Multiplayer Online Games over Scale-Free Networks: a Viable Solution?

Stefano Ferretti
Department of Computer Science,
University of Bologna
Mura Anteo Zamboni 7, 40127
Bologna, Italy
sferrett@cs.unibo.it

Gabriele D'Angelo
Department of Computer Science,
University of Bologna
Mura Anteo Zamboni 7, 40127
Bologna, Italy
gdangelo@cs.unibo.it

ABSTRACT

In this paper we discuss the viability of deploying Multiplayer Online Games (MOGs) over scale-free networks. We employ a general peer-to-peer overlay network; nodes have a number of neighbors which follows a power law distribution, $p_k \sim k^{-\alpha}$, the usual degree distribution that characterizes scale-free nets. Game events generated by nodes during the game evolution are disseminated through the network, based on some (push) gossip protocols run over the created overlay. We experiment with different gossip protocols. Results demonstrate that the employed gossip protocol may greatly influence the ability of disseminating the game data through the scale-free network. In particular, when gossip is performed using a small dissemination probability, a non-negligible percentage of the network is not able to receive the message. This implies that not all players might be able to perceive the game event. Hence, parameters of gossip protocols must be properly tuned to guarantee a full network coverage. Concurrently, it is shown that, due to their low diameter, the use of scale-free networks allows to disseminate game events in very few steps. This could ensure a high level of responsiveness on the dissemination of game events, which is the main objective to pursue when dealing with MOGs.

Categories and Subject Descriptors

K.8.0 [**Computing Milieux**]: PERSONAL COMPUTING—*General, Games*; C.2.4 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Experimentation, Performance

Keywords

Multiplayer Online Games, Scale-Free Networks, Simulation, Parallel and Distributed Simulation, Performance Evaluation

DISIO 2010 March 15, Torremolinos, Malaga, Spain.

1. INTRODUCTION

Multiplayer Online Games (MOGs) are characterized by very demanding requirements, in terms of responsiveness and scalability. The game evolution must be perceived at players as fluent as if the game is played in the real life, and responsiveness should not degrade when the number of users grows [24]. Of course, several technical issues make these requirements very difficult to be achieved, since networks introduce delay latencies during the transmission of game updates that could slow down the game advancements. Moreover, the higher the number of participants, the higher the time to collect all messages from players and process them to correctly compute updates of the game state [14].

Several architectures have been proposed and employed in practice to support MOGs. The simpler (and mostly utilized) solution is the client/server architecture. Here, a single server is in charge of receiving the game events generated by clients (i.e. hosts where players are playing the game), computing the game state and periodically notifying the novel game state to all these players. The graph corresponding to this architecture is a classic star network, the server being a possible bottleneck of the system [17, 22]. To augment the scalability of these approaches, several auxiliary solutions have been devised. For instance, mechanisms allow to partition the game state and assign to different servers different portions of the game state, hence reducing the number of players connected to the same server [3, 23, 25]. Alternatively, mirrored server architectures may be employed which replicate the game state at different servers; then, clients may connect to different servers (which are kept synchronized) so as to reduce the workload at each server for managing the interactions with clients [10, 20, 22].

On the other hand, peer-to-peer architectures may be exploited, where peers locally manage their own copy of the game state, without the intervention of centralized servers. In this case, peers should be organized in some overlay, and then the dissemination of game events is performed by passing messages through the overlay [1, 13, 19, 25]. Of course, identifying the best overlay is a crucial aspect for the success of the game management. A typical approach is that of employing some structured peer-to-peer network, according to which peers are organized and already know how the communication among nodes should flow, once a game event has been generated. Conversely, to the best of our knowledge,

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

only few proposals considered unstructured peer-to-peer architectures, which employ probabilistic approaches to disseminate data [19, 25]. Of course, a structured network provides an (almost) constant number of hops to perform a path between two nodes; conversely, only statistical measures can be evaluated in unstructured nets.

Even in presence of such uncertainty of complete network coverage within a fixed time deadline (which is indeed a primary constraint in MOGs), there are some features that make unstructured networks interesting for MOGs. In fact, these mentioned approaches do not require a costly nodes configuration at the beginning of the application, that should be managed and maintained during the game evolution. In MOGs, players may join and leave the gaming network quite dynamically. Hence, avoiding a reconfiguration of the network each time a player joins or leaves the game could be quite proficient. It is thus interesting to evaluate how gossiping approaches might perform over some complex network overlay, and if their behavior provides viable performances for being utilized in MOGs. Of course, such chosen method must guarantee that: i) each peer receives (almost) all game data disseminated over the network, in order to guarantee that the game state evolution may be consistently perceived at each node participating to the distributed game; ii) game updates are received in short time intervals, so as to make sure the game evolves quite fluently.

