Proc. of ASIAN'98, LNCS 1538,
pp. 300-304, Springer, 1998.

Mobile Computation: Calculus and Languages (A Tutorial)

N.Raja* and R.K.Shyamasundar**

School of Technology & Computer Science
Tata **I**nstitute of **F**undamental **R**esearch
Mumbai 400 005, INDIA

Abstract. This is a brief survey of current research directions in mobile computation. In particular we provide an overview of the various abstract calculi for mobility, an overview of the programming languages that have been developed for mobile computation, and provide a comparative study of the language abstractions via the calculi and discuss the underlying challenges.

1 Introduction

Mobility denotes the physical or virtual movement of computational resources (comprising hardware and software) across a local or global network. The term *mobile computing* is used for environments where the hardware is mobile, while the term *mobile computation* is used in situations where software is mobile. In this survey we shall focus on mobile computation. With rapid advances in Internet technology, and the availability of powerful laptop and palmtop computers, the concept of mobility is undergoing a metamorphosis from a mere theoretical curiosity to become one of the essential requirements of every computing environment. The phenomenon of mobility on a global scale has the potential of causing wide ranging effects on the practice of computing. Further, by providing a completely new paradigm for understanding the nature of computation, it forces us to reconsider our conceptual understanding of the notion of computation as well.

In this survey, we shall provide an overview of the calculi for mobility, an overview of the programming languages that have been developed for mobile computation, and provide a comparative study of the language abstractions via the calculi and discuss the underlying challenges. In the following, we shall provide a brief discussion of the various calculi for mobility and their evolution, and the widely used programming languages for mobile computation purposes.

* Email: raja@tifr.res.in; WWW: http://www.tcs.tifr.res.in/~raja
** Email: shyam@tcs.tifr.res.in; WWW: http://www.tcs.tifr.res.in/~shyam

2 Calculi for Mobility

What is an appropriate abstract model for mobile computation? Traditional research has focussed along three independent strands – functional, concurrent, and distributed computing. However none of these models adequately captures the significant features of mobile computation [7]. Mobile computation demands an entirely new paradigm.

A number of calculi have been proposed over the years to study concurrent computation. New calculi are being evolved from these, with the addition of notions to capture features associated with mobile computation.

One of the earliest abstract framework for concurrent computing was provided by Actors [2]. Though the framework had an intuitive appeal to model massively parallel computation, it was not amenable to formal reasoning. Recent proposals have sought to rectify this situation with notions of nested structures of arbitrary depth, which mirrors the nesting of administrative domains on the web [25]. The earliest proposals of formal calculi for concurrent computing could deal only with static connectivity among processes, and in certain cases even the total number of processes had to be fixed in advance. Among these are Petrinets [26], CSP [18], and CCS [21]. In the setting of the Internet, communication links appear and disappear frequently. This is due to wide fluctuations in bandwidth of links, unpredictable failures and recovery of links giving rise to connections through alternative routes. So calculi which are restricted to static connectivity are clearly inadequate. The next generation of process calculi could model networks of processes with dynamic connectivity. Among these π-calculus [22], and Asynchronous π-calculus [17] directly modeled channel mobility, while Linda [10] and CHOCS [29] modeled process mobility. These calculi lack the notion of distributed locations of processes, which cannot be ignored in mobile computation over the web. These calculi have in turn been extended with primitives to capture distributed concurrent computing. LLinda [11] extends Linda, while Join-calculus [13] extends π-calculus. The next logical step towards developing a calculus for mobile computation was to incorporate explicit notions of locality in these calculi [4]. The notion of locations is important in modeling barriers caused due to administrative domains which impose their own restrictions on the movement of messages across them. Such extensions gave rise to the Distributed Join-calculus [14] and the Ambient calculus [6, 7]. The Ambient calculus can also model mobility of active computations.

Apart from these calculi, a number of others have been proposed to study various orthogonal issues in mobile computation. The Spi calculus [1] is a preliminary attempt to model security aspects, while Mobile Unity [27] incorporates techniques for reasoning and specification of mobile computations.

