Experiences Deploying a Large-Scale Emergent Network

Bryce Wilcox-O'Hearn

Zooko.Com Software Engineering
zooko@zooko.com

Abstract. "Mojo Nation" was a network for robust, decentralized file storage and transfer. It was first released to the public in July, 2000, and remained in continuous operation until February, 2002. Over 100,000 people downloaded and used the Mojo Nation software. We observe some surprising and problematic behavior of the users as a group. We describe several specific problems in the design of Mojo Nation, some of which appear to be soluble with simple practical improvements, and others of which are not yet addressed in the literature, suggesting opportunities for further research.

Introduction

Mojo Nation[1] was not a file-sharing system (like Gnutella or Napster), but an "emergent file store", in which the storage, transfer and naming of files was performed in a decentralized manner, independent of any individual node. It had much in common with systems like CFS[2], PAST[3] and OceanStore[4], both in goals and in design. Mojo Nation was designed from the start with ambitious goals of attack-resistance and scalability.

The first version of Mojo Nation was released to the public in July of 2000. It had many advanced features, but deployment to large numbers of end users inevitably revealed its architectural deficiencies. During its lifespan, its developers deployed literally hundreds of changes to the protocol in response to observed behavior and in order to take advantage of newly discovered techniques. For example, in August of 2001, shortly after reading a pre-print of [5], we deployed a new version that used consistent hashing to locate a block in a set of servers.

We typically observed more than 10,000 downloads of the Mojo Nation software per month, as shown by statistics published by SourceForge.net[9]. (Note that before August of 2001 downloads were not hosted by SourceForge, although some of the web pages were. The higher number of page views in October of 2000 were a result of Mojo Nation being featured on slashdot.org, the consequences of which will be described below.)

Unfortunately, the vast majority of these users who tried Mojo Nation were not satisfied by the service it offered, as indicated by the fact that they permanently stopped using the network after trying it only briefly. At its largest, Mojo Nation never exceeded 10,000 simultaneously connected nodes, and during the majority of its 19-month lifespan it had between 100 and 600 persistent nodes. This paper is motivated by the desire to learn from this failure.

There are many important aspects of the Mojo Nation product which we must omit from consideration in this paper. These include Mojo Nation's user interface, marketing, distributed search engine service, lightweight resource accounting scheme, agnostically blindable digital tokens and more. In this paper we will focus on the basics: the individual nodes, connecting them into a network, and building a decentralized file store on that network.

Observed Behavior

Frequent Join / Leave

The most surprising and problematic behavior that users of Mojo Nation displayed was frequent joining and leaving. We observed that the most common behavior was to join the network, stay connected for less than an hour, then leave the network and never return. Measurements taken from two particular 1-month periods (October, 2000 and February, 2001) indicated that between 80% and 84% of the users fell into this group of "one-time, less than one hour" group, and that of the remaining 16% to 20%, a significant fraction stayed connected for less than 24 hours then permanently disconnected.

Even among the remaining persistent nodes (those that recurringly connected to the network over a period of weeks), the typical node remained connected for only a short consecutive time, and only a few times per week. One measurement taken in April 2001 showed that the average node was connected 0.28 of the time, and other, less systematic observations suggest that the distribution was highly skewed, with approximately 1/6 of the nodes connected almost all the time, and the rest connected approximately 3 hours per day.

The overall picture suggested by these observations is that the "network half-life", or the time for replacement of half of the nodes in the network by new arrivals, was usually less than 1 hour, and at times it was much less than 1 hour.

Varying Space Allocation

The default disk allocation per node in Mojo Nation was originally 100 MB. In April of 2001 we raised the default to 500 MB. Users can manually adjust that setting. The Mojo Nation software did not report to us what settings the user chooses, but we do know from support mail and user feedback that no users have complained about the default setting, and that many users are quick to point out that they have raised their limit to a high setting, usually in the range of 10 GB to 60 GB.

Varying Connection Quality

Market research reports (e.g. [7]) typically suggest that around 13% of Internet users have broadband connections, and the rest use relatively slow and intermittent dial-up connections. Anecdotal evidence from Mojo Nation was consistent with this. However, there was an active minority of users with very high quality connections (including academic and corporate networks). These users also tended to be in the minority that stay consistently connected and in the minority that allocate large amounts of disk space.