In this context, scale-free networks have recently gained interest in the field of distributed systems [6, 8, 12, 11, 16]. These networks possess the distinctive feature of having nodes which tend to link to each other, following a degree distribution (i.e. number of neighbors) that can be well approximated by a power law function. Therefore, while the majority of nodes tends to have an average number of neighbors that is relatively low, a non-negligible percentage of nodes exists which has a high degree, i.e. a high number of neighbors, quite above the average node degree. These *hubs* play a fundamental role in the network, since they are those which maintain the network connected, and contribute to keep a small diameter of the network, thus allowing to propagate information in a low number of hops. Such features of scale-free networks represent interesting aspects in the context of MOGs, and suggest that they may represent the solution for a scalable, responsive, server-less solution. Indeed, hubs in the network could be thought as a sort of super-peers, able to manage a high number of connections in the distributed game system.

Motivated by this consideration, in this work we discuss and evaluate the viability of adopting a scale-free network as the basis for building an efficient peer-to-peer gaming architecture. To assess the ability of these networks to disseminate game events generated at peers during the game evolution, we utilize different push gossip protocols [11, 18]. Specifically, we run three gossip algorithms that manage game events on top of the scale-free networks. (Then, game events received by peers are to be locally processed based on some suitable synchronization protocol for peer-to-peer MOGs, e.g. [10, 13, 20].)

To perform simulations, we employ PaScaS (Parallel and distributed Scale-free Network Simulator), a distributed simulator we built that is able to represent large scale-free networks, and manage them in a responsive way [11]. Simulation results show that while the adoption of an underlying scale-free network represents a viable solution to disseminate game events in peer-to-peer MOGs, the employed gossip protocols may have very different outcomes in terms of ability of delivering game events to all peers. In fact, when the probability of gossiping is kept low, and we employ the classic approach of constraining nodes to gossip a message at most once, then an important percentage of nodes in the game system may not receive all game events. This means that those peers miss some information on the evolution of the game; as a consequence, inconsistent game state may be calculated, depending on the non-received game events [10, 13, 20, 22]. This outcome is explained by the fact that we use the preferential attachment method to construct the network [5]. It has been recognized that such types of scale-free nets have a clustering coefficient which depends on the size of the network, $\sim N^{-0.75}$, if N is the size of the network [4]. In our experiments, we employed networks with a number of nodes up to 500 (that is quite reasonable for gaming infrastructures), whose clustering results as quite low. These tree-like structures hence require that the probability of gossiping a message from a node to its neighbor kept as quite high, otherwise it will become difficult that these nodes will receive such messages from other peers (also because we limit the dissemination by imposing a time-to-live limit, to not overwhelm the network). By tuning parameters of these push gossip approaches, such as the probability of dissemination, the message time-to-live, the nodes' cache size, it is possible to cover a majority of nodes in the network.[1]

Of course, the work is meant to support MOGs. However, the same considerations and final remarks can be extended to those applications that present the following concurrent features: i) a high number of nodes that may join and leave the system while the application is on-going (churning); ii) the need for a rapid dissemination of events produced at different, distributed sources (responsiveness); iii) the need for disseminating generated messages to all the participants present in the system (consistency).

The remainder of this paper is organized as follows. Section 2 reviews the main principles at the basis of scale-free networks and delineates those characteristics that suggest these nets could be adopted for supporting peer-to-peer MOGs. The system architecture follows, based on mentioned considerations. Section 3 outlines some gossip algorithms that could be employed on top of the network to disseminate game events among peers. In Section 4 we report on an extensive simulation we performed to assess the proposed approach. We outline the main characteristics of PaScaS, the distributed simulator we built to simulate scale-free networks, and discuss the obtained results when the mentioned gossip protocols are utilized over these simulated networks. Finally, in Section 5 we provide some final remarks.

2. SCALE-FREE NETWORKS IN MOGS

In this section we review some main principles at the basis of scale-free networks, and discuss why these may have a good impact for the support of MOGs.

2.1 Notation

We model the network as a set of distributed nodes cor-

[1]Then, the use of additional pull-based gossip approaches to obtain missing information may solve possible inconsistencies.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

responding to peers participating in a distributed online game. The topology of the network is defined as a graph $G = (\Pi, L)$, where $\Pi = \{n_1, n_2, \ldots\}, |\Pi| = N$, is the set of peer nodes, and L denotes the set of edges among peers in the MOG. Two peers n_i, n_j are neighbors if an edge $l_{ij} \in L$ exists connecting the two peers in G. The set of neighbors of n_i is denoted with Π_i.

The degree of a node is the number of neighbors of that node. The probability that a node has k neighbors, i.e. its degree is equal to k, is denoted as p_k. The average degree in a network is denoted with $\langle k \rangle$.

2.2 Scale-Free Networks

Scale-free networks are gaining more and more attention in the research community [6, 11, 21]. These networks are characterized by the fact that their nodes have a degree k which is distributed according to a power law distribution, i.e. $p_k \sim k^{-\alpha}$, for some constant α. Sometimes, a cutoff is introduced to force that no node may have a degree higher than a given threshold value k_{max}. This may be useful when dealing with peer-to-peer MOGs, since in certain cases it might be difficult to assume that some peer is in charge of managing a huge amount of concurrent connections with other peers, that would introduce excessive computational and communication burdens for that node.

