A Generic Approach to Dependability in Overlay Networks

Barry Porter and Geoff Coulson

Computing Department, Lancaster University, Lancaster, UK
(barry.porter; geoff)@comp.lancs.ac.uk

Overlay networks are virtual communication structures that are logically "laid over" underlying hosting networks such as the Internet. They are implemented by deploying application-level topology maintenance and routing functionality at strategic places in the hosting network [1, 2]. In terms of *dependability*, most overlays offer proprietary "self-repair" functionality to recover from situations in which their nodes crash or are unexpectedly deleted. This functionality is typically orthogonal to the purpose of the overlay, and a systematic and complete approach to dependability is rarely taken because it is not the focus of the work. We therefore propose to offer dependability as a *service* to any overlay.

Dependability is a well-studied field in distributed applications, but many of the existing approaches for applications are unsuitable for overlays; a common approach is to submit an application to a fault-tolerant framework for controlled execution within that framework [3]. We suggest instead that a dependability service for overlays is built only from decentralized, lightweight agents which exist *alongside* overlay nodes, operating at the same level and with the same resources available to their nodes. In addition, such agents should maintain only soft state which can be re-built automatically simply by existing in the environment, making the service inherently self-repairing. An architectural model of this is shown in figure 1.

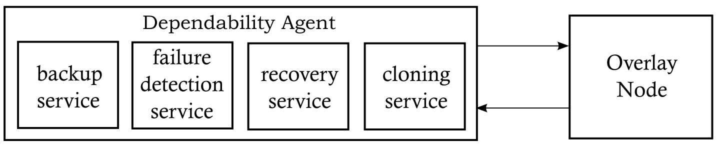

Fig. 1. An example configuration of the proposed overlay dependability service.

This decentralization creates some interesting challenges, not least of all that recovery of a failed node can be initiated by multiple different instances of the recovery service that notice the failure of a neighbouring node. The service instances must then ensure that exactly one proposed recovery is chosen from potentially many in order to maintain a sensible system. We also note that overlays often operate in a purely end-user host environment, which can be quite limited in resources (or at least resources that users are happy to give up). This makes some classic approaches infeasible, like the use of dedicated checkpointing servers. Checkpointing is a well-known method of making elements of a distributed system survivable—their state is periodically saved on dedicated servers so that it can be restored if they fail. Without the ability to provision checkpointing servers, checkpoints must be stored on the hosts used by the overlay itself, and must be distributed across those hosts in such a way that multiple failures are survivable.

Checkpointing itself, though often used in distributed applications, is not very suitable for overlays. Checkpointing is best used in deterministic systems, a class that many overlays are not part of—the data stored by Chord and the members of a multicast tree are both driven by users, for example, which makes those aspects appear random. Since overlays can, then, be in a near-constant state of non-determinism, their nodes would require frequent checkpointing, which can amount to a significant performance overhead.

To help alleviate issues like this we propose that overlays be loosely defined by two basic types from the point of view of the dependability service, which we term *accessinfo* and *nodestate* records; the former provides the service with the neighbours of a node to both save and communicate through, and the latter gets any part of a node that needs to be backed up (both are defined by the overlay and their internals are transparent to the service). If supported by the overlay, nodestate records may themselves be divisible into nodestate *units* which could, for example, map to individual resources stored at a node, again as determined by the overlay. This finer-grained abstraction gives us opportunities to back up only the elements of a node which change.

We are also interested in the performance of overlays when they are operating across highly heterogeneous hosts, as many overlays assign equal responsibility to each node regardless of the capabilities of its host. Re-using our definition, we can migrate nodestate units to "cloned" versions of their original nodes instantiated on alternative hosts, re-routing messages appropriately, to help alleviate pressure on the more sparsely resourced hosts that are part of an overlay. This behaviour is encapsulated in our cloning service, which monitors resources and attempts to prevent resource exhaustion by migrating and tracking nodestate units.

We intend to investigate solutions to all of the presented problems and expand on our loose overlay definition in our effort to create a generic, configurable and efficient dependability service for overlay networks.

References

1. Mathy, L., Canonico, R., Hutchinson, D.: An Overlay Tree Building Control Protocol. Proceedings of the 3rd International COST264 Workshop on Networked Group Communication, London, UK, 2001
2. Stoica, I. et al: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Proceedings of SIGCOMM'01, California, USA, 2001
3. Bagchi, S., Whisnant, K., Kalbarczyk, Z., Iyer, R.: The Chameleon Infrastructure for Adaptive, Software Implemented Fault Tolerance. Proceedings of the Symposium on Reliable Distributed Systems, Indiana, USA, 1998