Mobility Aware Adaptation of Space Based Coordination Patterns

Reliable Mobile Computing by Means of Wireless Link Quality Sensing and Prediction

Karin Anna Hummel

Department of Computer Science and Business Informatics
University of Vienna
Lenaugasse 2/8
A-1080 Vienna

karin.hummel@univie.ac.at

ABSTRACT. *Networked mobile devices ranging from lightweight smartphones and PDAs to powerful notebooks benefit from persistent distributed object management and asynchronous communication means provided by space based middleware. The demanding requirements for successful coordination of processes on mobile devices are seamless operation even during offline periods and dependable synchronization when reconnecting. In order to adapt the space accurately and to hide the different semantics of coordination primitives while roaming, this work proposes a mobility aware coordination layer. This layer uses sensor information about the current quality of the wireless link and predicts the link quality in the near future by means of heuristically derived mobility patterns. A prototype implementation on CORSO is applied to the producer/consumer coordination pattern and experimentally evaluated.*

RÉSUMÉ. *Les périphériques mobiles, des plus légers des téléphones et PDA aux ordinateurs portables les plus puissants peuvent bénéficier d'une gestion persistante des objets distribués comme des moyens de communications asynchrones offerts par les intergiciels orientés espace (space based middleware). La coordination des processus sur ces périphériques mobiles est une opération délicate, même lors des périodes déconnectées et pour les synchronisations lors des reconnexions. Afin d'adapter l'espace précisément et cacher les différentes sémantiques des primitives de coordination lors des déplacements, ce travail propose une couche intermédiaire de coordination prenant en compte cette mobilité. Cette couche utilise les informations issues de senseurs pour apprécier la qualité actuelle de la liaison et prédit son comportement futur à l'aide d'heuristiques de schéma de mobilité. Un prototype implémenté en CORSO est appliqué pour la coordination producteur/consommateur et des expérimentations concluent le travail.*

KEYWORDS: *virtual shared object space, wireless networking, ubiquitous coordination.*

MOTS-CLÉS : *espace objets partagés virtuellement, réseau sans fil, coordination pervasive.*

1. Introduction

In distributed systems, a vast manifold of mobile computers support mobile users to connect to Web-services, *ad hoc* collaborative applications and legacy systems. The use of mobile devices, like PDAs (Personal Digital Assistants), notebooks or smartphones, heavily depends on their capabilities in terms of display size, computing power, available memory, and connectivity support.

Over the last decades, object based middleware systems targeted stationary and stable systems where network quality degradation has been mainly caused by overload. In addition to traditional coordination issues, like discovery services, communication, synchronization, and replication, in distributed mobile computing systems it has to be considered that components roam in and out of the other participants’ transmission range or reconnect at another topological part of the network, which may cause additional addressing overhead. Furthermore, due to lower bandwidth availability on radio links and considerable fluctuations in network quality, mobile distributed systems will have to consider different QoS (Quality of Service) levels depending on their location.

As a consequence, the flexibility of the mobile components will be best supported using concepts of mainly asynchronous communication, reliable and persistent data management, and ubiquitous data access. One possible solution targeting these issues is the use of space based middleware systems, which use the abstraction of a virtual shared data space for coordination and communication purposes [KUE 98b, GEL 85]. In order to support offline operation on shared data, processes on mobile devices will often operate on copies and cached data. Accuracy while creating the copy, efficient synchronization, and data lock management have been identified as key challenges [IMI 94].

This work presents the concept of a *mobility aware coordination layer* on top of space based middleware supporting basic principles to connect, read from, and write to the space. The mobility aware coordination layer adapts the behavior of the coordination principles depending on *coordination states* which are pro-actively derived from the current network quality and an estimation function about the future link quality. Furthermore, the time domain is modeled and used for additional prediction of the retention period at a specific link state. The estimation function relies on heuristics about the mobility behavior of the mobile device (termed *mobility patterns*). The first pattern presented is based on a two order Markov model and the assumption about continuous movement, while the second one considers smart information about the movement of the mobile device.

The feasibility of the approach is demonstrated by a use case implementation of the producer/consumer coordination pattern based on Java, CORSO, and the Java&Co API [KUE 02], and both estimation techniques for mobility patterns. Experiments simulating different movement types and movement speed are carried out while roaming in and out of the transmission range of an access point in a WLAN 802.11b network. The experiments include the two prediction based algorithms, a network state driven approach, where solely the current link state is considered, and a case without any mobility awareness support.

The paper is structured as follows: in Section 2 related work on space based and context aware middleware used for mobile settings is presented. Section 3 introduces the approach of mobility aware coordination and the mobility patterns used, while Section 4 discusses the design and architecture of the mobility aware coordination layer. In Section 5 the approach is applied to the space based technology CORSO and the Java&Co API [KUE 02], and the producer/consumer coordination pattern is presented. In Section 6 the mobility aware coordination layer is evaluated.

2. Related Work

Space based computing originates from the Linda Tuple Space coordination model and language [GEL 92, GEL 85] and provides advanced means to support coordination of distributed mobile processes, mainly by means of persistent data management and discoupled coordination. JavaSpaces, which is included in the Jini framework [BIS 03, EDW 99] and T-Spaces [LEH 99] are widely referred tuple space approaches based on associative search for data tuples. However, these approaches do not explicitly provide specific support for mobile devices. In contrary to associative search in a tuple space, the distributed CORSO implementation provides object ID based access to data items and advanced means for reliable replication, caching, garbage collection and transactional processing [KUE 98b, KUE 98a, KUE 02]. Furthermore, APIs (Application Programming Interfaces) for a set of programming languages are provided, like Java&Co and C++&Co. In case a mobile device is not able to run a full CORSO site locally, it may use the API libraries to connect to a remote site and thus access the virtual object space. Similarly, the Odyssey prototype implementation provides remote data access for mobile devices using dedicated servers [NOB 97].

