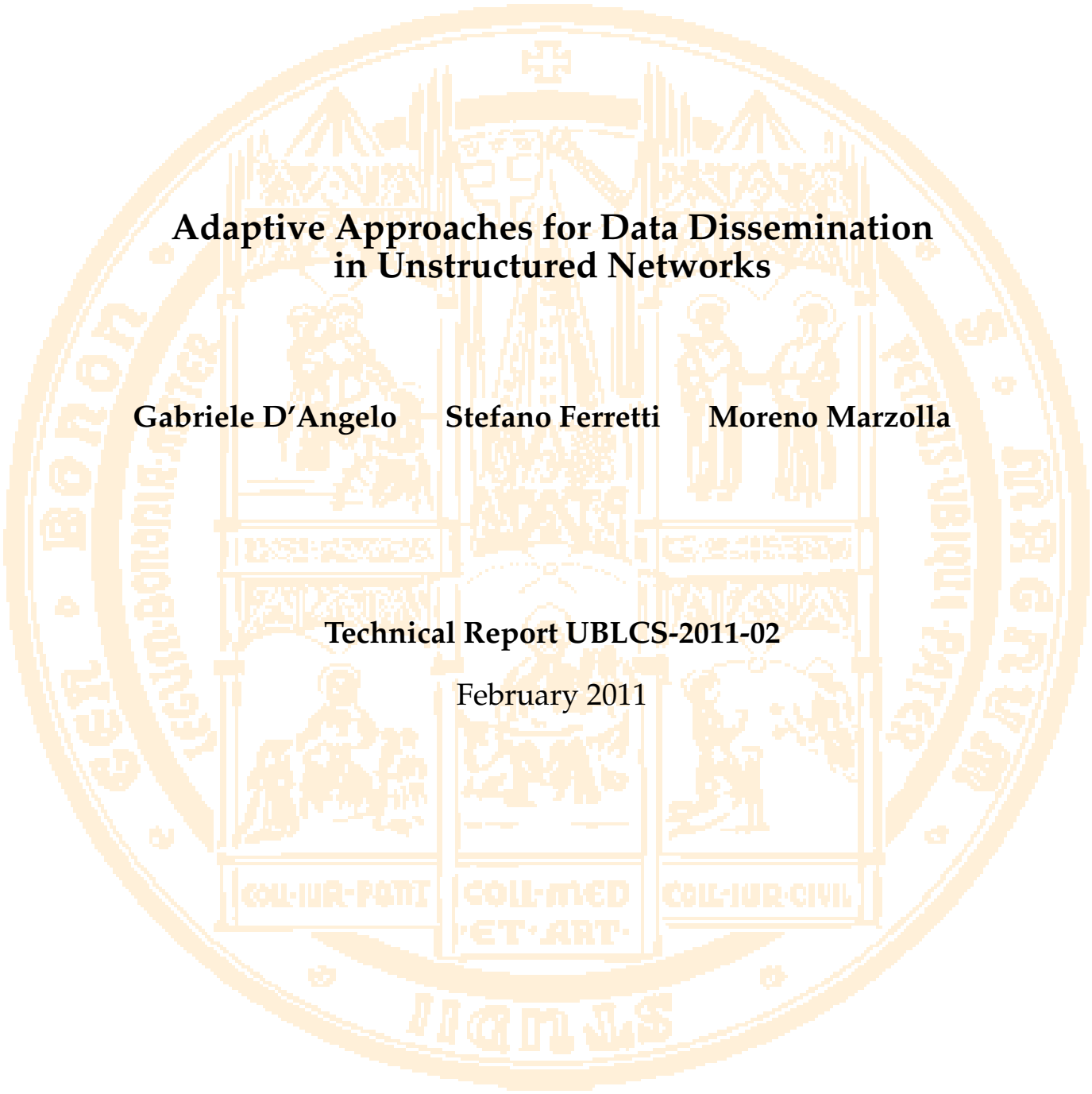

Adaptive Approaches for Data Dissemination in Unstructured Networks

Gabriele D'Angelo **Stefano Ferretti** **Moreno Marzolla**

Technical Report UBLCS-2011-02

February 2011

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in PDF and gzipped PostScript formats via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/`. Plain-text abstracts organized by year are available in the directory `ABSTRACTS`.

Recent Titles from the UBLCS Technical Report Series

2009-08 *3-Dimensional Protein Reconstruction from Contact Maps: Complexity and Experimental Results (Ph.D. Thesis)*, Medri, F., February 2009.

2009-09 *A core calculus for the analysis and implementation of biologically inspired languages (Ph.D. Thesis)*, Versari, C., February 2009.

2009-10 *Probabilistic Data Integration*, Magnani, M., Montesi, D., March 2009.

2009-11 *Equilibrium Selection via Strategy Restriction in Multi-Stage Congestion Games for Real-time Streaming*, Rossi, G., Ferretti, S., D'Angelo, G., April 2009.

2009-12 *Natural deduction environment for Matita*, C. Sacerdoti Coen, E. Tassi, June 2009.

2009-13 *Hints in Unification*, Asperti, A., Ricciotti, W., Sacerdoti Coen, C., Tassi, E., June 2009.

2009-14 *A New Type for Tactics*, Asperti, A., Ricciotti, W., Sacerdoti Coen, C., Tassi, E., June 2009.

2009-15 *The k-Lattice: Decidability Boundaries for Qualitative Analysis in Biological Languages*, Delzanno, G., Di Giusto, C., Gabbrielli, M., Laneve, C., Zavattaro, G., June 2009.

2009-16 *Landau's "Grundlagen der Analysis" from Automath to lambda-delta*, Guidi, F., September 2009.

2010-01 *Fast overlapping of protein contact maps by alignment of eigenvectors*, Di Lena, P., Fariselli, P., Margara, L., Vassura, M., Casadio, R., January 2010.

2010-02 *Optimized Training of Support Vector Machines on the Cell Processor*, Marzolla, M., February 2010.

2010-03 *Modeling Self-Organizing, Faulty Peer-to-Peer Systems as Complex Networks* Ferretti, S., February 2010.

2010-04 *The qnetworks Toolbox: A Software Package for Queueing Networks Analysis*, Marzolla, M., February 2010.

2010-05 *QoS Analysis for Web Service Applications: a Survey of Performance-oriented Approaches from an Architectural Viewpoint*, Marzolla, M., Mirandola, R., February 2010.

2010-06 *The dark side of the board: advances in Kriegspiel Chess (Ph.D. Thesis)*, Favini, G.P., March 2010.

2010-07 *Higher-Order Concurrency: Expressiveness and Decidability Results (Ph.D. Thesis)*, Perez Parra, J.A., March 2010.

2010-08 *Machine learning methods for prediction of disulphide bonding states of cysteine residues in proteins (Ph.D. Thesis)*, Shukla, P., March 2010.

2010-09 *Pseudo-Boolean clustering*, Rossi, G., May 2010.

2010-10 *Expressiveness in biologically inspired languages (Ph.D. Thesis)*, Vitale, A., March 2010.

