ABSTRACT

Peer-to-Peer computing is a popular, relatively new, distributed computing paradigm. It allows for a flexible set of participants to coordinate their resources with little overhead or reliance on central servers/services and is becoming particularly relevant in mobile computing environments, where peers come and go. Communication between an (unknown) number of peers, which may or may not be online at the same time, is greatly facilitated by the publish/subscribe model. In this chapter, the authors review the state-of-the-art in publish/subscribe systems, focusing on the support for mobile peers in infrastructure-based networks. They propose a novel handoff approach that proactively distributes pub/sub-related information to brokers/superpeers ahead of a peer’s movement. They show through extensive experiments in a small testbed that the new approach has significant performance benefits, compared to the more typical reactive approach, in which pub/sub context is only established after a handoff event occurred.

MOTIVATION

Peer-to-Peer (P2P) computing has gained both scientific and social importance recently due to the success of systems such as Freenet, Gnutella and Napster. Harnessing P2P technology has the potential to produce systems that combine good scalability with minimal infrastructure cost. P2P
systems are designed to start out small and seamlessly evolve to very large distributed systems with thousands of participants.

To coordinate the communication between entities, the publish/subscribe model is very promising (Kangasharju, 2005). As P2P applications grow, the need for selective data dissemination increases, in particular for many-to-many aggregate queries, which is at the core of the publish/subscribe approach. Many publish/subscribe systems have been implemented, using either centralized or distributed matching and routing algorithms (Burcea et al., 2004), (Cugola et al., 2001), (Fiege et al., 2003), and (Muthusamy et al., 2005). More recently, a number of researchers have explored how to implement publish/subscribe over large P2P networks in a scalable fashion. This is done either using structured P2P overlays such as CHORD or CAN, or unstructured P2P overlays, see for example (Bender et al., 2007), (Chand & Felber, 2005), or (Choi & Park, 2006).

We are interested in particular in the performance of Pub/Sub approaches in a cellular/infrastructure-based wireless network. In such networks, publishers and subscribers are free to roam randomly, attaching to a fixed core network at different locations over time. Example networks are cellular networks (GSM or CDMA), WiFi-based wireless LANs, or WiMAX/Wireless Mesh Access networks. One important characteristic of such networks is the fact that they contain a fixed core, which enables these networks to scale to many users. This contrasts to the other popular form of wireless networks, Mobile Ad-Hoc Networks, or MANETs, which are completely infrastructure-less and typically rather small-scale (at most 100 users/nodes).

Efficiently supporting publish/subscribe-style applications in such infrastructure-based mobile networks raises some interesting challenges. Typically, mobile devices, connected over a wireless link, are severely resource-limited in terms of CPU power, memory size, and/or network connectivity. One option to implementing a P2P-based pub/sub middleware in such an environment is to treat the mobile devices simply as weak peers but to otherwise not distinguish further between mobile and stationary devices, as advocated in (Kellerer et al., 2005). This will, in general, not result in good overall application performance. As our own experience (Kunz & Omar, 2002) and others such as (Barr et al., 2002), (Fox et al., 1998), (Zenel & Duchamp, 1997) have shown, the key to increasing the performance of mobile applications is to partition the application. Code/application logic that requires lots of computational power and/or high communication bandwidth is offloaded to the fixed infrastructure.

A similar division of the application logic can be seen in many P2P designs. For example, the Gnutella network originally consists of a number of equal nodes, called peers or servants. These peers are connected by an application level overlay network that provides routing and forwarding of Gnutella messages. A newly participating servant can connect to Gnutella network by handshaking with the already connected node whose address is learned somehow out-of-band. Once a servant has connected successfully to the network, it communicates with the other servants by sending and receiving Gnutella protocol messages. Having random connections with the other servants results in, among other things, routing inefficiencies. To address this problem, the ultrapeer system has been introduced by organizing nodes in a hierarchical fashion with ultrapeers and leaves. A leaf keeps only a small number of connections with ultrapeers. On the other hand, an ultrapeer maintains many leaf connections as well as a small number of connections to the other ultrapeers. It acts as a proxy to the Gnutella network for the leaves connected to it and shields leaves from the majority of message traffic (Park et al., 2007). KaZaA has a similar notion called superpeers, which are well-connected, resource-rich peers, providing service to regular peers.

Publish/Subscribe applications also support this approach to application partitioning well by