2012 41st International Conference on Parallel Processing Workshops

Detecting Tree Distributed Predicates

Min Shen, Ajay D. Kshemkalyani, and Ashfaq Khokhar
University of Illinois at Chicago, Chicago, IL 60607, USA
{mshen6,ajay,ashfaq}@uic.edu

***Abstract*—In a large-scale locality-driven network, knowing the state of a local area is sometimes necessary due to either interactions being local and driven by neighborhood proximity or the users being interested in the state of a certain region. We propose locality-aware predicates that aim at detecting a predicate within a specified area. We model the area of interest as the set of processes that are within distance k from the initiator process. By associating the predicate with a tree topology, we force the set of processes satisfying the predicate to form a tree with height no more than k. This enables the detection of the predicate within the area of interest. We also formalize several classes of locality-aware predicates, which deal with strong stable and stable predicates for both conjunctive and relational types. The algorithms to detect each class are also proposed. These algorithms associate a tree topology constraint with the predicate to be detected. Since a locality-aware predicate detects predicates only within the specified area, the complexities of the corresponding algorithms are thus scale-free. These properties make locality-aware predicate a natural fit for detecting distributed properties in systems such as modular robotics and wireless sensor networks.**

Keywords - predicate detection; locality-aware; scale-free; modular robotics; wireless sensor networks

I. Introduction

In recent years, distributed systems have found applications in new areas such as modular robotics and wireless sensor networks (WSNs). These emerging areas share some common properties such as large scale and dynamic topologies. These properties lead to the need for robust and scalable algorithms to manage, monitor, and reason about the distributed execution in these applications. A major problem in reasoning with a distributed execution is the detection of distributed properties. The dynamism and nondeterminism of executions present challenges to observing the distributed states of the system. To solve this problem, many distributed predicate detection algorithms have been proposed [10]. Detecting an unstable predicate is an NP-complete problem. Thus most of the research in predicate detection has been on detecting stable predicates [2], [9].

However, in networks such as modular robots [1], [5], [6], [7] and WSNs, where the number of processes can be large and the events are locality driven, users are sometimes more interested in the state of a local region rather than the entire network. Hence, it makes sense to detect predicates based on part of the system rather than its entirety. For this purpose, we propose the concept of locality-aware predicates. Locality-aware predicates are similar to classical predicates. They can also be classified as conjunctive/relational based on the function on the variables involved in the predicate [4], or be classified as stable/unstable etc. based on their detectability. (A *stable predicate* is a predicate that remains true once it is found true by a global snapshot [3]. Deadlock, termination, and garbage collection are examples of stable predicate. A *strong stable predicate* is a stable predicate that, if true on some consistent cut, must remain true on all subsequent consistent cuts [11]. Termination and deadlock are strong stable, though distributed garbage collection is not.) The difference lies in that locality-aware predicates detect a predicate in a sub-network (area) of the system.

Inspired by the algorithm in [8], which can detect a stable predicate *only* on a linear chain or a ring topology, we design a set of algorithms to detect several classes of locality-aware predicates: (i) strong stable conjunctive predicates, (ii) strong stable relational predicates, (iii) stable conjunctive predicates, and (iv) stable relational predicates.

II. Locality-Aware Predicates

Locality-aware predicates aim at specifying predicates in a large-scale locality driven network such as WSNs or modular robotics. In a system like this, the users are usually more interested in the state of a certain region, rather than the entire system. This is because: (i) The number of processes in the system is large, thus knowing the state of the entire system can be quite costly, (ii) Interactions are local, driven by neighborhood proximity, (iii) In a large-scale system, the state of a certain region can contain more information than that of the entire system. One example is the following. In a token-passing system, the detection of a predicate Φ =*number of tokens is larger than 5*, defined for the global system might not contain any useful information, since the system contains many processes and the total number of tokens can easily exceed 5. However, if Φ is defined on a local region, then the detection of this predicate provides insight towards this particular region, thus contains more information. In a large-scale locality-driven system, such as WSNs, users are usually interested in such kind of properties within a certain region. Examples are number of patients in one particular area in a WSN monitored hospital environment, and number of hostile entities in a certain region in a WSN monitored battlefield.

The key aspect here is to be able to specify the area of interest. We want to detect the predicate in an area centered at the process p the user interacts with and the "radius" of the area is k, meaning that processes in the area are within distance k from p. To achieve this, we associate the predicate with a tree topology rooted at p with height no more than k. This forces the detected set of processes satisfying the predicate to form such a tree topology. We use tree topology because for any process located in such a tree, it has to be within the area of interest. Thus the detected predicate is also within the area of interest. Notice that the set of processes satisfying the predicate need not contain all processes in the area of interest.

We note that, the formalism and algorithm given in [8] can only detect a predicate on a chain or ring topology. This linear topology is not enough for specifying the area of interest. We extend the topology to include tree, thus making the topology constraint a tool to specify the area of interest.

III. Algorithms

Detecting a locality-aware predicate is not a trivial task, especially for a stable predicate. When detecting a stable predicate, we need to build a consistent cut among the processes over which the predicate is detected. For locality-aware predicates, the set of processes over which the predicate is to be detected is not the entire network. Although one trivial solution is to construct a global consistent cut and detect the locality-aware predicate based on this cut, the complexity of such a solution is affected by the size of the network. To better solve this problem, that is, to design algorithms that are scale-free, we need to build a consistent cut of only the group of processes satisfying the predicate within the area of interest.