2010-11 *Performance-Aware Reconfiguration of Software Systems*, Marzolla, M., Mirandola, R., May 2010.

2010-12 *Dynamic Scalability for Next Generation Gaming Infrastructures*, Marzolla, M., Ferretti, S., D'Angelo, G., December 2010.

2011-01 *Server Consolidation in Clouds through Gossiping*, Marzolla, M., Babaoglu, O., Panzieri, F., January 2011.

Adaptive Approaches for Data Dissemination in Unstructured Networks

Gabriele D'Angelo[1] **Stefano Ferretti**[2] **Moreno Marzolla**[3]

Technical Report UBLCS-2011-02

February 2011

Abstract

In this technical report we study the performance of gossip protocols for data dissemination in unstructured networks. More in detail, the goal is to propose a methodology for the assessment of such kind of protocols and to evaluate new methods for data dissemination that are based on adaptive approaches.

In particular, we show that gossip algorithms may be effectively used to disseminate updates in many systems such as the Peer-to-Peer (P2P) video conferencing, in which messages are disseminated through an overlay network. The proposed scheme exploits the typical behavior of such kind of systems to tune the data dissemination. In fact, it can be assumed that users participating in a video conference typically generate update events at a rate that can be approximated using some (system dependent) probability distribution. Hence, as soon as a given node experiences a reception rate, for messages coming from a given peer, which is lower than expected, it can send a stimulus to the neighbor that usually forwards these messages, asking it to increase its dissemination probability. Three variants of this approach will be studied. According to the first one, upon reception of a stimulus from a neighbor, a peer increases its dissemination probability towards that node irrespectively from the sender. In the second protocol a peer increases only the dissemination probability for a given sender towards all its neighbors. Finally, the third protocol takes into consideration both the sender and the neighbor in order to decide how to increase the dissemination probability. We performed extensive simulations to assess the efficacy of the proposed scheme, and based on the simulation results we compare the different dissemination protocols. The results confirm that adaptive gossip schemes are indeed effective and deserve further investigation.

1. Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy, Email: g.dangelo@unibo.it
2. Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy, Email: sferrett@cs.unibo.it
3. Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy, Email: marzolla@cs.unib.it

1 Introduction

As the bandwidth available on networks and the speed of computers increases, real-time transmission of "rich media content" among large sets of participants becomes more and more common. For example, this is the case of videoconferencing [13] and networked virtual environments [11]. In both cases, the traditional approach based on the client-server architecture is inadequate due to scalability constraints. A much more promising approach is Peer-to-Peer (P2P), in which an overlay network is built among participants. The very dynamic and variable nature of the generated overlay network makes impossible to employ the traditional "routing based" methods for data dissemination in such kind of networks. Much more dynamic and adaptive approaches are necessary.

Previous works have already demonstrated that gossip strategies can be proficiently employed to disseminate data in P2P overlay networks [4, 6]. The topology of the overlay is a critical aspect, since the different characteristics of the network correspond to different features that influence the message dissemination. For instance, when peers organize themselves as a scale-free network, the network as a low diameter; hence, a low number of hops is required to cover the whole network with a broadcast message. However, certain nodes must act as hubs, i.e. they maintain a large number of neighbors (degree). This corresponds to an unbalanced workload among peers. Conversely, overlays with uniform degree distribution result in balanced workload required to forward messages in the network.

Regardless of the network structure employed as the P2P overlay, the use of a dissemination strategy based on a static spanning tree built on top of such overlay is not appropriate for highly dynamic scenarios, such as P2P systems where the number of nodes (and hence the network itself) frequently changes. At the same time, it is not possible to employ pure broadcasts to disseminate update events. In fact, the number of updates generated during a (video) conference session is quite high. Hence, it is necessary to avoid as much as possible redundant transmissions to not congestion the network.

The considerations above motivate the need to devise novel, adaptive decentralized algorithms for update events distribution in dynamic P2P systems. In this technical report, we propose an adaptive gossip scheme that exploits the typical behavior of video conferencing systems to optimize the message distribution among nodes. It is known that such kind of system (such as many others) generate update events according to some (application-specific) inter-generation probability distribution between successive update. This feature can be employed to dynamically tune the dissemination probability of events coming from a given peer in the overlay.

Specifically, based on our approach, messages are gossiped through an overlay network (a mesh) using a completely decentralized approach. Once a peer receives a message from a neighbor, it forwards the message to other neighbors based on a dissemination probability. As soon as a node p observes that it is receiving messages from another peer q at a rate lower than expected, it activates a countermeasure, asking its neighborhood (actually, the neighbor n from which it usually receives messages originated from q or a random neighbor if it did not receive any message at all) to increase its dissemination probability of update events. Three variants of the this scheme are considered. According to the first one, upon reception at n of a request from p to increase the event flow, n increases the probability of dissemination of update events towards n, independently of the originator of the events to be forwarded. In the second approach, n increases its dissemination probability only for update events originating from q (independently from the receivers). In the third variant, the node n increases its dissemination probability only for update events coming from q and that will be delivered to p, that is the specific sender that has requested the probability increase.

The request from p to n to increase the dissemination probability can be interpreted as a stimulus that remains active at n for a limited period of time. Then, the dissemination probability returns to the original value (i.e. the stimulus decades in time). This approach is adopted to avoid that in time all dissemination probabilities reach the maximum value and thus the gossip scheme becomes a pure broadcast algorithm.

We assessed the algorithms above using simulation experiments. The results demonstrates

the effectiveness of our approach; in particular, we we observe that adaptive gossip schemes improve the update event dissemination with limited additional overhead. Additional experimental and methodological aspects will be discussed in detail.

The reminder of this technical report is structured as follows. Section 2 presents the system model. In Section 3 we discuss the adaptive gossip algorithms. Section 4 reports on simulation experiments we carried out to assess the viability of our proposal. Finally, concluding remarks are reported in Section 5.

2 System Model

We consider a video conferencing system organized as a P2P network. Peers communicate through an overlay, which means that only nodes which are directly connected in the overlay can exchange messages. Nodes which are not directly connected must resort to multi-hop communication. We do not impose any restriction on the overlay, which can be generated using any kind of algorithm and attachment protocol when peers join the network. Several alternatives exist in the literature [2, 4, 7].

Each node produces update events which must be disseminated to all other nodes in the network. The inter-generation time of events follows a node-specific probability distribution.

It is clear that the topology of the overlay has a strong influence on the performances of the content dissemination. For instance, if a scale-free network is employed, then the network has a low diameter (in general it ranges from $\log \log N$ to $\log N$, being N the number of nodes). This means that a message requires very few hops to travel from a node to any other node, assuming that routing happens only along shortest paths. Also, scale-free networks are known to be resilient against random failures [12]. However, they contain a non-negligible fraction of peers with high degree; these nodes (hubs) have a high number of neighbors, and thus must maintain a high number of active connections [2, 4, 10]. Hubs will likely sustain a higher workload than the other low-degree nodes.

