

GhostShare- Reliable and Anonymous P2P Video Distribution

Srinivas

September 29, 2005

1 Summary

GhostShare [1] is an application built over the Pastry substrate, to distribute video content. The main design issues are load balancing, system reliability, indexing the content for efficient search, maintaining the anonymity of both the source and the destination. The following sections explain the different techniques proposed to solve the design issues considered.

1.1 Keyword Lookup Service

When a user Z wants to share a file on the Pastry network, the hash function is applied to the file in two different ways. 160-bit hash is applied to each of the keywords in the name of the file i.e., for the file “Matrix Revolutions”, we get two distinct hashes X_1 and X_2 , referred as token hashes for each of the words in the file name. After this, a hash Y is generated from the content of the file. Thus the user Z forms a tuple $\langle X_i, Y, Z \rangle$ and sends each tuple for all i , to the Pastry node whose nodeID is numerically closest to the token hash X_i . Thus, each node in the Pastry network will have a collection of all the tuples whose token hash is closest to its ID. For the purpose of indexing the list is maintained as two separate mappings—one from filename token hashes \rightarrow content hashes, and another content hashes \rightarrow node IDs. Therefore the node whose ID is numerically closest to X_i knows that Y is shared by Z and returns this information to the requested user. In an event that a node fails or disconnects then the list that it is maintaining is lost. To avoid this, some kind of duplicacy is provided by storing each tuple $\langle X_i, Y, Z \rangle$ not only at the node whose ID is closest to X_i , but also at the node whose node ID is second closest to X_i .

1.2 Load Balancing

suppose a given node A is numerically closest to token hash X_i which belongs to a very popular file on the network. Every time this file is searched for, a message will travel to this node asking for nodeIDs sharing this file. The node will be

overwhelmed by huge number of requests and can break down if it has limited bandwidth. An obvious way to do load balancing is to have the replicas of the list in d number of nodes, but then every time the request will be routed to the node numerically closest to X_i because of the basic pastry routing algorithm. To avoid this and have a fair load balancing, it is proposed that every time a node sends a request asking for nodeID closest to X_i , the node permutes the first d bits of X_i and sends the request onto the pastry network. Assuming that pastry nodeIDs are uniformly distributed, the requests are thus uniformly distributed among the 2^d nodes.

1.3 Disjoint MultiPath

Pastry routing mechanism helps us to find the optimal path between two nodes A and B , B being the closest to the token hash X_i that A is searching for. If a bottleneck exists between path, considering the fact that in an overlay network, nodes can join, leave and fail unpredictably, finding an alternate route can drastically improve performance. Thus multiple paths that are disjoint over the interior nodes, i.e., the only shared nodes between them are the source and destination nodes are build for reliable communication. So, the first path between A and B is set using the traditional routing technique of Pastry. While establishing this path, each interior node on it sends the hash of its nodeID to A . Node A collects these nodeIDs and forms an exclusion set, which it uses to construct multiple disjoint paths.

1.4 Anonymity

Anonymous transfer of files implies that the source doesn't know the destination's IP address and destination doesn't know source IP address. This is done by allowing the node to perform searching, downloading and publishing through a representative. For eg., the node A has the file with the token hash X_i and wants to share the files, it then request a random node B selected from its Routing Table|Leaf Set|Neighborhood Set and sends a request asking the node to be its representative. B with probability p will advertise itself as the owner of A 's files and with probability $1-p$ forward the request to a randomly selected node from it's node set. Thus the source and destination will choose representatives to publish and download files and thereby maintaining the anonymity.

2 Review

The performance of GhostShare with the inclusion of multiple path and disjoint functionality has shown to be efficient by simulation in terms of number of hops required for searching the content and balancing of load over the network when compared to the basic pastry. It also implements good strategy to maintain anonymity of the users.

2.1 Strengths

1. The paper gives an insight on the ability of building several applications over the pastry network. It has considered an application of video distribution and tried to bring out the inefficiency of pastry substrate to address some issues like load balancing, reliability.
2. The indexing of filenames with the use of two different hashes, one for keywords of the file name and other for the content is well presented.
3. The reliability of getting the information is ensured by disjoint multiple paths. And, also Index reliability is obtained by storing the index not only at the closest node to the token hash X_i , but also at the second closest to X_i .
4. The anonymity of both the sender and destination is maintained by choosing representative.

2.2 Issues Unanswered

1. Keyword Search:

Is it the efficient and sufficient way to store the hash of the keywords in the filename. If the movie name contains general words like “The”, “Of”, “Under” etc, do we also need to do hash of these words?

If we should, then it would result in large number of result sets, which would be difficult to handle over the network. And, also while searching for video we generally also specify the format of the video file we are looking for, in which case how does the indexing help us?

To maintain index reliability, GhostShare is replicating the index in the second closest node to the token hash X_i . What if both the nodes fail simultaneously?

2. Reliability:

It is not specified how the index list is replicated over 2^d nodes i.e. what is the criteria used and in index reliability section it is mentioned that only two nodes share the same index list which is sort of creating confusion.

Disjoint multiple paths is suggested for finding alternate paths quickly and improve the performance. The simulations are carried out considering 3 multiple paths, but the reason for choosing 3 is not clarified.

And, to choose the first path, pastry’s routing technique is used, but while choosing the second disjoint path, the criterion for choosing the next hop is not specified and the only criterion mentioned is the next nodeID at every hop should be different from those mentioned in the exclusion set. Is this sufficient information?

3. Load Balancing:

Does permuting the first d bits ensure that the load is distributed uniformly. And, is this process done only when the destination node is overwhelmed or every time node A is looking for token hashes X_i ?

4. Anonymity:

What is the incentive that a node will agree to be the representative for source or destination?

And, also the content is downloaded at the source through the forwarding chain, thus incentives need to be provided for all the nodes to be participate in the chain and perform store and forward operation.

References

- [1] A. Nandan, G. Pau, and P. Salomoni. GhostShare - Reliable and Anonymous P2P Video Distribution. *IEEE GlobeCom Workshops 2004*, pages 200–210, 2004.