Basically, the structure of such kind of networks is characterized by a notable (i.e. non negligible) presence of nodes, usually referred as *hubs*, which have a number of edges really higher than the average degree in the network. These peers play a fundamental role in the network, as they contribute in maintaining a low diameter of the network (i.e. the maximum distance between two nodes in the net). In particular, it has been mathematically proven that when $2 < \alpha < 3$, the diameter of the network $d \sim \ln \ln N$, smaller even than small world networks, which remains almost constant while the network is growing [9]. This should have an important consequence on the responsiveness provided to the game, since in theory very few hops are required to disseminate a message, independently of the size of the network (i.e. game participants). Massive MOGs could hence take some benefit from this feature.

The theoretical model of these networks is probabilistic. As a consequence, in general it is not possible to assume that the whole network is connected, especially when node failures are possible in the system. This is certainly a problem if one wants to support a game application over such a kind of graphs, where links between peers are created based on a probabilistic nature. However, theoretical and empirical results have demonstrated that in scale-free networks a major fraction of the set of nodes remains connected (usually of the order of $\Theta(N)$), even when random node faults arise in the network. In particular, it has been shown that these networks are quite resilient to random node faults since the presence of hubs guarantees that the network remains connected. Indeed, the majority of nodes are those with smaller degrees; thus, it is more likely that these ones will fail/leave the network, while the probability that all hubs are eliminated (during a short time interval) is almost negligible.

This consideration plays a fundamental role in our conjecture of exploiting scale-free nets for MOGs, since it is not unusual that players leave the game while the game session is still ongoing. They are in the gaming network to play and they may decide to leave the system at their best convenience. Hence, mechanisms are needed to guarantee that no partitions occur among game participants. Moreover, this is a problem that typically arises in all peer-to-peer overlays [27]. In fact, when the peer-to-peer overlay is managed using some structured network, e.g. a tree or a mesh, a node failure would imply a network reconfiguration. Instead, due to the mentioned theoretical result, we could simply assume that the scale-free network remains connected after a node departure; then, some (lazy) control mechanism could be activated to manage (and repair) possible network disconnections.

2.3 The Preferential Attachment

A practical method to build a scale-free network was introduced by Barabási and Albert in [5]. The scheme starts with a number of nodes N_0. At each time step, a new node is added to the network, with initial degree m. For each edge of the new node, a neighbor is selected and a link between the two nodes is created. The important characteristic of such construction method is that the neighbor node is selected with a probability proportional to the degree of that vertex (*preferential attachment*). In other words, the higher the degree of a node the more likely it will be selected as a neighbor of the newly added node. Such approach models the *rich get richer* phenomenon, arising when the amount an entity gets in time, goes up with the amount it already has [21]. Indeed, the preferential attachment is responsible for the generation of a power law node degree distribution. The average degree of a given node $\langle k \rangle \simeq 2m$, for large networks (each link counts for two node edges). This is due to the fact that each new node is added at each time step, with new m links. Hence, at time t the network has $N_0 + t$ nodes, with mt links. N_0 can be considered as a negligible term for large nets (or large times).

Such construction method creates a scale-free network, whose degree distribution follows a power law with a parameter $\alpha \simeq 3$. The diameter of such network, with this exponent, can be approximated as $\sim \frac{\ln N}{\ln \ln N}$ [9]. It has been observed that networks generated using such an approach exhibit a coefficient clustering, i.e. a coefficient measuring how much nodes tend to cluster together, which depends on the network size as $\sim N^{-0.75}$ [4]. This means that the larger the network the lower the clustering coefficient. This is an important factor to consider, when one tries to implement some gossip approach on top of the network. In fact, assume that a low probability of gossip is employed. This means that it is possible that once a node n_i has a message, it does not forward it to a given neighbor n_j; in a network with a low clustering, the probability that one of the nodes that receive the message is also neighbor of n_j is low; hence, it is possible that n_j does not receive such message from n_i's neighborhood Π_i.

3. GOSSIP PROTOCOLS FOR GAME EVENT DISSEMINATION

In this section, we describe the protocols we consider to gossip game events generated at peers, over a scale-free network exploited as the overlay for disseminating game data. They are all push-based approaches, meaning that nodes locally decide to forward a given game event to a set of their neighbors, based on independent and local decisions, without any coordination with these receiving nodes. The im-

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

plementation of these protocols is simple. They differ from pull-based protocols, which instead require that nodes wishing receive some content randomly contact some of their neighbors to assess if these have the desired content. We try to exploit push gossip approaches for two reasons. First, the aim is to distribute the whole event trace to all game participants. Hence, the idea is to disseminate each message as much as possible, without waiting for explicit requests by some receiving peers. (At the same time, it should be avoided that a message reaches a given node several times, to avoid network congestion.) Second, each step in a push gossip approach is generally faster than a pull approach, since in the former case a single message is necessary, while in the latter a message exchange is needed to let the receiver know which game events it can ask (and then ask and receive them).