MARS [CAB 00] is a tuple space implementation compliant to the JavaSpaces specification, which adds reflection, that is, the ability of a system to reason about and alter its own behavior, and thus allows to invoke actions upon events. Actions are used to change the tuple spaces' content and provide logical and space context information to mobile agents. In order to support mobile agents on mobile devices with physical context information, the TOTA middleware has been proposed [MAM 02, MAM 03]. TOTA focuses on holding context information in the space, on agent-to-space interactions, and on propagation of tuples in a peer-to-peer manner. A typical scenario is the support of swarm intelligence where each unit implicitly coordinates its actions reflecting on the others' behavior. In the LIME model, the notion of an interface tuple space is used to define a distributed global space, where mobile agents merge their local interface tuple space (agent context) while entering the global space [MUR 01, PIC 00]. The in() and out() operations from the basic Linda model are altered by adding a location address variable for the tuple's destination, which allows the tuple to migrate to a host and agent as soon as they are available. In L^{2}imbo, data consistency maintenance in a tuple space supporting mobile agents is delegated to proxy agents, which manage the access to the tuple space [DAV 98]. Since in this work only logical mobility is assumed, the disconnection of the tuple space sites is not treated.

In CORSO [KUE 02], AspectIX [HAU 01], or GLOBE [BAK 00], distributed object replication is used for performance and fault tolerance purposes. In [GEI 98], mobility of replicated objects is proposed for AspectIX by automatic migration and relaxation of the object. Although the work includes reasoning about replication and mobility, this concept neither targets replication inconsistencies caused by disconnecting devices, nor the case of dependent objects. Similar to our approach, XMIDDLE addresses copy, synchronization, data locking and release of data locks in a mobile environment [MAS 02]. Here, versioning is used to solve inconsistencies while synchronizing. In case of disconnection, XMIDDLE depends on finishing the disconnection procedure. In contrast the work presented in this article, no means for accurate or pro-active detection of possible disconnections are presented.

Recently, context-aware middleware approaches try to bridge the gap between the (physical) context and traditional data processing and management. In [ELI 99], requirements for efficient next generation middleware systems are stated and reflection is used to enable context-dependent middleware adaptations. Following the same approach as MARS [CAB 00], a meta-space model is proposed to provide reflection. The significant context information may be location or QoS parameters. Applications benefit from context-aware middleware because different context-models and positioning systems can be hidden from the application [CAP 02]. The Gaia OS and application framework uses a set of managers to discover, observe and deliver context information to ubiquitous applications while utilizing a distributed object space [ROM 02]. In contrary to traditional distributed object and space based approaches, CARISMA opens the middleware to be programmable by applications using reflection [CAP 03]. The drawbacks of this approach are the loss of transparency and the occurrence of conflicts whenever applications require different middleware behavior at the same time. However, applications benefit from adapting middleware behavior based on application specific information, for example in terms of performance, or reliability.

3. Network-Aware and Mobility Pattern Driven Coordination

In a mobile environment, devices are expected to roam between different locations and thus, communicate over wireless networks with varying bandwidths. A distributed mobile system relies on the performance of the communication channel and may have to adapt various parameters, for example timeout settings, communication, and coordination behavior to provide seamless and reliable services. Here, the following issues have to be considered:

Disconnected mobile devices. A participant in a distributed system, who is currently disconnected from the network, may solely work locally. To provide utmost support, a disconnected mobile entity should be able to work on local copies. As a consequence, the data needed should have been fetched prior to disconnection and has to be synchronized with the global view as soon as the mobile device reconnects. The probability of working on an accurate copy may be increased by pro-active actions, that is, actions utilizing knowledge about a future disconnected state.

Frequent movement of mobile devices. Adapting the coordination behavior depending on the link state causes overhead in terms of synchronization and copy actions. In case of frequent network state changes, the benefit gained for reliability will be diminished by the overhead costs. Thus, including prediction of retention periods is a potential strategy to reduce overhead costs.

Replication. Due to the need to use redundancy for fault-tolerant distributed systems, data replicas and distributed replication management protocols are used to facilitate mutually exclusive and non-exclusive operations on replicated data, like the replication protocol used by CORSO [KUE 02]. In case of mobile participants, locking mechanisms needed for mutual exclusion may cause significant problems. In case the mobile participant holding a lock disconnects, deadlocks and delays may occur. One solution is the introduction of proxies managing the interface between the global state and the mobile participant. Another solution, which does not need a specific proxy for a mobile entity, is the release of locked data and replicas, whenever an offline state is predicted.

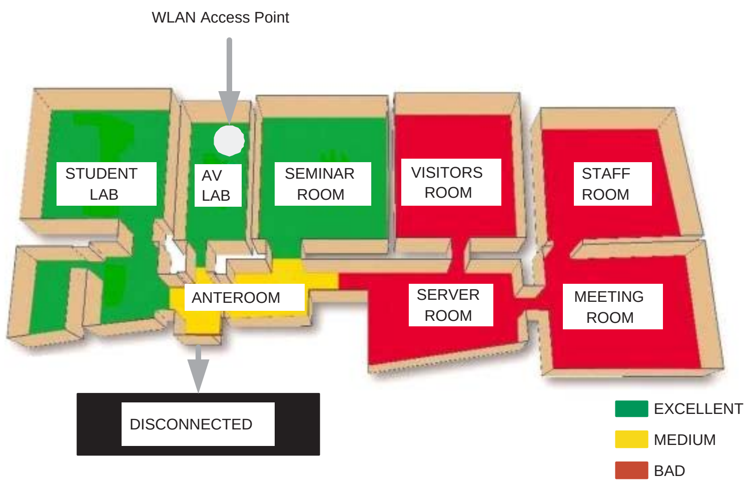

Figure 1. *Mapping of Rooms to Link Quality Categories*

In order to increase the reliability of a distributed mobile system, the wireless link may be sensed in terms of the SNR (Signal to Noise Ratio), which is a mean for characterizing wireless link quality by comparing the signal strength to the sensed noise. Based on measures of the wireless LAN quality, a set of rooms can be mapped to

WLAN link quality states, like shown in Figure 1. Due to our setting, and based on experiments using the ORINOCO WLAN client manager [AGE 02], the WLAN 802.11b SNR value range is split into four categories: EXCELLENT (SNR > 25dB), MEDIUM (15dB < SNR $\leq$ 25dB), BAD (5dB < SNR $\leq$ 15dB) and DISCONNECTED (SNR $\leq$ 5dB). In EXCELLENT and MEDIUM state the data rates remain at 11Mbit/s while in network state BAD, the data rates automatically fall back to 5.5 Mbit/s and 2.0 Mbit/s. (The term *link state* will be used to refer to one of the described intervals.)

