Enabling Rapid and Cost-Effective Creation of Massive Pervasive Games in Very Unstable Environments

Bartosz Wietrzyk, Milena Radenkovic
School of Computer Science and IT
University of Nottingham
Nottingham, NG8 1BB, UK
{bzw, mvr}@cs.nott.ac.uk

Abstract—**Pervasive gaming is a new form of multimedia entertainment that extends the traditional computer gaming experience out into the real world. Through a combination of personal devices, positioning systems and other sensors, combined with wireless networking, a pervasive game can respond to player's movements and context and enable them to communicate with a game engine and other players. We review our recent deployment examples of pervasive games in order to explain their distinctive characteristics as wireless ad-hoc networking applications. We then identify the network support challenges of scaling pervasive games to include potentially mass numbers of players across extremely heterogeneous and unreliable networks. We propose a P2P overlay capable of storing large amount of game related data, which is the key to combating the loss of coverage and potential dishonesty of players. The proposed protocol decreases the deployment costs of the gaming infrastructure by self organization and utilizing storage space of users' devices. We demonstrate scalability and increased availability of data offered by the proposed protocol in simulation based evaluation.**

Keywords-DHT; MANET; Peer-to-Peer; Pervasive Gaming

I. Introduction

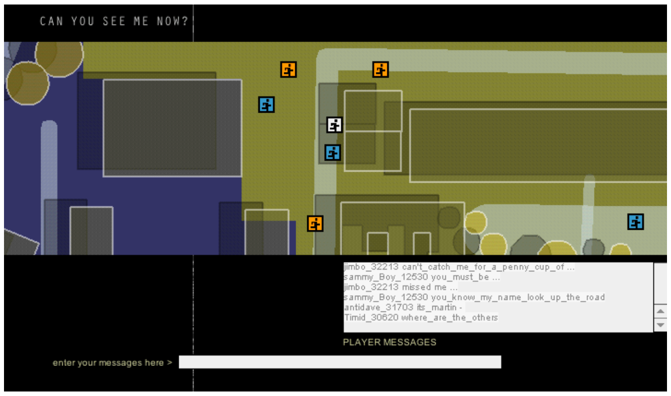
The wide spread usage and low cost of mobile and wireless computing predict sizes for future applications of Mobile Ad hoc Networks (MANETs) several magnitudes larger than current protocols can handle. Such applications are often location-based and they exploit positional information to support mobile interactivity in domains as diverse as tourism [1], information retrieval [2], resource discovery [2], workplace awareness [3] and games [4].

The focus of this paper is on the gaming domain. Figures quoted in the press increasingly show patterns of massive investments and consumption budgets in this domain. For example the European Union's Sixth Framework Program and British Engineering and Physical Sciences Research Council (EPSRC) are investing millions of euros in projects and initiatives such as the Integrated Project of Pervasive Games (iPerG) [5] and Equator [6]. Major companies such as Nokia, Sony, and Microsoft have assembled large advertising, marketing, and development teams to promote and realize these technologies on a massive scale. More than 300,000 players participated in the Nokia Game in 2003, approximately 48 percent of whom were 25-40 years old. This points to a significant trend: rather than simply toys for teenagers, mixed-reality games are increasingly becoming an entertainment and communication medium for adults. Together with BBC, BT, Microsoft Research and the University of Bath we have recently started Participate [7], a multi-million project that explores convergence of pervasive, online and broadcast media to create new kinds of mass-participatory events. Participate will involve a number of massively multiplayer pervasive game trials.

Pervasive games extend the traditional gaming experience out into the real world. They become increasingly massive in terms of number of players, their geographical distribution, amount of data, duration of the game, heterogeneity of players' devices and wireless platforms. That makes the demand of these systems very different from the demands of the current systems designed for hosting pervasive games of limited scale or current massively multiplayer online games. For example mass scalability, extreme heterogeneity management and decreasing deployment costs are becoming essential challenges.

One of the goals of this paper is to propose a peer-to-peer overlay approach that is able to provide network connectivity and basic network services in a self-organizing fashion, opportunistically using available infrastructure when and where it exists. Such an overlay would further enable rapid and cost-effective creation and staging of massive pervasive games that is one of the main concerns identified today in the networked gaming community [8].

Pervasive games are often location based [8], potentially utilizing touristically attractive remote areas where the coverage of WLAN, UMTS and GPRS is typically limited. UMTS coverage is still relatively constrained even in urban areas and GPRS offers only limited bandwidth. Costs of using UMTS and GPRS are high due to high infrastructure maintenance cost. Therefore, MANETs are promising in terms of extending wireless coverage, improving bandwidth and lowering usage and deployment costs but the temporal



jimbo_32213 can't_catch_me_for_a_penny_cup_of ...
sammy_Boy_12530 you_must_be ...
jimbo_32213 missed me ...
sammy_Boy_12530 you_know_my_name_look_up_the_road
antidave_31703 its_martin -
Timid_30620 where_are_the_others

Figure 1. An on-line player's interface

disconnections in MANETS are difficult to avoid. They can degrade the playability of the game because players may not receive the updates of the game state, their communication with the other players and game operators may be hindered. This motivates work on the dedicated storage and retrieval protocols that effectively deal with disconnections, such as one proposed in this paper.