Routability

Measurements taken at various times over the life of the Mojo Nation network always returned the same answer: 1/3 of Mojo Nation nodes were not directly reachable from the Internet, as observed by the fact that they did not have routeable IP addresses. In addition, some unknown number of users may have had routeable IP addresses, but may have been behind firewalls that did not allow incoming TCP connections.

Which Parts Worked?

Mojo Nation was a complex system and it is difficult to ascribe its successes to individual components. It could be described in general as a file storage and transfer network in which there is a mechanism for global coordination without communication (e.g. consistent hashing to locate nodes and data blocks in a ring), and in which individual nodes use local information to decide how to store, transmit, replicate and cache data. When Mojo Nation worked, it was a demonstration that such a network can be deployed and operated in an environment made up of unmanaged volunteers.

When Mojo Nation failed, its failures can more easily be ascribed to particular design elements.

Which Parts Failed? (Open Problems)

Original Introduction

The only failures which rendered the network completely unusable for all new users (not counting occurences of the authors releasing a new version with fatal bugs), are failures of original introduction. "Original introduction" is the problem of how a node connects to the network for the first time, when it does not yet have any connections to any other nodes in the network. The first version of Mojo Nation used single central introducer. Each new node would contact that introducer and receive in response a list of other nodes.

The Great Slashdotting of October 2000 was a dramatic demonstration of the inherent weakness in this design. In October 2000 an entry was posted on the popular web site slashdot.org headlined: "Forget Napster & Gnutella: Enter Mojo Nation" [8]. The next day our web server reported that downloads of the software had rocketed from 300 copies per day to almost 10,000 copies per day. The central introducer was totally overloaded and was not returning any responses to any users. We struggled for days to make the server operate, but it wasn't until the flash crowd had died down and we took the time to implement a new system of introduction (involving multiple redundant but still centrally managed introducers), that the network became usable again.

The issue of original introduction is largely ignored by the extant literature. There are several solutions to the problem in use on currently deployed networks including redundant centrally-administered introducers (FastTrack, Mojo Nation), bundling a list of original contacts with the download of the software (Limewire, Freenet), asking users to manually configure the original connection (Freenet), and combinations of more than one of these techniques (Limewire).

The scalability, security and attack-resistance trade-offs implicit in these design decisions have not been publically analyzed as far as we know.

Data Availability

Even when the network as a whole was working, a very common failure was that the data that a user sought was unavailable. We ascribe the source of that problem to our design's failure to accomodate the highly unreliable behavior of the nodes. Simultaneously, we believe that the primary reason for frequent join/leave behavior was that the data users sought was unavailable. This constitutes a “chicken-and-egg” problem, which was exaccerbated instead of solved by other elements of our design.

We repeatedly tuned our replication and information dispersal design in order to counteract this problem, but even at the best, data availability was variable, and appeared to depend upon which server nodes were connected at the time an observation was made.

There were two significant mistakes that Mojo Nation made which can be easily avoided.

The first was that it did not discriminate against newly joined nodes. As described in [6], the length of time that a node has been continuously connected to the network is a good predictor of the length of time that it will remain connected into the future. A simple heuristic to favor long-lived nodes, such as that proposed in [6], would have reduced the problems caused by frequent join/leave behavior.

The second was that Mojo Nation used an erasure code to split the data into a set of shares such that any sufficiently large subset of the shares would suffice to rebuild the data. The number of shares required to rebuild the data was equal to 1/2 of the total number of shares generated. This was intended to increase data robustness, but if the availability of the underlying shares is less than the “required to total” ratio (in this case, less than 1/2), then such an erasure code has the opposite of the intended effect, dramatically reducing the robustness of the data. We believe that the other problems of data availability were thus compounded by the addition of an erasure code with a “required to total” ratio that was too high for the actual behavior of the nodes.

As noted in the “Future Research” section of [2], the issue of how to manage block storage in the face of servers joining and leaving remains mostly open. More sophisticated caching and replication strategies will hopefully ameliorate this problem. In addition "reputation" or "trust metric" techniques such as described the section on "Attack Resistance" below might help by discriminating against unreliable servers. Mojo Nation deployed software which attempted to do exactly that, but the interaction between this discrimination and other design goals is not well analyzed.

Other Open Problems

Bypassing Firewalls and NAT

The challenge of enabling nodes that live behind firewalls or NAT to act as servers is a challenge that most deployed systems do not yet attempt to address. It is also likely to become more rather than less important in the future as the size of the Internet grows and as application-level connections cross more administrative boundaries.

Mojo Nation used a "relay" technique in which a third node helps two firewalled nodes to communicate with one another, similar in principle to [10].

