. In: Proceedings of 2005 Internet and Multimedia Systems, and Applications, pp 353–358
24. Gu X, Wen Z, Yu PS, Shae ZY (2005) Supporting multi-party voice-over-IP services with peer-to-peer stream processing. In: Proceedings of the 13th Annual ACM International Conference on Multimedia, pp 303–306

25. Hosseini M, Georganas ND (2006) End System Multicast routing for multi-party videoconferencing applications. Comput Commun 29:2046–2065
26. Tseng S, Huang Y, Lin C (2006) Genetic algorithm for delay- and degree-constrained multimedia broadcasting on overlay networks. Comput Commun 29:3625–3632
27. Akkus IE, Civanlar MR, Ozkasap O (2006) Peer-to-peer multipoint video conferencing using layered video. Image Processing, 2006 IEEE International Conference, Antalya, pp 3053–3056
28. Venkataraman V, Yoshida K, Francis P (2006) Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference, Santa Barbara, CA, pp 2–11
29. Luo C, Wang W, Tang J, Sun J, Li J (2007) A Multiparty Videoconferencing System Over an Application-Level Multicast Protocol, *Multimedia*. IEEE Transactions on 9:1621–1632
30. Fahmy S, Minseok K (2007) Characterizing Overlay Multicast Networks and Their Costs,. Networking, IEEE/ACM Transactions on 15:373–386
31. Banik SM, Radhakrishnan S, Sekharan CN (2007) Multicast Routing with Delay and Delay Variation Constraints for Collaborative Applications on Overlay Networks, Parallel and Distributed Systems. IEEE Transactions on 18:421–431
32. Lao L, Cui J, Gerla M, Chen S (2007) A Scalable Overlay Multicast Architecture for Large-Scale Applications, Parallel and Distributed Systems. IEEE Transactions on 18:449–459
33. Tu W, Sreenan CJ, Jia W (2007) Worst-Case Delay Control in Multigroup Overlay Networks, Parallel and Distributed Systems. IEEE Transactions on 18:1407–1419
34. Tseng S, Lin C, Huang Y (2008) Ant colony-based algorithm for constructing broadcasting tree with degree and delay constraints. Expert Syst Appl 35:1473–1481
35. Zimmermann R, Liang K (2008) Spatialized audio streaming for networked virtual environments. In: Proceeding of the 16th ACM International Conference on Multimedia. Vancouver, British Columbia, Canada, pp 299–308
36. Nari S, Rabiee HR, Abedi A, Ghanbari M (2009) An efficient algorithm for overlay multicast routing in videoconferencing applications. Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th International Conference, San Francisco, CA, pp 1–6
37. Chia-Hui H, Kai-Wei K, Ho-Ting W (2009) An application layer multi-source multicast with proactive route maintenance. TENCON 2009–2009 IEEE Region 10 Conference, Singapore, pp 1–6
38. Liu Y (2010) Delay Bounds of Chunk-Based Peer-to-Peer Video Streaming, Networking. IEEE/ACM Transactions on 18:1195–1206
39. Liang C, Zhao M, Liu Y (2011) Optimal Bandwidth Sharing in Multiswarm Multiparty P2P Video-Conferencing Systems. Networking, IEEE/ACM Transactions 19:1704–1716
40. Chen X, Chen M, Li B, Zhao Y, Wu Y, Li J (2011) Celerity: Towards low-delay multi-party conferencing over arbitrary network topologies. In: Proceedings of the 21th International Workshop on Network and Operating Systems Support for Digital Audio and Video (ACM NOSSDAV 2011), Vancouver, Canada
41. Chen X, Chen M, Li B, Zhao Y, Wu Y, Li J (2011) Celerity: A low-delay multi-party conferencing solution. In: Proceedings of the 19th ACM international conference on Multimedia. Scottsdale, Arizona
42. Akkus IE, Ozkasap O, Civanlar MR (2011) Peer-to-peer multipoint video conferencing with layered video. J Netw Comput Appl 34:137–150
43. Verma S, Wei Tsang O (2005) Controlling gossip protocol infection pattern using adaptive fanout. In: Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference, Columbus, OH, pp 665–674
44. Georgiou C, Gilbert S, Guerraoui R, Kowalski DR (2008) On the complexity of asynchronous gossip. Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing, Canada, Toronto, pp 135–144
45. Ozkasap O, Xiao Z, Birman KP (1999) Scalability of two reliable multicast protocols. Cornell University, Ithaca, NY
46. Tang C, Chang RN, Ward C (2005) GoCast: Gossip-enhanced overlay multicast for fast and dependable group communication. In: Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference, pp 140–149
47. Melamed R, Keidar I (2004) Araneola: A scalable reliable multicast system for dynamic environments,. Network Computing and Applications, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium, Cambridge, MA, USA, pp 5–14
48. Zhang X, Liu J, Li B, Yum TSP (2005) CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol 3. Proceedings IEEE, Miami, p 2102
49. Padmanabhan VN, Wang HJ, Chou PA, Sripanidkulchai K (2002) Distributing streaming media content using cooperative networking. In: Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM New York, NY, USA, pp 177–186
50. Gupta I, Kermarrec AM, Ganesh AJ (2006) Efficient and adaptive epidemic-style protocols for reliable and scalable multicast, Parallel and Distributed Systems. IEEE Transactions on 17:593–605
51. Karp R, Schindelhauer C, Shenker S, Vocking B (2000) Randomized rumor spreading. Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium, Redondo Beach, CA, pp 565–574
52. Eugster PT, Guerraoui R, Kermarrec A, Massoulié L (2004) From Epidemics to Distributed Computing. IEEE Computer 37:60–67
53. Kermarrec AM, Massoulie L, Ganesh AJ (2003) Probabilistic reliable dissemination in large-scale systems, Parallel and Distributed Systems. IEEE Transactions on 14:248–258
54. Demers A, Greene D, Hauser C, Irish W, Larson J, Shenker S, Sturgis H, Swinehart D, Terry D (1987) Epidemic algorithms for replicated database maintenance. In: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing. ACM New York, NY, USA, pp 1–12
55. Chandra R, Ramasubramanian V, Birman K (2001) Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. Distributed Computing Systems, 2001. 21st International Conference, pp 275–283
56. Luo J, Eugster PT, Hubaux JP (2003) Route driven gossip: Probabilistic reliable multicast in ad hoc networks, INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IIEEE Societies 3:2229–2239. San Francisco, CA
57. Haas ZJ, Halpern JY, Li L (2006) Gossip-based ad hoc routing, Networking. IEEE/ACM Transactions on 14:479–491
58. Chun B, Culler D, Roscoe T, Bavier A, Peterson L, Wawrzoniak M, Bowman M (2003) PlanetLab: an overlay testbed for broad-coverage services. SIGCOMM Comput. Commun Rev 33:3–12
59. Fei H, Ravindran B, Jensen ED (2008) RT-P2P: A scalable real-time peer-to -peer system with probabilistic timing assurances,. Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference, Shanghai, China, pp 97–103
60. Kai H, Ravindran B, Jensen ED (2007) RTG-L: Dependably scheduling real-time distributable threads in large-scale, unreliable networks. Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium, Melbourne, Qld, pp 314–321
61. Kempe D, Dobra A, Gehrke J (2003) Gossip-based computation of aggregate information. Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium, Cambridge, MA, USA, pp 482–491
62. Mujtaba MK (2003) Push-pull gossiping for information sharing in peer-to-peer communities. In: Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). Las Vegas, pp 1393–1399
63. Leitao J, Pereira J, Rodrigues L (2007) Epidemic broadcast trees. Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium, Beijing, China, pp 301–310
64. Zhang X, Liu J (2004) Gossip based streaming. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. ACM New York, NY, USA, pp 250–251
65. Li HC, Clement A, Wong EL, Napper J, Roy I, Alvisi L, Dahlin M (2006) BAR gossip. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation. USENIX Association, Berkeley, CA, pp 191–204
66. Frey D, Guerraoui R, Kermarrec AM, Monod M (2010) Boosting gossip for live streaming. Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference, Delft, pp 1–10
67. Liang J, Ko SY, Gupta I, Nahrstedt K (2005) MON: On-demand overlays for distributed system management. Proceedings of USENIX WORLDS, Edinburgh
68. Carvalho N, Pereira J, Oliveira R, Rodrigues L (2007) Emergent structure in unstructured epidemic multicast. Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference, pp 481–490