Conversely, if a network has uniform degree distribution, meaning that all nodes have approximately the same number of neighbor nodes, then the workload is equally shared among all peers. However, the diameter of the network increases, and so does the number of hops needed to cover the whole network with a broadcast [5]. Moreover, random networks are more prone to partitioning after random failures. Hence, some additional control mechanism is needed to cope with this issue.

We assume that in the P2P system, every peer knows all other peers. This is quite reasonable in this kind of video conferencing system, since all user want to participate in the same conference session.

3 Adaptive Gossip

The adaptive gossip algorithm[4] is a basic push scheme: nodes which have novel information to disseminate are responsible for generating or forwarding messages to other peers [6]. Each node forwards messages (update events) to a random subset of its neighbors. The peculiarity of our proposal is that the dissemination probability of messages at node p varies depending on the communication performances perceived at p during the conference session.

As mentioned, each peer p knows the list of peers that are participating in the conference session. For each peer, p maintains statistical information on received messages, such as the average reception rate and the times of last received events. These metrics allow p to estimate whether it is receiving updates from other peers at the "correct" rate.

The gossip protocol is executed when p receives a novel update event generated at the application layer, or when an update event is received from another peer q. In either case, p randomly

4. In the following of this technical report, when referred to the proposed adaptive gossip schemes the terms "algorithm" and "protocol" will be used interchangeably.

Algorithm 1 Adaptive Gossip #1: gossiping procedure executed by p

Require: *msg* generated at $p \vee$ *msg* received from a peer q
1: $N_p \leftarrow p$'s neighbors $\setminus q$ {$q =$ NULL if *msg* originated at p}
2: **if** *msg* is a duplicate **then**
3: Return
4: **end if**
5: **for all** $n \in N_p$ **do**
6: currentTime $\leftarrow$ GETTIME()
7: $\upsilon_n \leftarrow$ COMPUTETHRESHOLD(n, currentTime)
8: **if** RANDOM() $< \upsilon_n$ **then**
9: SEND(*msg*,n)
10: **end if**
11: **end for**

Algorithm 2 Adaptive Gossip #1: Monitoring procedure executed by p

1: **loop**
2: SLEEP(monitoringPeriod)
3: peerList $\leftarrow$ RETRIEVEPEERSLOWRATE() {retrieve peers with low reception rate}
4: **for all** $j \in$ peerList **do**
5: $q \leftarrow$ FORWARDER(j) {neighbor that sends msgs from j}
6: SEND(q, "low rate from j")
7: **end for**
8: **end loop**

selects some neighbors to which the event will be propagated [9, 14]. In what follows, we consider three different protocols that exploit a stimulus to increase the probability of dissemination of messages at a given peer, based on measured performances.

3.1 Algorithm #1: Stimuli Associated to Receivers

The first protocol works as follows. When a node p receives a new message *msg* from one of its neighbors q, then p considers all its neighbors (except q). For each neighbor n, the message is forwarded to n with probability υ_n, where υ_n is a threshold which is dynamically computed (details will be given shortly). Note that each message is processed only once: we assume that each node p maintains a LRU cache of recently seen messages, and drops all duplicates which are received after the first message has been processed. The pseudo-code for this protocol is shown in Algorithm 1.

The threshold υ_n plays a fundamental role for the message dissemination, since after such a gossip p will never reconsider msg for dissemination. All values υ_n are initially set to a given constant υ_0. Then, all values are updated periodically by a monitoring procedure, which runs on each peer. The monitoring procedure periodically measures the reception rate of update events originated at each node i in the overlay network. As soon as p observes a lower update event reception rate from a peer (say j), it selects the neighbor q from which it usually receives messages containing update events generated by j;[5] then, p sends q a stimulus message to request the increase of the value of υ_p stored at q. This procedure is sketched in Algorithm 2.

The RETRIEVEPEERSLOWRATE() procedure (line 3, Algorithm 2) collects the list of nodes i, in the whole network, from which R_i messages has been received in the monitoring period T_{mon} such that

$$R_i < \alpha\omega T_{mon}$$

where ω is the expected event generation rate, and α is a parameter that can be tuned.

The stimulus that increases the dissemination probability υ_p decays over time. This means that after a certain deadline, its effect terminates and υ_p comes back to the default value υ_0.

5. When multiple peers are possible forwarders of messages originating from j, q is selected as a random neighbor of p.

Algorithm 3 Adaptive Gossip #1: Procedure executed by p upon stimulus reception

Require: stimulus received from q
1: *timeLastStimulus*$_q$ = GETTIME()

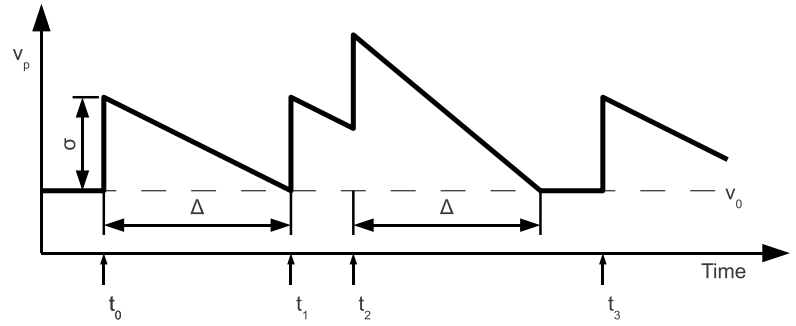

Figure 1. Pictorial representation of function υ_p. A stimulus σ is received at times t_0, t_1, t_2 and t_3. At time t_2, the stimulus adds σ to the current value of υ_p. υ_p decays linearly to υ_0 after time Δ from the last received stimulus.

Algorithm 3 describes what happens at q upon reception of a request from p to increase its dissemination probability: a variable *timeLastStimulus*$_p$, which contains the last received stimulus from p, is updated to the current time. In fact, the measure of the threshold υ_p depends on the time elapsed since *timeLastStimulus*$_p$ as we will discuss shortly.

Different implementations of the COMPUTETHRESHOLD() procedure (line 7 in Algorithm 1) are possible. In the simulations, we adopted the following function: all thresholds are initialized to a default value υ_0. Upon reception of a stimulus from a peer p at time *timeLastStiumlus*$_p$, the actual value of υ_p is increased by a fixed quantity σ. Then, υ_p decades linearly over a time interval of length Δ, such that at time *timeLastStimulus*$_p + \Delta$ its value is back to υ_0. If another stimulus is received during the decaying phase, the stimulus adds to the current value of υ_p and *timeLastStimulus*$_p$ is updated accordingly; in any case, υ_p decays linearly to υ_0 after time Δ from *timeLastStimulus*$_p$. We also remark that the value of υ_p is limited to 1. See Figure 1 for a pictorial explanation.

