Using Moving Object Databases to Provide Context Information in Mobile Ad-hoc Networks

Joos-Hendrik Böse[1], Katharina Hahn[1], Manuel Scholz[1], Heinz Schweppe[1], Agnès Voisard[1,2]
[1]*Institute of Computer Science, Freie Universität Berlin*
[2]*Fraunhofer Institute for Software and Systems Engineering (ISST)*
{boese,khahn,mscholz,schweppe,voisard}@inf.fu-berlin.de

Abstract

Applications deployed in mobile peer-to-peer systems have to cope with mobility and fluctuation of nodes. These dynamics restrict the reliability of communication links as well as the availability of certain nodes, services, and resources. Exploring the context of nodes can be used to anticipate these shortages. Context-awareness enables improvement of performance of systems and enables adaptation of applications in order to make their behavior more relevant with respect to their current situation. Pre-evaluation of context can be used to estimate future availability of crucial resources and reliability of communication links. Which information is considered to be relevant context information varies according to the application scenario. In this paper, we present a model for context information along with a mobile service that provides a local view of surrounding nodes in a Mobile Ad-hoc NETwork (MANET). The service is designed to enable efficient prediction of future stability of the vicinity of a node.

1. Introduction

Handheld devices with wireless communication capabilities now allow users to offer and use services in their local environment without being connected to a wired network. Users spontaneously form so called MANETs to make use of their connectivity. These mobile peer-to-peer networks can be used to corporately reach individual or shared goals. An example is an eLearning campus in which students carry laptops, PDAs or smartphones and share and trade information or electronic goods, for example minutes of lectures or collectively solved homework. Also in disaster areas, where structured networks are not provided, spontaneous wireless communication can be used in order to manage recovery measures.

In this paper, we present a service to locally provide context information about surrounding nodes. This system is used to reason on the stability of the vicinity of a node based on the provided information. As we want to evaluate information of several nodes we introduce a multi-level context model to distinguish between levels and types of context information. This distinction is made in order to classify information according to the originating nodes. Nodes interested in their vicinity gather information on the situation, capabilities, and status of surrounding nodes.

The stability of the MANET is the main criterion responsible for success or failure of communication between nodes. We consider the collaboration of heterogeneous devices as we assume that devices differ in their technical configuration as well as in mobility. Mobility and fluctuation of nodes lead to instability of communication links and availability of resources. These are the shortages that need to be overcome (in opposition to the fixed wired scenario) in order to be able to support successful interaction.

We want to enable the evaluation of the stability of a region of the network by locally presenting relevant information and thereby enable pre-evaluation about the future. To this end we present a system that adapts concepts of Moving Object Databases (MOD) of the fixed server/mobile client scenario. Those are well suitable for the pre-evaluation of future situations in dynamic environments. The presented service is the foundation to establish a context pre-evaluation strategy to reason on local stability. It focuses on modeling and storing information, as well as efficiently exchanging information between nodes.

The paper is structured as follows. Section 2 presents background information, which includes the common notion of context and its adaptation to our particular case. Section 3 presents our data model to capture context information as well as the way to query it. The exchange of context information of involved nodes and establishing a local view on the MANET is described in Section 4. Section 5 introduces a representative application scenario. Finally, Section 6 draws our conclusions and briefly depicts future work.

2. System Model

We assume that nodes participating in the MANET act cooperatively to reach individual or shared goals. This means that all nodes participate in providing, storing and forwarding information according to their capabilities. Such behavior can be incited by repudiation or incentive systems. We also assume the MANET to be heterogeneous according to the participating nodes. Devices mainly differ in capabilities, e.g., laptops have more processing power than PDAs, as well as in mobility, e.g., users using smartphones will constantly change their location opposed to users using laptops. Based upon sensing these attributes and differences, an evaluation of behavior and prediction of future behavior can be done.