Once a game event is generated at the application (game) level, it is passed to the module which performs the gossip of the message (see Algorithm 1). The message including the game event is created and then gossiped through the net, using a GOSSIP() procedure (line 4 of the algorithm). The message is also inserted in a cache (line 3).

Algorithm 1 Generation of a Game Event

```
1: function GENERATE(event)
2: msg ← CREATEMESSAGE(event)
3: CACHE(msg)
4: GOSSIP(msg)
```

Algorithm 2 Reception of a Message

```
1: function RECEIVE(msg)
2: if (NOTCACHED(msg) ∧ msg.ttl > 0) then
3:    CACHE(msg)
4:    msg.ttl ← msg.ttl − 1
5:    GOSSIP(msg)
6: end if
```

Upon reception of a given message (see Algorithm 2), the receiving node forwards the message to its neighbors using the gossiping protocol by calling the GOSSIP() function (line 5 in the algorithm). This is accomplished only if the message is not already in the node's cache. The idea is that if the message is in cache, it has already been gossiped; hence, the node has nothing to do with the message *msg* (line 2). Conversely, *msg* is gossiped and cached (line 3 of Algorithm 2). Note that, when gossiped, we avoid that a message is sent through the link it has been previously received. Based on the protocol, the game event has only one chance to be gossiped by the considered node; as we will show, this has an important influence on the ability/difficulty shown by gossip protocols to disseminate game events. Needless to say, due to the possible memory constraints of a node, the cache is limited in size (*cache.size*). Hence, upon insertion of a message in the cache, a control is performed on the cache; if it is full, an old message is removed. We do not provide here the complete description of the CACHE() procedure; we simply implemented an aging policy to free memory in cache.

A time-to-live parameter is inserted within each message (i.e. *msg.ttl* in the algorithm), in order to avoid that old messages are indefinitely propagated among peers, due to the limited size of the cache which cannot contain the complete list of messages forwarded in the past. Such *ttl* is progressively decreased (line 4) until it reaches a 0 value; in this case the message is not forwarded (see the second part of the condition in line 2). Needless to say, it is important to assess if the particular choice of the time-to-live allows to disseminate the game event among all players in the network.

We consider three different algorithms which implement the GOSSIP() procedure. These are shown in Algorithms 3, 4 and 5. All algorithms require the definition and initialization of a parameter at each peer, defined through the INITIALIZATION() procedure reported at the beginning of these algorithms.

3.1 Gossip #1: Fixed Probability of Dissemination

The first gossip protocol is shown in Algorithm 3. Based on it, the node n_i executing the protocol randomly selects those edges through which the message *msg* must be propagated [16, 26]. Specifically, all n_i's neighbors (i.e. Π_i) are considered and a threshold value $v \leq 1$ is maintained, which determines the probability that *msg* is gossiped to the neighbor (when $v = 1$ we obtain a flooding algorithm).

At each step the game event is propagated from n_i to $v|\Pi_i|$ other nodes. (In a scale-free network, on average a given node will propagate the message to $v\langle k\rangle \simeq 2vm$ nodes.) Hubs will send a higher number of messages to their neighbors, with respect to others. This is in perfect accordance with the nature of scale free networks, since each node contributes to disseminate the game message in accordance with its degree. This also means that the work (in terms of computation and communication) performed at hubs is higher than at other nodes.

Algorithm 3 Gossip: Fixed Prob. of Dissemination (at n_i)

```
1: function INITIALIZATION()
2: v ← CHOOSEPROBABILITY()
3:
4: function GOSSIP(msg)
5: for all n_j ∈ Π_i do
6:    if RANDOM() < v then
7:       SEND(msg,n_j)
8:    end if
9: end for
```

It is clear that the probability of dissemination v plays a fundamental role here, since after the gossip n_i will never reconsider *msg* for dissemination. For instance, let say a node with a single neighbor generates a novel game event. It passes such event to its single neighbor that decides not to disseminate the game event. This means that no further players will be enabled to receive and process the game event corresponding to *msg*. Moreover, when such an approach is employed over a tree-like structure, the gossip may exclude some branches that would never receive the message. In these cases, a gossip probability near 1 should be exploited (i.e. flooding).

3.2 Gossip #2: Fixed Fanout

The second gossip scheme we consider is reported in Algorithm 4. In this case, a message is sent to a fixed number of nodes (i.e. a fixed fanout is exploited), selected at random among the n_i's neighbors, Π_i [16]. This means that the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

higher the degree of n_i, the more unlikely a n_i's neighbor will receive a gossip message at each step.

A constant *fanout* is chosen and shared among all nodes in the network, during the INITIALIZATION() procedure. When a message *msg* is to be gossiped, n_i selects a number of neighbors equal to the *fanout*. A list of nodes (*toSend* in the algorithm) is filled up by iteratively selecting a node among the neighbors not already in the list (see lines 8-12).[2] If the number of neighbors is lower than the selected fanout, the message is sent to all the n_i's neighbors, see lines 5-6).

