1 Network-aware middleware for multi-server data distribution

The enormous popularity of Peer-to-Peer (P2P) file sharing has shown that it is indeed possible for a large number of people (well in the millions) to exchange terabytes of data over a significant period of time. Over the course of the last few years well over a dozen popular protocols have been developed and several are open-source. Recent estimates have shown that P2P traffic is easily over half the number of bytes in the Internet and has only increased. As recently as four years ago HTTP was responsible for 75% of the bytes. The rapid shift in traffic is mainly due to the popularity of the content exchanged in P2P networks. A very large fraction of P2P content exchanged is questionable in nature in terms of content ownership. Television-quality copies of movies, DVD images, entire CDs, individual songs in MP3 format etc. are the primary content formats exchanged. Popular protocols include KaZaa/Morpheus, eDonkey/eMule, BitTorrent, DirectConnect etc. There is a reasonable separation of protocol and content; specific protocols appear to be used for specific content. A significant amount of work has gone into optimizing the delivery of the 'shared' content from a smaller number of sources to a large number of users with varying degrees of connectivity. For example, the primary problem noticed with Gnutella, the protocol that was popular early on in the P2P world, was that of free-riding. There were many 'peers' who downloaded data but did not necessarily share the data. Freeloading has been largely solved in the recent P2P protocols (e.g., eMule) where peers essentially keep track of the upload/download ratio and downgrade delivery to peers who do not have a fair ratio. The other problem in P2P world has been the introduction of 'decoys' on behalf of the content owners who want to reduce the 'free' downloads ('stealing'). eDonkey has enabled checksum comparison to reduce the risk of downloading of decoys. KaZaa has maintained control of content downloading by resorting to encrypting the transfers (including many of the headers).

A key technical advance in P2P has been the breaking the large media files into chunks ('parts') and allowing the P2P clients to download them from multiple servlets and assembling them. This has allowed for parallel downloads to speed up the fetching of the large objects. The selection of chunk sizes is driven by individual site considerations although there has been a rough consensus in certain protocols. The chunks have individual signatures (often MD4 hashes, computed offline) and the headers include

resource size to ensure content integrity.

A key point in the existing work on P2P has been that it is driven largely by a open-source friendly community (with the signal exception of KaZaa) and the primary motivation has been to get around the difficulties of large scale delivery of fat content in the face of legal troubles, bandwidth constraints, freeriding etc. Our proposal faces none of the above constraints and is more closely related to the problem of content distribution. Traditional Content Distribution Networks (CDNs) arose in the context of the World Wide Web to reduce the overhead on busy Websites. If cnn.com receives tens of millions of hits (each of which can be a separate TCP connection in the absence of HTTP/1.1 persistent connections) for the many small images it has on its Website, it may be unable to handle the load. CDNs *offloaded* this work and using DNS as the request routing load balancing mechanism, delivered the small images on behalf of the busy Web sites. The various models of CDN delivery and their effectivness in reducing the latency perceived by the user has been examined (cite bala's IMW01 'On the Use and Performance of Content Distribution Networks' paper). Of late CDNs have been delivering streaming media content as well. But the motivation of CDNs has never been to deliver large files to many users.

The advantage of grid oriented computing is that the user base is known in advance and thus their connectivity. Similarly the file size ranges are generally known and their chunking can be determined easily keeping in mind the classification of users' connectivity. Several informed algorithms can be tried to find the right chunking size and their placement location. The set of algorithms to try based on user's connectivity class, the distributions of delays on the respective paths, and the ability to efficiently replicate the chunks in the right sites are all problems not examined in the P2P or the CDN worlds.