Peers communicate using wireless technologies such as IEEE 802.11 WLAN. Nodes are equipped with a positioning sensor, for example GPS, so that they are able to sense their position within the system. In general, scenarios for MANETs are found at places where fixed-wired infrastructure is not available at all or it is not available at every place. Wireless ad-hoc communication will be used instead. We do not assume any previous knowledge about the spatial surroundings of nodes (e.g., a map).

In order to be able to evaluate the stability of certain regions of the MANET we gather knowledge of surrounding nodes. Therefore every node interested in context information carries a local database in which relevant data of its neighbors is stored. We use a mobile database that is able to monitor moving objects and therefore easily enables pre-evaluation for knowledge about future situations. This is the foundation for forecasting future availability and reliability of communication links and nodes.

Considering mobile and wireless communicating devices most of the existing context systems assume fixed powerful servers that collect and evaluate context data of mobile nodes [10,11]. Other existing approaches are developed precisely for one target platform [17]. A few attempts on context frameworks distributed on mobile nodes have been made in the past few years [12, 15]. Our design focuses on a multi-level context model to distinguish between different context information originating from different nodes. This model is integrated in an architecture that is well suitable to monitor mobile nodes. Our system focuses on efficiently gathering and exchanging context information in order to establish a local view on the network. The proposed system is designed to be part of a middleware suite that supports transactional collaborations in MANETs (see Section 5).

2.1. What is Context?

Gathering and evaluating context information is currently a broadly studied field as it can be profitably used to advance mobile services. The general goals of examining context and providing context-awareness are to improve performance of systems and make application behavior more relevant with respect to the situation it is being used in [1,6].

Several definitions of context have been made. Schilit et al. [16] refer to context as the answer to the questions: “Where are you?”, “Who are you with?”, and “What resources are nearby?” They distinguish between three categories of context sources, namely computing environment (e.g., communication costs, network connectivity, etc.), user environment (e.g., user’s profile, preferences, etc.) and physical environment (e.g., temperature, lightning, etc.). A similar context definition has been made by Pascoe [14] who refers to context as the subset of physical and conceptual states of interest to a particular entity. A more general definition of context is introduced in [7]. The authors state context *being any information that can be used to characterize the situation of an entity*.

Which information is considered to be valuable to characterize the situation widely varies according to the application scenario. In wireless sensor networks, context often refers to neighboring nodes, their signal strength and distance. In MANETs, the goal of context-awareness often focuses on the availability of resources and services. In ubiquitous computing and location-based services, context information also includes the user’s real-life environment such as “user is within a certain room” as well as device configurations and user profiles [9]. Those can be also used to derive situation-awareness as formalized in [13]. In client-server applications, context also refers to the environment where the system is deployed (e.g., certain company, buildings, etc.).

We concretize the cited definitions to match our ambition to evaluate the stability of nodes within a certain region of the MANET. We therefore add goal-orientation and refer to context being *any information that can be used to characterize the situation in order to reach a certain goal;* in our case pre-evaluating the availability and stability of surrounding nodes.

2.2. A Multi-level Context Model

We use a model-oriented context management approach (as opposed to an ontology-oriented approach like in [9]) to formalize context. In our model we distinguish between different levels of context (see Figure 1). At the *local level*, context information refers

to information of a single node. This can be either any *atomic* attribute sensed by any kind of sensor (e.g., CPU load), or *derived* attributes that have been derived by one or several atomic attributes (e.g., speed or remaining battery time).

