Service Duality: Vitalizing the Commons

UNPUBLISHED DRAFT

Bryan Ford
M.I.T.

July 24, 2003

Abstract

Service duality is a design principle for peer-to-peer protocols, whose purpose is to create direct incentives for nodes to provide good service, and to minimize the network's vulnerability to denial-of-service attacks based on attempts to overload good nodes. In protocols designed this way, nodes providing naturally complementary services, or *duals*, form symmetrical, self-reinforcing relationships in which they use each other to provide a common larger service. Each node constantly monitors the performance of the partner nodes it depends on during the course of its normal operation, in order to find the best partners and provide good performance to its local user. When serving requests from other nodes, each node gives preferential service to its *primary partners*: those nodes that it relies on and have given it the best service in the past. This paper explores the principle of service duality in the abstract and in terms of how it might be applied in two specific protocols: a distributed routing protocol and a file sharing protocol.

1 Introduction

A persistent thorn in the side of Peer-to-Peer (P2P) systems, commonly described as "the tragedy of the commons," is that their incentive structures often encourage users to behave selfishly to the detriment of the system as a whole. For example, Napster and Gnutella users often lie about their connection bandwidth in order to discourage incoming connections [9], or simply choose to share few or no files [3], since these "selfish" behaviors do not have any direct negative impact on the service they receive from others and can leave more bandwidth available for their own downloads.

In systems such as Freenet [1] and Chord/DHASH [4], in which nodes can "push" data into the network to be stored on other nodes, there seems to be no satisfactory way to ensure, or even encourage, users to store and reliably serve back as much as much data or metadata as they themselves insert. In the worst case, this flaw can leave such systems open to denial-of-service attacks where malicious nodes flood the network with insertions of useless data. PAST [8] suggests a smart card-based quota scheme to solve this problem, whereas OceanStore [5] avoids it by assuming centralized administration. Neither of these solutions seem satisfactory from a purely decentralized peer-to-peer perspective.

MojoNation [2] attempted to create a global "virtual currency" for peer-to-peer services such as storage space and download bandwidth, in which nodes would "earn" currency by providing services and "spend" currency when consuming services provided by other nodes. This system failed to take off, however, perhaps in part due to the technical complexity of the currency scheme, but no doubt also due to its *perceived* complexity to users, who would be forced to set prices and effectively barter for on-line resources that are usually considered "too cheap to meter" [10].

Freeloading by itself is not necessarily a fatal flaw, as long as there are enough "good citizens" to compensate for the freeloaders and ensure that everyone gets a decent level of service. Freeloading can even be a good thing, because it allows new users (potential "future good citizens") to try the system before investing substantial resources of their own into it, thereby increasing the overall growth rate of the user community. But as a system grows and diversifies, the likelihood of malicious denial-of-service attacks increases, and current peer-to-peer systems make such attacks all too easy.

1.1 The Solution

What we need is a way to design peer-to-peer systems so that they will *directly reward* good citizenship, thereby encouraging users to contribute resources to the system, and limit the effects of denial-of-service attacks based on "maliciously bad citizenship." This incentive structure must be accomplished without substantially complicating the user's experience or preventing the harmless variety of freeloading. This paper presents a novel (and untested) design principle called *service duality*, which could enable peer-to-peer systems to provide this property. The principle is potentially applicable to a variety of different types of peer-to-peer services, including distributed routing and data storage.

The central idea of service duality is to design peer-to-peer systems so that a larger distributed service is built out of a *pair* of smaller "sub-services." Nodes that can provide one of the sub-services seek and form bonds with nodes that reliably provide the complementary sub-service (its "dual"). Once that bond is established, each node can then reliably provide the larger service, by serving requests for "its half" directly and forwarding requests for the other half to its partners. This larger service may in turn have a dual, for which the node in turn seeks reliable providers in order to build an even larger service, and so on.

These pairwise bonds between nodes need not be based on any explicit agreement; rather, these bonds are merely emergent properties resulting from the nodes independently pursuing "rational self-interest." For a given sub-service, each node seeks other nodes that provide the complementary sub-service, and in making or forwarding requests for that service, relies primarily on the complementary nodes that have given the fastest and most reliable service in the past. In return, each node *provides preferential treatment* for requests from (or forwarded by) the nodes it relies on primarily. For example, a node might service requests from its primary partners before serving queued requests from unknown or less reliable nodes, and drop requests from its primary partners last in the event of overload. By giving preferential treatment to its most reliable partners, the node will in turn be seen to be more reliable by those partners, and will be more likely to receive preferential treatment from them. Each node is merely pursuing its own self-interest: namely, the goal of providing the larger service reliably to its local user. But since the node can only provide half of this service directly and must rely on other nodes to provide the complementary half, it is in the node's interest to favorably treat other nodes that provide good service, in order to increase its own likelihood of receiving good service from them.

