Brief Announcement: Autonomous Virtual Mobile Nodes*

Shlomi Dolev[†] Seth Gilbert[‡] Elad Schiller[†||] Alex Shvartsman[‡§] Jennifer Welch[¶]

Categories & Subject Descriptors: F.1.1 [Models of Computation]: Automata
General Terms: Algorithms, Design, Reliability, Theory, Verification.

We present a new abstraction for mobile ad hoc networks designed to cope with inherent difficulties caused by processors arriving, leaving, and moving according to their own agendas, as well as with failures. An Autonomous Virtual Mobile Node (AVMN) is a robust and reliable entity that serves as part of an ad hoc middleware infrastructure. The AVMN extends the focal point abstraction in [1] and the virtual mobile node abstraction in [2]. The new abstraction is that of a virtual general-purpose computing entity, an automaton, that can make autonomous on-line decisions concerning its own movement. We describe a self-stabilizing implementation of this new abstraction that provides automatic recovery from any corrupted state of the system.

At any given point in time, the AVMN resides at a distinct location. The AVMN is implemented by the processors that happen to be near the AVMN's current location. The set of processors implementing the AVMN changes over time as it moves and as the implementing processors move (not necessarily in the same direction). The AVMN decides to move based on its current state and sensor inputs from the physical environment. For instance, if the area to the west of the AVMN appears deserted, then it may decide not to move west. Or, the AVMN may decide to "hitch a ride" with a subset of the processors currently emulating it.

In order for the algorithm to be self-stabilizing, it must be able to tolerate starting from an arbitrary configuration, in which there may be several (undesired) copies of the same AVMN, or none at all. In the former case, we must eliminate all but one copy, and in the latter case we must generate a single copy. Since the AVMN moves autonomously, there is no predictable location to search for the AVMN, making it difficult to determine the required corrective action.

Ensuring exactly one instance of an AVMN. Three schemes are considered. (1) Use a Virtual Stationary Automaton (VSA) [3] to keep track of the AVMN. The AVMN uses a GeoCast (e.g., [4]) service to send *alive* messages to the fixed known location of the VSA. If the VSA does not receive *alive* messages for too long a period, the VSA creates a new AVMN. The VSA is also responsible for the elimination of undesired copies of an AVMN. Each *alive* message carries the location of the AVMN and the timestamp at which the message was sent. The VSA can easily detect that more than one AVMN exists and send an elimination message to all but one of them. (2) The AVMN sending *alive* messages in a random walk fashion. A processor that doesn't receive an *alive* message for, say, twice the expected cover time generates a formation token that carries the identifier of the processor and traverses the network in a random walk fashion. When formation tokens collide, the lists of initiator ids carried by the formation tokens are merged. When a token contains $\lceil (N+1)/2 \rceil$ ids (where N is an upper bound on the number of processors), the (single) processor that holds the token creates a new AVMN. (3) Processors periodically send *stay alive* messages that should arrive at the AVMN by means of a random walk. The AVMN must collect at least $\lceil (N+1)/2 \rceil$ *stay alive* messages in order to survive.

Self-Stabilizing Implementation. Each participating processor keeps a replica of the AVMN's current state and a buffer of input events waiting to be applied to the state. To ensure that the replica states remain identical among all the processors that emulate the AVMN, in spite of faults, each processor, at a fixed interval, sends its replica state to all the other emulating processors. Upon receiving all the messages for the current round, a predetermined (reset) function, such as majority, is used to resolve conflicts, in case the states are not identical.

*This work is supported in part by NSF (grants CCR-0098305, NSF ITR-0121277, 64961-CS, MIT9904-12, 9988304, 0311368, 0098305, and CAREER Award 9984774), AFOSR #F49620-00-1-0097, DARPA #F33615-01-C-1896, IBM faculty award, Israeli ministry of defense, Rita Altura trust chair in computer sciences, and EU COMBSTRU.
[†]Ben-Gurion University, {dolev,schiller}@cs.bgu.ac.il [‡]MIT CSAIL,{sethg,alex}@theory.lcs.mit.edu [§]University of Connecticut, aas@cse.uconn.edu [¶]Texas A&M University, welch@cs.tamu.edu [||]CTI, schiller@cti.gr

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

References

[1] S. Dolev, S. Gilbert, N. Lynch, A. Shvartsman, J. Welch. GeoQuorums: Implementing Atomic Memory in Mobile Ad Hoc Networks. DISC'03: 306–320.
[2] S. Dolev, S. Gilbert, N. Lynch, E. Schiller, A. Shvartsman, J. Welch. Virtual Mobile Nodes for Mobile Ad Hoc Networks. DISC'04: 230–244.
[3] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, T. Nolte. Virtual Stationary Automata for Mobile Networks. T.R. MIT-LCS-TR-979, Jan. 2005.
[4] J. Navas, T. Imielinski Geocast – geographic addressing and routing, MobiCom'97.