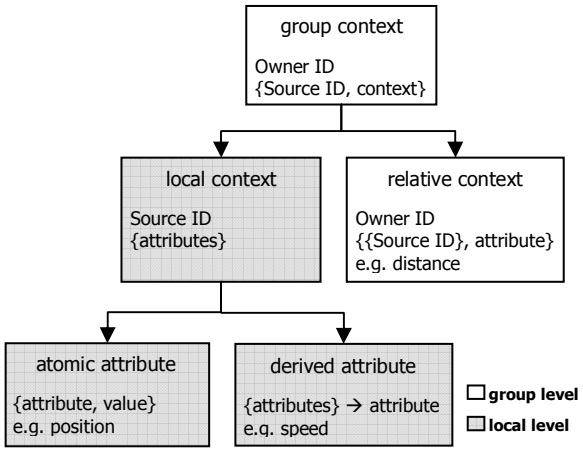

Figure 1: Multi-level context model

At the *group level*, context information of several nodes is considered. At this level we distinguish between purely *local* and *auxiliary* context information. Purely local context information refers to context information of the local level, e.g., several attributes of one node. Auxiliary context data includes the aggregation of attributes originated by several nodes (e.g., distance between two nodes).

We define, according to this model, at the local level the *local context LC* of a node to be:

$$LC := (id, t, \{av\})$$

with *id* being a unique identifier of the node, *t* the timestamp of the local context and *{av}* being a list of attribute-value-pairs considered relevant to characterize the situation. Attributes can be either atomic or derived attributes. At the group level, the *context C* of a node is defined as:

$$C := (id, t, \{LC\})$$

with *id* being the unique identifier of the node, *t* is the timestamp, and *{LC}* a list of local contexts that characterize considered nodes. The *LCs* a node stores within its *C* are either those of a certain region of interest or those of a certain group of nodes. Context evaluation to reason on the stability of the local surrounding is done locally based on the context *C* a node acquired.

3. Storing Context

We propose that every node interested in evaluation of context information stores *LCs* of surrounding nodes within its local database. This section explains how *LCs* are modeled and stored. Which attributes (within *{av}*) are considered to be relevant depends on the device (the attributes/services it can offer) as well as the used strategy to evaluate.

We adapt the concepts of MOD (Moving Object Databases) in fixed server/mobile client scenarios. Those are often applied in transportation networks where moving vehicles equipped with positioning sensors are monitored. Moving objects are used as data sources which transfer their attributes to a server. Mobile nodes in a MANET also serve as data sources. However these nodes are also part of the MOD as they are involved in collecting, forwarding and evaluating information of their neighborhood in order to establish a local view of the network.

In the following subsections we briefly recall the concepts of modeling moving objects as introduced by [18] and describe how these are adapted to our case.

3.1. The MOST Data Model

The concepts of MOD of Client/Server environments are well suitable to model moving objects in mobile environments. A common data model for the conventional system model is the MOST (Moving Object Spatio Temporal) data model [18]. It contains the possibility of modeling attributes dynamically (that is to consider the value and the change of the value), submitting different query semantics (“may” and “must”) and query types (namely instantaneous, continuous, and persistent) as well as the query language FTL (Future Temporal Logic) that enables temporal queries.

Values of conventional attributes are explicitly changed while locking the attribute accordingly. Modeling the position of a moving object dynamically (because the value continuously changes) enables to model location changes without having to lock and update the attribute every time a new position is sensed. Considering the position as a composite attribute (consisting among others of the position and the movement) instead of one value enables interpolating between times when no explicit update has taken place. Even extrapolating the present or future positions according to knowledge of past points in time is possible. This is not the case for conventional static attributes. Querying those, one can only get the value of the last update.

Querying extrapolated and interpolated values of those strongly time-dependant attributes, the MOST data model distinguishes between two different query semantics: may and must, which refer to the estimated whereabouts of moving objects. If the whereabouts intersect the region of interest of a query, the actual position might ("may") be within the specified region whereas if the whereabouts are completely within the region of interest, the actual position will ("must") be within the specified region.

Considering the database states that are queried, MOST distinguishes between instantaneous, continuous, and persistent queries. Instantaneous queries are those that refer to the current database state whereas continuous queries are a sequence of instantaneous queries. They have a given starting and an ending point. Persistent queries also refer to a certain time interval. As opposed to continuous queries, they relate to database states from the starting point on until the query is finished. Therefore persistent queries save database states during evaluation of the query. (e.g., "Select all objects that will double their speed within the next 5 minutes.").

