# Increasing Scalability and Reliability of Self-Organizing Grids for Life Science

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

## Abstract

*This paper proposes a novel approach for increasing the scalability and reliability of self-organizing Grids comprising a large number of mobile Grid services and one or more mobile users. The devices hosting these Grid services are placed on live animals. The Grid services provide access to the sensors monitoring the animals' behavior. Each of them has a state containing the collected measurements. The increased scalability is achieved by application of a multi-hop routing protocol based on Distributed Hash Tables (DHTs). The reliability and availability of the state associated with Grid services is improved by exploiting the awareness of the underlying DHT routing to replicate the state data. In this paper we propose our approach and present a protocol based on our concept of state replication. We then prove the correctness of our approach by an extensive simulation comparing its performance with existing solutions.*

## 1. Introduction

The recent development of mobile network technologies together with increasing availability of fast, energy efficient microcontrollers and high capacity batteries offer numerous incentives for scientist doing field research involving monitoring of wild or domestic animals [1]. Providing scientists with access to field instruments opens new areas for the Grid technologies. However the mobility of users and instruments puts numerous challenges in front of the current Grid middleware which in most cases is meant to be deployed on the fixed network infrastructure.

In this paper we are interested in a scenario with Grid services put on wild or domestic animals and one or more mobile users. These services provide access to sensors monitoring animals' activity and are connected over radio communication. The collected measurements become a dynamically changing state of the Grid services. The major challenges include a large number of network nodes, limited range of their transceivers, their limited power and possible dispersion on a large area.

As we proved in [2], to provide a mobile user with an access to services behind the range of her radio transceiver we have to use the multi-hop communication. We need a communication protocol fulfilling the following requirements: (1) scalability to the extent of tens or even hundreds of Grid services, (2) self organization to handle the constantly changing network topology, (3) high reliability, (4) availability of state of Grid services, being temporarily out of range of multi-hop communication.

The most typical solution for providing self-organization and multi-hop communication in the mobile ad-hoc networks (MANETs) are multi-hop routing protocols like DSR [3], AODV [4], DSDV [5], TORA [6]. Unfortunately they do not provide sufficient scalability to support large network topologies [7].

The solution for providing self-organization and scalability are distributed hash tables (DHTs) like Chord [8], CAN [9] and Pastry [10] but in their original form they are targeted at fixed wired networks. The recent proposal for adapting Pastry for MANET environment, Ekta [11], is a cross layer system ranging from the network layer to the application layer where it exposes the DHT application programmer interface (API).

Our simulation shows that Ekta does not provide sufficient reliability for our scenario and does not address the problem of partial interconnectivity. The current approach for dealing with partial interconnectivity is Epidemic Routing [12, 13], which is not sufficient for us as we require more instant access to data than it can provide. We cannot use any existing solution based on exploring mobility [14, 15] as we cannot enforce any controlled mobility pattern on the monitored animals. Our solution is to support Ekta with an adapted data replication initially used in the PAST file storage system [10].

This paper proposes using Ekta for increasing scalability of the self-organizing Grid comprising a large number of mobile Grid services and one or more mobile users. We improve the reliability achieved in this approach and address the problem of partial interconnectivity by providing routing aware data replication. We propose a novel state retrieval and replication protocol built on top of Ekta using our approach.

## 2. Background

Since our study concentrates on using Ekta [11] as a routing protocol and adapting the replication strategy from PAST [16], which is a system based on Pastry [10] it is relevant to give a brief overview of Pastry, the PAST's replication strategy and Ekta.

### 2.1. Pastry

Pastry [10] is one of several application-level DHT routing protocols designed for wired networks. In a Pastry network each node has a unique 128-bit id and every routed message has an associated 128-bit key. The objective of Pastry is to route the message to a node with an id numerically closest to the key associated with the message.

In a Pastry network comprising 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. Every node in Pastry has state consisting of the leaf set, routing table and neighborhood set. Two first of them are used for routing messages and are set of node ids associated with IP addresses. The routing table is a two dimensional array having $\lceil \log_{2^b} N \rceil$ rows and $(2^b - 1)$ columns. At a row *n* are placed ids sharing first *n* and only *n* digits with the present node's id. The *n*+1 digit of an id in the routing table defines the column where it is stored. Each entry in the routing table can have more than one id/IP pair. The leaf set contains numerically closest ids to the present node: *L*/2 greater and *L*/2 smaller, where *L* is a configuration parameter.