The contributions of this paper are three fold: (1) identification of challenges for applying MANET approach to pervasive games in order to decrease costs, improve wireless coverage and bandwidth, (2) a novel fully distributed self-organized protocol based on a hybrid distributed hash table (DHT) approach, (3) simulation based evaluation of the proposed protocol.

This paper is organized as follows. Section II reviews our recent deployment examples of pervasive games. Section III identifies the requirements and challenges of massive pervasive games. Section IV proposes our novel store and retrieval DHT protocol. Section V presents simulation based evaluation of the proposed protocol and discuses its security. Section VI briefly reports on the related work. Finally, Section VII gives conclusions.

II. Pervasive Games

In pervasive games players equipped with mobile devices move through the world – be it on the city streets or in the remote wilderness. Sensors capture information about their current context, including their location, and this is used to deliver them a gaming experience that changes according to where they are, what they are doing and potentially even how they are feeling. We begin by presenting recent examples of pervasive games developed within the Equator [6] and iPerG [5] projects including Can You See Me Now? [9] and Uncle Roy All Around You [10]. Drawing on these examples, we explore the key research challenges for how to provide networking support for pervasive games that allows disseminating, storing and searching for data in unstable networks with frequent disconnections.

A. *Example 1: Can You See Me Now?*

Can You See Me Now? [9] was one of the early games that we developed and staged in which up to two hundred on-line players logged in over the Internet were chased across a map of a city by three performers who were running through its streets. Central to Can You See Me Now? was a relationship between up to twenty on-line *players* (members of the public using the Internet) who were moving across a map of Sheffield, and three *runners* (members of Blast Theory) who were moving through the streets of Sheffield. The runners chased the players. The players avoided being 'seen'. Everyone, runners and players, saw the position of everyone else on a shared map. Players sent text messages to each other, which were also received by the runners. In turn, runners talked to one another over a shared radio channel, which was also overheard by the players.

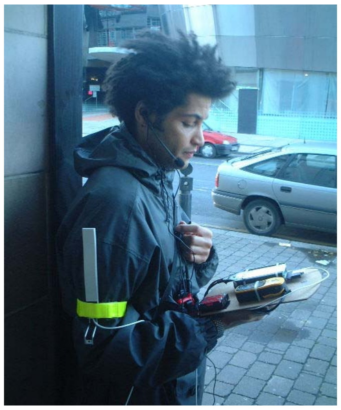


Figure 2. A runner ready to go

Fig. 1 shows an example of the player interface. A simple white icon showed the player's current position according to her local client. Other players were represented as blue icons. The runners were shown as orange icons.

The runners also saw the map of Sheffield showing their positions as well as the players' positions and text messages. This was delivered to them on a Compaq iPAQ from a server in a nearby building over a 802.11b local area network. A GPS receiver plugged into the iPAQ registered the runner's position as they moved through the streets and this was sent back to the server over the wireless network via an armband antenna. The runners also used walkie-talkies with earpieces and head-mounted microphones (see Fig. 2).

B. *Example 2: Uncle Roy All Around You*

Uncle Roy All Around You [10] sent the public out on to the streets of London equipped with handheld computers in search of the elusive Uncle Roy (see Fig. 3). These street players followed a series of clues that led them through St James Park and the city streets of Westminster to Uncle Roy's office in Piccadilly. From there they were directed to a nearby telephone box and from there to a waiting limousine that drove them back through London to the Institute of Contemporary Arts, during which they were asked whether they would be willing to commit to help a stranger if called upon.

At the same time, online players were exploring a parallel virtual model of London, also in search of Uncle Roy's office. Once they found it they could then help (or hinder) the street players by sending them instructions and directions as text messages, to which the street players could respond with short audio messages. Online players were also invited into

Figure 3. A street player explores central London in search of Uncle Roy

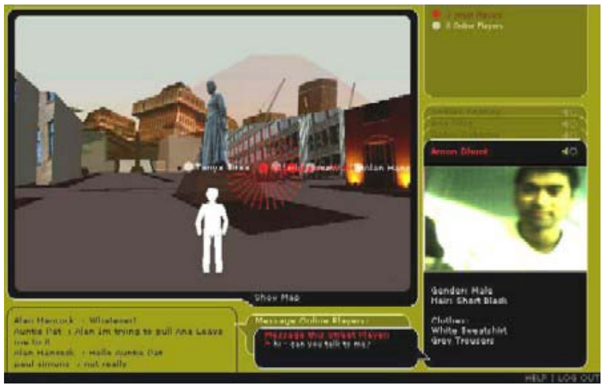


Figure 4. An online player explores central London

Uncle Roy's office - via a webcam - whenever a street player entered inside and were asked the same question - would they be willing to commit to help a stranger if called upon. At the end of the game, we paired up street players and online players who had made such a commitment and passed on their contact details.

Uncle Roy was staged in central London for two weeks during late May and early June in 2003 during which time it was played by nearly three hundred street players and a similar number of online players. Studies of Uncle Roy shed light on how people use position information within mobile experiences. In particular, Uncle Roy exploited a technique called self-reported positioning in which street players would report their known position, either explicitly by declaring their position to Uncle Roy or implicitly by their PDA sending information about which area of the map they were looking at to remote online players.