MOST proposes to use FTL (future temporal logic) as query language. FTL is based upon SQL and expands it by two temporal operators (*f Until g* and *Nexttime g*). More time-dependant operands can be derived from them.

3.2. Dynamic Attributes

Most of the attributes that can be used to predict future connection status can be conventionally modeled. These attributes within *{av}* of a node's *LC* are stored in a relational database. Their values are explicitly updated. The attributes that can be considered are either provided by the hardware of the node, represent a node's state and might as well, include information about basic services that can be offered (e.g., safe storage, atomic and derived context attributes, and so on). In the following, we refer to the source of any attribute as a sensor.

Dynamic attributes have been introduced for attributes whose values constantly change and depend on previous values. To avoid update overhead and permanent locking of the attribute, the value of the database is not updated any time the sensor senses a new value. In order to still have accurate values within the database, the actual value is stored along with its change. We store a dynamic attribute *A* as a composition consisting of four sub-attributes:

1. The timestamp of the last update, in order to know when the value has been sensed, is stored in *A.updatetime.*
2. *A.value* denotes the value of *A* at the time *A.updatetime.*
3. The change of the value is modeled as a function of time. *A.function(t)* indicates the change of *A.value* according to the last known update. The value of *A* at time t_i according to the last known update ($t_i > A.updatetime$) can be determined by:
$$A.calculated(t_i) = A.value + A.function(t_i)$$
4. The change of the value (*A.function(t)*) is an estimation about future behavior of *A.value.* Therefore we also observe the allowed deviation between $A.calculated(t_i)$ and the sensed value at that time $A.sensed(t_i)$. It is defined by the nonnegative value stored in *A.uncertainty*. If no updates are missing, one can assume that at the current time (t_c) the following is true:
$$A.calculated(t_c) - A.uncertainty < A.sensed(t_c)$$
and
$$A.sensed(t_c) < A.calculated(t_c) + A.uncertainty$$
Whenever the deviation is greater than *A.uncertainty*, an update of the attribute is created.

Since mobile environments are dynamic as far as mobility and stability are concerned, several attributes should be modeled dynamically. For example the remaining battery time (rbt) of a node and its position should be stored dynamically.

For the position, *position.function()* can be determined according to a node's current speed and direction each time an update is sent (e.g. as proposed in [19]). For the remaining battery time (rbt), *rbt.function()* will always be constant. As it is a countdown-value, the remaining battery time is decremented by 1s per second. Note that *rbt.function()* is constant however *rbt.value* can still be explicitly updated.

We propose to update the uncertainty values of dynamic attributes whenever an update is being sent. This enables adaptation of the update precision to the node's behavior and reduction of the information cost as described in [18,20]. The information costs which model a degree for information validity at cost for retrieving information are determined by considering deviation costs, update costs and uncertainty costs. For example, a greater uncertainty value leads to imprecise values but saves update costs.

Allowing changes of the uncertainty value whenever an update is being sent enables adaptation of the uncertainty to the current behavior. We propose to adapt the uncertainty to the semantics of an attribute. This can for example include choosing more sensible uncertainty values whenever the movement of a user is fast or the distance to a certain node is critical in terms of its transmission range. It also includes choosing a more robust uncertainty value for the remaining battery time, whenever a node's remaining power is low to

avoid frequent updates that charge the power spares even more.

3.3. Querying Context

Context queries will be submitted on the local database. Distributed query processing is not intended. Each node interested in the stability of its surrounding will query its local context database. We propose that a set of temporal queries will be implemented within the context database. Therefore it will be possible to either submit those given temporal queries or query conventionally using the standard database interface (e.g., SQL).