For state and retention period prediction, two different patterns are used. The *Continued Move Pattern* is a second order Markov predictor assuming that the next state depends only on the last two history states [CHE 03]. Both the probabilities for state transitions and retention periods can be derived by observing the person's habits. The probability transitions described in Figure 2 are meant as a starting point, when no observed roaming habit is available. In [ASH 03], for example, it is shown how GPS (Global Positioning System) data of roaming persons can be used for incorporation into a similar two order Markov model. Since only four network states are distinguished, the Markov model is sufficiently simple for our purpose. The probabilities are derived heuristically considering continued movement. It is assumed that a person walking with a mobile device will more likely change to neighbor locations – and thus adjacent network states (that is, states with a current SNR value corresponding to the next higher or lower interval) – than roaming like jumping randomly in a discontinuous manner. Furthermore, a moving person is expected to continue in her direction. For example, in case the last two history states are EXCELLENT–MEDIUM, the person seems to roam away from the access points and the prediction for the next network state will be BAD. In case no continuous roaming has been observed, one of the adjacent locations/states is assumed.

Estimates for the retention periods are dependent on the history state, the prediction of the next network state and its most likely successor. Whenever it is predicted that the next network state will be left in a continued movement, the minimum retention period interval should be selected. The minimum retention period possible must not be smaller than the accuracy of the sensing application in order to avoid missed state changes. If the mobile device is expected to remain in the predicted network state, one of the other possible durations is assigned randomly. Based on the office case, four different classes of durations are proposed. The classes describe the behavior of a walking person (D1, for example 30 seconds needed for roaming to the next link state), short movement breaks (D2, for example 5 minutes), average working session (D3, for example 2 hours) and a stationary behavior lasting longer than 2 hours (D4).

The second pattern chosen is a *Smart Office Pattern*, which utilizes personal information, like a persons' calendar entries for more accurate retention period and location estimation. The model used is a timeline describing the personal schedule for a period like a day. The schedule information is used to determine intermediate states based on an accurate mapping of link states to locations.

The more accurate mobility patterns describe the behavior of a mobile entity, the more useful they are. Thus, further improvements are expected when using learning and self-adapting patterns, for example based on neural networks. On the other hand, the processing overhead caused by these adaptations has to be reasonable.

	EXC.	MEDIUM	BAD	DISC.
EXCELLENT – EXCELLENT	2/3	1/3	0	0
EXCELLENT – MEDIUM	1/4	1/4	1/2	0
EXCELLENT – BAD	0	1/3	1/3	1/3
EXCELLENT – DISCONNECTED	0	0	1/2	1/2
MEDIUM – EXCELLENT	2/3	1/3	0	0
MEDIUM – MEDIUM	1/4	1/2	1/4	0
MEDIUM – BAD	0	1/4	1/4	1/2
MEDIUM – DISCONNECTED	0	0	1/2	1/2
BAD – EXCELLENT	0	1/2	1/4	1/4
BAD – MEDIUM	1/2	1/4	1/4	0
BAD – BAD	0	1/4	1/2	1/4
BAD – DISCONNECTED	0	0	1/3	2/3
DISCONNECTED – EXCELLENT	1/2	1/2	0	0
DISCONNECTED – MEDIUM	1/3	1/3	1/3	0
DISCONNECTED – BAD	0	1/2	1/4	1/4
DISCONNECTED – DISCONNECTED	0	0	1/3	2/3

Figure 2. *The Continued Move Mobility Pattern Transition Probability Matrix*

4. Adaptable Space Based Coordination Primitives

Coordination patterns for space based computing are built upon sets of only a few operations originating from the Linda Tuple Space [GEL 85]. The space abstraction allows to keep these primitives simple and easy to use. Thus, distributed processes write, read, and synchronize by accessing one virtual space. This abstraction hides the distributed nature of the space, which consists of a multitude of local space fragments. (The term *global space* will be used to refer to the virtual space and *local space* when referring explicitly to space fragments located on the local site.)

The mobility aware coordination layer keeps this beneficial reduction of coordination complexity. While the semantics of the space primitives will be changed, the coordination patterns can be built upon the primitives like in the stationary case. The primitives will either operate on the global space or on a copy at the local space, depending on a mobility aware decision.

Figure 3 introduces the terminology of the basic and the derived primitives. While the basic primitives operate on the global space, the derived primitives operate on either the global or the local space. In order to provide consistent views of the global space, local operations have different semantics. Instead of executing the operations they mark the concerned objects indicating the actions that are postponed until a synchronization with the global space is possible. Therefore, *activity flags* are used to identify postponed operations (that is, *CREATE*, *READ*, *WRITE*, *DELETE*). In case of consistency violations while synchronizing, local operations will be discarded unless required otherwise by an application program.

create(space, object)	$\longrightarrow$	create_local (localspace, object, CREATE)
	$\longrightarrow$	create_global (space, object)
read (space, object)	$\longrightarrow$	read_local (localspace, object, READ)
	$\longrightarrow$	read_global (space, object)
write (space, object)	$\longrightarrow$	write_local (localspace, object, WRITE)
	$\longrightarrow$	write_global (space, object)
delete (space, object)	$\longrightarrow$	delete_local (localspace, object, DELETE)
	$\longrightarrow$	delete_global (space, object)

Figure 3. *Adapted Coordination Primitives*

4.1. *Determining the Semantics of the Coordination Primitives*

Primitives will be interpreted according to the *Coordination State* of the application process. Four states are proposed to solve the characteristics of mobile entities and are changed on reasoning about the current network state and a prediction for the next state and retention duration periodically. Figure 4 shows the coordination states used and the state transitions of the mobility aware coordination layer. The transitions are mainly time driven but include also the event driven change in case of error (transition *E*). At the beginning of a transition, a decision is taken which state to enter next (transitions *A* and *B*). Then, the methods initializing the state have to be executed. *Synchronize Space* leads to the execution of the postponed actions and is executed whenever a state transition causes a change from accessing the local space to accessing the global space, *Copy Space* creates a copy of the needed objects in the global space, and *Release Space* releases all locks held.

