Performance Analysis and Prediction of Physically Mobile Systems

Antinisca Di Marco
Dipartimento di Informatica
Universitá dell'Aquila
via Vetoio 1, L'Aquila, ITALY
adimarco@di.univaq.it

Cecilia Mascolo
Computer Science Department
University College London
Gower Street, London, U.K.
c.mascolo@cs.ucl.ac.uk

ABSTRACT

The market of portable computational devices is expanding more and more rapidly. The systems created by the interactions of these devices among themselves and with the surrounding infrastructure result in being quite different from existing traditional systems in terms of connectivity, dynamicity and resource availability. As a consequence, existing performance evaluation and prediction techniques appear to be inadequate to the application to mobile systems. While some adaptations have been proposed for systems presenting some logical mobility (i.e., software mobility), very little has been attempted to provide useful performance prediction methodologies for physically mobile systems.

In this paper we present a methodology for modeling performance of physically mobile systems: our aim is to provide guidelines for the designer of the system on how particular physical mobility patterns affect the system performance and on how these measures can be taken into account in the early stages of development of the system.

Categories and Subject Descriptors: B.8.2 [HARDWARE]: Performance Analysis and Design Aids; C.4 [PERFORMANCE OF SYSTEMS]: Modeling techniques; D.2 [Software]: Software Engineering

General Terms: Performance, Design.

Keywords: Software Performance, Mobile Systems, Modeling.

1. INTRODUCTION

Mobile devices we carry with us during the daily activities now come in all sorts of shapes and provide all sorts of functionality. The networks that these devices create by interacting both among themselves (e.g., in peer to peer mode, for instance through Bluetooth) and with the fixed backbone network (e.g., through the cellular connection) may be very dynamic. The resources which these devices have are usually quite limited (e.g., little memory and, often, limited energy). Even the connectivity which is provided to these devices may be quite intermittent and often quite low.

Software engineering techniques for the analysis and design of distributed systems seem therefore quite inadequate to deal with these very different characteristics offered by mobile systems. However, given how challenging the development of mobile systems is, due to all the parameters and dynamicity involved, tools for the support of the design and development of mobile application are greatly needed. In particular tools for the design-time assessment and prediction of performance of these systems would be very useful, for example, to predict situation of possible network congestion, to devise a profitable (possibly dynamic) deployment of software system over the hardware platform which guarantees high system QoS by carefully exploiting the available hardware resources.

This paper presents a methodology for modeling performance of physically mobile systems. It concentrates on component-based software systems and evaluates figures of merit by generating an Layered Queuing Network (LQN) performance model starting from the description of the Software Architecture (SA) of the application, of the considered user mobility patterns, and of the context (hardware plus software) the application meets during the user mobility.

The proposed approach offers metrics to the users to allow them to judge how the performance of the system would be in the various mobility contexts. Moreover, it provides a support for the designer in the predictive evaluation of the impact that the identified mobility patterns have on the system performance. Such analysis helps designers in different ways: to design a suitable software deployment with good performance in presence of user mobility conform to the identified patterns; it indicates which physical mobility behavior should be avoided/minimized in order to guarantee certain performance under specific conditions.

Attempts which keep mobility into account in the assessment of performance have been presented in [1, 2, 3], where software mobility was considered. In these approaches, software mobility was not only the assessed factor but also a means to improve performance (as software could be moved from one location to the other). Moreover, some of these approaches have considered physical mobility in the modeling. Among them, the closest work to ours is probably [1], where an integrated approach to modeling and performance evaluation of mobile systems at the architectural level based on simulation is devised. Both physical mobility and code mobility are considered. Differently from our approach, this work does not define metrics that characterize the physical mobility patterns of the system users, vital for mobile system analysis, and use a different (non analytical) performance model.

2. OUTLINE OF THE APPROACH

The predictive performance analysis of mobile software system we propose deals with component-based software systems and uses, as target performance model, an LQN model. It concentrates on the physical mobility and omits the treatment of the logical mobility.

Physical mobility is a requirement which developers have not yet considered with the due care despite it having a huge impact on the performance of the system. It is applied at the SA level.

We assume that the SA and the Physical Mobility (PhM) are described by means of UML 2.0 diagrams annotated with additional information related to performance aspects needed to carry out the analysis. Such additional information is provided on the diagrams by means of stereotypes and tags defined by the UML Profile for Schedulability, Performance and Time (SPT) [5].

The approach generates LQN models from the SA and the mobility behavior of users and evaluates the obtained models in order to predict the performance indices of interests. Indeed, the LQN generation algorithm derives a set of LQN models, one for each system configuration identified in the PhM description, and calculates a performance metrics that estimates the performance of the software system when a user/device has one of the PhM behaviors described.