Although the implemented set of queries limits the evaluation process that can be done, we propose this procedure in order to keep the system light-weight. In [18], the authors describe an FTL query processing algorithm which enables to process temporal queries on the described data model. The mobile moving object database system we propose serves the specific purpose to build the foundation for a context evaluation service that evaluates gathered data to reason on the stability of a certain group of nodes or region of the MANET. We suppose that the set of pre-evaluated information queried by different prediction methods to evaluate the stability will strongly overlap. The questions, context evaluation schemes possibly answer, are for example:

- "How long will we stay connected?"
- "Where will node A be in 5 minutes?"
- "How many stable nodes are within transmission range?"

Answering these questions always relies on prediction of attributes of mobile nodes and their future behavior. Providing the prediction of crucial information and dynamic attributes within the implemented set of queries will therefore serve different prediction schemes. That is why we assume that different stability prediction schemes will have most queries (querying those future states) in common. We therefore set aside to completely implementing FTL as a query language, as we think that we can thereby save unnecessary costs in the mobile peer-to-peer environment

The MOST data model distinguishes between two query semantics (must/may). These state whether certain regions of interest have to overlap or include each other. Both query semantics are relevant for the introduced purpose. Considering context evaluation and communication links, intersection and overlapping of possible whereabouts with transmission ranges of nodes will be used as the foundation for a probability that nodes will still be connected. This can be based on whether the transmission range includes, overlaps or excludes future whereabouts. The degree of overlapping of future whereabouts with possible transmission ranges will be important to quantify the future availability of nodes.

4. Exchanging Context

The last section described how to store and query *LCs*. This section deals with the exchange of *LCs* between mobile nodes in order to establish context at the group level (establish *C* at a node's database).

We propose to use a simple reactive context update scheme. This means that only when context evaluation is demanded by an application the nodes involved collect, store, and forward the according context data. In [5], different location services are studied which deal with exchanging position information over a MANET in order to provide position information about every node. Those services can be used by routing algorithms that demand for location information. Three different services are compared: two proactive services and one reactive service. The conclusion that is drawn is that the simplest proactive service, which states that a node will transmit its location package to its neighbors at a given rate that adapts to its movement, is to be preferred over the others as it offers higher performance and lower overhead than the others.

The scheme we propose to disseminate *LCs* between nodes is similar. However, several aspects are handled differently since we do not assume a constant need for context information of any node at any place of the network. In order to prevent message overhead, we propose only to exchange *LCs* when demanded by an application. This leads to a greater response time to gather and evaluate a group context than proactively gathering group context information. However it saves messages that need to be transferred.

We classify the nodes involved in exchanging *LCs* into three main categories:

- Nodes that require to gather a group context of nodes broadcast and gather information about involved nodes as long as context evaluation is necessary. They are called *initiators*.
- Nodes that are contacted in order to collect and broadcast information as long as any initiator is within transmission range are categorized as *sources*. Those nodes have to collect their local context and communicate it.
- Nodes that are currently not involved in any context dissemination process are classified as *inactive*.

As a simple update scheme, we propose to store and exchange *LCs* of nodes in single-hop environment.

Therefore a group context involves local context data of nodes that are currently within its single-hop range. When interaction between nodes in multi-hop distance is part of the supported application, we assume that multi-hop communication is available. It will therefore be possible to demand *LCs* of specific nodes within the MANET to establish a group context of a specific group of nodes.

Whenever any node initiates to collect information to build a group context, it broadcasts its *LC* along with its category (*initiator)* to its neighbors in order to trigger those to participate in exchanging *LCs*. Whenever a node being *inactive* receives an *LC* of an *initiator*, it will change its state to *source.* This way, the initiator informs nodes to sense and exchange their *LCs*. In the next two subsections, we describe how and when *LCs* are exchanged. We imply that initiation has been taken place; therefore the *initiator* and nodes being *sources* already know about their category.

4.1. Event-Triggered Context Exchange

Updates of conventionally modeled attributes are detected whenever the sensor senses a new value. It is therefore dependant on the scale factor of the sensor whether an update is detected or not. Whenever the value is updated, a node being *initiator* or *source* will inform the according nodes.