Algorithm 4 Gossip: Fixed Fanout (at n_i)

```
 1: function INITIALIZATION()
 2: fanout ← RETRIEVESHAREDFANOUT()
 3:
 4: function GOSSIP(msg)
 5: if fanout ≥|Π_i| then
 6:    toSend ← Π_i
 7: else
 8:    toSend ← ∅
 9:    for i = 1 to fanout do
10:       select n_j ∈ Π_i ∩ toSend, i ≠ j
11:       toSend ← toSend ∪ n_j
12:    end for
13: end if
14: for all n_j ∈ toSend do
15:    SEND(msg,n_j)
16: end for
```

As mentioned for gossip #1, *fixed probability*, also this approach may present some inefficiencies regarding the full coverage of the network, depending on the type of the network.

3.3 Gossip #3: Probabilistic Broadcast

The third distribution protocol we consider is a probabilistic broadcast scheme (see Algorithm 5). Once the GOSSIP() procedure is called, the possible receivers of the message are computed, i.e. all the node's neighbors except the node which has sent the message *msg* to be gossiped (in case *msg* has been generated at the considered node, NEIGHBORTHATSENT(msg) should return a null node, line 5). Then, if the message has been locally generated at the node and *msg* still needs to be spread to the network (we assume this check is performed in FIRSTTRANSMISSION(), line 6), *msg* is sent to all node's neighbors (lines 7-9).

Conversely, if *msg* has been received from someone else, the node decides to forward *msg* with a certain probability p_b (defined at the beginning of the protocol, line 6). In the positive case, the message is sent to all node's neighbors (excepted the node that sent *msg* to it, line 5).

4. SIMULATION ASSESSMENT

This section is devoted to present the simulation study we performed to assess the efficacy of exploiting gossip protocols over scale-free nets, for the support of MOGs.

4.1 PaScaS

The simulations have been performed by exploiting PaScaS (Parallel and distributed Scale-free Network Simulator),

[2] $\overline{toSend}$ represents the complement of the set *toSend*.

Algorithm 5 Probabilistic Broadcast

```
 1: function INITIALIZATION()
 2: p_b ← PROBABILITYBROADCAST()
 3:
 4: function GOSSIP(msg)
 5: recvs = Π_i \ NEIGHBORTHATSENT(msg)
 6: if (RANDOM() < p_b ∨ FIRSTTRANSMISSION()) then
 7:    for all n_j ∈ recvs do
 8:       SEND(msg,n_j)
 9:    end for
10: end if
```

a tool specifically designed for the modeling of scale-free networks [11]. It offers the possibility to implement both sequential and parallel/distributed simulations, using the ARTÌS simulation middleware [2] and GAIA adaptive framework [7, 15]. Specifically for this simulation study, some new features have been developed and integrated in the tool, to efficiently analyze the simulation results and compute the metrics of interest. PaScaS is going to be made freely available as part of the ARTÌS software distribution [2].

For the creation of the scale-free network, PaScaS runs the preferential attachment method proposed by Barabási and Albert in [5], starting with an initial number of nodes $N_0 = 1$; when not differently stated, once generated each node attaches to 2 nodes. Once built the network, PaScaS allows to simulate communications among nodes in the network. We hence tested different gossip communication protocols among nodes of the network, being considered here as peers in a peer-to-peer architecture for MOGs. Specifically, during the simulations, gossip protocols were run up to a simulation time s^* needed to generate a given, predetermined amount of game events at different nodes. After that time, the simulation continued for other *ttl* (time-to-live) steps, while no message generation was allowed during these final simulation steps. This way, we gave chance to game events generated at simulation time s^* to be distributed till reaching their maximum number of hops. Indeed, no gossip procedure would have been possible after $s^* + ttl$.

In the following performance evaluation, the simulation time was set to 1000 timesteps and each node in the network was generating new events following an exponential distribution with mean 50. In the simulated scenarios, we varied the number of nodes present in the network and for each configuration 9 different runs were executed. The reported outcomes are obtained by averaging these results.

4.2 Performance Metrics

We analyze two specific metrics, that clearly characterize the viability of adopting gossip protocols over scale-free networks for supporting MOGs. The first metric is the ability of the tested gossip protocols to cover the whole scale-free network. We already stressed the fact that in general it is important that all nodes involved in the game receive the whole list of game events, so that they may consistently update their local game state as the game evolves in time.[3] It is indeed true that some game data may become obsolete in

[3] We remind that the considered model is a peer-to-peer architecture where each node locally manages its own copy of the game state which is updated using some synchronization algorithm, based on game events received during the game evolution.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