The *PARTICIPATING* state is entered whenever the network conditions – and the prediction about the next state – allow full participation in the global space. Here, the coordination primitives are applied to the global space similar to a stationary space participant. This state is left upon network changes or because an error occurred. The latter results in changing to the *OFFLINE* state. In *OFFLINE* state, if a local copy is available, space manipulations are executed locally. In *BY_PROXY* state, the local space is not used. The process on the mobile device connects to a remote space entity and accesses the global space in a Client/Server way. In this state, the local site of the space may not cause lock problems. Working *ON_COPY* allows to create a copy of the used global space before going offline and allows to change space objects locally in order to prevent ongoing network traffic over a limited bandwidth. However, in this state triggered by a refreshing timeout, synchronization and copy of the local space are carried out in order to support working with the most recent global space view possible. Finally, *WAITING* is used to prevent time consuming transition actions in case of frequent changes between states of different link quality like very short disconnection times. After a waiting period, the previous coordination state is entered (transition *B*) and the next transition takes place from the previous coordination state to the next coordination state. In order to prevent long process stalls the *WAITING* state has to be left after a defined timeout.

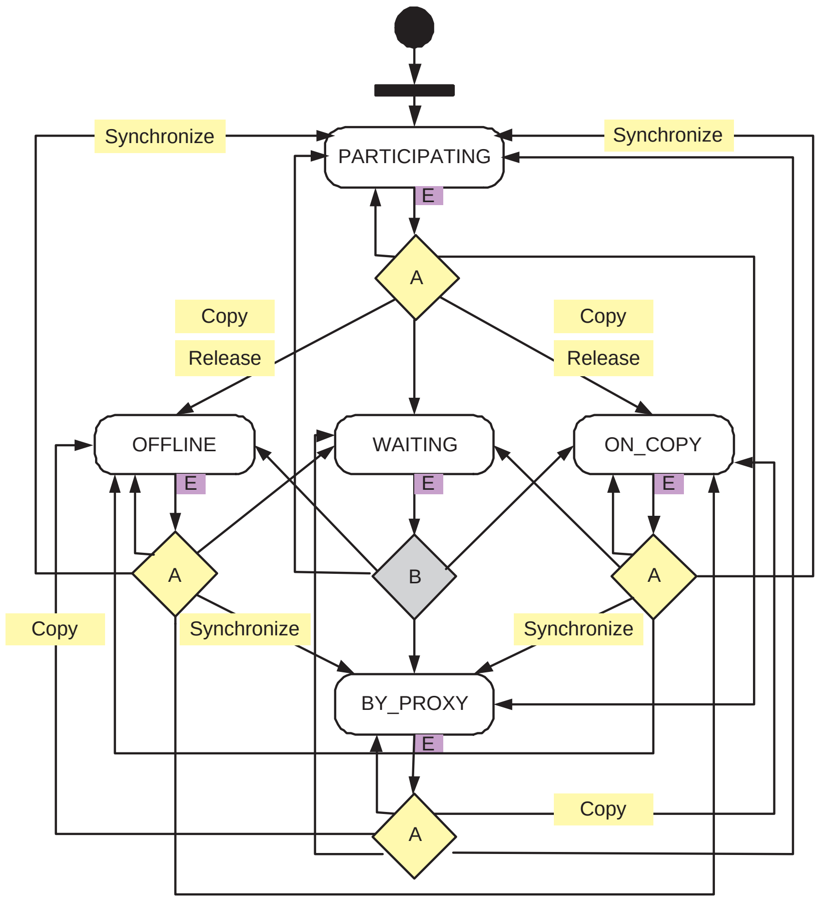

Figure 4. *State Transitions of the Mobility Aware Coordination Layer*

4.2. *Calculating the Next Coordination State*

The decision in the state transition diagram (Figure 4, transition *A*) depends on the location of the entity and on the timing constraints. As Figure 5 shows, the current state of link quality corresponding to the location, its successor and the expected retention periods in the current and the next link state are used to derive the current coordination state (*current_co_state*). Based on the duration estimations, the controlling link state (*controlling_link_state*) is chosen, which is primarily important for decisions about the next coordination state. In the default case, the current link state carries more weight. The estimated retention period in the current and the next link state are compared to a relaxation threshold (*D_RELAX*). In case the retention period of the current link state is lower than or equal to this threshold, the link state is assumed to be left soon and the next predicted link state is considered for controlling the coordination state prediction algorithm. In order to avoid non-beneficial frequent state changes, the following exceptions to the rule are used: A *waiting_flag* indicates frequent link state changes and is thus set to true if the estimated retention period in the current and the next link state are smaller than the relaxation threshold. However, if the time already spent in the *WAITING* state has already expired (*waiting_time_expired*), the *waiting_flag* is reset to false.

```
if (current_duration > D_RELAX)
   controlling_link_state = current_link_state
else
  controlling_link_state = next_link_state;
  if ((current_duration + next_duration <= D_RELAX) OR
      (current_link_state != next_link_state))
     waiting_flag = true

if (waiting_flag AND waiting_time_expired)
  waiting_flag = false
```

Figure 5. *Determining the Controlling Link State and the Waiting Condition*

In case the *waiting_flag* is not set or because no coordination state change happens (that is, *previous_co_state* and proposed *current_co_state* are the same), Figure 6 shows how the link state value of the controlling state is used to derive the coordination state *current_co_state*.