Dynamically modeled attributes (described in section 3.2.) are stored along with their uncertainty value. Whenever *dyn_attribute.value* of a node of the class *initiator* or *source* differs strongly from the values its neighbors expect, it updates its local context and communicates it. According to the last sent *LC* a node can determine which values its neighbors evaluate. The allowed deviation depends on the previously exchanged *dyn_attribute.uncertainty*. Whenever a node exceeds the deviation it updates its *LC* and broadcasts it. The receiving neighbors of class *initiator* or *source* update the according *LC* in their local database. Using this event-triggered-update strategy, a node can be sure (unless it is missing any updates) that the nodes it examines within its local database are where it expects them to be.

If any node, being *inactive*, receives the local context of the initiator, it will set its current state to *source* and respond to the initiator by sending its information. This way, information about current neighbors will be kept up-to-date.

An inactive node receiving local context information of a node of class *source* will not trigger any information exchange as it is not involved in the region of the network to be included in the group context. The categories of nodes are used to determine the border of the evaluated region.

4.2. Time-based Context Exchange

Whenever no event-triggered information exchange has taken place, a timeout T_{ex} will be in charge to initiate an update of the local context to be sent. This way, one can learn about newly arrived nodes within the neighborhood (as the inactive nodes will respond by sending their *LC*) as well as determine which neighbors might have disappeared. If nodes, stored in the local database, do not send any information within another previously chosen timeout (denoted by T_{del} which is to be chosen greater than T_{ex}), their *LC* will be deleted, as one can assume that they have moved out of transmission range or were shut-down. If a node of class *source* does not receive any local context information of the *initiator* within this timeout, it will change its state from *source* to *inactive*.

If update messages get lost due to the unreliable wireless communication channel, the timeout T_{ex} is in charge that at least one additional update message is sent before the information of a surrounding node is locally deleted (as $T_{ex} < T_{del}$). Depending on the value of T_{del}, a node N_i will trigger one or several updates before surrounding nodes will delete the local entry for N_i.

Figure 2 shows the three categories of nodes within a network. The *initiator* sends its *LC* as well as its status to its neighbors. They change their state from *inactive* to *source* (e.g., nodes A and B in Figure 2.a). As the initiator moves and its position differs greatly from the prediction, it will trigger to send an update of its *LC*. The *inactive* nodes receiving this *LC* change their state from *inactive* to *source* and start storing and sending their local context (e.g., nodes C and D in Figure 2.b.).

Since node A is not in transmission range of the coordinator anymore, it will not receive any update of initiator's *LC*. The timeout T_{ex} is responsible for sending updates even when no significant changes have taken place. Therefore node A will soon assume not to be in the initiators surrounding anymore. It changes its state from *source* to *inactive*.

These concepts can easily be adapted to establish group context information in a multi-hop environment. In order to do so, a few categories have to be added as well as a parameter that denotes the number of hops in order to limit the distance in which nodes are involved in exchanging their local context.

5. Application Scenario

The goal of this system of collecting and evaluating context information is to adapt application behavior to the stability of the local surroundings in the MANET. We want to give application support by evaluating the stability of the local MANET surroundings.