III. Requirements and Challenges

Section II showed how games are becoming increasingly massive in terms of number of players, their geographical distribution, amount of data, duration, heterogeneity of players' devices and wireless platforms. This increases the turnover of players (churn) who come and go without a warning. These massive pervasive games are being deployed in increasingly challenged environments. Despite the requirements, across the multiple game generations, the infrastructure support has been mostly centralized. This is very limiting in terms of: cost, scale, network awareness and this in turn influences playability of the games.

As we can see in Section II Pervasive games are often location based, utilizing touristically attractive areas potentially not only London but also remote wilderness e.g. highland in northern Scotland, where the coverage of WLAN, UMTS and GPRS is typically limited. UMTS coverage is still relatively constrained even in urban areas and GPRS offers only limited bandwidth. Costs of using UMTS and GPRS are high due to high infrastructure maintenance cost. Therefore, MANETs are promising in terms of extending wireless coverage, improving bandwidth and lowering usage and deployment costs but the temporal disconnections in MANETS are difficult to avoid. They can degrade the playability of the game because players may not receive the updates of the game state, their communication with the other players and game operators may be hindered. This can also undermine the consistency of the game and make its logic unclear for the players.

The deployment of the games is only feasible when its total cost is kept low. This includes the cost of the deployed hardware, the cost of configuring the infrastructure before the beginning of the game and the cost of reconfiguring it when the number of players changes. Self organization of such systems is important because it decreases effort necessary to configure users' software and potentially complex game engine.

These challenges are different from the current massive multiplayer on line games (MMOGs), as the target hardware and typical design of the games in both cases are different. In pervasive games the amount of multimedia data produced during the gameplay both by players and operators is potentially much higher. The heterogeneity of the players' devices and network links, the probability of temporary disconnections are also higher.

Considering these requirements, we see that they conform to the peer-to-peer (P2P) paradigm. Distributed Hash Tables (DHTs) [11-14] are a very promising P2P approach to cover these requirements. With DHTs the data is distributed among the nodes in a network and the hash is calculated to determine the address of the node with certain information. The whole architecture is strongly decentralized. It has been already demonstrated [15] that DHTs can support on-line massively multiplayer games in the wired networking environment. Design of a DHT for massive pervasive games that have wireless ad hoc users poses key additional challenges:

High churn rate [16, 17] – users come and go without any warning, e.g. due to lack of battery capacity or leaving area with the coverage. This can lead to loss of the game data.

Cheating [15] – dishonest players can alter data they store and functionality of their nodes to gain advantage in the game.

Heterogeneity of user devices and their wireless platforms – some of the users can use devices with constrained computational power and storage space, the wireless platform available to them can have very limited bandwidth.

Cost of the links - some of the users' devices can be connected over expensive links such as UMTS. These links can be financially expensive for the users if they are charged for the payload. If the operator charges fixed rate for using them or participating in the game, the increased game related traffic may affect other services and users.

Disconnections [18] – the coverage of wireless networking platforms often used today can be limited in remote areas, which are potentially attractive for staging pervasive games. Even extending coverage of the wireless infrastructures with MANETs does not eliminate temporary (and potentially long) disconnections of players, which can affect their gaming experience.

IV. Storage and Retrieval Protocol

In this section, we propose a novel storage and retrieval protocol for the massively multiplayer pervasive games in the highly heterogeneous environment – Hybrid Pastry (HP). This protocol provides users with improved coverage and bandwidth by opportunistically utilizing the temporarily optimal available wireless networking platform.

Our protocol is based on a peer-to-peer overlay capable of storing large amount of game related data. The peers in this overlay comprise both distributed dedicated servers and users' devices. In order to decrease the deployment costs and increase scalability data is stored by some of the users' devices. In order to address churn and high cost of certain links our protocol differentiates between different classes of users, each of them has different tasks related to application-level routing and storage of game data. To make it less vulnerable to data loss, churn and dishonesty of players, game data stored by the users is replicated, i.e. each piece of game data is stored by several users. When a node requests data, multiple copies are fetched and compared. Nodes having data different from other replicas are accused of cheating and blacklisted. This can potentially lead to the explosion of replies. In order to avoid this, it is necessary to carefully select the number of replicas. If this number is too high the explosion of replies can occur and cause increased network congestion. If there are too few replicas the invulnerability to cheating, churn and disconnections may be compromised.

Our protocol recognizes three classes of nodes:

Regular Clients (RCs) are connected over expensive (e.g. GPRS, UMTS), low bandwidth (e.g. GPRS) or unreliable (MANET over e.g. Bluetooth or 802.11) links or are resource constrained devices. These nodes do not receive or send traffic not directly related to their participation in the game in order to limit utilization of the expensive/low bandwidth links and storage space of constrained devices. In particular they neither store game related data which is accessed by other users nor, if possible, forward queries issued by other nodes or answers to these queries. If the number of RCs is much larger than nodes of other types described below, for the reasons of scalability, it may be inevitable for them to forward some of the queries.

