A Ring Infrastructure for Neighbor-Centric Peer-to-Peer Applications

Oliver Haase, Alfred Toth, and Jürgen Wäsch

HTWG Konstanz - University of Applied Sciences, Constance, Germany

Abstract. We propose a peer-to-peer system that supports distributed virtual world applications. For these applications, the connections between directly neighboring peers are of the utmost importance. To minimize wide area network traffic and average latency, peers that belong to the same subnet, are grouped together, and these groups are interconnected via wide area connections. To build up and maintain this optimized peer-to-peer structure, we developed a set of protocols that efficiently handle the joining and leaving of peers as well as failure situations. Peers are aranged in the logical ring structure using a two-step discovery and join procedure. The first step uses broadcast messages to discover peers in a local subnet, followed by a local join. If no peer answers in the local subnet, a remote join is performed. With the implemented recovery procedures, our peer-to-peer system can survive multi-node failures in a local subnet as well as the failure of an entire subnet.

1 Introduction

Peer-to-peer networks support a variety of different applications, including file sharing, telecommunication, multimedia streaming, web caching, distributed collaboration, and shared virtual world implementations. Evidently, a good peer-to-peer infrastructure has to efficiently support the needs of its applications [1]. For file sharing applications, e.g., the pre-dominant operation is the retrieval of a (key, value) pair for a given key. Many peer-to-peer infrastructures, such as distributed hash tables [2,3,4,5,6], are optimized for exactly this retrieval operation.

For distributed shared virtual world applications, however, the situation is different. Typically, the virtual world is divided into separate, neighboring areas that are distributed among the participating peers. Data exchange mainly takes place between neighboring areas, because the beings that inhabit the virtual world can move from one area to a neighboring one.

One example of a shared virtual world is Aqualife, a peer-to-peer application that simulates a distributed ecosystem. In Aqualife, each participating peer runs a part of the virtual global aquarium, and hosts fish that interact with each other, and that can swim from one peer to another. A peer has a preceeding and a succeeding neighbor that its fish can swim to and from, so that as a result all peers together form a logical ring. When a new peer joins the community, it needs to connect to a succeeding and to a preceeding neighbor, but its actual

T. Enokido, L. Barolli, and M. Takizawa (Eds.): NBiS 2007, LNCS 4658, pp. 355–364, 2007.

position in the ring is irrelevant from the application point of view. Thus, to minimize network traffic and latency, it is advisable to build the peer ring upon the topological proximity of the peers.

Topological proximity takes into account not only the geographical distance between two peers, but also the characteristics of the interconnecting network including bandwidth, throughput, and latency. Evidently, determining the topological distance between any two peers is a complex and costly task; what is even more, it is a metric that varies over time, as the network load and other parameters change. Also, it is generally not possible to map the surface of the earth onto a ring while at the same time preserving topological distances.

On the other hand, the single one type of proximity that has the greatest impact on both network load and latency, is whether or not two peers belong to the same subnet. Taking that differentiation into account is very beneficial and, at the same time, feasible with comparably simple and robust procedures.

Our peer-to-peer infrastructure groups peers that belong to the same subnet together in a chain, and interconnects these local chains to a global ring, see Figure 1. This approach minimizes the amount of wide area network traffic and the average latency. In addition, it reduces the number of connections that have to traverse firewalls and NAT boxes, and that need to be taken special care of.

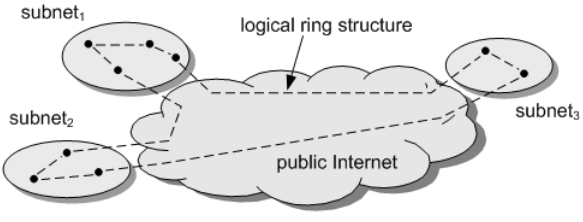

Fig. 1. Global ring structure interconnecting local peer chains

We achieve this optimized peer-to-peer structure by a two-step discovery and join procedure. In the first step, the new peer broadcasts a discovery request into its subnet. If at least one local peer replies, the new peer initiates a *local join* procedure to have itself inserted into the local peer chain.

If no local peer responds to the discovery broadcast, the new peer performs the second step of the discovery procedure and uses a bootstrap server to be put in contact with any one peer in the community. The new peer uses the contact to request a *remote join* procedure. During this procedure, care is taken not to place the new peer between two peers that belong to the same subset, so as not to corrupt the optimal structure shown in Figure 1.

The bootstrap server is the only central entity in an otherwise serverless peer-to-peer infrastructure. It helps new peers to contact the existing community by maintaining a partial list of known peers in its peer cache. To ensure scalability, the bootstrap server caches the addresses of a constant number of peers only and operates completely statelessly on a simple request-response protocol. Its