The main steps of the methodology are:

1. **Modeling of the application**: the software system and the user (physical) mobility patterns are modeled by means of UML 2.0 diagrams;

2. LQN generation and performance analysis:
   (a) **Meta-LQN generation**: this step starts from the SA description and generates a partial LQN, called meta-LQN. This models the software system statics and dynamics. This step is performed once at the beginning of the generation process, since we assume that the SA description remains unchanged despite the physical mobility.
   (b) **LQN models generation**: this step generates a set of LQNs, one for each deployment diagrams defined, and evaluates them to obtain values/functions for the performance indices of interests.
   (c) **PhM pattern characterization**: this step aggregates the obtained performance indices to provide a metrics for the PhM patterns, indicating how these influence the performance of the system.

3. **Results interpretation**: from the performance indices and the PhM metrics obtained from the performance analysis step, indications to both designers and final users are formulated.

2.1 Modeling of the Application

Interesting features and aspects of the system are modeled by means of UML 2.0 diagrams. We use Use Case diagrams to identify the system services and the system users. We model the SA behavior through *Sequence Diagrams* that describe the evolution of the scenarios which are critical for the performance. The structure of the software system is described by a *Component Diagram* emphasizing the interfaces provided and required by the software components and how these combine to provide services. The information related to the PhM is modeled by *State Diagrams* and *Deployment diagrams*. The state diagrams describe the PhM patterns of specific user types, where the states represent the system configurations (hardware plus software) the user/device interacts with during his/its moves. Such contexts, or system configurations, are described by deployment diagrams. One deployment diagram is associated to each different physical context. The association of a state in the state diagram with the corresponding system configuration is specified by introducing a reference to the deployment diagram in the state.

The information used to enrich the diagrams needed to analyze the performance of the system is annotated with SPT Profile stereotypes used as notes in the diagrams.

Use Case Diagram - Through a use case diagram, the critical system use cases for which the performance analysis is needed are described. Hence the diagram used in the approach can be smaller than the one produced in the software modeling phase. We call such diagram *Performance Use Case Diagram*. The associations between the actors and the use cases has information on the operational profile of the mobile system. From the operational profile the approach infers the system workload for each system use case. The workload intensity, provided to the system by a customer class, is specified through the `<<PAopenLoad>>` or `<<PAclosedLoad>>` stereotypes. We extend the usage of such stereotypes to annotate the workload information in the Performance Use Case Diagram. The rational of such decision is that in our approach each system use case represents a different request type entering the system, hence in the approach this request type is mapped as a customer class of the LQN.

Component Diagram - The component diagram considered in the approach is a portion of that built during the software life cycle. It only contains the software components involved in the selected critical performance We use the `<<PAhost>>` stereotype (`PAschdPolicy` tag value) to annotate each component with the scheduling policy of its waiting queue. We annotate each component interface with the execution time required by a software component to perform the corresponding operations. This information is specified by either `PAdemand` or `PAdelay` tag values of the `<<PAstep>>` stereotype. The `PAdelay` tag is used when the corresponding operation is not shared among multiple (concurrent) accesses. To simplify the annotation of the component operations, we assume that each interface contains just one operation from which it inherits the name.

Sequence Diagrams - The behavior of the mobile application is modeled by means of *Sequence Diagrams*, where the lifelines model component instances and the arrows represent the interaction among them. The information about the time needed to execute such interactions is annotated in the component diagram. The sequence diagram is annotated only with information related to the fragment operators and the network usage in remote method invocation. For all of them we use the `<<PAstep>>` stereotype.

Physical Mobility Description

To describe the PhM patterns we use UML State Diagrams where each node represents a context and the arrows among states represent the probability that the user will be moving

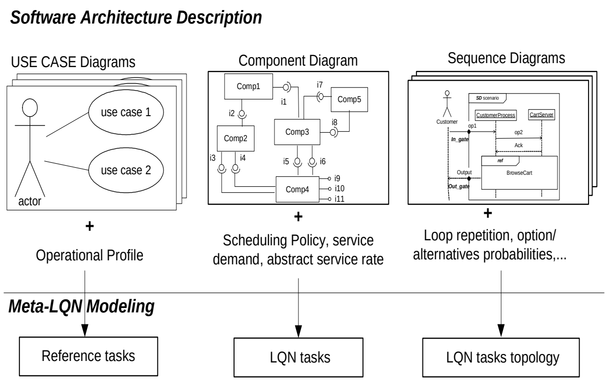


Figure 1: Meta-LQN Model Generation.

from the starting context to the destination one.

Each state is annotated with: (i) the name of the deployment diagram describing the context (software and hardware) represented by the state, and (ii) the estimated total time the device/user spends in this context during the observation interval. To this aim we introduced a new stereotype `<<PHcontext>>` with two tag values: `PHdeploymentRef` and `PHTOTresTime`. This information is used to combine the homogeneous performance indices evaluated for each system configurations in the state diagram, in order to provide a performance measure for each physical mobility pattern.