Super Clients (SCs) are clients connected over cheap, high bandwidth links e.g. wired network or 802.11g (infrastructure mode) and are powerful devices such as desktop or laptop computers. They store shared game data, forward queries and replies but they are considered not trusted so each unit of data is stored on several SCs and compared upon retrieval. Sending data to a set of nodes and comparative retrieval of replicated data is performed only by SCs. Therefore, an RC has only to receive or send each of the data units once, which potentially considerably limits the bandwidth on expensive links and usage of resources on constrained devices. An RC can be upgraded to SC and SC can be downgraded to RC depending e.g. on the network congestion or moving from/to WLAN coverage.

Application Servers (ASs) are dedicated servers deployed for the purpose of the game. They store data, route queries and answers to these queries. The data they store is not replicated as these nodes are not affected by churn and are trusted.

As addressing and application level routing in our protocol are motivated by Pastry DHT [11, 19] for better understanding we briefly review the original Pastry protocol.

A. *Pastry*

Pastry [11, 19] is a structured p2p routing protocol that implement the DHT abstraction. In a Pastry network, each node has a unique, uniform, randomly assigned nodeId in a circular 128-bit identifier space. Given a message and an associated 128-bit key, Pastry reliably routes the message to the live node whose nodeId is numerically closest to the key.

In a Pastry network consisting of N nodes, a message can be routed to any node in less than $\log_{2^b} N$ steps on average, where b is a configuration parameter responsible for the number of rows, columns in the routing table and maximal number of routing steps. Each node stores only O(log$N$) entries, where each entry maps a nodeId to the associated node's IP address. Specifically, a Pastry node's Routing Table is organized into $\lceil \log_{2^b} N \rceil$ rows with (2^b-1) entries each. Each of the (2^b-1) entries at the row n of the Routing Table refers to a node whose nodeId shares the first n digits with the present node's nodeId, but whose (n+1)th digit has one of the (2^b-1) possible values other than the (n+1)th digit in the present node's nodeId. Pastry stores multiple candidates per Routing Table entry to increase availability. In addition to a Routing Table, each node maintains a Leaf Set, consisting of L/2 nodes with numerically closest larger nodeIds and L/2 nodes with numerically closest smaller nodeIds, relative to the present node's nodeId. L is another configuration parameter. In each routing step, the current node forwards a message to a node whose nodeId shares with the message key a prefix that is at least one digit (or b bits) longer than the prefix that the key shares with the current nodeId. If no such node is found in the Routing Table, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node but is numerically closer to the key than the current nodeId. Such a node must exist in the Leaf Set unless the nodeId of the current node or its immediate neighbor is numerically closest to the key.

B. *Hybrid Pastry*

We propose Hybrid Pastry (HP) in which data in the form of key-value pairs is stored by SCs and ASs with nodeIDs closest to the keys. NodeIDs are calculated by hashing IP addresses of the nodes. The key identifying data is a hash value calculated over a subset of data. This can be, for example, the name of the object related to these data, such as the name of the place that a stored picture shows. As the time plays an important role for the queries, it can also be used for the hash calculation. It is particularly important not to calculate hash over the same data associated with different objects. If the same hash values identify different objects the fair distribution and thus scalability of the whole system can be seriously affected [20]. In particular, some of the users' devices can be forced to store much more data than the others.

There are also two types of messages – directed to a node from any group (e.g. answers to queries) or directed to a node which is an SC or AS. Routing of the messages issued by any class of nodes is initially performed only by SCs and ASs and reaches RCs only at last hop(s). Therefore, the state of nodes in HP differs from Pastry in the following way. Neighborhood Set and Routing Table contain only SCs and ASs. There are also two Leaf Sets – one for SCs and ASs (SC/AS Leaf Set) and one for RCs only (RC Leaf Set). This separation is necessary to assure that a message will finally reach an RC regardless of the proportions between the numbers of RCs, SCs and ASs. Otherwise it could happen that the SCs or ASs with nodeId closest to the key of the forwarded message has its Leaf Set full of other SCs and ASs and is not aware of the existing RC that should be the next hop.

At the initial stage, when the message is routed only by SCs and ASs the Leaf Set with RCs is ignored. Only when the message is directed to any class of nodes and already reached the SC or AS with nodeId numerically closest to the message's key, it can be further forwarded to an RC from the RC Leaf Set if it has a nodeId numerically closer to the message's key than the forwarding node.

If the number of RCs is much higher than the number of SCs and ASs it may happen that not all RCs will fit into the Leaf Sets of ASs and SCs. In that case a very limited number of last hops may be performed by RCs. For that reason an RC after receiving a key-data pair checks if it has in its RC Leaf Set a node with nodeId numerically closer to the key of the message.

Note that SCs and ASs are potentially much less affected by churn and disconnections than RCs. Therefore, limiting the number of hops performed by RCs potentially considerably improves the probability of successful delivery and decreases delays/overhead necessary to forward the message.