3.2 Algorithm #2: Stimuli Associated to Generators

We now consider a protocol which differs from the previous one in the method to adapt the dissemination threshold. Each peer p maintains an array of dissemination thresholds, one for each node in the network. As soon as a new message *msg* generated by s has to be disseminated by p, all p's neighbors (except the peer q that sent *msg*, if *msg* has not been originated by p itself) are considered and a threshold value $\gamma_s \leq 1$ is computed.[6] The value γ_s is employed as the threshold to determine if *msg* has to be gossiped to a given neighbor. The pseudo-code of the described protocol is outlined in Algorithm 4.

The monitoring procedure remains the same as the in previous scheme (Algorithm 2), as well as the function used to compute the actual value of the threshold (COMPUTETHRESHOLD(), line 7 in Algorithm 4). The only difference is that values γ_s, γ_0 must be employed instead of υ_p, υ_0. The same happens for what concerns the request from q to p to increase the dissemination probability related to peer j, i.e. the procedure is equal, however, the variable *timeLastStimulus*$_j$ is employed to update the variable γ_s.

3.3 Algorithm #3: Stimuli Associated to Generators and Receivers

This version of the adaptive dissemination protocol is derived from Algorithm #2 and also in this case the difference is in the mechanism used to adapt the threshold employed to gossip messages.

6. Since the message is gossiped through an overlay, it is possible that $q \neq s$.

Algorithm 4 Adaptive Gossip #2: the gossiping procedure executed at p

Require: *msg* generated at $p \vee$ *msg* received from a peer q
1: **if** *msg* is a duplicate **then**
2: Return
3: **end if**
4: $N_p \leftarrow p$'s neighbors $\setminus\, q$ {$q =$ NULL if *msg* originated at p}
5: $s \leftarrow$ peer that generated *msg*
6: currentTime $\leftarrow$ GETTIME()
7: $\gamma_s \leftarrow$ COMPUTEPROB(s, currentTime)
8: **for all** $n \in N_p$ **do**
9: **if** RANDOM() $< \gamma_s$ **then**
10: SEND(*msg*,n)
11: **end if**
12: **end for**

In this variant, each peer p has not a single array of dissemination thresholds (such as in Alg. #2), rather it maintains a set of arrays (one for each neighbor). In this way, each stimulus that is received will cause a very selective update: it will change the probability to disseminate the messages originated by a specific node which should be forwarded to a given neighbor. The other parts of the dissemination algorithm remain unaltered. The aim of this protocol is to generate much more stimuli but each one is very specific and targeted.

4 Simulation Assessment

In this Section we investigate the performance of the dissemination protocols described above using a simulation model. First of all some aspects about the comparison of dissemination protocols will be discussed. Then the proposed algorithms will be compared to some well known gossip protocols. Finally, some variants (i.e. different setups) of the protocol that gives the best outcomes will be shown.

4.1 Testbed and metrics

All dissemination protocols have been run on a set of 100 different overlay networks that have been randomly generated using an Erdos-Renyi generator. All graphs are undirected, and they have been constructed to ensure that they are also connected. Furthermore, we ensure that the diameter of the graphs is always less than a predefined value (i.e. that will be used to set the Time-To-Live in dissemination messages). As said before, also other graph structures (i.e. scale-free and small-world) would be interesting to evaluate but are left as future work.

We now define some metrics under which the protocols will be evaluated. Informally, a desirable property of a dissemination protocol is that of being able to reach all nodes, and this should happen as quickly as possible. Thus we define a metric called **coverage**, which denotes the fraction of nodes which actually received the messages. Ideally we wish to obtain 100% coverage, meaning that all nodes received all the generated messages. The second metric is called **delay**, and represents the average number of hops that a message traverses before reaching a node (lower is better). The delay is computed as follows: when a message is received by a node for the first time, that node records the number of hops the message traversed from its generation. The delay is computed as the number of hops, averaged over all nodes which received the message, and over all messages sent during a simulation run.

It is important to also define appropriate cost metrics, so that all dissemination protocols can be compared in the same conditions. We define the "overhead ratio" ρ as follows:

$$\rho = \frac{\textit{Delivered messages}}{\textit{Lower bound}}$$

where "delivered messages" is the total number of messages that are delivered in a simulation run by a specific dissemination protocol and the "lower bound" is the minimum number of messages (in each graph) that are necessary to obtain a complete coverage. Thus, the lower bound represents the number of messages sent by a broadcast protocol which deliver events along the edges of a spanning tree, and never sends duplicates. The lower bound depends on the graph and is independent from the dissemination protocol to be used. For example, in a graph of n nodes and in which m different events are generated, the lower bound to the number of delivered messages is $\Omega(nm)$. Each newly generated message has to traverse at least $n-1$ links to eventually reach all nodes in the graph. Observe that $n-1$ is precisely the number of edges on any spanning tree on a graph with n vertices.

4.2 Simulator

The performance evaluation of the adaptive gossip algorithms described in Section 3 has been conducted using discrete-event simulation based approach. The simulator, called LUNES (Large Unstructured NEtwork Simulator) [1] has been rewritten from scratch after our previous work on PaScaS [4].

The main goal of LUNES is to offer an efficient and easy-to-use tool for the simulation of complex protocols on top of large graphs. In practice, LUNES is able to import the graph topologies generated by other tools (e.g. igraph) and provides the functionalities that are needed for the performance evaluation of simulated protocols. One of the main goals of the simulator redesign is to obtain a tool that clearly splits the fundamental phases:

- network topology creation;
- protocol simulation in a specific testbed;
- traces analysis (i.e. performance evaluation).

This modular approach permits the easy integration of external software tools. In practice, such integration is based on very simple template files (such as the graphviz dot language [8]) and a provides a good level of extensibility. Under the performance and scalability viewpoint, the most demanding points are the protocol simulation and the traces analysis. The first one is demanded to a specific simulator that will be discussed shortly. The second one, that is the traces analysis, has been excluded from the simulation tasks and some specific software tools have been implemented.

The amount of traces generated by a single run of the protocols described in this technical report, in a medium size graph, is pretty large. For example, to evaluate the performance of a dissemination protocol all the different messages seen by each node have to be accounted for. Therefore, efficiency is essential in order to obtain timely results. In the current version of LUNES this task is implemented via a mix of shell scrips and dedicated tools written in C language (for performance reasons). All such tools have been designed and implemented to work in parallel and therefore are able to exploit all the computational resources provided by parallel (multiprocessor or multi-core) architectures.

As said above, the simulation services are demanded to the ARTÌS middleware and the GAIA framework [1]. In this way, the LUNES user does not need to deal with low-level simulation details and can transparently take advantage of all the features offered by GAIA/ARTÌS. In particular, to allow the simulation of large models, a parallel and distribution simulation approach can be followed and some advanced features such as the dynamic model partitioning and load-balancing features are implemented transparently. For example, clusters composed of very heterogeneous nodes (in terms of hardware) can be employed for the simulation of large networks: the model partitioning is dynamic, adaptive and totally demanded to the simulation tools without any tuning to be done by the simulator user [3].