We propose algorithms *TDP-conjunct* and *TDP-relational* that detect strong stable conjunctive predicates and strong stable relational predicates, respectively. We then push one step further by studying the problem of detecting stable locality-aware predicates, which have stronger detectability compared to strong stable locality-aware predicates. The satisfiability of stable predicates depends on channel states [8], unlike the satisfiability of strong stable predicates which does not depend on channel states.

TDP-conjunct and *TDP-relational* both can only detect predicates whose satisfiability does not depend on channel state. To capture stable locality-aware predicates using basically the same algorithms, we need to empty all channels and inhibit communication message transmission within the area of interest. By doing this, we guarantee that, for any stable locality-aware predicate, it now depends purely on process states. With the help of *Channel Freezing* algorithm, *TDP-conjunct* and *TDP-relational* can detect stable locality-aware predicates. Then, we can observe this predicate via a consistent sub-cut collected by either *TDP-conjunct* or *TDP-relational*.

IV. Complexity Analysis

We analyze the complexity of *TDP-conjunct* and *TDP-relational*. We evaluate their complexities using two metrics: message complexity and storage cost. The complexity of the algorithm in [8], which can detect a predicate only on a chain or ring topology, is shown to be related to the degree of the network and the associated topology, rather than the size of the network. Our algorithms share the same scale-free property. We evaluate the complexity under two different settings of the network. One is the fully connected network; the other is the degree-d bounded network. To make the analysis more meaningful, we compare the complexity of our algorithms with the Chandy-Lamport (CL) snapshot algorithm [3] and the algorithm (*LDP-Basic*) proposed in [8].

TABLE I
Complexity Measures in Fully Connected Network.

Metric	C-L	LDP-Basic	TDP-conjunct	TDP-relational
Message	$O(n^2)$	$O(n^{k-1})$	$O(n^2)$	$O(n^2)$
Storage	$O(n)$	$O(k)$	$O(k)$	$O(k)$

TABLE II
Complexity Measures in Degree-d Bounded Network.

Metric	C-L	LDP-Basic	TDP-conjunct	TDP-relational
Message	$O(n^2)$	$O(d^{k-1})$	$O(d^{k+1})$	$O(kd^{k+1})$
Storage	$O(n)$	$O(k)$	$O(k)$	$O(k)$

The size of the network is n, the maximum degree of the network is d, and the height constraint of TDP is k.

Message complexity is measured by counting the total number of messages generated by the algorithm. Notice that the size of the network n has no impact on the message complexity of both algorithms in the degree-d bounded network. Storage cost measures the space complexity at each process. This again is independent of n. Thus, unlike classical snapshot algorithms such as the Chandy-Lamport algorithm, the complexity of both our algorithms are not affected by n. Instead, the complexity depends on the degree of the network (d) and the parameter of the topology (k). Both *TDP-conjunct* and *TDP-relational* share the same scale-free property with the algorithms in [8]. This property gives system designers the ability to control the complexity of the algorithm's execution by changing parameter k.

References

[1] M. Ashley-Rollman, M. De Rosa, S. Srinivasa, P. Pillai, S. Goldstein, "Declarative programming for modular robots," IROS Workshop on Modular Robots, 2007.

[2] Ö. Babaoglu and K. Marzullo, "Consistent global states of distributed systems: Fundamental concepts and mechanisms," In: S. Mullender (ed.) Distributed Systems, 97-145, 1993.

[3] K. M. Chandy, L. Lamport, "Distributed snapshots: Determining global states in distributed systems," ACM Transactions on Computer Systems 3, 1, 63-75, February 1985.

[4] R. Cooper, K. Marzullo, "Consistent detection of global predicates," Proc. ACM/ONR Workshop on Parallel and Distributed Debugging, 163-173, May 1991.

[5] M. De Rosa, S. Goldstein, P. Lee, J. Campbell, P. Pillai, "Distributed watchpoints: Debugging large modular robotic systems," International Journal of Robotics Research 27, 3, 2008.

[6] M. De Rosa, S. Goldstein, P. Lee, P. Pillai, J. Campbell, "Programming modular robots with locally distributed predicates," IEEE ICRA, 2008.

[7] M. De Rosa, S. Goldstein, P. Lee, P. Pillai, J. Campbell, "A tale of two planners: Modular robotic planning with LDP," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[8] M. De Rosa, S. Goldstein, P. Lee, J. Campbell, P. Pillai, "Detecting locally distributed predicates," ACM Transactions on Autonomous and Adaptive Systems, June 2011.

[9] A. Kshemkalyani and B. Wu, "Detecting arbitrary stable properties using efficient snapshots," IEEE Transactions on Software Engineering 33, 5, 330-346, 2007.

[10] A. Kshemkalyani, M. Singhal, Distributed Computing: Principles, Algorithms, and Systems, Cambridge University Press, 2008.

[11] A. Schiper, A. Sandoz, "Strong stable properties in distributed systems," Distributed Computing 8, 2, 93-103, 1994.