There may be many nodes providing a given sub-service, as well as many nodes providing its dual, all of which are seeking to form relationships in order to provide the larger service to their local users. Suppose the nodes within each group exhibit a wide variation of service quality, as is inevitably the case in real peer-to-peer networks [9]. If nodes give preferential treatment to their primary partners, then the best-connected and most reliable nodes will naturally gravitate toward each other over time, limiting their vulnerability to unreliable nodes. Less reliable nodes will gravitate toward other nodes with similar levels of reliability from whom they can obtain preferential treatment, and also toward more reliable nodes that have sufficient excess service capacity to provide good service to non-primary partners. It is in every node's interest to be as close to the "core" as possible, in order to receive preferential treatment from highly reliable nodes—but the only way for a node to get there is by staying up and providing good service.

Service duality is ultimately just an application of the principle of *quid pro quo*, the foundation on which all manner of self-reinforcing, mutually bene-

ficial relationships are built in human and biological societies—the ultimate "peer-to-peer networks." As such the plausibility of this principle should not be surprising; the trick is in its implementation. The rest of this paper therefore focuses on how this design principle might be applied to specific peer-to-peer services. These protocol designs are not complete, implemented, or tested; their purpose is merely to suggest possible directions and promote discussion on how we might design future P2P systems to provide more robust incentive structures.

2 A Routing Protocol

As a starting point, consider a P2P routing service akin to Chord [4] or Pastry [7], where the goal is to route message to nodes according to node identifiers that have no relationship to geography or the underlying network topology. Assume that the distribution of node IDs is reasonably uniform, and that there is some way to verify that a node has a "right" to a given node ID. For example, the node ID may be a SHA-1 hash of the node's IP address, in which case the test is to send a challenge message to that IP address; or the node ID may be the hash of a public key, in which case the test is to send a challenge message asking the node to sign a response with the corresponding private key.

The overall service each node wishes to provide to its local user, then, is the ability to route a message to any node in the network by its node ID. Applying the principle of service duality, the obvious way to subdivide this "large" service into smaller services is by subdividing the node ID space. For any n-bit pattern p, S_p is the service of routing messages to any node whose ID has prefix p. If n is less than the length of the node ID, then the service corresponding to an n-bit prefix p can therefore be viewed as the aggregation of two smaller (more specialized) services S_{p0} and S_{p1}: services for prefix p with a 0 bit and a 1 bit appended, respectively.

The only "atomic" service a node can provide without any help from other nodes is the service of routing messages to itself: i.e., the service S_q where q is the node's own ID. But if the node can provide service S_{p0}, and find other nodes who provide the complementary service S_{p1}, then the node can use these relationships to provide the service S_p; and similarly if the node starts with service S_{p1}. If no other nodes can be found providing the complementary service for a given prefix length, which is likely to be the case for long prefixes, the node merely assumes that there are no nodes having that prefix.

To route a message to a given ID other than its own, the node picks one or more of its most reliable partners providing routing service for the longest matching prefix, which will be at least one bit longer than the prefix of any service the node itself provides. As long as the addressed node exists and the network is sufficiently connected, the message will eventually be forwarded to the proper node. Each node monitors the results of requests it forwards to other nodes in order to determine which of its peers at a given prefix length are providing the best service, and uses that information in future routing decisions. The node also uses this information to prioritize requests from its most reliable peers above requests from unknown or less-reliable nodes. Each node therefore has a strong incentive to monitor its partners carefully and make good routing decisions for a given prefix length, because its own reputation is on the line for the higher-level services it provides to other nodes, and its reputation has a direct impact on the service it receives from its partners in turn.

Kademlia[6] already implements a routing protocol very similar in its basic operation to the one described here, so the general structure of the protocol has already been proven to work. The key element introduced by the service duality principle in this case is the idea of giving preferential treatment to partners that have proven reliable in the past, thereby promoting the formation of long-term, self-reinforcing relationships and reducing the system's vulnerability to denial-of-service attacks. It is the natural symmetry of the interactions between nodes in this protocol—the property that each node seeks out other nodes providing exactly complementary services—that enables this incentive structure.