The routing table and leaf set are used to find a next node on the route of the message being forwarded. Such a node should share with the message's key more digits than the id of the forwarding node or be at least numerically closer to the key than the id of the forwarding node. The nodes located closer to the forwarding node (in the sense of network distance) are preferred.

### 2.2. Replication in PAST

PAST [16] is a file storage system based on Pastry. A file is stored in *k* nodes with ids numerically closest to the key associated with the file. As the nodes' ids are assigned randomly, nodes with numerically close ids are very likely to be well distributed over the network, which offers good resilience against hardware problems and heavy network traffic. Since Pastry tries to route the message to a node with an id closer to the message's key, preferring nodes closer (in the sense of network distance) to the forwarding node, there is a great possibility that the message will be forwarded to a nearby node holding the replica. That increases speed of accessing the file.

### 2.3. Ekta

Ekta [11] is an efficient adaptation of Pastry to the MANET environment. It is based on integrating Pastry with the underlying multi-hop routing protocol. In Ekta every node has a leaf set and a routing table but every entry contains apart from the target node's id, the whole multi-hop route to this node, not only its IP address. If a lookup to the leaf set and the routing table returns a node to which the route is not known the node performs flooding based route discovery.

The procedure of joining a new node to the Ekta network is simplified in comparison to Pastry. The new node *a* generates its id by hashing its IP address. Then it floods a *Join* message, which is replied only by a node *b,* which is numerically closest to *a*. The *JoinComplete* message sent by *b* to *a* contains the *b*'s leaf set. The node *b* then notifies all the nodes from its leaf set about the arrival of the node *a*.

Ekta provides the functionality of a routing protocol but also provides applications with an API similarly as Pastry.

## 3. Data retrieval in DHT-based self-organizing Grids

### 3.1. The approach

In this section we describe our approach to data retrieval and replication in self-organizing Grids. It is based on using Ekta [11] as a routing protocol and adding into the protocol stack a new layer between Ekta and the Grid service application as shown in Figure 1. Our state retrieval and replication protocol replicates every period $p_1$ the state of the Grid service to the $k$ other Grid services with numerically closest Ekta ids. The ids of these services are obtained from the underlying Ekta.

Parameter $p_1$ should be set according to the frequency with which the Grid service's state changes. Too short period $p_1$ can cause high power usage, while too long one can decrease the advantage of using replication. The value of the parameter $k$ should depend on the relation between size of the Grid services' state and their storage capabilities.

When a user issues a query to retrieve state of a selected Grid service, it is sent over a possibly multi-hop route to the target Grid service. Every forwarding Grid service checks if it has the required state obtained by the replication and if so sends it to the user instead of further forwarding the query.

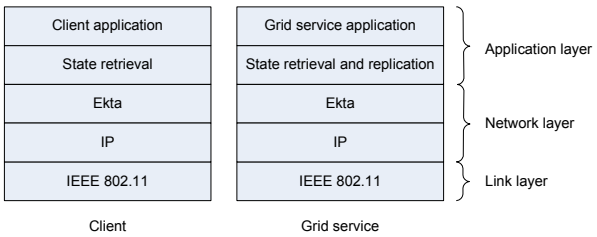


**Figure 1. Protocol stack**

### 3.2. The protocol

This section gives details about our state replication and retrieval protocol. As the protocol uses Ekta as its routing protocol, we use the following existing functions from the Ekta's API:

**route(Message, IP Address)** – Called by a higher level protocol to deliver the Message to the particular IP address

**route(Message, Key)** – Called by a higher level protocol to deliver the Message to a node with an id numerically closest to the Key

**deliver(Message)** – Called by Ekta to deliver the Message to a higher level protocol

We have to extend Ekta with the following API functions:

**LeafSet getLeafSet()** – Called by a higher level protocol to obtain the node's Leaf Set

**boolean forward(Message, Source Id, Destination Id)** – Called by Ekta before forwarding the Message to a next hop. A higher level protocol can cancel the delivery by returning false.