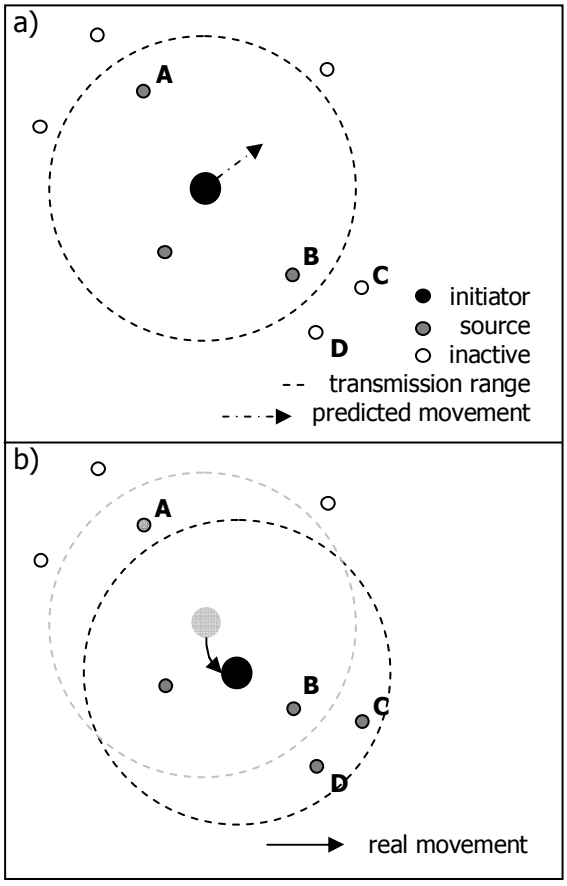

Figure 2: Categories of nodes involved in the group context.

In [4], an eLearning scenario is described in which students form a MANET using their devices equipped with wireless communication capabilities. Electronic values are introduced so that students are able to share and trade electronic goods, e.g., minutes of lectures or homework. Purely mobile transactions in MANETs are likely to fail as their participants might disappear. In order to still guarantee transactional qualities, such as an atomic commit, information about the transaction has to be preserved within the network. In [4] a middleware platform for such transactional support is described. Transaction logs are distributed within the MANET to enable recovery strategies for failing nodes. Nodes that suffer from disconnection are able to learn about missed protocol steps or decisions later on and are therefore able to get back to the current context of the transaction.

Different log dissemination strategies are considered to avoid flooding the network [3]. In order distribute reasonably, the dissemination strategies are adapted to the stability of the nodes' context. One dissemination strategy, the directed dissemination, aims at evaluating the stability of the surrounding of nodes involved in the transaction. This is to be done in order to find highly available nodes. One wants to estimate the probability of at least one of those nodes, and therefore when using those for log preservation, at least one log item, being accessible within the network at a later point in time. Such an estimation of future availability demands for context-evaluation of the current context of nodes.

The proposed context model suits the requirements of this scenario gather local context information in order to evaluate information at the group level. On top of this architecture, evaluation and prediction schemes can be easily established in order to give such availability predictions as demanded in [3].

6. Conclusion

In this paper we have introduced a context model and a corresponding service that enables to model, store, gather, and query context information of a certain group of nodes in a region of the MANET. The context model defines certain levels of context that are needed to reason on the stability of the vicinity of a node. The system architecture uses a mobile MOD to store context data of mobile nodes. It establishes a local view on a certain region of the MANET. This enables pre-evaluation of context information in order to support prediction of future availability of nodes and stability of communication links. The system is designed to be integrated into a middleware platform that supports mobile transactions. In the proposed application scenario, context evaluation is needed to evaluate the probability of an atomic commit as well as to support the recovery mechanism to reach this probability at possibly low costs.

As part of future work, we want to integrate a prediction scheme for context data into the proposed service. Several strategies which focus on prediction of context information based on the current situation already exist [2,8]. We plan to adapt those to our infrastructure. The idea is to use these predictions in order to evaluate certain quality of service parameters according to the current surrounding of a node. These are used to evaluate the stability based on the availability of nodes and reliability of communication links.

7. References

[1] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith and P. Steggles: "Panel: Towards a Better Understanding of Context and Context-Awareness". In *Proceedings of the 2000 Conference on Human Factors in Computing Systems,* The Hague, The Netherlands, April 2000.

[2] C. Anagnostopoulus, P. Mpougiouris and S. Hadjiefthymiades: "Prediction intelligence in context-aware applications". In *Proceedings of the Sixth International Conference on Mobile Data Management 2005* (MDM), Ayia Napa, Cyprus, May 2005.