Depending on the type of the storing node stored data can be replicated or not. The key-value pair which is to be stored is routed to the SC or AS with the nodeId numerically closest to the key. The node stores it and if it is an SC it replicates the data to M-1 SCs with nodeIds numerically closest to the key of the message. Similarly, when the data is retrieved from SC, the SC with nodeId numerically closest to the key of the message is responsible for retrieving and comparing all the replicas. This is necessary for detecting nodes with tampered data. If the overall number of the collected replicas is lower than M because the nodes storing them left or were downgraded to RC, the retrieving node disseminates the 'missing' replicas. Optionally it may be allowed to retrieve and compare less than M replicas in case of increased network traffic in order to improve network friendliness for the cost of decreased security. The node comparing or updating replicas also controls if a node, which requested update or retrieval, is allowed to do that.

Note that the node retrieving the replicas sends them directly to the querying node, not using application level multi hop routing. This is very important for saving the bandwidth, especially in the case when the number of RCs is much higher than the number of ASs and SCs. More precisely in that case RCs do not have to forward queried data.

Downgrading a node from SC to RC is similar to leaving the DHT in the proposed protocol, which is perform similarly as in the original Pastry [19]. The Leaf Sets are maintained actively by periodical sending keep-alive messages to all nodes in both Leaf Sets. The new status of the downgraded node is advertised in the acknowledgments it sends for the received keep-alive messages. The downgraded node also propagates its status in the lazy way by starting to refuse providing access to the public data it stores. Nodes which discover that this node has been downgraded remove it from the Neighborhood Set and Routing Table and move it from the AS/SC Leaf Set to the RC Leaf Set. The repopulation of the Routing Table, AS/SC Leaf Set and Neighborhood Set in order to cover the gap after downgraded node is performed similarly as in the original Pastry upon the node's departure.

When a node is upgraded from the RC to SC it informs about that all the nodes in both its Leaf Sets, Routing Table and Neighborhood Set so they can upgrade their states accordingly. If any of these nodes removes the upgraded node from the RC Leaf Set, the RC Leaf Set is repopulated by retrieving the RC Leaf Set from the live node with the largest index on the side of the failed node.

Lazy repairing of the Routing Tables from the original Pastry also helps in the propagation of the information about downgrades and upgrades of the nodes. More precisely, during message forwarding, when a Routing Table entry is found to be either empty or containing a failed node, the message is routed to another node with numerically closest nodeId. If a downstream node has a routing table entry that matches the next digit of the message's key, it automatically informs the upstream node of that entry.

Arrival and departure of nodes is performed similarly as in the original Pastry DHT [11].

C. *Synchronization*

As we can see in Section II pervasive games are characterized by higher versatility than the current Massively Multiplayer On Line Games (MMOGS), which typically have very similar architectures [15]. Therefore, the synchronization model has to be adjusted to the construction of the game. Let us consider the games described in Section II as en example. They are location based so we use the cell-based synchronization similarly as proposed by Knutssen et. al. [15]. In particular the game area is divided statically or dynamically into cells. The data associated with the cell (i.e. references to players located there, references to multimedia data) can be stored (and replicated) as key-value pairs. This data comprises keys of the multimedia data associated with the cell, stored as key-value pairs and ids of the players located within the cells both physically as *runners* and virtually as on-line players (from Can You See Me Now, see Section II). Note that another form of handling keys of the multimedia data is passing them as a part of the communication among players and between players and game operators.

Players occupying the cell inform about their actions the node numerically closest to the id of the cell (cell coordinator), which updates the state of the cell. The cell id can be a name of the place, map coordinates (in case of self reporting, see Section II), or id of a gsm cell (in case of GSM positioning [8]). Cell coordinator also acts as the root of the application level multicast tree (such as Scribe [21]) informing other players about the changes in the cell they occupy. Note that Scribe multicast based on Pastry DHT is congruent with the proposed protocol. If a player is disconnected for some time she may miss the updates sent by the cell coordinator. After a reconnection such player polls the coordinator for the current state of the cell.

V. Analysis of the Proposed Protocol

A. *Simulation Based Evaluation*

This section discusses simulation based evaluation of the proposed protocol. We concentrate here on a single MANET that can be a part of a larger topology. In the basic case 20 wireless nodes (802.11) move according to Random Waypoint Model [22] with the random speed from 1 to 2 m/s (uniform distribution) which reflects the walking speed of humans [23]. Nodes make random stops from 0 to 180 seconds (uniform distribution). They move on the square area 400m × 400m and communicate with the rest of topology through the stationary gateway located in the middle of one of the sides of the simulation area. The communication range of both wireless nodes and the gateway is 250m. The simulated mobile nodes represent street players (e.g. Uncle Roy All Around You) or runners from Can You See Me Now.

We assume that the total number of nodes in the topology is 1000. It comprises several MANETs and dedicated servers. We simulate one of the MANETs being the part of the overall topology. All the nodes issue queries every 60 seconds and receive replies to these queries with 5KB payload of data. This represents data concerning other players and objects from the 'virtual' world including location data, text, images, sounds etc. Every 10 seconds each node sends 1KB of data for storage (an update) and in case of success it receives an acknowledgement [24]. This represents spatial data, measurements from the sensors carried by players, communication with other players or information about player's interaction with the objects from the 'virtual' world.