69. Vishnumurthy V, Francis P (2006) On overlay construction and random node selection in heterogeneous unstructured P2P networks. In: Proceedings of IEEE INFOCOM'06. , Barcelona, Spain
70. Boyd S, Ghosh A, Prabhakar B, Shah D (2006) Randomized gossip algorithms, Information Theory. IEEE Transactions on 52:2508–2530
71. Ram SS, Nedic A, Veeravalli VV (2009) Asynchronous gossip algorithms for stochastic optimization. Decision and Control, 2009 Held Jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, Shanghai, China, pp 3581–3586
72. Deb S, Medard M, Choute C (2006) Algebraic gossip: a network coding approach to optimal multiple rumor mongering, Information Theory. IEEE Transactions on 52:2486–2507
73. Birman KP, Hayden M, Ozkasap O, Xiao Z, Budiu M, Minsky Y (May, 1999) Bimodal multicast. ACM Trans Comput Syst 17:41–88
74. Luk VWH, Wong AKS, Ouyang RW, Lea CT (2008) Gossip-based delay-sensitive N-to-N information dissemination protocol. IEEE Global Communications Conference IEEE GLOBECOM 2008, pp 1–5
75. Mills D (1992) RFC1305. Internet Engineering Task Force
76. Jelasity M, Montresor A, Babaoglu O (2005) Gossip-based aggregation in large dynamic networks. ACM Trans Comput Syst 23:219–252
77. Wikipedia (2010) Producer consumer problem. [Online]. Available: http://en.wikipedia.org/wiki/Producer-consumer_problem
78. Jim M, Paul E (2010) Usleep(3) - linux man page. [Online]. Available: http://linux.die.net/man/3/usleep
79. Park K, Pai VS (2006) CoMon: a mostly-scalable monitoring system for PlanetLab. SIGOPS Oper Syst Rev 40:65–74
80. PlanetLab (2010) Sirius calendar service. [Online]. Available: https://www.planet-lab.org/db/sirius/index.php
81. PlanetLab (2010) Sirius upgrade. [Online]. Available: http://www.planet-lab.org/node/5
82. Liang C, Guo Y, Liu Y (2008) Is random scheduling sufficient in p2p video streaming?. Distributed Computing Systems, 2008. ICDCS'08. the 28th International Conference, Beijing, China, pp 53–60
83. Ponec M, Sengupta S, Chen M, Li J, Chou PA (2009) Multi-rate peer-to-peer video conferencing: A distributed approach using scalable coding. In: Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, pp 1406–1413

doi:10.1186/1869-0238-4-14
Cite this article as: Luk *et al.*: RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication. *Journal of Internet Services and Applications* 2013 **4**:14.