Our state replication and retrieval protocol provides following API functions:

**query(IP Address, Timestamp)** - Called by a client application to retrieve state from a Grid service with the given IP Address. Timestamp describes the maximal age of the state to be delivered.

**getData()** – Called by our protocol to get data from the local Grid service

**deliver()** – Called by our protocol to deliver state data to the client application which earlier invoked the query(IP Address) function.

Our protocol works in the following way. Every period $p_1$ it replicates the state of its local Grid service to $k$ other Grid services, the ids of which are numerically closest to the id of the local Grid service. These ids are obtained from the underlying Ekta by the getLeafSet() function. Every pair of subsequent replications is divided by period $p_2$, where $p_2$ is much smaller than $p_1$. The period $p_2$ is introduced to alleviate the danger of a flurry of control packets sent due to a possible sequence of route discoveries. The target Grid service saves the replicated state together with the timestamp of the moment when it is received.

Whenever a client application queries state of a selected Grid service, the query is forwarded by Ekta over a possibly multi-hop route. At every hop, which is a Grid service (not a user), our protocol checks if the current node has the required Grid service state not older than the timestamp specified by the client application. If it is the case, instead of further forwarding the query, the requested state is sent back to the user over Ekta.

## 4. Evaluation

### 4.1. Modeling methodology

This section describes the methodology we used to evaluate our approach. We simulated one mobile user and a herd of 70 domestic cows, each with one Grid service. The simulation area is a square of a 700m dimension. The cows move according to the random waypoint model [3]. Their speed varies from 0.6 m/s to 1.0 m/s as in the case of the real cattle [17]. When changing the direction of their movement they make

stops lasting from 0 to 100 seconds. The user moves from one of the corners to the opposite one at the speed of 1.5 m/s, which is an equivalent of a moderate speed walking [18]. The size of the retrieved Grid service state data is assumed to be 1KB. The initial layout of the user and cows is shown in Figure 2.

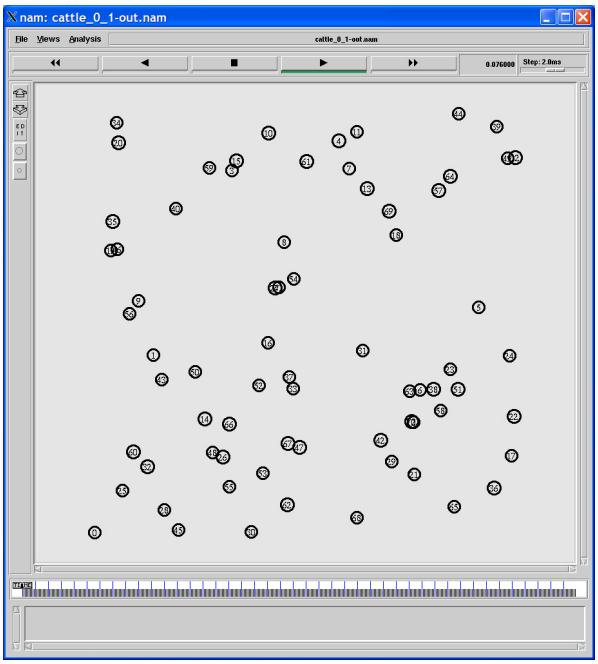


**Figure 2. Initial layout of cows (1-70) and the user (0)**

We use following state retrieval techniques: over AODV [4] and with our protocol on top of Ekta with the *k* parameter (number of replicas) equal to 0 (no replication), 2, 4 and 8. Initially we use DSR [3] instead of AODV. However DSR is not scalable enough for our scenario. During the state retrieval the massive flurry of control packets crashes the simulator. This situation is depicted in Figure 3. Finally we choose AODV [4] for our simulation as it is one of the most scalable from currently available MANET multi-hop routing protocols. The sequence of events in our simulation is presented in Table 1.

We use the ns-2 network simulator [19] and its AODV implementation. We implement Ekta as a routing agent and our protocol is implemented as an agent. We use CanuMobiSim [20] to generate movement patterns for cows and mhash [21] to generate Ekta ids from IP addresses. We choose MD5 hashing algorithm [22] because it generates 128-bit output, which has the length suggested for an Ekta id [11].