We simulate both original Pastry [11, 19] and the proposed Hybrid Pastry (HP). In the simulations Pastry nodes send queries/data with probability P_{MANET} (0.02 in case of MANET with 20 nodes) to other nodes within the simulated MANET and with probability $1\text{-}P_{MANET}$ (0.98 in case of MANET with 20 nodes) over the gateway. $P_{MANET}=N_{MANET}/N$, where N is total number of nodes in the topology and N_{MANET} is the number of nodes within the simulated MANET. In both cases data and queries are routed by all nodes in the multi hop manner. The queries arrive from the gateway every 3 seconds and the data updates every 0.51 seconds. Both these frequencies (f_{GTW}) are calculated from the following formula:

$$f_{GTW} = f_N (N - N_{MANET}) P_{MANET} = \frac{f_N (N - N_{MANET}) N_{MANET}}{N} \qquad (1)$$

where f_N is the frequency for a single node. In case of Hybrid Pastry all nodes within the MANET are RCs so no data updates and no queries are sent to them. All the queries and updates they issue are routed to the gateway in the multi hop manner.

We perform two sets of simulations. In first we are altering the density of the nodes and in second the number of nodes in the simulated MANET, with the constant density 125 nodes/km^2. In the second set of simulations we assume that the total number of nodes in the overlay is 1000 so we are recalculating the P_{MANET} and frequency of updates/queries coming from the gateway for each simulation. We perform the simulation in ns-2 [25] and the MANET routing protocol is AODV [26].

We monitor Success Ratio (SR) and average delays. SR is the percentage of successfully received answers to the queries sent by mobile nodes within the simulated MANET and successfully delivered data updates sent by these nodes. Delays are measured as average time between sending a query/data update and receiving response/data update. SR and delays are important for player's experience because they have influence on how often and how fast a player is informed about changes in state of the game and how quickly her actions affect other players. The results of the simulations are presented in Fig. 5.

In Fig. 5a we can see that denser topologies are not challenging for both protocols. Hybrid Pastry outperforms Pastry in terms of SR of queries by to 13.8% for average densities. The SR of data updates is similar for both compared protocols. In Fig. 5b we can notice that the average delays of queries and updates are similar for both protocols in case of all tested densities.

In Fig. 5c we can see that Hybrid Pastry achieves better SR than Pastry for both updates and queries (by up to 7%) for numbers of nodes higher than 5. The measured SR is pessimistic as we do not consider here neither multiple replicas nor repeating queries after regaining connectivity or end of increased network traffic. Note that the considered number of nodes refer to a single MANET, not to the size of the overall topology.

The delays shown in Fig. 5d are slightly higher for the Hybrid Pastry than Pastry but this can be attributed to the higher SR of Hybrid Pastry. In particular packets that take most time to deliver for Hybrid Pastry would not be delivered by Pastry.

To summarize, we demonstrated that the proposed protocol, Hybrid Pastry, offers better scalability in terms of size of a single MANET, offering better SR which affects players' experience. It also deals better with sparser topologies of MANETs. We achieved that without any significant increase of

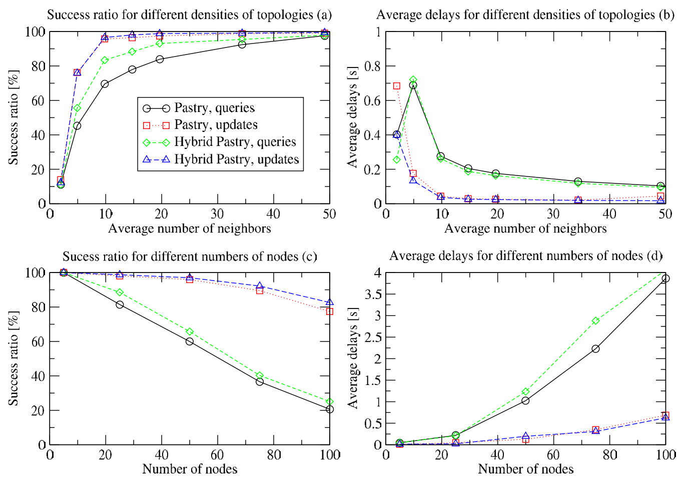


Figure 5. Simulation results

delays.

B. Security

This section discusses the possibilities of tampering data and/or functionality of the system using the proposed protocol in order to get advantage in the game. Dishonest players can *modify data stored by their node* to gain advantage in the game. This behavior can endanger consistency of the game. In order to prevent this each key-value pair stored by SCs is replicated to *M-1* other SCs. *M* is a configuration parameter selected as a compromise between network/computational overhead and limiting vulnerability to disconnections, churn and cheating. Optionally it can be adjusted dynamically depending on the available bandwidth.

On retrieval all the replicas are compared by the SC, which has nodeId numerically closest to the key identifying data. If compared data is different the most common value is assumed as not tampered and the other values as tampered. Nodes which provided tampered data are removed from the game and blacklisted. The blacklist can be kept on one of the ASs.

We can assume that the probability of tampering a data unit stored by a user is P_T and the probability of tampering data stored by AS is negligible. Then, for the original Pastry the probability of retrieving the tampered data (P_{TR}) is

$$P_{TR} = (P_{SRC} + P_{SSC})P_T = \frac{N_{RC} + N_{SC}}{N_{RC} + N_{SC} + N_{AS}} P_T \tag{2}$$

where P_{SRC}, P_{SSC} are probabilities of storing a key-value pair by an RCs and SCs respectively. N_{AS}, N_{RC}, N_{SC} are numbers of ASs, RCs and SCs.