Deployment diagrams are used to represent the portion of the system accessible to the user, after one of her transfers. We call such portion of the system a *visible configuration* and we annotate it with additional information about hardware devices and available connectivity. We use `<<PAhost>>` stereotype with `PArate` and the `PAschdPolicy` tag values to annotate each node that represents a CPU with its processing rate and scheduling policy. Whereas, we use the `<<PAresource>>` stereotype with `PArespTime` and `PAschdPolicy` tag values to annotate nodes representing (server) communication networks. The tag values contain the latency of the network to transmit a packet and the scheduling policy of the network, respectively.

2.2 Meta-LQN Generation

Figure 1 sketches the step of the generation of Meta-LQN. It starts from the SA description and produces a partial LQN, called meta-LQN. The meta-LQN models the software system statics and dynamics: however it does not consider the PhM patterns and the set of contexts the system has to adapt to during mobility. The meta-LQN generation is a preliminary step performed once at the beginning of the generation process. The meta-LQN remains unchanged despite the physical mobility. The meta-LQN contains:

- reference tasks, one for each use case in the use case diagrams, that generate the different request types;
- LQN tasks, one for each component in the component diagram. The LQN tasks have as many entries as the provided interfaces of the software components;
- a set of interconnections among tasks that are created by traversing the sequence diagrams. The technique used in this step is very similar to the one defined by Petriu et al. in [6] or can be extracted from the technique described by Woodside et al. [4].
- an additional task for each resource type that executes external operations localized in the sequence diagrams by the previous step. Such task is reached by a connection starting from the entry that invokes the (entry corresponding to the) component interaction in the sequence diagram annotated by `<<PAextOP>>` stereotype.

The parametrization of the LQN tasks and reference tasks is done by extracting the relevant information from the annotations in the UML diagrams. In particular:

- the *operational profile* of the system is extracted by the annotation of the use case diagrams. Such information is used to parameterize the LQN reference tasks and allows the definition of the arrival processes (one for each use case) in the performance model;
- the *scheduling policy* of software components, and the *service demand* of the provided interfaces are retrieved from the annotations in the component diagram. Such information is needed to parameterize the LQN tasks;
- the *loop repetition factors* and *behavioral alternative probabilities* are extracted from the annotations in the sequence diagram. These annotations represent i) the number of visits an LQN job makes to the involved LQN tasks when a repeated behavior is modeled in the sequence diagram; ii) the routing probabilities of a job type when the sequence diagram models behavioral alternatives;
- the *host demand* that each task entry requires to the device hosting the relative task, is extracted from the annotation associated to the interfaces in the component diagram.

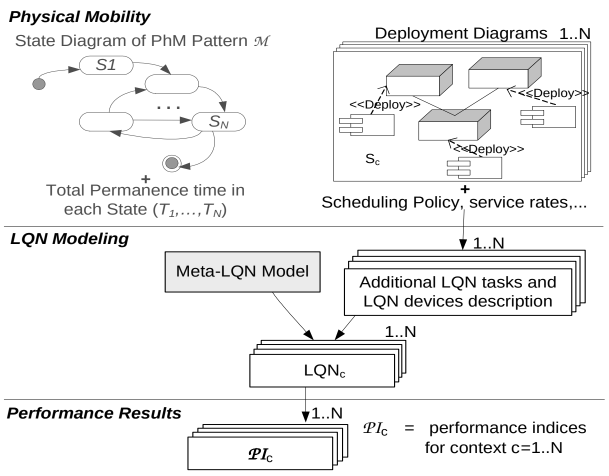
2.3 LQN Models Generation

This step generates a set of LQN models associated to the defined deployment diagrams and calculates the relative performance metrics. This step is reiterated as many times as the number of system configurations identified. Figure 2 outlines this step for N system configurations or contexts.

For each PhM pattern, the required modeling comprises: the description of the possible physical contexts of the device and the specification of the mobility behavior, i.e., the way the device moves among the contexts. The former describes the overall system in terms of hardware and software components available after a movement and is described by means of deployment diagrams. The latter, instead, is modeled by a UML state diagram where each state points to a deployment diagram describing the system configuration visible (or reachable) from that location.

For each deployment diagram the approach generates an LQN model as follow:

- it identifies the hardware components in the deployment diagram and instantiates an LQN devices for each of them;
- it adapts the meta-LQN model according to the software components reachable/visible in the location.



Physical Mobility
State Diagram of PhM Pattern M
S1
SN
Total Permanence time in each State (T1,...,TN)
Deployment Diagrams 1..N
<<Deploy>>
Sc
Scheduling Policy, service rates,...
LQN Modeling
Meta-LQN Model
Additional LQN tasks and LQN devices description
1..N
LQNc
Performance Results
PIc
PIc = performance indices for context c=1..N

Figure 2: LQN Model Generation and Performance Indices.