We measure routing overhead, packet delivery ratio (PDR) and average query time. The routing overhead is measured as the number of control packets sent by the routing protocol (AODV or Ekta) and as the total size of these control packets. In both cases each hop-wise transmission of a control packet is counted as one transmission. PDR is defined by the following formula:

$$PDR = \frac{answers\ \ received}{queries\ \ sent} * 100\%$$

The average query time is the average time from issuing a query to receiving results.

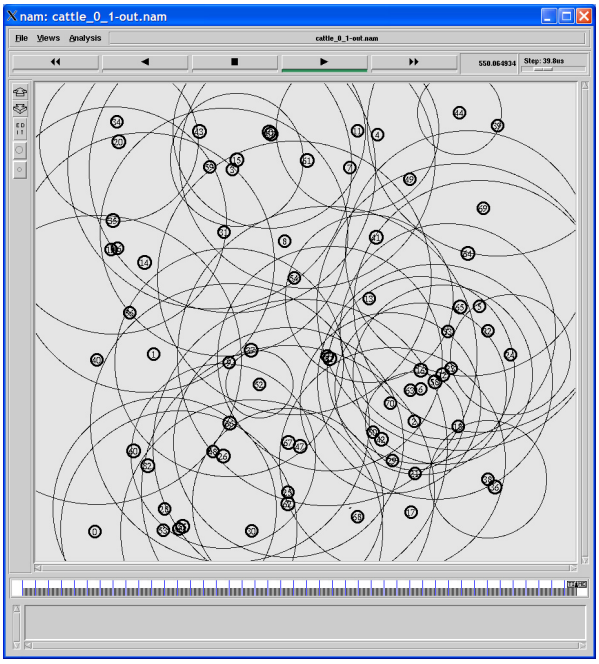


**Figure 3. Effects of using DSR protocol – broadcasts are depicted as circles**

**Table 1. Events in our simulation**

| Simulation time [s] | Event |
|---|---|
| 0.0 | Cows start moving and Grid services join the Ekta network (when applicable) – one service every 2 seconds |
| 200.0 | Replication of services' state (when applicable) – a service starts replication every 1 second |
| 250.0 | Starting point for packet statistics |
| 450.0 | Failure of the 10% of Grid services (when applicable) |
| 500.0 | The user joins Ekta network (when applicable) |
| 550.0 | The user issues a query (1 query every 0.1 second) and starts moving |
| 1000.0 | Simulation end |