[3] J. Böse, S. Böttcher, K. Hahn, M. Scholz and H. Schweppe: „A Probabilistic Model for Transaction Persistency in MANETs". In *Proceedings of Mobilität und Mobile Informationssysteme 2006* (MMS), Passau, Germany, February 2006.*To Appear*.

[4] J. Böse, K. Hahn, L. Pelz and M. Scholz: „Optimistic Fair Transaction Processing in Mobile Ad-Hoc Networks.". *Technical Report B-05-22*. FU Berlin, December 2005.

[5] T. Camp, J. Boleng and L. Wilcox: "Location Information Services in Mobile Ad Hoc Networks". In *Proceedings of the IEEE International Conference on Communications* (ICC), 2002.

[6] A. Dey: "Understanding and Using Context". *Personal and Ubiquitous Computing*, Volume 5, Issue 1, Feb. 2001, Pages 4-7.

[7] A. Dey and G. Abowd: "Providing Architectural Support for Building Context-Aware-Applications". PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[8] K. Gopalratnam and D.J. Cook: "Active Lezi: An Incremental Parsing Algorithm for Sequential Prediction". In *Proceedings of the Florida Artificial Intelligence Research Symposium*, 2003.

[9] T. Gu, X. Wang, H. Pung and D. Zhang: „An Ontology-based Context Model in Intelligent Environments". In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference* (CNDS). San Diego, 2004.

[10] H. Höpfner: *Towards Update Relevance Checks in a Context Aware Mobile Information System*, In Proceedings of *INFORMATIK 2005 - Informatik LIVE!*, vol. 2, *Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V.* (GI), Bonn, September 2005.

[11] G. Judd and P. Steenkiste: "Providing Contextual Information to Pervasive Computing Applications." In *First IEEE International Conference on Pervasive Computing and Communications* (PerCom), p. 133-142. IEEE Computer Society Press, 2003.

[12] Mayrhofer R.: "An Architecture for Context Prediction". *Advances in Pervasive Computing, Part of Second International Conference on Pervasive Computing* (PERVASIVE), Austrian Computer Society (OCG), vol. 176., April 2004.

[13] U. Meissen, S. Pfennigschmidt, A. Voisard and T. Wahnfried: „Context- and Situation-Awareness in Information Logistics". In *Proceedings of International EDBT Workshop on Pervasive Information Management* (PIM), LNCS No. 3268, Spring Verlag, Berlin/Heidelbertg, New York, 2004.

[14] J. Pascoe: "Adding Generic Contextual Capabilities to Wearable Computers". In *Proceedings of 2nd International Symposium on Wearable Computers*, 1998.

[15] Preuveneers and Berbers. "Adaptive Context Management Using a Component Based Approach". In *Proceedings of the 5th International Conference on Distributed Applications and Interoperable Systems* (DAIS), Athens, Greece, 2005.

[16] B. Schilit, N. Adams and R. Want: "Context-aware Computing Applications". In *Proceedings of 1st International on Mobile Computing Systems and Applications*, 1994.

[17] A. Schmidt and K. Laerhoven: "How to build smart appliances". IEEE Personal communications, August 2001, pp. 66-71.

[18] A. Sistla, O. Wolfson, S. Chamberlain and S.Dao: "Modeling and Querying Moving Objects". In *Proceedings of 13th International Conference on Data Engineering*, Birmingham, UK,April 1997.

[19] W. Su, S. Lee and Mario Gerla "Mobility Prediction and Routing in Ad Hoc Wireless Networks" *International Journal of Network Management,* vol. 11, no. 1, John Wiley & Sons, January/February 2001.

[20] O. Wolfson, A. Sistla, S. Chamberlain and Y. Yesha: "Updating and Querying Databases that Track Mobile Units". *Distributed and Parallel Databases*, v.7 n.3, p.257-387, July 1999.