- it adds LQN tasks to LQN device interconnections according to the Deploy association in the deployment diagram.
- it adds additional LQN tasks to LQN device interconnections according to the resource name executing the external operation the additional task represents.

The LQN devices are hence parameterized by means of information annotated in the deployment diagram. Annotations in such diagrams refer to hardware component features and specify the scheduling policy of the hardware components (i.e. the strategy used by the component to extract job from its waiting queue), the service rates of each hardware component, in case the component is an active resource (e.g., a CPU) or the latency introduced by the component in case it is a passive resource (e.g. network).

2.4 PhM Patterns Characterization

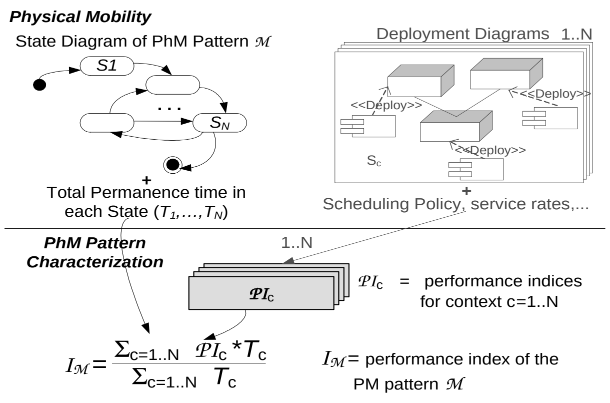
Referring to Figure 3, the PhM pattern M describes N different physical contexts. The previous generation step devised N different LQN models starting from the meta-LQN and the N deployment diagrams modeling the system configurations. Then it evaluated the performance indices

Since a mobility pattern is a combination of the system configurations representing the contexts the device/user interacts with while roaming, as final step, the methodology aggregates the performance indices evaluated in each physical context. The aggregated value represents figures of merit of the mobility pattern itself.

2.5 Analysis Scope and Results Interpretation

Predictive performance analysis becomes of primary importance in the context of mobile applications, given the dynamicity of these systems and the often scarce resources involved.

The performance indices of interests in mobile applications are mainly service response time and device utilization. The utilization of devices, in particular of the bandwidth in wireless networks, can be extremely useful for the identification of the hardware performance bottlenecks and



Physical Mobility
State Diagram of PhM Pattern M
S1
SN
Total Permanence time in each State (T1,...,TN)
Deployment Diagrams 1..N
<<Deploy>>
Sc
Scheduling Policy, service rates,...
PhM Pattern Characterization
1..N
PIc
PIc = performance indices for context c=1..N
IM = Σc=1..N PIc * Tc / Σc=1..N Tc
IM = performance index of the PM pattern M

Figure 3: PhM pattern characterization.

to evaluate potential alternatives, including the strengthening of the hardware, its better positioning/allocation in the physical contexts, or a better deployment of the software components over the available hardware platforms.

The service response time, instead, is the primary element for the metrics related to the mobility pattern. The idea here is to calculate the response time of a given service in all the identified system configurations and then to combine these values on the basis of the mobility patterns considered through a function that we call aggregation function. In this way we can devise a metric that is associated to the mobility patterns. The metric will represent the system response time for a given service if the user follows the mobility pattern.

When the performance indices and the mobility pattern metrics are calculated, such results have to be interpreted to gathered useful advises to both the designer and the user of the mobile application.

Acknowledgments: This work has been supported by the IST EU projects PLASTIC, RUNES and SEGRAVIS.

3. REFERENCES

[1] S. Balsamo and M. Marzolla. Towards Performance Evaluation of Mobile Systems in UML. In *Proc. of ESMc'03 Conference*, pages 61–68, Naples, Italy, Oct. 27–29 2003.

[2] V. Grassi and R. Mirandola. PRIMAmob-UML: A Methodology for Performance Analysis of Mobile Software Architectures. In *Proc. of WOSP'02*, pages 262–274, Rome, Italy, July 2002.

[3] V. Grassi, R. Mirandola, and A. Sabetta. A UML Profile to Model Mobile Systems. In *Proc. of UML Conference 2004*, pages 128–142, October, 2004.

[4] T. A. Israr, D. H. Lau, G. Franks, and C. M. Woodside. Automatic Generation of Layered Queuing Software Performance Models from Commonly Available Traces. In *WOSP*, pages 147–158, July 2005.

[5] OMG. *UML Profile, for Schedulability, Performance, and Time.* OMG document ptc/2002-03-02, http://www.omg.org/cgi-bin/doc?ptc/2002-03-02.

[6] D. C. Petriu and X. Wang. From UML Descriptions of High-Level Software Architectures to LQN Performance Models. In *Proc. of AGTIVE'99. Springer-Verlag*, pages 47–62, 1999.