3 A File Sharing Protocol

Consider next a file sharing protocol built on top of the routing protocols above, in which files are addressed by a content hash such as SHA-1. Chord/DHASH [4] and PAST [8] simplify this problem by "pushing" data to the nodes whose node IDs most closely match the content hash of the data. As mentioned above, however, this scheme provides little direct incentive for selfish nodes to store much, if any, of the data that is actually pushed to them. Furthermore, the scheme assumes that all nodes provide roughly similar amounts of storage, which is unlikely except under highly controlled conditions.

An alternate data sharing scheme, based on service duality, is for each node to choose independently which parts of the content hash space it wants to serve directly, and build relationships with other nodes in order to construct services for larger portions of the content hash space. Suppose a node wishes to publish a particular set of files. For each file to be published, the node offers a service S_q, where q is the complete content hash of the file. Then for each n-bit proper prefix of q, the node searches the network for other nodes offering the complementary service, serving files whose content hash shares the first n bits but differs in the immediately following bit. As in the routing protocol, each node builds upward from more specialized services with longer prefixes toward more general services with shorter prefixes.

Each node evaluates its partners not only according to their speed and reliability but also by their hit rate. If the node forwards a request for a file with content hash h to two of its partners, and one responds quickly indicating the requested block doesn't exist, but the other responds (perhaps more slowly) with the contents of a block matching the requested content hash, then the rating of the first partner is reduced in favor of the second. As with the routing protocol, each node provides preferential service to partner nodes that have provided the best service in the past, creating an incentive for nodes to be both accurate and reliable. Since the ultimate responsibility for storing and serving a given data file remains with the node that wants to publish that file, however, the storage incentive problem common to other schemes does not apply.

One obvious potential problem with this scheme is the sheer number of relationships each node may have to maintain, since the node must keep a table of partners at various prefix lengths *for each file to be published*. This issue is only likely to be of concern to nodes that publish a very large number of files, however. To address this problem, it may be possible to layer yet another protocol on top of this one which well-connected nodes can use to "trade" responsibility for publishing data sets of similar size, so as to narrow the range of content hashes they are directly responsible for into a specific portion of the content hash space and thereby reduce the number of relationships they must maintain.

4 Potential Weaknesses

Servers or centrally-administered server clusters that have provided very good service for a long time may become a weakness in a system based on service duality, in the sense that they may cause considerable disruption if they suddenly disappear or start behaving maliciously. An attacker with substantial resources could conceivably insert a cluster of well-behaving, reliable servers into the network for long enough for them to become widely trusted and trickle into the core of the network, then suddenly cause these servers to misbehave all at once in the worst possible way in order to cause the maximum disruption of the overall system. Such an attack would be inherently somewhat self-defeating, however, because in order to infiltrate these nodes into the network the attacker has been forced to feed substantial resources into the system, creating a large net benefit to the system up to the point at which those nodes become malicious. In other words, the attacker has effectively amortized the harm caused by his attack by all the good service he has provided up to that point. For this reason, such a method of attack is likely to be effective only in situations in which it is the *timing* of the disruption that is critical for some reason (e.g., if the attack is timed to coincide with some other important event) rather than merely to disrupt the system in general.

5 Conclusion

Service duality is a design principle for peer-to-peer protocols, in which self-reinforcing relationships between nodes providing complementary services are used to form an incentive structure that directly rewards "good citizenship." The specific protocols outlined here have not been implemented or tested, and may need considerable modification to be made practical—though the similarity of the routing protocol described here to Kademlia provides some evidence of plausibility. At any rate, it seems clear that *some* design principle of this general variety will need to become a pervasive element in any peer-to-peer protocol that is intended to survive truly "in the wild," among untrusted nodes not under any common administrative domain and whose users are primarily motivated by self-interest.

References

[1] http://freenetproject.org/.

[2] http://mojonation.net/.

[3] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[4] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, May 2001.

[5] John Kubiatowicz et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.

[6] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[7] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[8] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[9] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN) 2002*, San Jose, CA, USA, January 2002.

[10] Clay Shirky. In praise of freeloaders. *openp2p.com*, December 2000.