In the proposed protocol the node comparing the replicas accepts the most common version of data as not tampered so we can pessimistically assume that the querying node receives a tampered data if more than half of the replicas are tampered. It is pessimistic because we assume that all the tampered replicas are modified in the same way. From the Bernoulli's Trials we can calculate:

$$\begin{aligned} P_{TR} &= P_{SSC} \sum_{i=\left\lceil \frac{M}{2} \right\rceil}^{M} b(i, M, P_T) = \frac{N_{SC}}{N_{RC} + N_{SC} + N_{AS}} \sum_{i=\left\lceil \frac{M}{2} \right\rceil}^{M} b(i, M, P_T) = \\ &= \frac{N_{SC}}{N_{RC} + N_{SC} + N_{AS}} \sum_{i=\left\lceil \frac{M}{2} \right\rceil}^{M} \frac{M!}{i!(M-i)!} P_T^i (1-P_T)^{M-i} \end{aligned} \tag{3}$$

For example for the P_T=0.01, 10 ASs, 100 SCs and 1000 RCs for original Pastry P_{TR}=0.099. For the proposed protocol the dependence between P_{TR} and M is shown in Fig. 6. As we can see, even a small number of replicas such as three considerably helps in fighting tampering but it is not advisable to have an even number of replicas. Fig. 6 clearly shows that using two replicas is particularly suboptimal for combating tampering. If one of the nodes tampers the replica the node retrieving replicated data will not know which replica is original. If both nodes tamper their replicas in the same manner the tampered version is accepted as original. The only possibility of retrieving the original data unit is when it is not altered by both nodes holding the replicas, which has lower probability than retrieving a single not tampered replica.

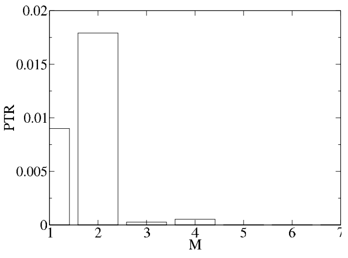


Figure 6. Relation between the number of replicas (M) and probability of retrieving tampered data (P_{TR})

Note that for the number of replicas higher than two it is relatively difficult to blacklist non-cheating players by tampering data. In order to achieve this it would be necessary to tamper in the same way more than one replica. This would require cooperation between randomly designated set of players.

A user can *request update or retrieval of data without having appropriate access rights*. That should be detected by a node, which is numerically closest to the key of the updated/requested data.

A user can tamper her SC's functionality so that it *does not retrieve and compare replicas stored by other SCs*. This gives only a short-lived advantage because the churn makes the closest numeric proximity between nodeId and the keys of the stored messages transient. At any time, such SC may be asked to provide its replica for comparison and if it provides altered data its cheating is immediately detected or if it provides legitimate data, its illegal advantage in the game is gone. Similar applies to illegal disabling of the access rights control.

A subset of nodes can be tampered in the way they *lie about their capabilities trying to qualify as a different class*. Pretending to be an SC instead of an RC is not profitable for a user as this potentially highly increases usage of computational power and network bandwidth, thus decreasing the node's performance and player's gaming experience. On the contrary, pretending to have a more constrained hardware and/or network connection in order to qualify as RC instead of SC can offer a user much better performance. This form of attack only decreases the performance of ASs, other SCs and in extreme cases RCs without affecting consistency of game data and functionality.

VI. RELATED WORK

The proposed protocol is based on the Pastry DHT [11, 19], which we extended in order to handle heterogeneity of peers, cheating and churn of possibly mobile players. There are many examples of heterogeneity management in unstructured p2p systems including JXTA [27], Guntella 0.6 [28] etc. To the best of our knowledge the only structured p2p system addressing heterogeneity is Hybrid Chord [16, 17] but this protocol is more oriented at accessing user data rather than application data. Therefore, it does not address limiting network traffic at the more constrained nodes. Contrary to the authors of Hybrid Chord we decided to base our protocol on Pastry rather than Chord. We believe that Pastry better supports replication and delegating traffic from more constrained to more powerful nodes as nodes have more knowledge about their neighbors from the address space.

Using replication to handle churn and unreliability of nodes in the structured p2p overlay was proposed in the PAST [29] storage system based on Pastry DHT [11, 19]. We proposed [18] using proactive caching, similar to replication to address partitioning of the wireless network. However in both these cases caching is not congruent with the heterogeneity of the nodes and does not address dishonesty of the users.

Using Pastry DHT for the purpose of Massively Multiplayer Online Games (MMOGs) was proposed in [15]. The networking challenges in MMOGs are considerably different than in mass scale pervasive games (see discussion in Section III) so the referenced work concentrates on the synchronization problems not addressing heterogeneity of the peers and their links. They also consider replication only in terms of reliability, not for combating dishonesty of players.

There is also research concerning utilizing MANETs for gaming [4] but typically scale of such games is limited in terms of number of users and scope of a single MANET.

