Session Types for Time-Sensitive Protocols

Laura Bocchi[1], Julien Lange[2], Rumyana Neykova[2], and Nobuko Yoshida[2]

[1]University of Kent, UK
[2]Imperial College London, UK

Distributed software systems, such as e-business and financial systems, have often real-time constraint where exchanges of agreements and data transmissions need to be completed within specified timeframes. The analysis of real-time models is supported by established foundations and tools based on timed automata [1]. However, when timed automata are composed into networks and allowed to communicate asynchronously via unbounded communication channels, such as communicating time automata (CTA) [7], tractability is compromised in the sense that, for instance, reachability is no longer decidable. Tractable verification of distributed systems can be attained using session types [9]. However, session types typically focus on *safety* (e.g., *'the server will reply with a string'*) and not on *punctuality* (e.g., *'the server will reply within 10 milliseconds'*).

In recent work [6, 5] we extended multiparty session types (MPST) [10, 4] with a model of time borrowed from CTA, and established a sound and complete correspondence between *timed* MPST and a subclass of CTA that satisfies progress. In [5] we gave, on the basis of this correspondence, decidable conditions for properties on CTA that are undecidable in the general case, such as safety (absence of orphan messages and communication mismatches), progress, non-zenoness and eventual reception of messages sent. Moreover, in [5] we gave a procedure and a tool to build, when possible, global timed MPST from collections of timed automata; the resulting global specification guarantees an well-behaved composition. The procedure given in [5] adds a timed perspective to a line of research (e.g., [8, 11]) directed towards a flexible engineering practice: the idea is to combine the usual top-down approach (1 - global type definition; 2 - projection; 3 - modular system implementation) with reverse-engineering (i.e., combining existing parts into a well-behaved ensemble).

While formalisms and tools based on timed automata are valuable for the analysis of real-time models, they do not provide a seamless bridge from correct models to programs. Reversely, session types offer, in principle, a methodology to validate *programs*. In [6] we gave a typing theory based on *timed* MPST for *static* verification of real-time programs. The theory of timed MPST in [6] abstracted programming languages as a process calculus with simple time primitives; an open challenge is embedding it into concrete languages for real-time programming and type checking tools. At present, timed MPST have then been embedded [12] into the Scribble toolchain [13], to support the design of time-sensitive protocols and offer a concrete tool for *dynamic* monitoring. Along a similar line of research, [2] extended binary session types with time (including

a timed notion of compliance) and embedded them into a concrete dedicated middleware for *dynamic* verification [3]. The tools in [12, 3] demonstrate the practicality of a modular approach to formal verification of timed constraints based on session types. However, as we have shown in [12], the applicability of run-time monitoring in time-sensitive scenarios is limited by the verification overheads that may compromise transparency (i.e., the ability of dynamic monitors to not interfere with well-behaved programs). We also observed that overhead depends on factors like latency and shape of the protocol [12]. The definition of rigorous metrics of applicability is a critical future direction.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.

[2] M. Bartoletti, T. Cimoli, M. Murgia, A. Podda, and L. Pompianu. Compliance and subtyping in timed session types. In *FMOODS/FORTE*, LNCS 9039. Springer, 2015.

[3] M. Bartoletti, T. Cimoli, M. Murgia, A. S. Podda, and L. Pompianu. A contract-oriented middleware. In *FACS*, volume 9539 of *Lecture Notes in Computer Science*, pages 86–104. Springer, 2015.

[4] L. Bettini et al. Global progress in dynamically interleaved multiparty sessions. In *CONCUR*, LNCS 5201. Springer, 2008.

[5] L. Bocchi, J. Lange, and N. Yoshida. Meeting deadlines together. In *CONCUR*, LIPIcs 42. Schloss Dagstuhl LZI, 2015.

[6] L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In *CONCUR*, LNCS 8704. Springer, 2014.

[7] L. Clemente, F. Herbreteau, A. Stainer, and G. Sutre. Reachability of communicating timed processes. In *FOSSACS*, LNCS 7794. 2013.

[8] P.-M. Deniélou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP*, volume 7966 of *LNCS*, pages 174–186, 2013.

[9] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP*, LNCS 1381. Springer, 1998.

[10] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL'14*, volume 43. ACM, 2008.

[11] J. Lange, E. Tuosto, and N. Yoshida. From communicating machines to graphical choreographies. In *POPL*, pages 221–232, 2015.

[12] R. Neykova, L. Bocchi, and N. Yoshida. Timed runtime monitoring for multiparty conversations. In *BEAT*, EPTCS 162, 2014.

[13] Scribble Project homepage. `www.scribble.org`.