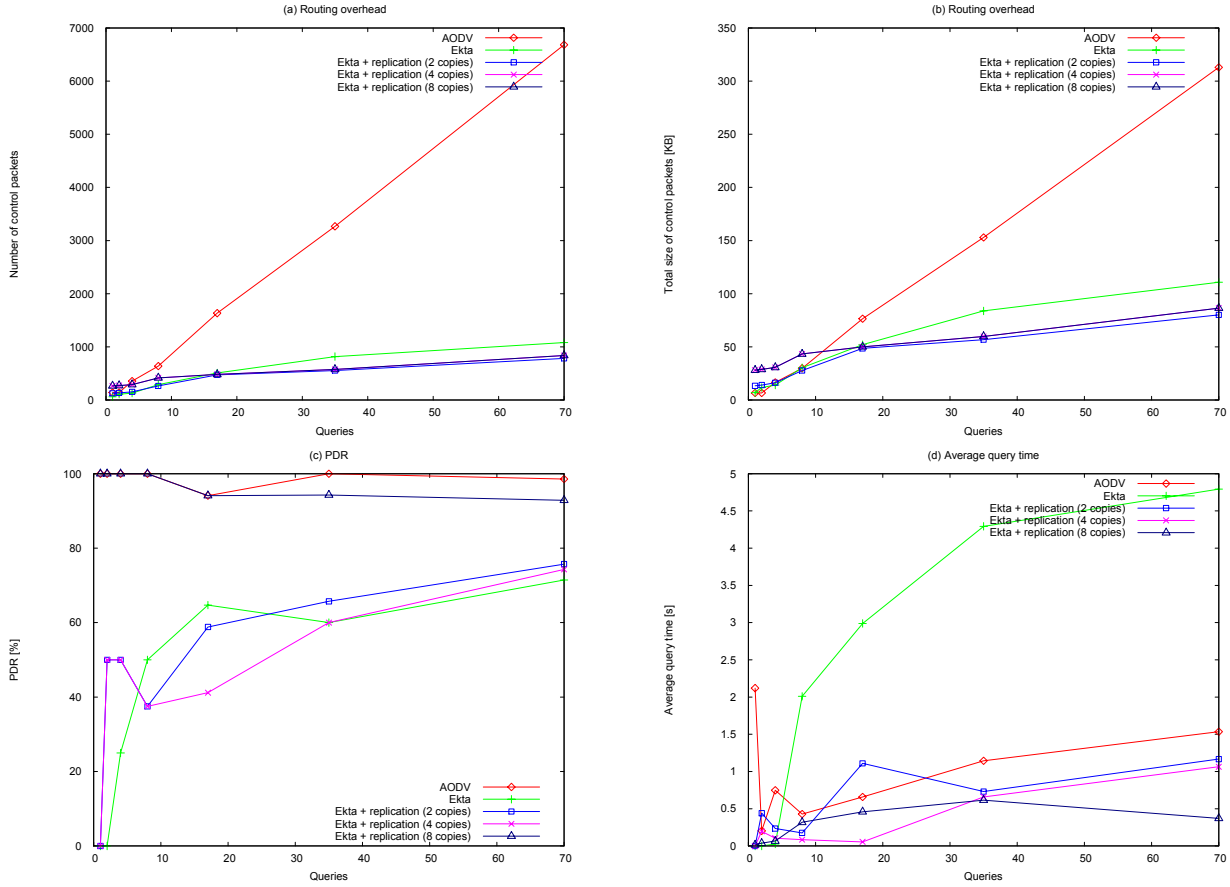


**Figure 4. Simulation results for the case with no failed Grid services**

We select a cutoff period of first 250 seconds of the simulation during which the Grid services' join the Ekta network and replicate their state. As mentioned in Table 1 we use certain time periods to divide similar actions performed by either Grid services or the user to avoid a flurry of control packets which is unlikely to happen in reality. For example the probability that many Grid services will join the Ekta network in one point of time is very small.

We perform our simulation for different number of queries issued by the user. We also consider two cases: (1) no failures of Grid services (2) failure of 10% of Grid services. The failures are simulated by moving cows with the 'failed' Grid services out of the simulation area. The results are presented in Figure 4 and Figure 5.

## 4.2. Performance results

In this section we discuss the results of our simulation. As we can see in Figure 4a and Figure 4b, the number and total size of control packets in case of AODV grow almost linearly with the number of queries. This means very high energy cost of queries and poor scalability. In contrast the number and total size of control packets in Ekta based communication grow much slower, which means better scalability. This is caused by the fact that the route discovery of AODV is fully on-demand (i.e. it is performed when it is necessary for routing a packet), while Ekta nodes have knowledge about the network acquired during the joining process and replication.

Although Ekta is better suited for our scenario than AODV, the biggest challenge in using it is clearly visible in Figure 4c. The reliability of pure Ekta in this scenario is much lower than in case of AODV. Fortunately it can be considerably improved by using the replication technique as we propose but to see the difference we must use a certain minimal number of replications (the *k* parameter). In this case the value of 8 offers a reasonable performance.

Another advantage of using replication is visible in Figure 4d. Using even a very small number of replications considerably decreases the retrieval time

in comparison to pure Ekta or AODV. It is due to the fact that the state of every Grid service is dispersed over the network nodes so in case of long routes, which mostly contribute to the high average of query times, the probability of finding a replica of the required state before reaching the original is high.

Comparing Figure 4 and Figure 5 we can notice the difference in dealing with partial interconnectivity achieved by using replication. The failure of 10% of Grid services has a considerable impact on PDR in case of AODV and pure Ekta but Ekta supported even with a small number of replications remains almost unaffected. The failure of Grid services does not even influence the Ekta's routing overhead.