Algorithm 5 Fixed Probability dissemination

Require: *msg* generated at $p \vee$ *msg* received from a peer q
1: $N_p \leftarrow p$'s neighbors $\setminus\ q$ {$q =$ NULL if *msg* originated at p}
2: **if** *msg* is a duplicate **then**
3: Return
4: **end if**
5: **for all** $n \in N_p$ **do**
6: **if** RANDOM() $< v_0$ **then**
7: SEND(*msg*,n)
8: **end if**
9: **end for**

4.3 Model parameters

In the following we have considered networks (i.e. graphs) composed of up to 100 nodes (i.e. peers) and generated as reported in Section 4.1. Each node has 2 edges, that is 200 edges in the whole network. Given the good scalability of the simulator, the evaluation of graphs with a larger number of vertices and edges is not a problem. This task is left as future work given that now we are more interested in a preliminary validation of the proposed approach.

Focusing again on the model parameters, each simulation run is 5000 time steps long and each node in the network can generate new messages during the whole simulation lifespan. The time between successive messages is generated according to a typical exponential distribution. The variability of the inter-generation between two successive update events generated by the same peer is of main importance, since peers send stimuli to their neighbors based on the variability of reception of messages. In other words, it is important to understand whether a low update event reception rate is due to a poor dissemination caused by the gossip algorithm, rather than a low generation rate by a given node. It is worth noting that other "more predictable" distributions (e.g. uniform) would largely increase the results of all adaptive dissemination algorithms. That's because each form of variability has the effect to "confuse" the evaluation heuristics implemented in the adaptive gossip protocols and to generate stimuli that are not necessary, with the side effect to increase the communication cost of the protocol.

Finally, as already introduced in Section 3, each node implements a cache structure with the aim to reduce the number of duplicate messages sent around. This cache is managed using the Least Recently Used (LRU) replacement algorithm; the cache size has been set to 256 items. To limit the lifetime of each message in the network, we implemented a Time-To-Live (TTL) scheme. When a new message is created, the TTL is set to 8, a value that we ensure is always greater than or equal to the network diameter. As usual, each hop will reduce this value up to discarding.

4.4 Results

We first compare the simplest adaptive algorithm (i.e. *Algorithm #1: Stimuli Associated to Receivers*) with respect to very common dissemination algorithms such as the *Fixed Probability* and the *Probabilistic Broadcast* disseminations. In the *Fixed Probability* dissemination scheme (see Algorithm 5), the node that receives a new message randomly selects those edges through which the message must be propagated. In the *Probabilistic Broadcast* (Algorithm 6), the node decides whether to forward the received message with a certain probability. If the message is forwarded, it is always sent to all neighbors. For more details, see [6, 4]. We have already shown that another very common dissemination algorithm, called *Fixed Fanout*, is unable to offer acceptable results in these conditions [6].

In this first test, the setup of the adaptive algorithm is the following: $monitoringPeriod = 100$, $\sigma = 0.2$, $\delta = 300$, $\alpha = 1/3$ (see Section 3 for the description of each parameter). For each algorithm, at least 10 different setups have been evaluated and the best results are shown.

The Figures in this Section show the results obtained by the proposed dissemination protocols (in terms of **coverage** and **delay**) with respect to the cost (in terms of "overhead", ρ) incurred by each protocol. Given a dissemination protocol and a specific setup, we varied a single parame-

Algorithm 6 Probabilistic Broadcast dissemination

Require: *msg* generated at $p \vee$ *msg* received from a peer q

1: $N_p \leftarrow p$'s neighbors $\setminus q$ $\{q =$ NULL if *msg* originated at $p\}$
2: **if** *msg* is a duplicate **then**
3: Return
4: **end if**
5: **if** RANDOM() $< \upsilon_0 \vee$ *msg* generated at p **then**
6: **for all** $n \in N_p$ **do**
7: SEND(*msg*,n)
8: **end for**
9: **end if**

ter: the default dissemination probability υ_0. Specifically, we considered 100 different values for υ_0, uniformly distributed in the range $(0, 1]$. For each value of υ_o, we executed each algorithm on multiple different random graphs, all of the same size, and computed average performance results on each set of runs. The resulting data set is shown in the Figure.

In Figure 2 we observe that, in terms of **coverage**, the adaptive gossip protocol (*Alg. #1*) is much better than the *Probabilistic Broadcast* and slightly better than *Fixed Probability*. The results obtained for $\rho < 1$ are not very interesting given that, in any case, all protocols would be unable to obtain a full coverage of the network (because the total number of messages used for the dissemination is below the lower bound seen above). On the other hand, when $\rho > 2.5$ the network is so full of messages that all the algorithms have very similar outcomes.

In terms of **delay**, the *adaptive protocol* is slightly worse than the *Fixed Probability*. It is worth noticing that, in this case, lower is better given that the delay is proportional to the average time that is necessary for the delivery of messages.