There are a number of issues that need further exploration, such as more structured notions of data, typing, and correctness proofs in a concurrent distributed mobile context. These notions need to be unified in a coherent semantic framework. Such an enterprise could lead to fundamental insights into the phenomenon of computation itself.

3 Programming Languages for Mobile Computation

One of the most fascinating outcomes of the development of mobile computation is the concrete realization of *mobile software*. Mobile software is programming language code which is shipped across a network from a source to a destination site, and is executed at the destination computer. The results (if any) are then returned to the source computer. The presence of mobile software in a framework where the source and destination computers are themselves mobile (due to hardware mobility) gives rise to a rich set of possibilities, and also gives rise to number of issues that need to be addressed.

The desirability of mobile software arises primarily from the fact that it can vastly reduce the communication overheads while implementing services between servers and clients across a network. The savings are particularly enhanced if the service happens to be an interactive one. Reduction in the messages exchanged between server and client is particularly meaningful in an environment where communication links experience unpredictable fluctuations in their effective bandwidth, are susceptible to frequent failures, and the messages have to traverse a number of distinct administrative domains which erect barriers to the flow of messages across them. Secondly, mobile software provides a great degree of flexibility to the server computers, wherein the kind and number of services offered by them need not be static and determined in advance. Third, the maintenance of networks of mobile computers is rendered easier when the clients can themselves control the installation of latest software updates as and when required. Finally, as a valuable side effect, mobile software also leads to savings in storage space since programs can be fetched and executed on demand.

The possibility of mobile software gives rise to at least three kinds of situations depending on the entity which takes the initiative of moving the software – the provider, the execution site, or the software itself.

A number of issues crop up in the design and implementation of mobile software. First among these is the obvious requirement of portability. The same software should be capable of being executed across a variety of architectures and computing environments. The second requirement is that of safety. Safety demands that errors in the mobile software should not affect the environment of the execution site. Third, the myriad security issues such as secrecy, message integrity, authentication of servers and clients, and non-repudiation of messages transmitted assume great significance. Finally, the need for efficiency is omnipresent. Various programming languages have been designed for mobile software. A few prominent ones among these are Obliq [9], Telescript [15], Limbo [20], O'Caml [19], Safe-Tcl [24], and Java [3]. A comparative study of these languages is contained in [30].

In spite of the availability of a number of languages for developing mobile software, there remains a wide scope for further improvements. For instance, transfer of active computations needs to be supported [7]. In the context of mobility, it is not sufficient to allow only the transfer of passive program code across computers; effective and efficient ways of transferring computation in progress, along with their environments, needs to be given careful consideration. The development of such programming paradigms could provide a positive feedback to the architecture of the Internet itself.

4 Discussion

The phenomenon of mobility represents a major revolution in the discipline of computing. Research on calculi for mobility is seeking to address the challenges arising from mobile computation by unifying various traditional computation models; while research on programming languages seeks to create tools which will translate theoretical possibilities to realistic systems which would comprise computational environments in the decades to come.

In order to fully translate the potential benefits of mobility to a computerized society, research is needed on a whole spectrum of issues ranging from the formulation of a formal basis for computation in a mobile world, to the development of cost-effective and user friendly software tools which translate theoretical possibilities to practical and realistic computational environments.

The scenario of autonomous entities freely traversing globally distributed networks, requires the development of entirely new proof systems, behavioral theories, and software development methodologies in order to specify and establish properties of such computational entities [23]. Software tools which comprise implementations of verification systems, and specification languages would result on the basis provided by such proof methodologies.

The area of network security acquires a prominent place when mobility is prevalent. Traditional research on security has focussed mostly on secrecy issues, which have been handled through cryptology. More recent work on security has dealt with ways of reasoning about authentication mechanisms using logics of belief [5]. However, the other aspects of security in mobile computation are yet to be addressed in a satisfactory manner [12].