To summarize, our simulation confirms the advantage of supporting Ekta with the state replication. It considerably improves reliability, lowers the effect of partial interconnectivity and shortens the duration of the queries.

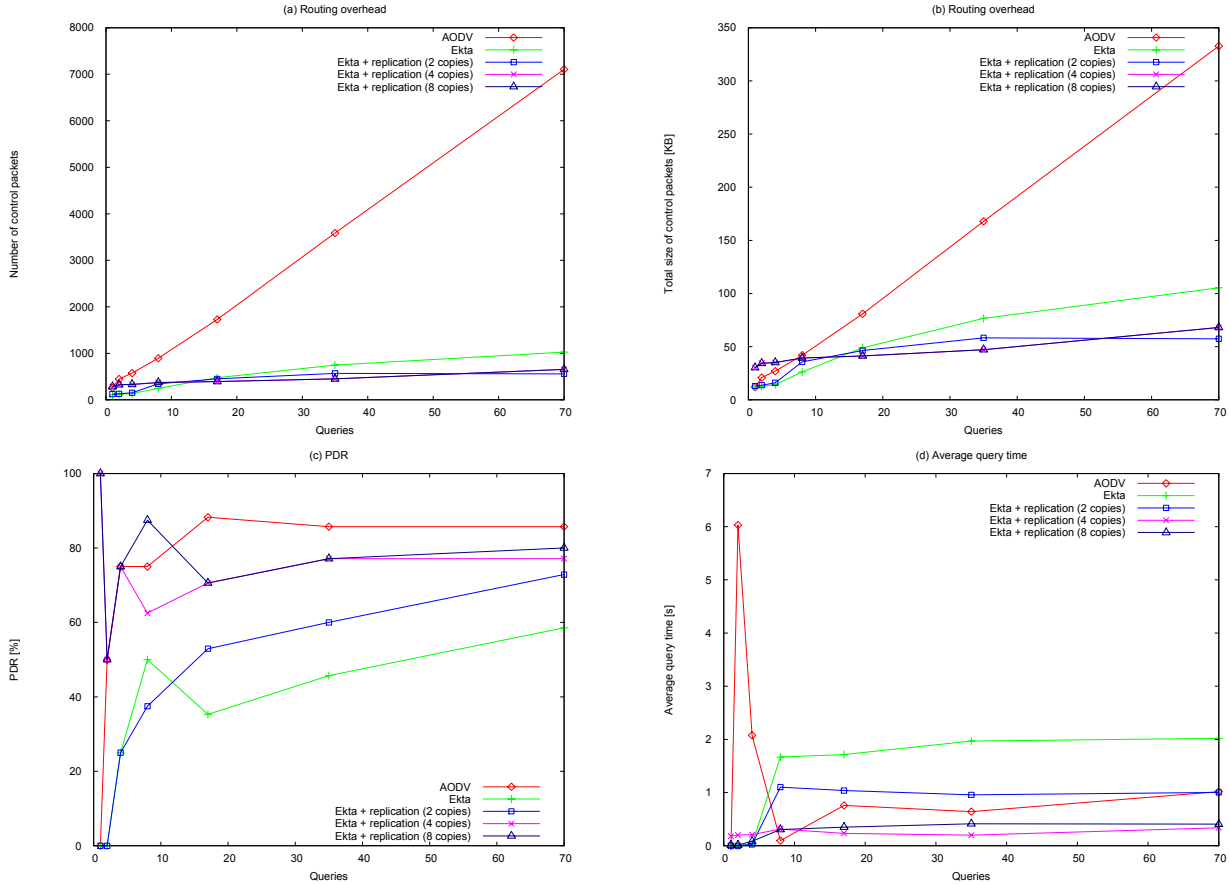


**Figure 5. Simulation results for the case with 10% of Grid services failed after the replication**

## 4.3. Further observations

This section presents additional observation we make analyzing results from our simulations. A large number of network nodes heavily increases the overhead caused by flooding, even when it is done in a controlled manner as in Ekta [11], which borrows the controlled flooding from DSR [3]. Therefore in larger MANETs flooding should be avoided, particularly a case when many nodes perform flooding simultaneously. This is a reason why in our scenario differentiating time of Ekta's *node joins*, or querying particular Grid services makes a difference.