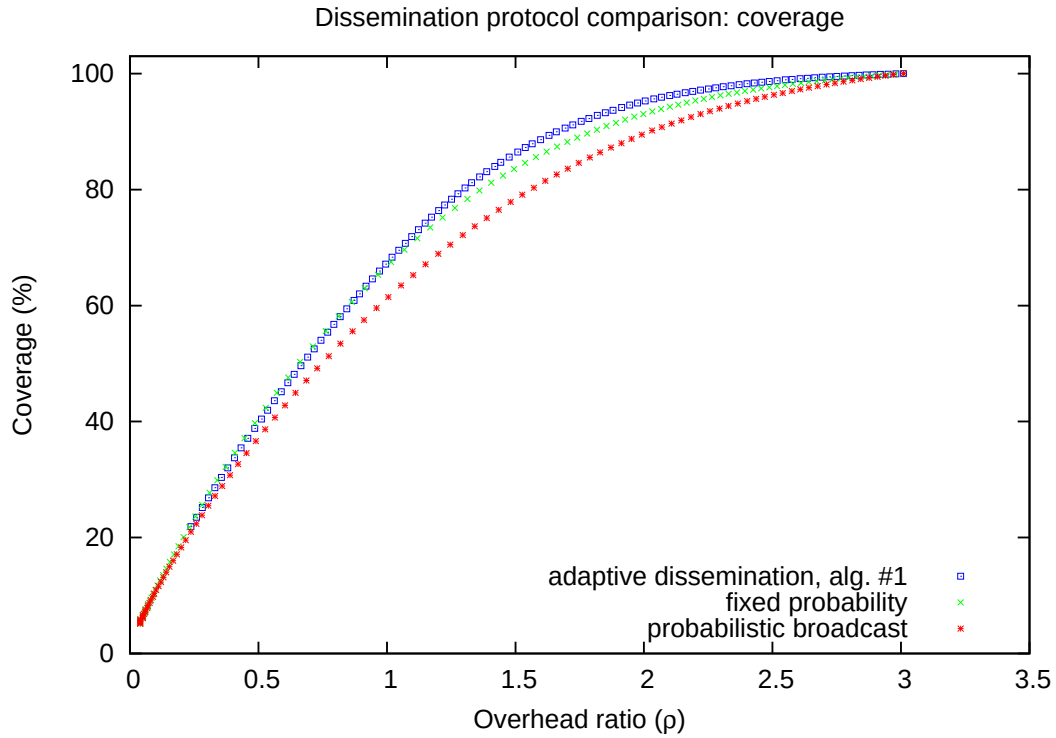

Figure 2. Dissemination protocols, coverage

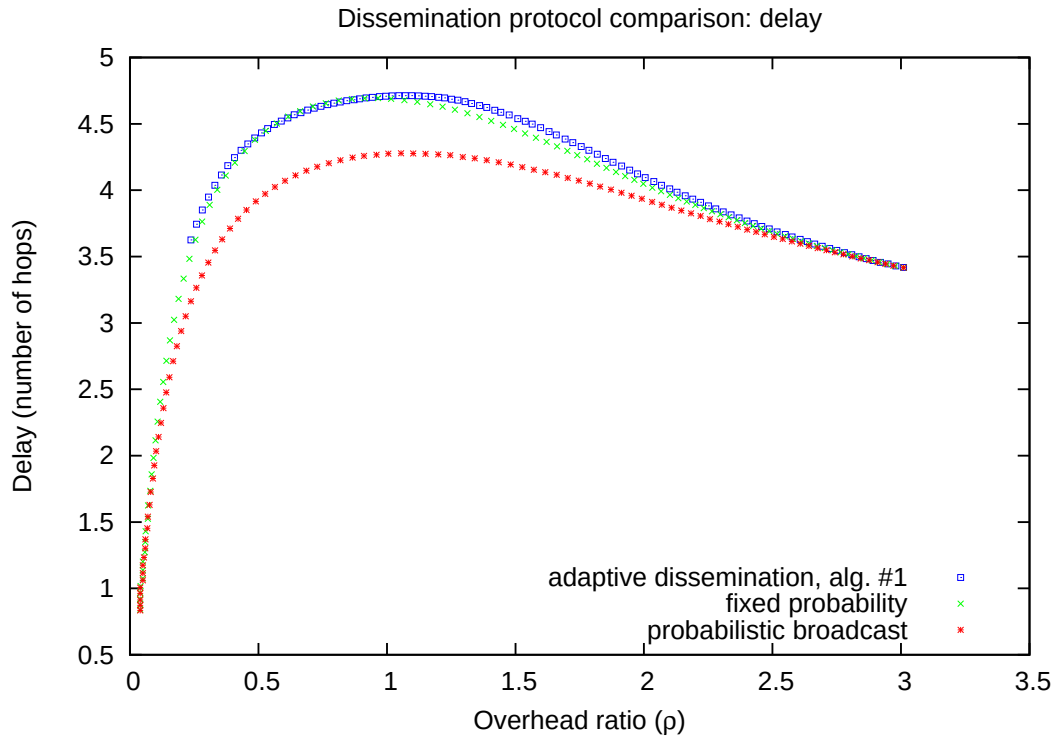

Figure 3. Dissemination protocols, delay

Table 1. Adaptive protocols, parameters in different setups

Algorithm	monitoringPeriod	σ	δ	α
#1	100	0.2	300	$1/3$
#2	50	0.5	1000	$3/4$
#3	50	0.7	10000	1

Given the results obtained above, in the following of this performance evaluation, we will consider only three different flavors of the adaptive gossip algorithms (as described in Section 3). Furthermore, for the reasons described few lines above, results will be shown only for $\rho \geq 1$.

As expected the more complex adaptive algorithm (i.e. *Algorithm #3: Stimuli Associated to Generators and Receivers*) is the clear winner (see Figure 4): especially in low overhead cases it is able to provide a much greater degree of **coverage** with respect to the other dissemination protocols (both adaptive or not). The results obtained by *Algorithm #2: Stimuli Associated to Generators* are very similar to those obtained by *Algorithm #1*. The difference is that, in this setup the *Algorithm #1* is unable to obtain "overhead" outcomes that are less than 1.37, this is due to its implementation and tuning. In Table 1 are reported all the setup parameters used to tune the algorithms considered in this part of the performance analysis.

Also under the **delay** viewpoint (see Figure 5), the *Algorithm #3* is almost always better than the counterparts. Only in case of a very low overhead (i.e. ρ that is a little higher than 1) the obtained delay is worse than *Algorithm #1*. This due to the characteristics of this dissemination algorithm: it starts with a very low dissemination probability for each couple (generators, receivers) and only when a reception rate that is too low is found then a probability increase is requested. What happens, due to overhead constraint, is that this is not sufficient to obtain a dissemination that is both efficient in terms of coverage and fast in terms of delay. Furthermore,

