Active and Concurrent Topology Maintenance

Xiaozhou Li*, Jayadev Misra**, and C. Greg Plaxton*

Department of Computer Science, University of Texas at Austin,
1 University Station C0500, Austin, Texas 78712–0233
{xli,misra,plaxton}@cs.utexas.edu

Abstract. A central problem for structured peer-to-peer networks is
topology maintenance, that is, how to properly update neighbor variables
when nodes join and leave the network, possibly concurrently. In this
paper, we first present a protocol that maintains a ring, the basis of
several structured peer-to-peer networks. We then present a protocol that
maintains Ranch, a topology consisting of multiple rings. The protocols
handle both joins and leaves concurrently and actively (i.e., neighbor
variables are updated once a join or a leave occurs). We use an assertional
method to prove the correctness of the protocols, that is, we first identify
a global invariant for a protocol and then show that every action of the
protocol preserves the invariant. The protocols are simple and the proofs
are rigorous and explicit.

1 Introduction

In a structured peer-to-peer network, nodes (i.e., processes) maintain some neigh-
bor variables. The neighbor variables of all the nodes in the network collectively
form a certain topology (e.g., a ring). Over time, membership may change: nodes
may wish to join or leave the network, possibly concurrently. When membership
changes, the neighbor variables should be properly updated to maintain the des-
ignated topology. This problem, known as topology maintenance, is a central
problem for structured peer-to-peer networks.

Depending on whether the neighbor variables are immediately updated once
a membership change occurs, there are two general approaches to topology main-
tenance: the passive approach and the active approach. In the passive approach,
a repair protocol runs in the background to periodically restore the topology.
Joins and leaves may be treated using the same approach or using different ap-
proaches (e.g., passive join and passive leave [12], active join and passive leave [6,
13], active join and active leave [2, 14]).

Existing work on topology maintenance has certain shortcomings. For the
passive approach, since the neighbor variables are not immediately updated, the
network may diverge significantly from its designated topology. And the passive
approach is not as responsive to membership changes and requires considerable
background traffic (i.e., the repair protocol). On the other hand, active topology

* Supported by NSF Grant CCR–0310970.
** Supported by NSF Grant CCR–0204323.
maintenance is a rather complicated task. Some existing work gives protocols without proofs [14], some handle joins actively but leaves passively [6,13], and some uses a protocol that only handles joins and a separate protocol that only handles leaves [2]. It is not true, however, that an arbitrary join protocol and an arbitrary leave protocol, if put together, can handle both joins and leaves (e.g., the protocols in [2] cannot; see a detailed discussion in Section 5). Finally, existing protocols are complicated and their correctness proofs are operational and sketchy. It is well known, however, that concurrent programs often contain subtle errors and operational reasoning is unreliable for proving their correctness.

In this paper, we first present a topology maintenance protocol for the ring topology, the basis of several structured peer-to-peer networks (e.g., [5,11,16,22]). We then present a topology maintenance protocol for Ranch, a structured peer-to-peer network topology consisting of multiple rings. Our protocols handle both joins and leaves concurrently and actively. To the best of our knowledge, our protocols are the first to handle both joins and leaves actively. Our protocols are simple. For example, the join protocol for Ranch, discussed in Section 4.2, is much simpler than the join protocols for other topologies (e.g., [2,6,13]). Our protocols are based on an asynchronous communication model where only reliable delivery is assumed.

As operational reasoning is unreliable, we use an assertional method to prove the correctness of the protocols, that is, we first identify a global invariant for a protocol and then show that every action of the protocol preserves the invariant. We show that, although a topology may be tentatively disrupted during membership changes, the protocols restore the topology once the messages associated with each pending membership change are delivered, assuming that no new changes are initiated. In practice, it is likely that message delivery time is much shorter than the mean time between membership changes. Hence, in practice, our protocols maintain the topology most of the time.

Unlike the passive approach, which handles leaves as fail-stop faults, we handle leaves actively (i.e., we handle leaves and faults differently). Although treating leaves and faults the same is simpler, we have several reasons to believe that handling leaves actively is worth investigating. Firstly, leaves may occur more frequent than faults. In such situations, handling leaves and faults in the same way may lead to some drawbacks in terms of performance (e.g., delay in response, substantial background traffic). To see this, note that only four messages is needed to handle an active leave (see Section 3.2), while a linear number of messages is needed to detect a passive leave. Secondly, while a node can leave the network silently, we consider it reasonable to assume that a node will execute a leave protocol, because nodes in peer-to-peer networks cooperate with each other all the time, by forwarding messages or storing contents. Thirdly, as an analogy, communication protocols like TCP have “open connection” and “close connection” phases, even though they handle faults as well.

Our work is only a first step towards designing topology maintenance protocols that have rigorous foundations. For example, a shortcoming of our protocols is that some of them may cause livelocks; see a detailed discussion in Section 4.4. We outline some future work in Section 6.