CSC2231 – Internet Systems and Services

Paper Review – Total Recall
Name: Alex Wun
Date: Nov. 23rd, 05

The authors propose a P2P data replication scheme that automatically determines the amount of replication needed to satisfy a certain level of availability. Their scheme is built on top of Chord and is intended for public overlay networks. Small files (<32KB) are replicated eagerly – complete copies are sent to new storage hosts if existing hosts of the file fail or go offline. Larger files use erasure coding (being able to reconstruct the entire file given that some blocks are missing) and lazy repair to ease bandwidth use.

Overlay networks still appear to be missing the "killer application", so this paper makes a useful contribution by experimenting with a potential P2P application. In three-tier architectures, the database is often the performance bottleneck. In this regard, leveraging P2P's natural scalability seems to be a good approach.

However, the authors acknowledge that a prior paper has shown that cooperative storage over P2P is infeasible due to the cost of dynamic membership. Although the authors claim that their research remains compelling, they never justify why TotalRecall is exempt from this limitation. Furthermore, public P2P networks suffer from a great deal of churn as the authors themselves point out. Studies of Overnet show that hosts join and leave the network over 6 times a day on average. While studies of Kazaa have shown that the median session duration is 60 minutes. However, the TotalRecall simulations are based on traces in which the average uptimes are 28.6 and 109 hours – orders of magnitude greater than what other studies have found. This brings into question whether their experiments accurately model the effects of short-term churn in public overlays.

Also, their traces contain 7 days' worth of host availability data. Considering that the average uptimes are greater than a day (4.5 days for the File System trace), there does not seem to be enough data to accurately observe the effects of long-term churn either. Additionally, the authors mention that they wait until the simulated system reaches a "steady-state" before the experiment begins. This further reduces the effective duration of their simulation.

Even though it seems that there may be issues with the feasibility of their approach and possibly their experimental results, this is still an interesting approach. The authors mention that the need for probing storage hosts is unscalable and suggest using "random sets". However, it's conceivable that hosts could select their neighbours as storage hosts – thereby incorporating storage host "awareness" into the existing DHT maintenance algorithm rather than treating it as a separate scheme. For instance, Pastry already maintains a neighbour set that could double as a storage host set – although that may cause problems when the network topology changes.