Brief Announcement: STALK — A Self-Stabilizing Hierarchical Tracking Service for Sensor Networks

Murat Demirbas, Anish Arora
Computer Science & Engineering
The Ohio State University
Columbus, OH 43210
{demirbas, arora}@cis.ohio-state.edu

Tina Nolte, Nancy Lynch
MIT Computer Science & Artificial Intelligence Laboratory
Cambridge, MA 02139
{tnolte, lynch}@csail.mit.edu

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems
General Terms: Algorithms
Keywords: sensor networks, self-stabilization, tracking, fault-containment, distributed data structures

We present STALK, a hierarchy-based fault-local stabilizing algorithm for tracking in sensor networks. Starting from an arbitrarily corrupted state, STALK satisfies its specification within time and communication cost proportional to the size of the faulty region instead of the network size. Local stabilization is achieved by slowing propagation of information as the levels of the hierarchy underlying STALK increase, enabling the more recent information propagated by lower levels to override misinformation at higher levels. While achieving fault-local stabilization, STALK also adheres to the locality of tracking operations: an operation to *find* a mobile object at a distance d away requires $O(d)$ amount of time and communication cost to intercept the moving object, and a *move* of an object to a distance d away requires $O(d * log(\text{network diameter}))$ amount of time and communication cost to update the tracking structure. Furthermore, STALK achieves seamless tracking of a continuously moving object by enabling concurrent executions of move and find operations.

Overview of STALK. For achieving scalability, STALK employs a hierarchical structure. For ensuring the locality of both find and move operations, STALK adopts a partial information strategy. The tracking information is maintained with accuracy related to the distance from the mobile object: Nearby nodes that are relatively cheap to update have more recent and accurate information about the object, whereas far away nodes that are relatively expensive to update have older and more approximate information about the object.

Tracking structure. We assume a hierarchical partitioning of the sensor network into clusters based on radius. The tracking structure is a path rooted at the highest level of the hierarchy. Each process in the *tracking path* has at most one child, either at its level or one below it in the hierarchy, and the mobile object resides at the leaf of the tracking path, at the lowest level. Each process in the path points to a process that is generally closer to the object and has more recent information about its location.

PODC'04, July 25–28, 2004, St. Johns, Newfoundland, Canada.
ACM 1-58113-802-4/04/0007.

Find operation. A find operation invoked at a process queries neighboring processes at increasingly higher levels of the clustering hierarchy until it encounters a process on the tracking path. Once the tracking path is found, the find operation follows it to its leaf to reach the mobile object.

Move operation. We implement move-triggered updates by means of two local actions, *grow* and *shrink*. The grow action enables a path to grow from the new location of the object to increasingly higher levels of the hierarchy and connect to the original path. The shrink action cleans branches deserted by the object. Shrinking also starts at the lowest level and climbs to increasingly higher levels. Despite that grow and shrink occur concurrently, we achieve the move operation successfully by using suitable values for the process timers, which actuates the execution of these actions.

Fault-local stabilization. After state corruption of a region of (potentially all) processes, our tracking path heals itself in a fault-local manner within work proportional to perturbation size. We use two concepts for achieving fault-locality: hierarchical partitioning and level-based timeouts for execution of actions. The key idea is to wait for more time before updating a wider region's view. We employ larger timeouts when propagating an update to a higher level of the hierarchy, and thus, more recent updates coming from lower levels can catch-up to misinformed updates at higher levels. We achieve this by delaying a shrink/grow action for longer periods as the level of the process executing the action increases. The latency imposed by delaying is a constant factor of the communication delay to higher levels and does not affect the quality and accessibility of the tracking structure.

Concurrent move and find operations. STALK achieves seamless tracking of a continuously moving object: An object can relocate before the effects of its previous move operations finish updating the tracking path, and a find operation may be concurrently in progress with these move operations. During concurrent move operations, it is not possible to achieve a complete tracking path; there will be discontinuities in the path. By giving an upperbound on the speed of the object, we prove a reachability condition on the tracking path and ensure that if a find encounters a dead-end while following a path, there is always an available newer path nearby.

FULL VERSION is available as a Technical Report, OSU-CISRC-5/04-TR38, Ohio State University, May 2004.