References

1. Abadi, M., Gordon, A.D.: A calculus for Cryptographic Protocols: the SPI Calculus. Proc. ACM Conference on Computer and Communications Security, ACM Press (1997) 36–47.
2. Agha, G.: Actors: A Model of Concurrent Computing in Distributed Systems. MIT Press (1986).
3. Arnold, K., Gosling, J.: The Java Programming Language. Sun Microsystems (1996).
4. Boudol, G., Castellani, I., Hennessy, M., Kiehn, A.: A Theory of Processes with Localities. Formal Aspects of Computing, **6** (1994) 165–200.
5. Burrows, M., Abadi, M., Needham, R.: A Logic of Authentication. ACM Transactions on Computer Systems, **8** (1990) 18–36.
6. Cardelli, L., Gordon, A.D.: Mobile Ambients. Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, Vol. 1378, Springer-Verlag (1998) 140–155.

7. Cardelli, L.: Abstractions for Mobile Computation. Manuscript (1998).
8. Cardelli, L., Davies, R.: Service Combinators for Web Computing. DEC Research Report (1997).
9. Cardelli, L.: A Language with Distributed Scope. Computing Systems **8** (1995) 27–59.
10. Carriero, N., Gelernter, D.: Linda in Context. Communications of the ACM **32** (1989) 444–458.
11. Carriero, N., Gelernter, D., Zuck, L.: Bauhaus Linda. Proc. Object-Based Models and Languages for Concurrent Systems, Lecture Notes in Computer Science, Vol. 924. Springer-Verlag (1995) 66–76.
12. Chess, D.M.: Security Issues in Mobile Code Systems. Mobile Agents and Security, Lecture Notes in Computer Science, Vol 1419. Springer-Verlag (1998).
13. Fournet, C, Gonthier G.: The Reflexive CHAM and the Join Calculus. Proc. POPL, ACM Press (1996) 372–385.
14. Fournet, C, Gonthier G., Lévy, J-J., Maranget, L., Rémy, D.: A Calculus of Mobile Agents. Proc. CONCUR'96, Lecture Notes in Computer Science, Springer-Verlag (1996) 406–421.
15. General Magic: The Telescript Home Page. Available at http://www.genmagic.com/Telescript.
16. General Magic: Mobile Agents White Paper.
17. Honda, K., Tokoro, M.: An Object Calculus for Asynchronous Communication. Proc. ECOOP'91, Lecture Notes in Computer Science, Vol. 521. Springer-Verlag (1991) 133–147.
18. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1988).
19. Leroy, X.: Objective Caml. Available at http://pauillac.inria.fr/ocaml/.
20. Lucent Technologies: The Inferno Home Page. Available at http://inferno.bell-labs.com/inferno/index.html.
21. Milner, R.: Communication and Concurrency. Prentice-Hall (1988).
22. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes, Parts 1–2. Information and Computation **100** (1992) 1–77.
23. Necula, G.C., Lee P.: Safe, Untrusted Agents Using Proof-Carrying Code. Mobile Agents and Security, Lecture Notes in Computer Science, Vol 1419. Springer-Verlag (1998).
24. Ousterhout, J.K.: Tcl and the Tk Toolkit. Addison-Wesley (1994).
25. Raja, N., Shyamasundar, R.K.: Actors as a Coordinating Model of Computation. Perspectives of System Informatics, Lecture Notes in Computer Science, Vol 1181. Springer-Verlag (1996) 191–202.
26. Reisig, W.: Petrinets. EATCS Monographs on Theoretical Computer Science (1990).
27. Roman, G-C., McCann, P.J., Plun, J.Y.: Mobile UNITY: reasoning and Specification in Mobile Computing. ACM Transactions on Software Engineering and Methodology, **6** (1997) 250–282.
28. Stamos, J.W., Gifford, D.K.: Remote Evaluation. ACM Transactions on Programming Languages and Systems **12** (1990) 537–565.
29. Thomsen, B.: Calculi for higher-order communicating systems, Ph.D. thesis, Imperial College, London University (1990).
30. Thorn, T.: Programming Languages for Mobile Code. ACM Computing Surveys, **29** (1997) (213–239).