5. Implementation, Architecture and Use Case

The implementation of the proposed approach uses the space based middleware CORSO, which provides advanced means for fault-tolerant space based coordination in a distributed system [KUE 02], mainly, reading and writing of arbitrary data structures (objects) to the space, advanced transaction mechanisms, notification, reliable object distribution protocols, and object persistence. The virtual CORSO object space is distributed between different sites, which are connected and manage consistent data access using TCP/IP and UDP/IP channels. An application process opens a connection to either the local or a remote site to connect to the global space. In case mobile devices do not run a local CORSO site – because they cannot provide enough mem-

```
switch (controlling_link_state)
  case EXCELLENT:
       if (waiting_flag AND
           (previous_co_state != PARTICIPATING))
          current_co_state = WAITING
       else
          current_co_state = PARTICIPATING
  case MEDIUM:
       if (waiting_flag AND
           (previous_co_state != BY_PROXY))
          current_co_state = WAITING
       else
          current_co_state = BY_PROXY
  case BAD:
       if (waiting_flag AND
           (previous_co_state != ON_COPY))
          current_co_state = WAITING
       else
          current_co_state = ON_COPY
  case DISCONNECTED:
       if (waiting_flag AND
           (previous_co_state != OFFLINE))
          current_co_state = WAITING
       else
          current_co_state = OFFLINE
```

Figure 6. *Determining the Current Coordination State*

ory, processing power, or required software support – they connect to a remote site via UDP sockets. Only mobile devices capable of running a local CORSO site have the potential to benefit from the mobility aware coordination layer.

The mobility aware coordination layer uses CORSO primitives and the Java&Co API, a Java library for space operations. Network sensing is provided by the WLAN 802.11b client [AGE 02]. The logging capabilities of the client are used to calculate the mean link quality of a defined recent time interval (for example 10 seconds). Each coordination pattern consisting of reliable read and write operations can be built using the mobility aware coordination layer.

Figure 7 shows the software architecture and a typical hardware setup. The mobility aware coordination layer is executed on a notebook (or any other device capable of running a local CORSO site). The layer is made up of a *Network Interface Monitoring* module accessing the network logs and calculating the current link state based on the modeled SNR mapping. The *Mobility Patterns* are used to derive a prediction for the next link state based on the history of visited link states and timing information and a pattern describing the assumptions of the mobility behavior. The *Co-State Machine* implements the algorithm to chose the current coordination state and to execute the

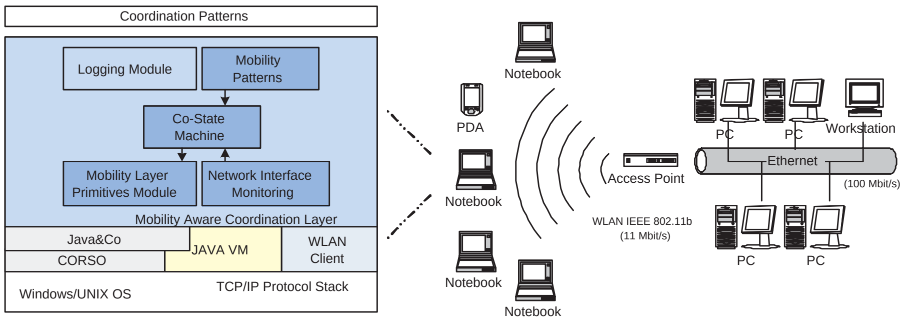

Figure 7. *Mobility Aware Coordination Layer Architecture*

state transition actions. The semantics of the coordination primitives are adapted in the *Mobility Layer Primitives Module*. An extended logging module allows to log, for example each access to the space for further evaluation.

Use Case: The Producer/Consumer Pattern

The producer/consumer pattern has been chosen because it makes use of main space operations and can be used for cooperation and competition purposes in various applications (see also [KUE 98a]). In a producer/consumer pattern, one or more processes produce information which is consumed by one or more processes. Figure 8 shows how CORSO data items are managed as a single linked list both in the global space (according to [TEC 03]) and the local space. In the global space, producers create new CORSO Objects (OIDs), link these new OIDs with the previous ones at the end of the list (*End of Stream*) and write the data to the new OIDs (for example *Item1*). Consumers read OIDs and delete them starting at the beginning of the list (*Start of Stream*) by means of atomic CORSO transactions. The local copy OIDs' data structure consists of additional *Activity Flags* indicating the action to perform on the original OID in the global space, which is either *NOTHING*, or flag values indicating the consumption of objects (*DELETE*), or the production of objects (*CREATE* and *WRITE*). A limit for the number of produced items can be used to model a case with limited resources.

6. Experiments

For all experiments, the limit for producer/consumer items in the space is set to 100 items. One producer adds 1 item per second to the space and one consumer reads one item per second from the space. At the beginning of each experiment, the space consists of 100 items previously entered (in order to prevent waiting periods

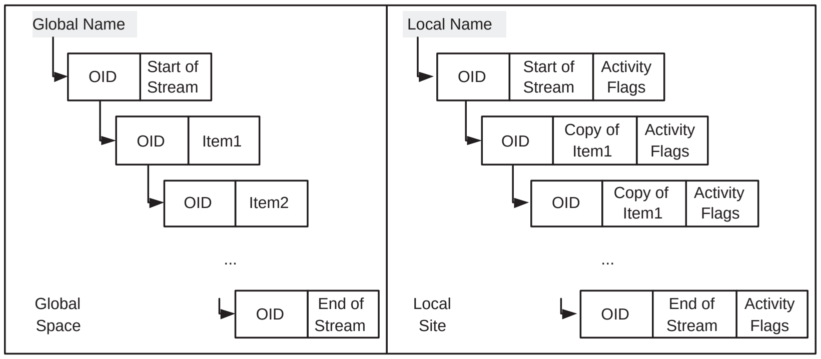

Figure 8. *Producer/Consumer Data Structure*

at the consumer site because no items are in the space). The consumer is executed on a notebook connected via WLAN 802.11b (11Mbit/s), while the producer remains stationary on a notebook connected via wired Ethernet connection (100Mbit/s). The consumer is the coordination part that is observed during the experiments.

Each experiment has been carried out using the same movement pattern. The chosen retention period intervals are: period < 30 seconds (D1), 30 seconds <= period < 60 seconds (D2), 60 seconds <= period < 180 seconds (D3), period >= 180 seconds (D4). Since D1 might lead to link state change misses, for the experiments only the intervals D2, D3 and D4 have been considered. Figure 9 shows the relative timeline and link state changes, starting with a sequence of periods which lie in interval D2, followed by periods which lie in D3 and finally, D4. The last two network changes are discontinuous changes, while the others follow the continuous model of roaming from one link state to the next.