Simultaneousness of Ekta's *node join* operations can also degrade the quality of the Ekta's state, resulting in longer routes or even failures in packet delivery to distant nodes. This is caused by the fact that new nodes retrieve from existing nodes leaf sets, which are not fully populated, so it is beneficial to disperse the *node join* operations over the time. Fortunately in reality it rarely happens that many nodes perform the *node join* operation at the same time.

## 5. Future work

Domestic cattle kept in large herds tend to break up into small groups of 10-12 animals, when enough space is available [17]. Such behavior can be exploited to select next-hops in a more reliable way and perform a more effective replication.

The knowledge about which ids belong to the members of a sub herd (mates) of a cow carrying a Grid service can be collected without additional data exchanges. It can be inferred from the long time statistics collected during analyses of the forwarded or overheard packets.

Preferring mates in choosing next-hops can improve the probability of successful delivery of a packet to the next hop, so delays caused by resending a packet to a different next-hop can be avoided. This can result in shorter query times.

State replication can be more effective if the replication targets are chosen from non-mates, which are Grid services with which the contact is more likely to be lost. This results in the higher availability of states from the Grid services which are beyond the range of the multi-hop communication. Such an approach combines the advantages of DHT based routing like Ekta with Epidemic Routing [12, 13].

As our future research we want to evaluate this ideas by performing simulations of longer scenarios and modeling cows' mobility using the Reference Point Group Mobility model [23].

## 6. Conclusions

In this paper we proposed the use of Ekta [11] to increase the scalability of a self-organizing Grid comprising services mounted on animals and one or more mobile users. We proposed a novel approach of using data replication to increase the availability of state associated with Grid services and reliability. We then presented a protocol using our approach and proved its validity over an extensive simulation.

## 7. References


[1] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet", In *Proc. of SenSys*, Baltimore, Maryland, USA, 2004.

[2] M. Radenkovic and B. Wietrzyk, "Life Science Grid Middleware in A More Dynamic Environment", In *Proc. of Grid Computing and its Application to Data Analysis (GADA'05)*, Ayia Napa, Cyprus, 2005.

[3] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", 1996.

[4] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing", In *Proc. of Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999.

[5] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", In *Proc. of ACM SIGCOMM'94*, London, England, UK, 1994.

[6] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", In *Proc. of IEEE INFOCOM'97*, 1997.

[7] Y. C. Hu, S. M. Das, and H. Pucha, "Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks", In *Proc. of Hot OS IX*, Kauai Marriott Resort & Beach Club, Lihue, Hawaii, 2003.

[8] 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.

[9] 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.

[10] 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.

[11] 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 Sixth IEEE Workshop on Mobile Computing Systems and Applications*, English Lake District, UK, 2004.

[12] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks", Duke University, 2000.

[13] J. A. Davis, A. H. Fagg, and B. N. Levine, "Wearable Computers as Packet Transport Mechanisms in Highly-Partitioned Ad-Hoc Networks", In *Proc. of ISWC'01*, 2001.

[14] Q. Li and D. Rus, "Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks", In *Proc. of ACM MobiCom*, 2000.

[15] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks", In *Proc. of MobiHoc'04*, Roppongi, Japan, 2004.

[16] 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.

[17] C. J. C. Phillips, *Cattle behaviour*, Farming Press, Ipswich, UK, 1993.

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

[19] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation", *IEEE Computer*, 2000, 33(5), pp. 59-67.

[20] I. Stepanov, J. Hahner, C. Becker, J. Tian, and K. Rothermel, "A Meta-Model and Framework for User Mobility in Mobile Networks", In *Proc. of 11th*

*International Conference on Networking 2003 (ICON 2003)*, Sydney, Australia, 2003, pp. 231-238.

[21] N. Mavroyanopoulos and S. Schumann, "mhash".

[22] R. L. Rivest, "Request for Comments: 1321, The MD5 Message-Digest Algorithm", Network Working Group, 1992.

[23] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks", In *Proc. of 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, Seattle, Washington, United States, 1999, pp. 53 - 60.