Multiparty Session Types for Mobility

Bogdan Aman and Gabriel Ciobanu

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.8, 700505 Iaşi, Romania
bogdan.aman@iit.academiaromana-is.ro, gabriel@info.uaic.ro

Mobility provides to the software applications the ability to move between locations and devices during their execution. This means that a running application migrates from one location to another depending on the proximity of the user. A mobile application is closely related to process migration in distributed systems modelled by process calculi such as distributed π-calculus [2] and TiMo [1]. Process migration represents the capability of a running process to be relocated from a location to another one in order to access and communicate data locally.

We propose session types for mobility to describe uninterrupted sequences of migrations for processes. We use a simple version of distributed π-calculus to define session types for mobility. The novelty of this approach is that we point out sequences of migrations as global types, and investigate scenarios in which processes are required to follow such a sequence of migrations along several locations.

Typing locations with mobility types is sufficient to detect subtle errors in the implementation of mobile applications and protocols. We do not insist on typing the local communication inside locations as it has been proposed by specific session types [4]. Communication session types are used to reason over communicating processes and their behaviour by abstractly representing the trace of the usage of the channels as a structured sequences of types. A detailed discussion and analysis of several versions of the π-calculi with session types is given in [5].

Our approach is in contrast with the typing system for distributed π-calculus [2] which uses types of the form $T@p$ and dependent type techniques. The syntax for processes is based on distributed π-calculus [2] and user-defined processes [3]. The operational semantics is given by a reduction relation for which we present the most important rules for migration and communication:

$$k[[go\ l(a).P \mid Q]] \mid l[[(a).R]] \rightarrow k[[Q]] \mid l[[P \mid R]] \quad (\textsc{Migrate})$$
$$l[[a!\langle v\rangle.P \mid a?(u).Q]] \rightarrow l[[P \mid Q\{v/u\}]] \quad (\textsc{Comm})$$

Rule (Migrate) describes the migration of processes between locations by using a port available at the destination location where the parties can communicate in order to establish necessary local interactions (by using this port). (Comm) is the standard communication rule derived from the π-calculus where the received value v replaces the existing free occurrences of u in Q.

The global types (ranged over by $G, G', \ldots$) describe the global behaviour of the system. For example, type $l \rightarrow l' : a.G'$ says that from location l a process migrates to location l' through the access port a, and at l' the process behaves accordingly to the mobility type described by G'.

Local types (ranged over by $T, T', \ldots$) describe the local behaviour of processes, acting also as a link between global types and processes. Type $a!.T$ represents the behaviour of sending a process on port a, and then behaving as it is described by type T. Similarly, $a?.T$ is for receiving.

We illustrate the connection between global and local types by defining the projection operation $G \uparrow l$ which, for a location l appearing in a global type G, provides the local type it has to conform to. For instance,

$$l_1 \to l_2 : a.G \uparrow l = \begin{cases} a!.G \uparrow l & \text{if } l = l_1 \\ a?.G \uparrow l. & \text{if } l = l_2 \end{cases}.$$

In the proposed typing system $\Gamma ::= \emptyset \mid \Gamma, G \mid \Gamma,\ X : \Delta$ describes the global type and the assignments to variables. On the other hand, $\Delta ::= \emptyset \mid \Delta, l : T$ describes the local types at each location involved in the typed system. We use the judgement $\Gamma \vdash N \rhd \Delta$ which says that "under the environment Γ, system N has typing Δ".

For example, the processes appearing in the operational semantics rule (Migrate) are typed using the following rules:

$$\frac{\Gamma \vdash l[[P]] \rhd l : T}{\Gamma \vdash l[[go\ l'(a).P]] \rhd l : a!, l' : T} \qquad \text{(Migration)}$$

$$\frac{\Gamma \vdash l[[P]] \rhd l : T}{\Gamma \vdash l[[(a).P]] \rhd l : a?.T} \qquad \text{(Accept Migration)}$$

As processes transform over time, their types need to follow their evolution. This is formalized by a type reduction relation $\Rightarrow$ over environments Δ, for which the typing of migrating processes is given by $l : a!, T_1, l' : a?.T_2 \Rightarrow l : T_1, l' : T_2$.

Such a typing system ensures certain properties, including type soundness.

Theorem 1. *(subject congruence and reduction).*

1. $\Gamma \vdash N \rhd \Delta$ *and* $N \equiv N'$ *imply* $\Gamma \vdash N' \rhd \Delta$.
2. $\Gamma \vdash N \rhd \Delta$ *and* $N \to N'$ *imply* $\Gamma \vdash N' \rhd \Delta'$, *where* $\Delta = \Delta'$ *or* $\Delta \Rightarrow \Delta'$.

References

1. G. Ciobanu. TiMo: Timed Mobility in Distributed Systems. *Proceedings SYNASC*, 5–10, IEEE Computer Society (2013).
2. M. Hennessy. *A Distributed π-calculus*. Cambridge University Press (2007).
3. K. Honda, N. Yoshida, M. Carbone. Multiparty Asynchronous Session Types. *Journal of the ACM* **63** (1), Article 9, (2016).
4. K. Takeuchi, K. Honda, M. Kubo. An Interaction-based Language and its Typing System. *Lecture Notes in Computer Science* **817**, 398–413 (1994).
5. N, Yoshida, V.T. Vasconcelos. Language Primitives and Type Discipline for Structured Communication-Based Programming Revisited. *Electronic Notes in Theoretical Computer Science* **171**(4), 73–93 (2007).