The considerable body of research addresses cheating in the current MMOGs [30, 31]. The referenced work mainly addresses cheating by delaying announcement of the players' positions. The applicability of this work to the pervasive games is limited as the position reported by players is associated with their physical location. Therefore, players can only cheat by reporting false positions, which typically can be easily detected from the physical interactions with other players and the environment.

VII. Conclusions

In this paper we identified network support challenges of scaling pervasive games to include potentially mass numbers of players across extremely heterogeneous and unreliable networks. Then we proposed a self organized p2p overlay addressing these challenges. In order to increase scalability and decrease deployment costs it utilizes bandwidth and processing power of the users' devices. The proposed protocol utilizes heterogeneity of the hardware and network connectivity available to users, using available infrastructure when and where it exists. It is optimized for high rate of churn, disconnections and limited coverage of the currently deployed wireless platforms. It is also resistant to dishonesty of users, making no assumptions about the fair use of the system. We demonstrated performance of the proposed protocol by simulation based evaluation. Such an overlay further enables rapid and cost-effective creation and staging of massive pervasive games that is one of the main concerns identified today in the networked gaming community [8].

References

[1] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide: some issues and experiences", In *Proc. of CHI*, The Hague, The Netherlands, 2000, pp. 17-24.

[2] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks", In *Proc. of WMCSA*, English Lake District, UK, 2004.

[3] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system", *ACM TOIS*, January 1992, 10 (1), pp. 91-102.

[4] D. Budke, K. Farkas, O. Wellnitz, B. Plattner, and L. Wolf, "Real-time multiplayer game support using QoS mechanisms in mobile ad hoc networks", In *Proc. of WONS*, Les Ménuires, France, 2006.

[5] "iPerG", Available: http://www.pervasive-gaming.org/.

[6] "Equator", Available: http://www.equator.ac.uk/.

[7] "Participate", Available: http://participateonline.co.uk/.

[8] M. Capra, et al., "The multimedia challenges raised by pervasive games", In *Proc. of International Multimedia Conference*, Hilton, Singapore, 2005, pp. 89-95.

[9] S. Benford, et al., "Can you see me now?" *ACM TOCHI*, March 2006, 13 (1), pp. 100-133.

[10] S. Benford, et al., "Uncle Roy All Around You: implicating the city in a location-based performance", In *Proc. of Proc. ACE*, 2004.

[11] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems", In *Proc. of Middleware*, Heidelberg, Germany, 2001, pp. 329-350.

[12] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: an infrastructure for fault-tolerant wide-area location and routing", Computer Science Division (EECS), Berkeley, Califorina, Technical Reprt UCB/CSD-01-1141, 2001.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network", In *Proc. of SIGCOMM*, San Diego, California, USA, 2001.

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for Internet applications", In *Proc. of SIGCOMM*, San Diego, California, USA, 2001.

[15] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games", In *Proc. of INFOCOM*, Hong Kong, 2004.

[16] S. Zöls, R. Schollmeier, W. Kellerer, and A. Tarlano, "The Hybrid Chord protocol: a peer-to-peer lookup service for context-aware mobile applications", In *Proc. of ICN*, Reunion Island, France, 2005.

[17] S. Zöls, S. Schubert, W. Kellerer, and Z. Despotovic, "Hybrid DHT design for mobile environments", In *Proc. of AP2PC*, Phoenix, Arizona, USA, 2006.

[18] M. Radenkovic and B. Wietrzyk, "Wireless mobile ad-hoc sensor networks for very large scale cattle monitoring", In *Proc. of ASWN*, Berlin, Germany, 2006, pp. 47-58.

[19] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks", technical report MSR-TR-2002-82, 2002.

[20] J. Eberspächer, G. Kunzmann, S. Zöls, and W. Kellerer, "Peer-to-peer networks, applications and services", in *ASWN*, Berlin, Germany, 2006, Available.

[21] M. Castro, et al., "An evaluation of scalable application-level multicast built using peer-to-peer overlays", In *Proc. of INFOCOM*, 2003.

[22] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks": Kluwer Academic, 1996.

[23] S. D. Pinna, *Forces and motion*, Raintree Steck-Vaughn Publishers, Austin, Texas, USA, 1998.

[24] S. Benford, et al., "The error of our ways: the experience of self-reported position in a location-based game", In *Proc. of UbiComp*, Nottingham, UK, 2004, pp. 70 - 87.

[25] L. Breslau, et al., "Advances in network simulation", *IEEE Computer*, 2000, 33 (5), pp. 59-67.

[26] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing", In *Proc. of WMCSA*, 1999.

[27] "jxta.org", Available: http://www.jxta.org/.

[28] T. Klingberg and R. Manfredi, "Gnutella 0.6", Available: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[29] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", In *Proc. of ACM SIGOPS*, 2001.

[30] N. E. Baughman and B. N. Levine, "Cheat proof playout for centralized and distributed online games", In *Proc. of INFOCOM*, 2001.

[31] A. S. John and B. N. Levine, "Supporting P2P gaming when players have heterogeneous resources", In *Proc. of NOSSDAV*, Stevenson, Washington, USA, 2005.