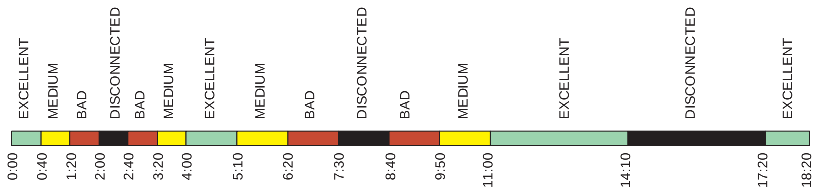

Figure 9. *Time Schedule of the Experiment*

According to the used mobility patterns, four different cases are considered. First, the example producer/consumer pattern is used without any mobility or link status awareness (*w/o*). In this case, the producer/consumer pattern is semantically equivalent to a traditional stationary case. Second, only the current link quality state is considered for choosing the coordination state and thus the semantics of the coordination primitives (*Network Driven*). Using this non-predictive pattern, the *WAITING* state cannot be entered. State predictive mobility patterns used are the continued move pattern (*Continued Move*) and the smart office pattern (*Smart Office*). In the smart office pattern case, a timeline similar to the experimental movement has been chosen to demonstrate the behavior of the state prediction algorithm in case of a perfect matching mobility state prediction.

All diagrams show values aggregated in periods of 30 seconds (that is, the cumulated occurrences in 30 seconds), which is the period assumed for coordination state changes. Related to the period, timeout intervals of 30 seconds are used for space operations. Logging of the current SNR is done once per second by the WLAN client manager. The mean of the last 10 logging entries is used to determine the current link state. In the *WAITING* state, the application process is suspended for 30 seconds.

6.1. *Evaluating the Effectiveness*

Here, effectiveness of the algorithm is measured by the amount of consumed items during the experiment.

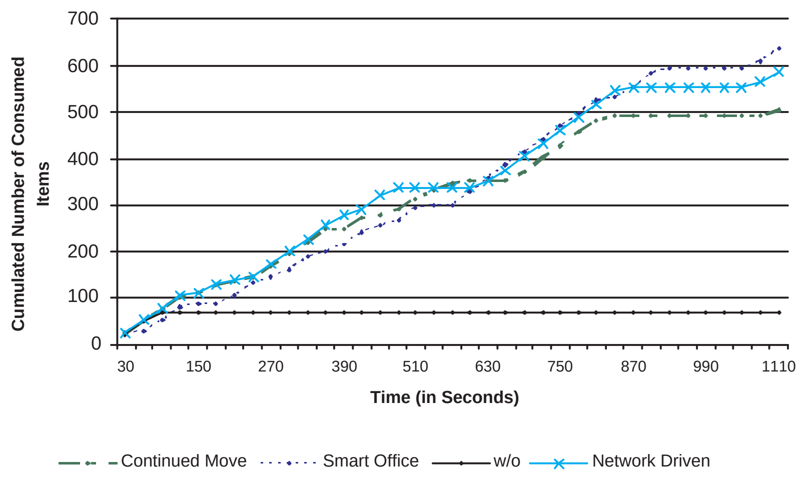

Figure 10. *Cumulated Items Consumed*

Figure 10 shows that the pattern without any reflection (*w/o*) cannot tolerate the first disconnected state and stops consuming elements at t=90s (total items consumed: 69). Since no fault tolerance mechanisms are used, the consumer cannot recover. In the *Network Driven* case, the consumption of items is very efficient due to the lack of the time consuming *WAITING* state and the fact, that data rate reduction at link state *BAD* – when not discovered in time – does not effect the transfer of the small amount of experimental data (total items consumed: 586). However, in case the notebook disconnects in a continuous way, only the continued move pattern (*Continued Move*, total items consumed: 504) and the smart office pattern (*Smart Office*) provide ongoing consumption by means of local copies (t=450s). At t=850s, only the smart office pattern (*Smart Office*) pattern can adapt and copy the space in time (total items consumed: 636). Each pattern operating with the mobility aware coordination layer was able to continue operation during or after *DISCONNECTED* states.

6.2. *Evaluating the Operation Efficiency and Failure Rate*

In order to compare the efficiency of the algorithm using different mobility patterns, global space operations have been counted. Figure 11 shows the cumulated values for each case. The erroneous pattern without support of mobility aware coordination layer (*w/o*, total number of operations: 1146) yields in permanently erroneous space operations. The network driven case (*Network Driven*, total number of operations: 5095) causes a higher number of operations as the continued move case (*Continued Move*, total number of operations: 4885), while being more effective (see Figure 10). At t=1040s when reconnecting to the network, the smart office pattern causes heavy load due to synchronization (*Smart Office*, total number of operations: 5884).