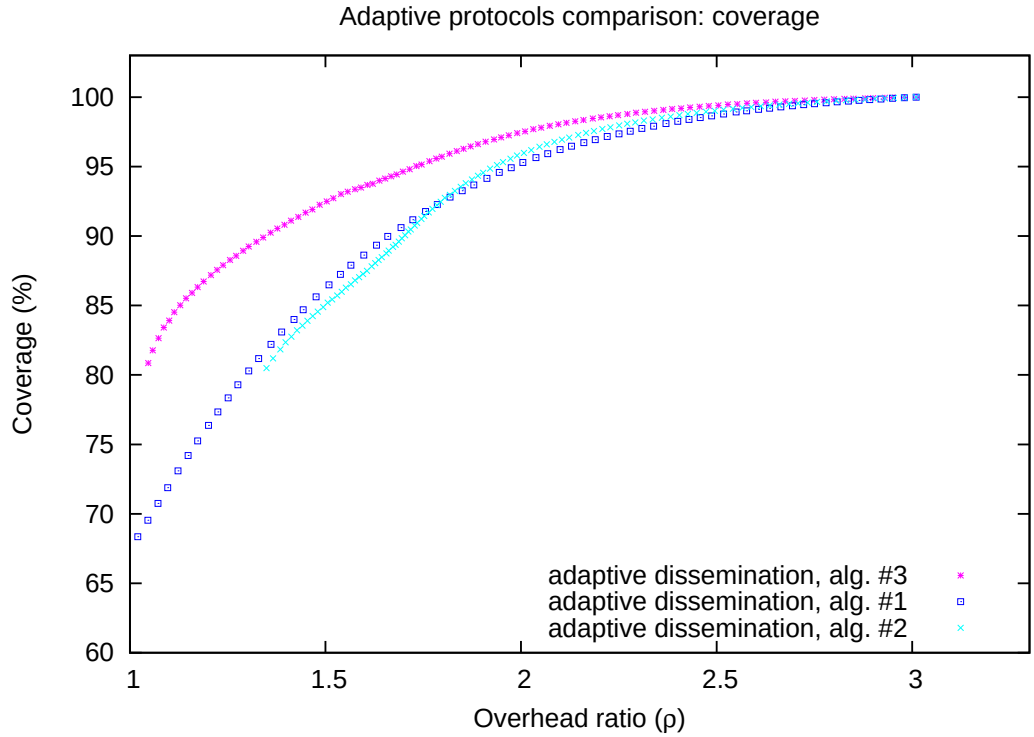

Figure 4. Adaptive protocols, coverage

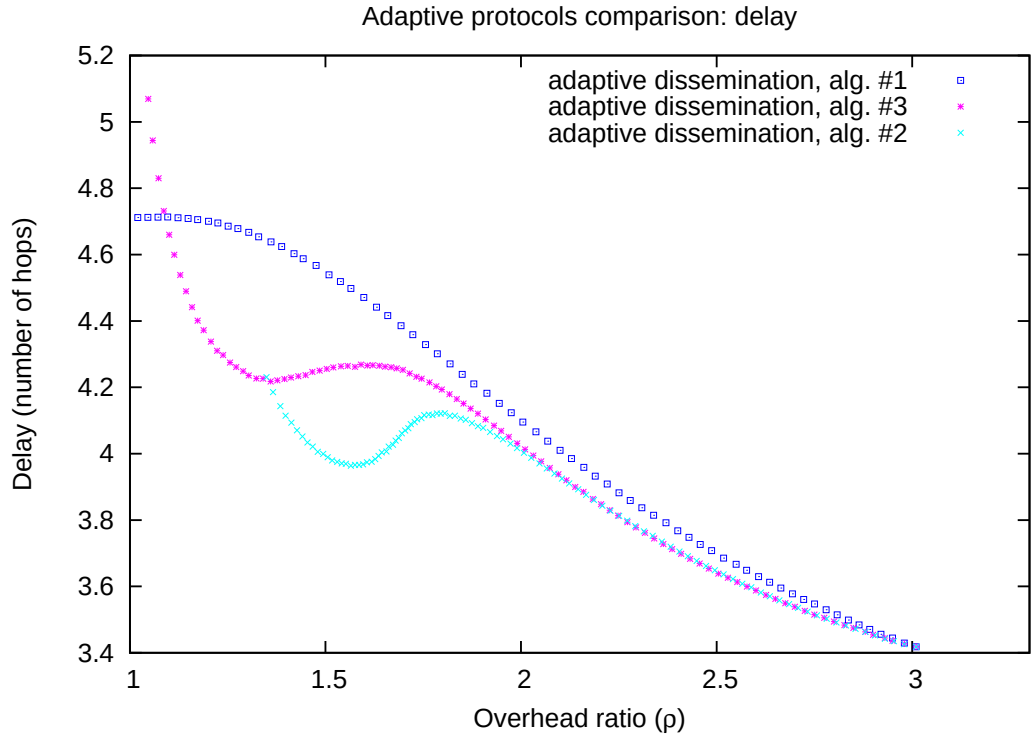

Figure 5. Adaptive protocols, average delay

Table 2. Algorithm #3, parameters in different setups

setup	monitoringPeriod	σ	δ	α
#1	50	0.5	1000	1
#2	50	0.5	5000	1
#3	50	0.5	1000	$3/4$
#4	50	0.7	10000	1
#5	30	0.25	10000	1
#6	30	0.25	10000	$1/2$

if ρ is in the range $[1.3, 1.7]$ then the behavior of this algorithm (in terms of **delay**) is quite odd but still better than *Algorithm #1*. The finding of a detailed and specific motivation for this behavior is a very hard task, given the complexity of the protocol and the many details to be considered. In the following of this Section it will be seen that this behavior is a characteristics of this adaptive dissemination protocol and that is not dependent on the parameters used to setup this specific test case. A deeper investigation is left as future work.

Finally, a few words about the *Algorithm #2*: in terms of **coverage** it is not outstanding but in terms of **delay** it shown some interesting aspects. Also if its **coverage** is very similar to *Algorithm #1*, its **delay** is much lower for almost all ρ values. In the comparison with *Algorithm #3*, its **coverage** is a lot worse but in some parts the experienced **delay** is quite good. This could be very interesting for user level applications that can tolerate some packet loss but require a very timely delivery.

In the last part of this simulation-based assessment, the focus will be on the dissemination *Algorithm #3*: many different setups of the algorithm will be compared. In this case, the aim is to demonstrate that, if necessary, the protocol can be finely tuned but in general it is pretty stable. In P2P networks, given their nature, it is quite hard to obtain at runtime all the necessary information to build accurate mechanisms for the fine tuning of the protocols and algorithms that are used for dissemination or other specific tasks. For this reason, the "stability" of the protocol in different conditions is a quite interesting property.

The parameters used for all setups are in Table 2, as usual in Figure 6 and 7 are shown the average **coverage** and **delay** that are obtained by this protocol with respect to the different overheads (i.e. ρ values). In Figures 6 and 7 it can be seen that the different setups can modify the obtained results but always in a limited manner. In other words, the main behavior of the adaptive algorithm is not altered in deep.

5 Conclusions

In this work we described some new adaptive gossip protocols for data dissemination in unstructured networks. In particular, we have proposed the usage of such kind of protocols in systems (such as the multi video conference) in which is possible to expect that each node in the network will follow a some predictable generation rate for updates. This simple property, that can be found in many real-word systems can be exploited to enhance the gossip protocol performance.

Simulation experiments showed that adaptive gossip protocols are quite promising in this scenario. After defining how to compare the outcomes of different dissemination protocols and proposing some new adaptive algorithms, we have compared their results with a couple of well-known dissemination strategies (i.e. *Probabilistic Broadcast* and *Fixed Probability*). The results show that simple adaptive strategies are better than non-adaptive ones both in terms of **coverage** and **delay** of data dissemination. The other main contribution of this work is LUNES, a new freely available simulator that is specifically aimed to the evaluation of complex protocols on top of network graphs. LUNES is a parallel and distributed simulator providing the necessary scalability