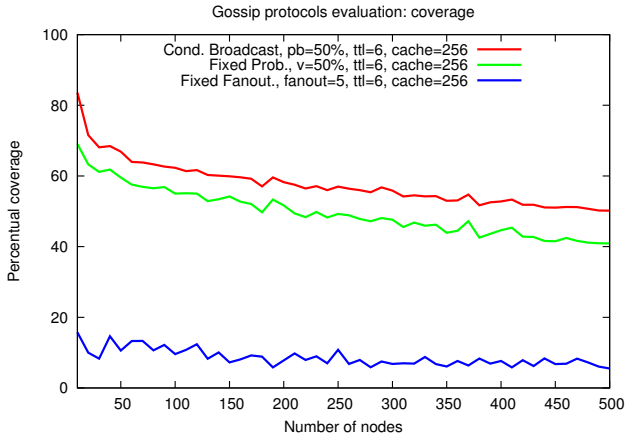

Figure 1: Network coverage using the gossip protocols; $v = 0.5, fanout = 5, p_b = 0.5$**.**

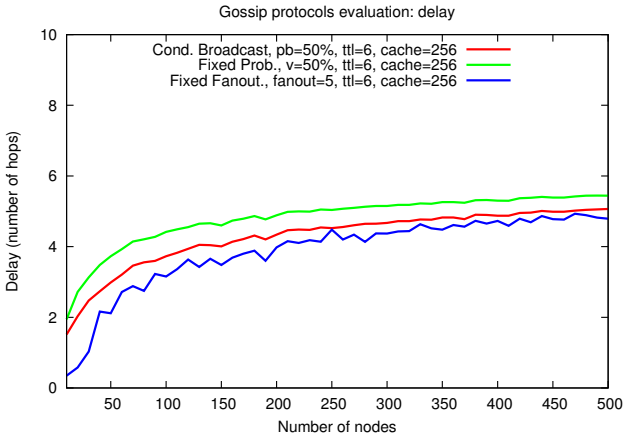

Figure 2: Network delay using the gossip protocols; $v = 0.5, fanout = 5, p_b = 0.5$**.**

time, and hence such obsolete information can be dropped [13, 22], but the underlying dissemination protocol must be nevertheless able to reliably distribute a given message to all game participants, so that "important" game information is spread through the whole network.

The other metric to consider is of course how fast messages are disseminated through the network. This in fact regulates the pace of advancements of the game. Since the dissemination protocols are executed in discrete steps, we measured the number of hops needed (on average) to distribute a game event to all nodes in the architecture. This is a general measure for the assessing the latency introduced by the network.

4.3 Results

Figure 1 reports the percentage of nodes that receive, on average, all game events generated during the game evolution. For these nodes, no holes are present in the game event list. The probability of gossiping where set as follows. As to the scheme *fixed probability*, $v = 0.5$; as to *fixed fanout*, $fanout = 5$; as to *probabilistic broadcast*, $p_b = 0.5$. A first important consideration is that, with these settings for the gossip probability, the protocols are not able to fully distribute to the whole network all the game events. Rather, an important percentage of nodes has holes in the event list. It is possible to observe that *fixed fanout* results as quite ineffective, since less than the 20% of nodes is able to receive the whole event trace. Moreover, the other two schemes *probabilistic broadcast* and *fixed probability* seem to behave differently as the number of nodes varies; hence, under these settings they do not seem to scale with the network growth.

Figure 2 reports the average number of steps required to disseminate a given message, when varying the number of nodes. Results demonstrate that game events may be quickly gossiped to the network. Indeed, a very low number of steps is required, that grows with the number of nodes in a logarithmic (or even less) way; hence, a high level of responsiveness may be obtained.

These two preliminary results may lead to different conclusions. Indeed, the choice of using a scale-free network surely leads to a quick dissemination of game events, hence confirming that a high level of responsiveness may be guarantee for the support of MOGs. On the other hand, the

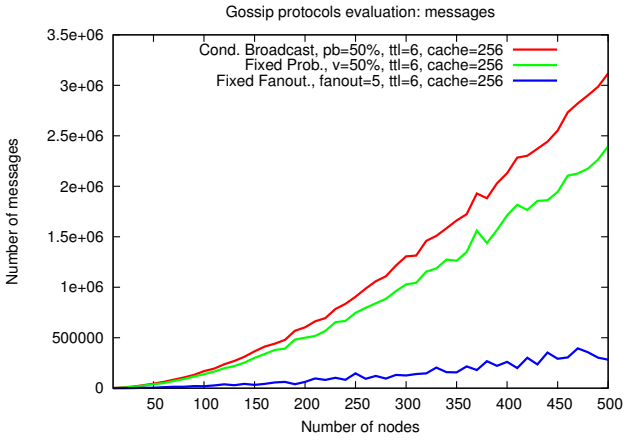

Figure 3: Number of messages using the gossip protocols; $v = 0.5, fanout = 5, p_b = 0.5$**.**

gossip protocols must be carefully selected since they may not be able to propagate the data to all the nodes participants.