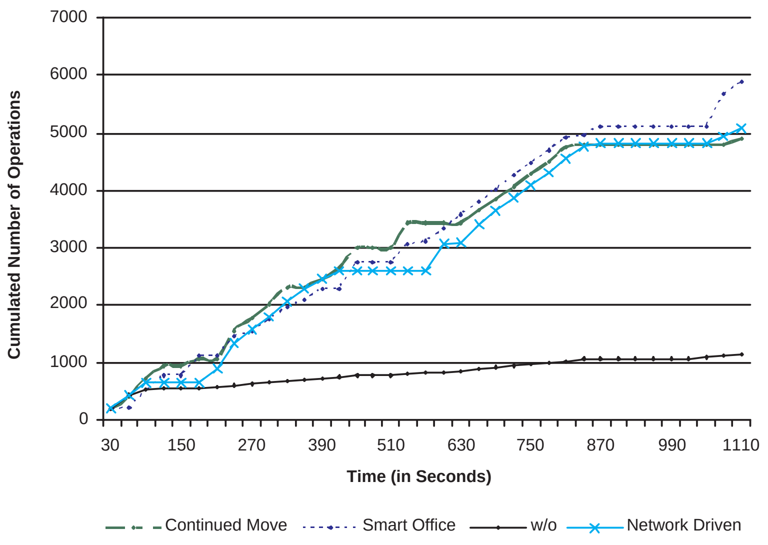

Figure 11. *Cumulated Number of Global Space Operations*

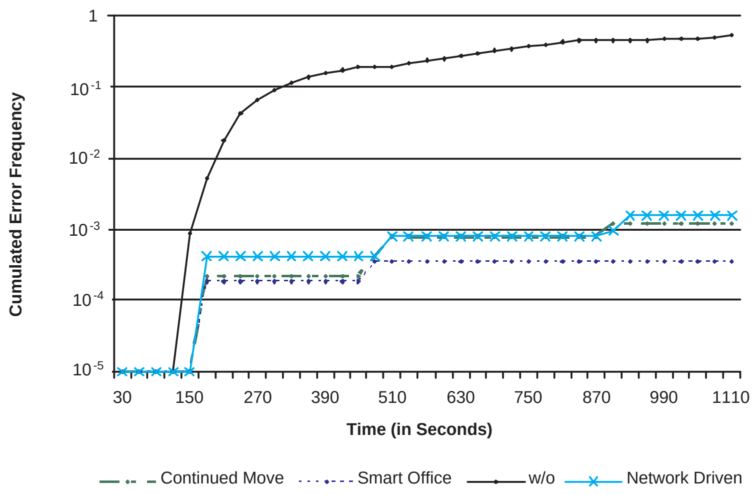

Figure 12. *Cumulated Error Frequency of Global Space Operations*

Figure 12 gives a detailed description of the cumulated relative error rate (that is, the number of erroneous operations in relation to the overall global operations) in logarithmic scale. Except the case without the support of the mobility aware coordination layer (where the cumulated error rate at the end of the experiment reached 0.52), all patterns show approximately linear behavior (on the logarithmic scale) and total error rate of each pattern is less than 0.01. The gradient of the approximated cumulated error curves is minimal in the smart office case (*Smart Office*).

6.3. *Evaluating the State Change Overhead*

Figure 13 shows the cumulated frequency of mobility aware coordination layer (*ML*) global space overhead operations (that is, the number of global space overhead operations related to the the total number of global space operations). In case *w/o*, naturally no overhead occurs. The other patterns show significant jump discontinuities, which reflect the overhead peaks at coordination state changes. Due to the intended highly mobile experiment, the total overhead ratios of the reliable patterns are 0.35 (*Network Driven*), 0.43 (*Smart Office*), and 0.53 (*Continued Move*). The discontinuity at t=1080 for the smart office pattern reflects the load caused by the synchronize operation after reconnecting (*Smart Office*).

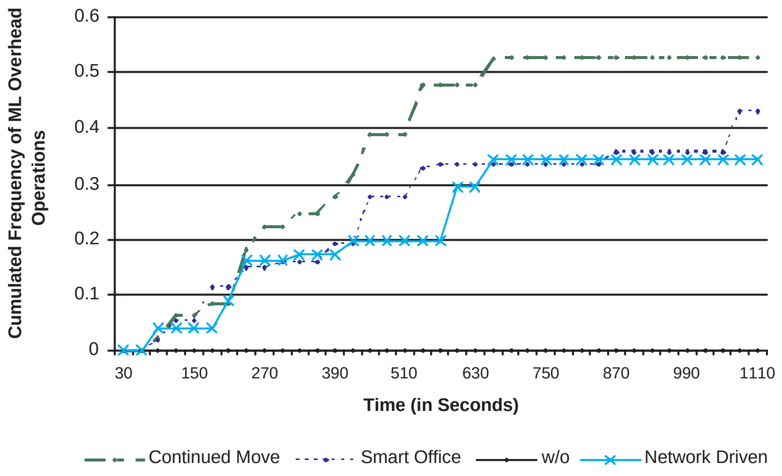

Figure 13. *Cumulated Overhead Frequency*

7. Conclusion

The increasing popularity of mobile computers fosters the formation of not only networked mobile devices, but mobile distributed systems. Space based middleware approaches offer advanced means to remain operational in case a mobile participant disconnects or the data transfer rate of a wireless link decreases. Mobile devices may either connect remotely to the distributed space using a remote server or run a fragment of the distributed space locally. This work has focused on the latter case. A mobility aware coordination layer has been presented targeting indoor wireless connectivity. This layer uses sensor information about the WLAN link's quality state (by interpreting the SNR logging) and mobility patterns for prediction of the next link state and retention periods. First, a two order Markov model has been presented for modeling continued movement. Second, calendar based roaming information about the owner of a device has been used for predicting location and thus the expected link quality.

A prototype of the mobility aware coordination layer has been implemented using CORSO and the Java&Co API [KUE 02]. Applied to the producer/consumer coordination pattern, the proposed layer has been evaluated experimentally for both predictive mobility patterns and compared to a producer/consumer pattern reflecting only on current link quality and a case without any environment reflection. The experiments included different speed and both continuous and discontinuous mobility behavior.

The results for the best matching mobility pattern (an optimal smart office case) showed the benefits in terms of most effective and seamless operation, while the results for the continued move pattern are worse than the case working with current link quality sensing only. Comparing the error rate, in both predictive patterns the number of erroneous space operations was slightly smaller than in the non-predictive case. Without using the mobility aware coordination layer, the producer/consumer pattern showed persisting errors after disconnecting for the first time. In contrast, for the other three cases the mobility aware coordination layer prevented the producer/consumer from any persisting error or inconsistency.

8. References

[AGE 02] AGERE, "Wireless LAN PC Card (Extended)", Product sheet, 2002, Agere Systems Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18109-3286.

[ASH 03] ASHBROOK D., STARNER T., "Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users", *Personal and Ubiquitous Computing*, vol. 7, num. 5, 2003, p. 275–286, Springer.

[BAK 00] BAKKER A., AMADE E., BALLINTIJN G., KUZ I., VERKAIK P., VAN DER WIJK I., VAN STEEN M., TANENBAUM A., "The Globe Distribution Network", *USENIX Annual Conference (FREENIX Track)*, San Diego, USA, 2000, USENIX, p. 141–152.

[BIS 03] BISHOP P., WARREN N., *JavaSpaces In Practice*, Addison-Wesley, 2003.