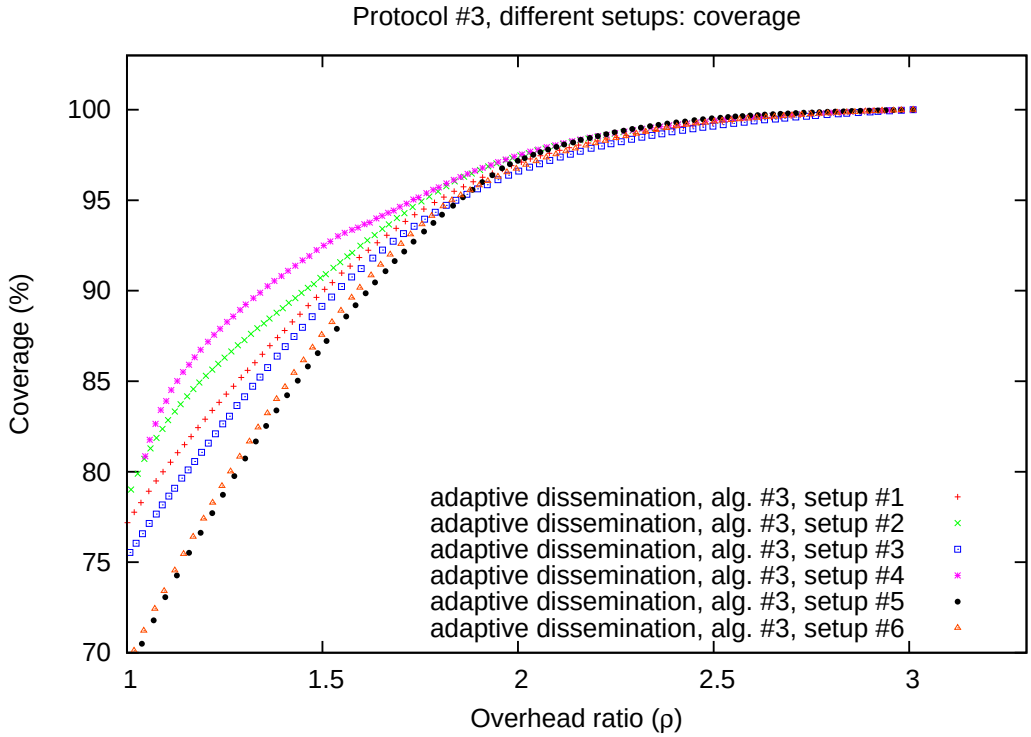

Figure 6. Algorithm #3, different setups, coverage

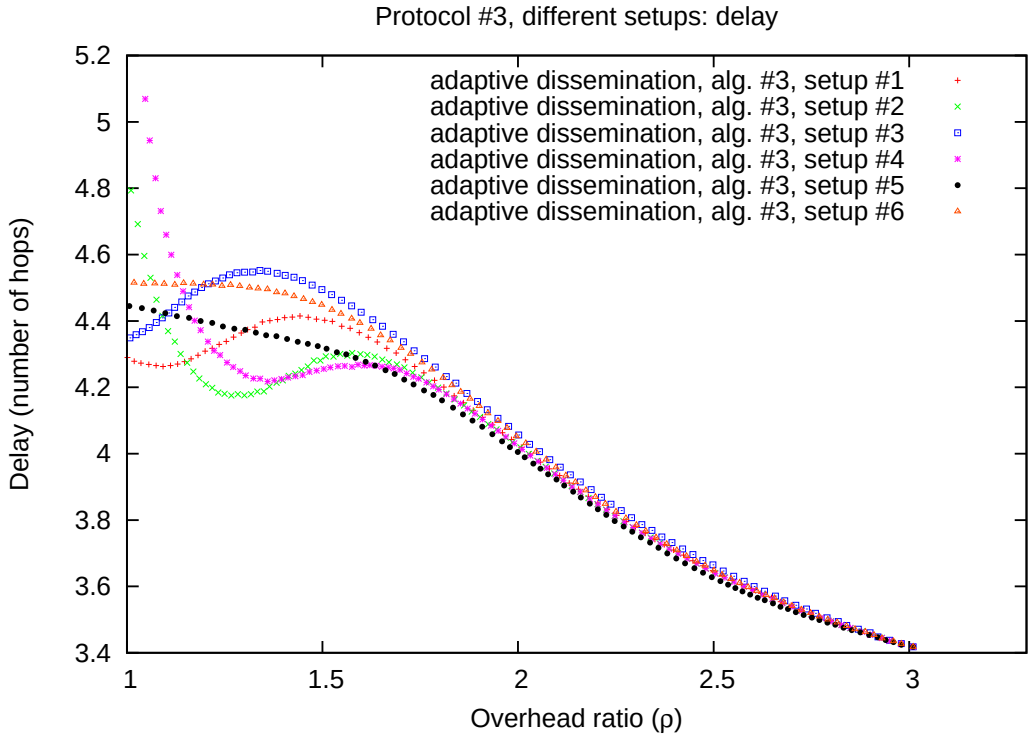

Figure 7. Algorithm #3, different setups, average delay

for the evaluation of very detailed and large-scale scenarios.

The obtained results are promising, and deserve further investigations. Specifically, in this technical report all the adaptive algorithms are based on a "positive" stimulus. Such a stimulus corresponds to a fixed increment of the dissemination probability, that does not depend on the performances of the dissemination algorithm and is activated only when they go below a certain threshold. Probably, a more granular approach that tunes the magnitude of the stimulus based on the performances, would permit a more fine control of the dissemination probabilities and consequently a lower dissemination overhead. As a further variation, we plan to implement a more complex adaptive gossiping scheme that will be able to use both positive and negative stimuli. In this way, each peer will be able to fine tune the dissemination probability of each of its neighbors. We also plan to evaluate the proposed adaptive protocols in graphs generated with different properties (e.g. scale-free and small-world networks) and in presence of a larger amount of nodes. Finally, we will investigate more in detail what is the impact of specific parameters such as the Time-To-Live (TTL) and the cache size that so far has not been addressed in sufficient detail.

References

[1] Parallel And Distributed Simulation (PADS) Research Group. `http://pads.cs.unibo.it`, 2011.

[2] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1-4):69–77, Jun 2000.

[3] G. D'Angelo and M. Bracuto. Distributed simulation of large-scale and detailed models. *International Journal of Simulation and Process Modelling (IJSPM)*, 5(2):120–131, 2009.

[4] G. D'Angelo and S. Ferretti. Simulation of scale-free networks. In *Simutools '09: Proc. of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–10, ICST, Brussels, Belgium, 2009. ICST.

[5] S. Ferretti. Modeling Faulty, Unstructured P2P Overlays. In *Proc. of the 19th International Conference on Computer Communications and Networks (ICCCN 2010)*. IEEE, August 2010.

[6] S. Ferretti and G. D'Angelo. Multiplayer online games over scale-free networks: a viable solution? In *Proc. of the International Workshop on DIstributed SImulation and Online gaming (DISIO 2010) - Conference on Simulation Tools and Techniques (SIMUTools 2010)*. ICST, 2010.

[7] G. H. L. Fletcher and H. A. Sheth. Unstructured peer-to-peer networks: Topological properties and search performance. In *3rd International Conference on Autonomous Agents and MUlti-Agent Systems*, pages 14–27. Springer, 2004.

[8] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Softw. Pract. Exper.*, 30:1203–1233, September 2000.

[9] B. Garbinato, D. Rochat, and M. Tomassini. Impact of scale-free topologies on gossiping in ad hoc networks. In *NCA*, pages 269–272. IEEE Computer Society, 2007.

[10] H. Guclu and M. Yuksel. Limited scale-free overlay topologies for unstructured peer-to-peer networks. *IEEE Trans. Parallel Distrib. Syst.*, 20(5):667–679, 2009.

[11] M. Macedonia and M. Zyda. A taxonomy for networked virtual environments. *Multimedia, IEEE*, 4(1):48 –56, 1997.

[12] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[13] T. Turletti and C. Huitema. Videoconferencing on the internet. *IEEE/ACM Trans. Netw.*, 4:340–351, June 1996.

[14] S. Verma and W. T. Ooi. Controlling gossip protocol infection pattern using adaptive fanout. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 665–674, Washington, DC, USA, 2005. IEEE Computer Society.