Figure 3 reports the number of messages which are sent through the network, required to disseminate the game events generated during the simulations. It is possible to observe that as the number of nodes grows, also the number of sent messages grows. Such growth is quite limited for *fixed fanout*, and this corresponds to its incapacity of distributing all messages to all nodes in the network. This behavior is quite reasonable. Indeed, such gossip does not take into account the degree distribution of nodes; rather, a fixed amount of nodes is selected. As the number of nodes grows in a scale-free net, also the number of hubs augments, together with their degrees. This is a direct consequence of the *preferential attachment* method for creating the network. The other two schemes, instead, show an important growth of sent messages though the network, as the number of nodes in the system increases. In these cases, in fact, the degree of the nodes i s taken into consideration when a node disseminates a message. We already discussed that in *fixed probability* gossip, the game event is propagated from n_i to $v|\Pi_i|$ other nodes, on average. As to probabilistic broadcast,

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

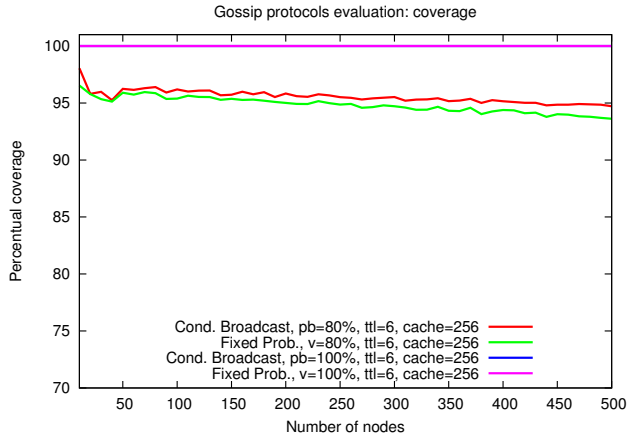

Figure 4: Network delay using the gossip protocols; $v = 0.8, v = 1, p_b = 0.8, p_b = 1$.

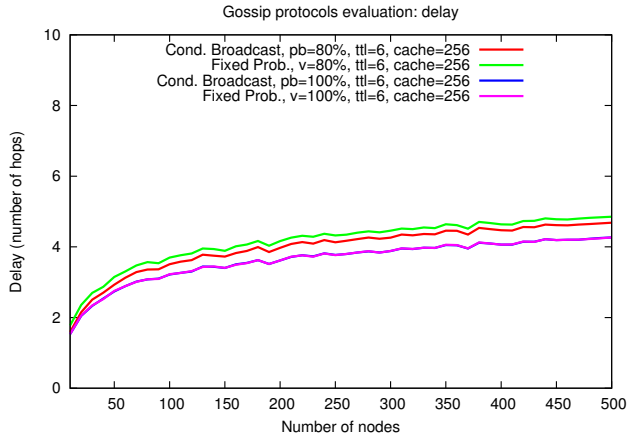

Figure 5: Number of hops using the gossip protocols; $v = 0.8, v = 1, p_b = 0.8, p_b = 1$.

instead, all neighbors are involved in the gossip, once a node decides to distribute the message.

The unsatisfactory results obtained using previously described gossip protocols are influenced by the configuration settings of the protocols themselves. If we indeed augment the probability of gossiping, then the coverage augments, yet exponentially augmenting the number of messages sent throughout the network.

Figure 4 shows the different behaviors of *Conditional Broadcast* and *Fixed Probability* with different settings. In particular, the probability of gossip is raised up to $p_b = v = 0.8$ and then 1. As a first consideration, it is possible to observe that the the two schemes have very similar performances. When $p_b = v = 0.8$ it is possible to appreciate a high coverage of the network, over 95%. Needless to say, when the dissemination probability is kept equal to 1, a full coverage is obtained using both schemes. Such high number of nodes is reached in very few hops (less than 5, even when large nets are considered), as demonstrated in Figure 5. As expected, the number of messages grows with the network size (Figure 6).

As a final consideration, these results demonstrate that when resorting to push based gossip algorithms, a high gossip probability must be exploited in order to cover the whole network. Of course a flood of the network allows to reach the whole peer set Π, yet at the cost of augmenting the number of messages.

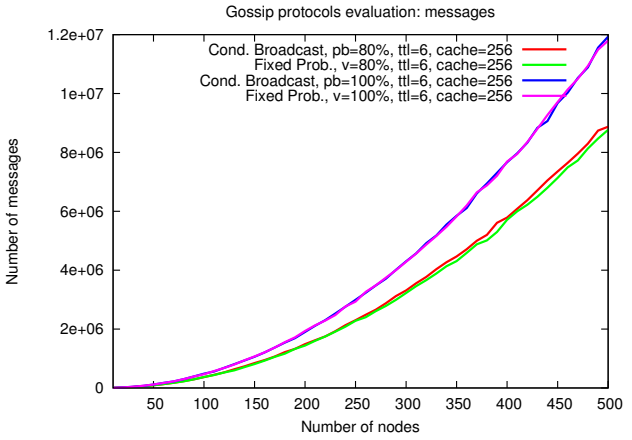

Figure 6: Messages sent using the gossip protocols; $v = 0.8, v = 1, p_b = 0.8, p_b = 1$.