Attack Resistance / Malicious Nodes / Mutual Distrust / Motivation to Cooperate

Perhaps the most challenging unsolved problem is that of mutual distrust. While a network architect is tempted to assume that all nodes in the system behave as he designed them to behave, this assumption may prove fatal once a network is deployed into multiple disjoint administrative zones.

A fundamentally related issue is that of "motivation to cooperate". Why does a node choose to offer services to the network as well as to make requests of the network? Is there anything preventing a user from altering their copy of the software, or writing their own compatible implementation, which uses the resources of the other nodes but refuses to provide its own resource to them?

Also closely related is the notion of "attack resistance". If a node can use the resources of other nodes without offering them service in return, then it is able to act as a drain on the resources of the network as a whole, possibly constituting a denial-of-service attack on the entire network.

On the other hand, if a node can be coerced into cooperating, perhaps by cutting that node off from the services of the network in retaliation for its lack of cooperation, how can we be sure that the same mechanism cannot be used to attack specific (innocent) nodes, or even to attack the network itself?

Hopefully the research pursued in papers like [11] and [12] will lead to a quantitatively justified method of gaining attack resistance without sacrificing other design goals.

Mojo Nation's experience shows that there are two kinds of attack that are likely to be encountered by any network that is deployed in a large scale on the Internet.

The first attack is when a user alters his client in the attempt to gain more advantage for himself. Several different users made such modifications to their

Mojo Nation software and then helpfully contacted us to describe what they did. Other users have made modifications, but we are aware of those changes only indirectly through observations of anomalous behavior.

The second kind of attack is when an enemy attempts to remove central components of the network through legal means. Legal action was recently initiated[13] against the Fast Track network even though the only centrally administered components are the original introducer service and the design, implementation and distribution of the software.

Conclusion

As an emergent file store, Mojo Nation was partially a success and partially a failure. The parts that failed were a centralized original introduction mechanism and a data storage scheme that proved too fragile when deployed on a network with a surprisingly short half-life.

These two problems can be straightforwardly solved in practice, and they also present possible directions for the advancement of theory.

In addition, we believe that any long term, large scale emergent network will need to address the “other open problems” of attack resistance, malicious nodes, mutual distrust, and motivation to cooperate.

References

1. Web Site: Mojo Nation.
`http://mojonation.net/`
2. Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP ’01) (To appear; Banff, Canada, Oct. 2001).
`http://citeseer.com/dabek01widearea.html`
3. Druschel, P., Rowstron, A.: PAST: A large-scale, persistent peer-to-peer storage utility.
`http://citeseer.com/439820.html`
4. Kubiatowicz, J., et al.: OceanStore: An Architecture for Global-Scale Persistent Storage. ASPLOS, December 2000.
`http://citeseer.com/kubiatowicz00oceanstore.html`
5. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT, March 2001.
`http://citeseer.com/stoica01chord.html`
6. Maymounkov, P., Mazières, D. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02).
`http://scs.cs.nyu.edu/~{}{}dm/kpos.pdf`
7. Web Site: Press Release: “U.S. Residential Internet Market Grows in Second Quarter”.
`http://www.isp-planet.com/research/2001/us_q2.html`

8. Web Site: Slashdot headline: "Forget Napster & Gnutella: Enter Mojo Nation". `http://slashdot.org/article.pl?sid=00/10/09/1826243`
9. Web Site: SourceForge Usage Statistics: Mojo Nation. `http://sf.net/project/stats/index.php?report=months&group_id=8340`
10. Ng, T. S. E., Stoica, I., Zhang, H.: A waypoint service approach to connect heterogeneous internet address spaces. Proceedings of the Usenix Technical Conference (June 2001), pp. 319–332. `http://citeseer.com/ng01waypoint.html`
11. Levien, R., Aiken, A. Attack-resistant trust metrics for public key certification. 7th USENIX Security Symposium, January 1998. `http://citeseer.com/levien98attackresistant.html`
12. Dingledine, R., Freedman, M.J., Molnar D. The Free Haven project: Distributed Anonymous Storage Service. Workshop on Design Issues in Anonymity and Unobservability, July 2000 (LNCS 2009). `http://www.freehaven.net/doc/berk/freehaven-berk.ps` `http://citeseer.com/dingledine00free.html`
13. News Article: "Suit hits popular post-Napster network", CNet News.Com. `http://news.cnet.com/news/0-1005-200-7389552.html`