Two Problems in Wide Area Network Programming*
(Position Statement)

Ugo Montanari

Dipartimento di Informatica, Università di Pisa, Italy
ugo@di.unipi.it

Motivations Highly distributed networks have now become a common platform for large scale distributed programming. Internet applications distinguish themselves from traditional applications on *scalability* (huge number of users and nodes), *connectivity* (both availability and bandwidth), *heterogeneity* (operating systems and application software) and *autonomy* (of administration domains having strong control of their resources). Hence, new programming paradigms (thin client and application servers, collaborative "peer-to-peer", code-on-demand, mobile agents) seem more appropriate for applications over internet.

These emerging programming paradigms require on the one hand mechanisms to support mobility of code and computations, and effective infrastructures to support coordination and control of dynamically loaded software modules. On the other hand, an abstract semantic framework to formalize the *model of computation* of internet applications is clearly needed. Such a semantic framework may provide the formal basis to discuss and motivate controversial design/implementation issues and to state and certify properties in a rigorous way.

Concern for the limited understanding we have of network infrastructure and its applications has been explicitly expressed in the US PITAC documents [9]. Also a theme on mobile/distributed reactive systems has been suggested as a new, proactive initiative for long-term, innovative research in a recent meeting promoted by the European Commission within the FET part of the V Framework programme.

Here we point out and shortly outline two issues which arise in the definition of such an abstract semantic framework.

Synchronization in a Coordination Framework Coordination [4] is a key concept for modeling and designing heterogeneous, distributed, open ended systems. It applies typically to systems consisting of a large number of software components, - considered as black boxes - which are independently programmed in different languages, and may change their configuration during execution.

While most of the activity is asynchronous, some applications, typically computer supported collaborative work or transactions among multiple partners, need primitives for synchronization. Synchronization might consist of complex computations to be performed by all partners on shared data before the global

* Research supported by TMR Network *G*ETGRATS and by MURST project *TOSCa*.

J. van Leeuwen et al. (Eds.): IFIP TCS 2000, LNCS 1872, pp. 609–611, 2000.

commit action takes place. While these primitives will be implemented in terms of asynchronous protocols in the lower levels of system software, it is important to offer to the user a high level model of them, to be employed for specification, validation and verification.

The concepts developed for concurrent access to data bases, like serializability or nested transactions, are not fully adequate in this setting, since they refer to restricted models of computation. Instead, such issues should be considered like name and process mobility, causal dependencies between interactions, and refinement steps in the design process. We see this as a challenging problem, with aspects of logic, semantics, concurrency theory, programming languages, software architectures, constraint solving and distributed algorithms.

As a contribution we just mention two pieces of work by the author and collaborators. The first is about a generalized version of Petri nets: A *zero-safe net* [1,2] is in a *stable* state when certain places are empty. Step sequences from stable states to stable states through non-stable states are transactions, and correspond to transitions of an *abstract* net. The second piece of work is a model, called *tile logic* [7,3] (see http://www.di.unipi.it/~ugo/tiles.html) which extends structured operational semantics (SOS) by Gordon Plotkin and rewriting logic by Jose Meseguer. Tiles are inference rules which can be combined horizontally to build transactions and vertically to build ordinary computations.

Finite State Verification with Name Creation Finite state verification is possible when threads of control are independent of the actual data. In this case an automaton encompassing the whole state space can be constructed, possibly minimized, and checked for properties of interest expressed in some modal or temporal logic. This technique has been successfully and extensively applied to hardware and protocols.

When the computation involves the creation of new names which can occur in transition labels, the ordinary finite state techniques cannot be applied, since dynamic allocation of names with reuse of the old ones is required if the states must remain finite. On the other hand, in the coordination approach control is actually often independent from data, since computation mostly consists of redirecting streams, connecting and disconnecting users, transferring them from an ambient to another, resolving conflicts and performing security checks. Creation of new names is actually quite common in wide area programming, since nonces generated during secure sessions, process locations, and causes of forthcoming events can also be represented as names. Finite representation techniques for such systems together with expressive logical frameworks and efficient algorithms are needed for checking security and other global and local properties of distributed applications.

Some experience has been described in the literature with π-calculus verification [10,5,6]. Also Marco Pistore and the author have introduced certain classes of automata, called *History Dependent* (HD) [8], which are able to allocate and garbage collect names. Behavioral properties related to dynamic network connectivity, locality of resources and processes and causality among events can be formally verified on finite HD-automata. However efficiency is not satisfac-

tory and lots of work remain to be done about the theoretical and practical applicability of these methods.

References

1. R. Bruni and U. Montanari. Zero-safe nets: Comparing the Collective and Individual Token Approaches. *Inform. and Comput.*, 156:46–89. Academic Press, 2000.
2. R. Bruni and U. Montanari. Executing Transactions in Zero-Safe Nets. *Proc. International Conference on Petri Nets 2000*, Aarhus, to appear.
3. R. Bruni, U. Montanari and V. Sassone. Open Ended Systems, Dynamic Bisimulation and Tile Logic, to appear in Proc IFIP TCS2000, Sendai. *Proc IFIP TCS2000*, Sendai, this volume.
4. N. Carriero and D. Gelenter. Coordination Languages and Their Significance. *Communications of the ACM*, 35(2), 97–107, 1992.
5. M. Dam. Model Checking Mobile Processes. *Information and Computation 129(1)*, 1996, pp. 35-51.
6. G. Ferrari, S. Gnesi, U. Montanari, M. Pistore and G. Ristori. Verifying Mobile Processes in the HAL Environment In: Alan J. Hu and Moshe Y. Vardi, Eds., *CAV'98*, Springer LNCS 1427, pp.511-515.
7. F. Gadducci and U. Montanari. The Tile Model. In: G. Plotkin, C. Stirling and M. Tofte, Eds., *Proofs, Languages and Interaction: Essays in Honour of Robin Milner*, MIT Press, to appear.
8. U. Montanari and M. Pistore. An Introduction to History Dependent Automata. In: Andrew Gordon, Andrew Pitts and Carolyn Talcott, Eds, Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II), ENTCS, Vol. 10, 1998.
9. President's Information Technology Advisory Committee. Information Technology Research: Investing in Our Future. Report to the President, National Coordination Office for Computing, Information, and Communications, February 1999, available from http://www.hpcc.gov/ac/report/.
10. B. Victor and F. Moller. The Mobility Workbench: A Tool for the π-calculus. *Proc. CAV'94*, Springer LNCS 818.