Brief Announcement: Two Classes of Communication Patterns

Ajay D. Kshemkalyani
EECS Dept., University of Illinois at Chicago
Chicago, IL 60607, USA

Mukesh Singhal
National Science Foundation
Arlington, VA 22230, USA

A distributed computation is viewed as a partial order on the events that occur at participating processes. Analyzing the structure of a distributed computation can lead to a better design of distributed applications, algorithms, and systems. To this end, this paper identifies two classes of communication patterns that occur in every distributed computation and examines their properties [2].

The first class of patterns consists of local patterns, termed *IO* and *OI intervals*, that occur at processes. These local patterns are specified in terms of messages sent and messages received by a process, and are distinguished by the order in which a pair of messages is sent and received by a process. Specifically, an *IO interval* corresponds to a receive event followed by a send event, and an *OI interval* corresponds to a send event followed by a receive event. (Likewise, there are II and OO intervals corresponding to two receive events, and two send events, respectively.) Clearly, there are numerous IO and OI intervals at each process in the computation. At each process there are also application-specific distinguished events, which define the durations between successive such events. An IO or OI interval of interest to an application is one that satisfies a certain application-specific relationship on the durations in which the send and receive events identifying the interval occur.

Application-specific predicates can be defined on how an IO or OI interval at one process is related to an IO or OI interval at another process. The use of such predicates on IO and OI intervals at different processes allows IO and OI intervals to be used as building blocks to formulate the second class of patterns, which is comprised of two global patterns, termed *segments* and *paths*. These two global patterns occur across processes in a distributed computation and signify the flow of information and coupling among the events at different processes. Segments and paths are generalizations of causal chains. While a causal chain captures only the causality relation, certain other message sequences in a distributed computation also play a significant role in the analysis of a distributed computation. By controlling the predicates on how IO and OI intervals at different processes are used to define segments and paths, different types of segments and paths can be defined.

Several key concepts and structures characterizing distributed computations are special cases of and can be expressed using these global patterns. Some examples are listed next.

(I) Measures of concurrency in a distributed computation can be expressed as functions of the length and number of the paths and segments in the computation. (II) It has been shown that computations using asynchronous communication can be realized under synchronous communication if and only if a certain graph structure called a crown does not exist in the computation. A crown can be expressed compactly using paths and segments in the computation. (III) Knowledge in a distributed system provides a formal method for reasoning about distributed protocols when different runs of the same protocol can yield different equivalent computations. Knowledge also plays a significant role in the evaluation of global predicates, debugging, monitoring, and establishing breakpoints and triggers. Paths and segments provide a means of identifying the extent of knowledge transfer in a distributed computation. (IV) Checkpointing is widely used in fault-tolerant computing and in parallel and distributed debugging. In order to optimize the number of checkpoints taken, it is useful to determine whether a certain local checkpoint could possibly be a part of a global checkpoint. It has been shown that a local checkpoint cannot be part of any global checkpoint if it is part of a "Z-cycle". However, a "Z-cycle" is just another term for a specific type of path/segment. (V) Specific types of segments and paths have already been used to define a deadlock in distributed systems [1]. Such a definition of deadlock is useful because prior definitions made the oversimplifying assumption of a global observer and a common clock. Consequently, several distributed deadlock detection algorithms based on these prior definitions were incorrect. The definition of deadlock using segments and paths overcame this drawback and clearly characterized the dynamics under which a deadlock forms and is resolved.

By controlling the predicates on how IO and OI intervals at different processes are used to define segments and paths, different types of segments and paths can be defined to capture key concepts and structures in other application areas also. See [2] for a full version of this announcement.

References

[1] A. D. Kshemkalyani, M. Singhal, Characterization and Correctness of Distributed Deadlock Detection, *Journal of Parallel and Distributed Computing*, 22(2): 44-59, July 1994.

[2] A. D. Kshemkalyani, M. Singhal, Universal Constructs in Distributed Computations, Technical Report 29.2136, IBM Research Triangle Park, March 1996.