5. CONCLUSIONS

In this paper we assessed the possibility of exploiting scale-free networks as models for organizing peer-to-peer MOGs. To disseminate the game events generated during the game evolution, we assessed how traditional gossip protocols may behave, when run on top of these networks. We obtained very clear results. Indeed, the very low diameter of generated scale-free networks allows to disseminate messages in very few hops. This means that a high level of responsiveness may be provided when resorting to these kinds of overlays. This is an important result, as responsiveness is the main goal to pursue when dealing with MOGs. However, we showed through simulation that gossip protocols may not be able to disseminate the whole event trace, when low gossip probabilities are exploited. In fact, situations may arise when some node is the unique host that received a given message and it decides not to propagate that message, which is then discarded (without being propagated by other nodes).

The conclusion of these outcomes is that smart gossiping strategies should be utilized to disseminate information, so as to guarantee that all nodes may receive the whole game event trace. We exploited here push gossip protocols, without considering pull mechanisms, where nodes may ask others for certain events. Probably, a viable strategy may consist in exploiting both push and pull approaches. For instance, a push gossip may be exploited to disseminate game messages, while concurrently allowing peers to directly ask for some contents they miss (pull gossip). This solution would add an important benefit for the provision of MOGs over scale-free networks.

6. REFERENCES

[1] D. T. Ahmed and S. Shirmohammadi. A dynamic area of interest management and collaboration model for p2p mmogs. In *DS-RT '08: Proceedings of the 2008 12th IEEE/ACM International Symposium on*

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655

Distributed Simulation and Real-Time Applications, pages 27–34, Washington, DC, USA, 2008. IEEE Computer Society.

[2] ARTÌS: Advanced RTI System Homepage. http://pads.cs.unibo.it, 2009.

[3] M. Assiotis and V. Tzanov. A distributed architecture for mmorpg. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 4, New York, NY, USA, 2006. ACM.

[4] A. Barabási, E. Ravasz, and Z. Oltvai. Hierarchical Organization of Modularity in Complex Networks. In R. Pastor-Satorras, M. Rubi, & A. Diaz-Guilera, editor, *Statistical Mechanics of Complex Networks*, volume 625 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 46–65, 2003.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[6] 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.

[7] L. Bononi, G. D'Angelo, and L. Donatiello. HLA-based adaptive distributed simulation of wireless mobile systems. In *Proc. 17th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*. IEEE Press, 2003.

[8] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33(1):309–320, June 2000.

[9] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *PHYS.REV.LETT*, 90:058701, 2003.

[10] E. Cronin, B. Filstrup, S. Jamin, and A. Kurc. An efficient synchronization mechanism for mirrored game architectures (extented version). *Multimedia Tools and Applications*, 23(1):7–30, May 2004.

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

[12] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *SIGCOMM*, pages 251–262, Aug-Sept. 1999.

[13] S. Ferretti. A synchronization protocol for supporting peer-to-peer multiplayer online games in overlay networks. In *DEBS '08: Proceedings of the second international conference on Distributed event-based systems*, pages 83–94, New York, NY, USA, 2008. ACM.

[14] S. Ferretti and M. Roccetti. Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 405–412, New York, NY, USA, 2005. ACM.

[15] 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.

[16] 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.

[17] J. Jardine and D. Zappala. A hybrid architecture for massively multiplayer online games. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 60–65, New York, NY, USA, 2008. ACM.

[18] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[19] S. Krause. A case for mutual notification: a survey of p2p protocols for massively multiplayer online games. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 28–33, New York, NY, USA, 2008. ACM.

[20] M. Mauve, S. Fischer, and J. Widmer. A generic proxy system for networked computer games. In *Proceedings of the 1st workshop on Network and system support for games*, pages 25–28. ACM Press, 2002.

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

[22] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, and M. Roccetti. Interactivity-loss avoidance in event delivery synchronization for mirrored game architectures. *IEEE Transactions on Multimedia*, 8(4):874–879, 2006.

[23] K. Prasetya and Z. D. Wu. Performance analysis of game world partitioning methods for multiplayer mobile gaming. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 72–77, New York, NY, USA, 2008. ACM.

[24] M. Roccetti, S. Ferretti, and C. E. Palazzi. The brave new world of multiplayer online games: Synchronization issues with smart solutions. In *11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008), 5-7 May 2008, Orlando, Florida, USA*, pages 587–592, 2008.

[25] C. Seeger, B. Kemme, P. Kabus, and A. Buchmann. Area-based gossip multicast. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 40–45, New York, NY, USA, 2008. ACM.

[26] 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.

[27] V. Vishnumurthy and P. Francis. A comparison of structured and unstructured p2p approaches to heterogeneous random peer selection. In *ATC'07: 2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8655
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8655