[CAB 00] CABRI G., LEONARDI L., ZAMBONELLI F., "MARS: A Programmable Coordination Architecture for Mobile Agents", *IEEE Internet Computing*, vol. 4, num. 4, 2000, p. 26–35, IEEE Computer Society.

[CAP 02] CAPRA L., BLAIR G. S., MASCOLO C., EMMERICH W., GRACE P., "Exploiting Reflection in Mobile Computing Middleware", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, num. 4, 2002, p. 34–44, ACM Press.

[CAP 03] CAPRA L., EMMERICH W., MASCOLO C., "CARISMA: Context-Aware Reflective Middleware System for Mobile Applications", *IEEE Transactions on Software Engineering*, vol. 29, num. 10, 2003, p. 929–945, IEEE Computer Society.

[CHE 03] CHENG C., JAIN R., VAN DEN BERG E., "*Location Prediction Algorithms for Mobile Wireless Systems*", p. 245–263, CRC Press, 2003.

[DAV 98] DAVIES N., FRIDAY A., WADE S., BLAIR G., "L2imbo: A Distributed Systems Platform for Mobile Computing", *ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks*, vol. 3, num. 2, 1998, p. 143–156, Kluwer.

[EDW 99] EDWARDS K., *Core JINI*, Prentice Hall, Upper Sadle River, NJ, USA, 1999.

[ELI 99] ELIASSEN F., ANDERSEN A., BLAIR G., COSTA F., COULSON G., GOEBEL V., HANSEN O., KRISTENSEN T., PLAGEMANN T., RAFAELSEN H. O., SAIKOSKI K. B., YU. W., "Next Generation Middleware: Requirements, Architecture and Prototypes", *7th IEEE Workshop on Future Trends of Distributed Computing Systems*, IEEE Computer Society, 1999, p. 60–68.

[GEI 98] GEIER M., STECKERMEIER M., BECKER U., HAUCK F. J., MEIER E., RASTOFER U., "Support for Mobilty and Replication in the AspectIX Architecture", *Object Oriented Technology, Workshop on Mobility and Replication*, vol. 1543, 1998, p. 325–326, Springer.

[GEL 85] GELERNTER D., "Generative Communication in Linda", *ACM Transactions on Programming Languages and Systems*, vol. 7, num. 1, 1985, p. 80–112, ACM Press.

[GEL 92] GELERNTER D., CARRIERO N., "Coordination Languages and their Significance", *Communications of the ACM*, vol. 35, num. 2, 1992, p. 97–107, ACM Press.

[HAU 01] HAUCK F. J., BECKER U., GEIER M., MEIER E., RASTOFER U., STECKERMEIER M., "AspectIX: A Quality-Aware, Object-Based Middleware Architecture", *3rd IFIP Int. Conference on Distributed Applications and Interoperable Systems, DAIS*, Krakow, Poland, 2001, Kluwer, p. 115–120.

[IMI 94] IMIELINSKI T., BADRINATH B. R., "Mobile Wireless Computing: Challenges in Data Management", *Communications of the ACM*, vol. 37, num. 10, 1994, p. 18–28, ACM Press.

[KUE 98a] KUEHN E., "How to Approach the Virtual Shared Memory Paradigm", *Journal of Parallel and Distributed Computing Practises, Special Issue on Vitual Shared Memory for Distributed Architectures*, vol. 1, num. 3, 1998, Nova Science Books.

[KUE 98b] KUEHN E., NOZICKA G., "Post-Client/Server Coordination Tools", *Coordination Technology for Collaborative Applications*, vol. 8, 1998.

[KUE 02] KUEHN E., "Coordinated Shared Object Spaces V1.2", White paper, 2002, TECCO Software Entwicklung, Prinz Eugen Strasse 58, A1040 Vienna.

[LEH 99] LEHMAN T. J., MCLAUGHRY S. W., WYCKO P., "T Spaces: The Next Wave", *32 Annual Hawaii International Conference on System Sciences*, vol. 8, IEEE Computer Society, 1999, Page 8037.

[MAM 02] MAMEI M., LEONARDI L., MAHAN M., ZAMBONELLI F., "Coordinating Mobility in a Ubiquitous Computing Scenario with Co-Fields", *Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, AAMAS 2002*, Bologna, Italy, 2002.

[MAM 03] MAMEI M., ZAMBONELLI F., LEONARDI L., "Programming Coordinated Motion Patterns with the TOTA Middleware", *Euro-Par 2003*, Klagenfurt, Austria, 2003, Springer, p. 1027–1037.

[MAS 02] MASCOLO C., CAPRA L., ZACHARIADIS S., EMMERICH W., "XMIDDLE: A Data-Sharing Middleware for Mobile Computing", *Wireless Personal Communications*, vol. 21, num. 1, 2002, p. 77–103, Kluwer.

[MUR 01] MURPHY A. L., PICCO G. P., ROMAN G.-C., "LIME: A Middleware for Physical and Logical Mobility", *The 21st International Conference on Distributed Computing Systems*, IEEE Computer Society, 2001, Page 0524.

[NOB 97] NOBLE B. D., SATYANARAYANAN M., NARAYANAN D., TILTON J. E., FLINN J., WALKER K. R., "Agile Application-Aware Adaptation for Mobility", *Sixteenth ACM Symposium on Operating System Principles – Operating System Review*, vol. 31, ACM Press, October 1997, p. 156–171.

[PIC 00] PICCO G. P., MURPHY A. L., ROMAN G.-C., "Developing mobile computing applications with LIME", *International Conference on Software Engineering*, IEEE Computer Society, 2000, p. 766–769.

[ROM 02] ROMÁN M., HESS C. K., CERQUEIRA R., RANGANATHAN A., CAMPBELL R. H., NAHRSTEDT K., "A Middleware Infrastructure to Enable Active Spaces", *IEEE Pervasive Computing*, vol. 1, num. 4, 2002, p. 74–83, IEEE Computer Society.

[TEC 03] TECCO, "Corso V3.2", Tutorial, 2003, TECCO Software Entwicklung, Prinz Eugen Strasse 58, A–1040 Vienna.