9

Gargantuan Computing—GRIDs and P2P

9.1 Introduction

In this chapter we are going to consider GCaP capacity planning techniques for gargantuan-scale computer systems such as so-called peer-to-peer (P2P) networks and computational GRID networks. One of the best known working examples of a GRID-style computing system is `SETI@Home` (Search for Extraterrestrial Intelligence at Home), where a scientific workload, viz., processing radio-telescope signals, is farmed out to a gargantuan number of floating-point operations per second (FLOPS) in the guise of millions of otherwise idle personal computers—many being home PCs.

One of the best known working examples of a P2P-style computing system is *Skype* (`www.skype.com`), which allows millions of people to use their PCs like a free telephone by forming its own gargantuan network (Fig. 9.1) which supports the voice over Internet Protocol (VOIP). Other well-known P2P architectures include Gnutella, Napster, Freenet, Limewire, Kazaa, BitTorrent, instant messaging, WiFi, PDAs and even cellphones. Mnay of these architectures have progressed from simple file transfer protocols to a viable means for distribution of applications such as games, movies, and even operating systems.

This class of system offers the potential for very large-scale implementations. Consequently, it is appropriate to draw on the concepts of system scalability developed in Chap. 4 as well as the concepts of virtualization developed in Chap. 7. The general goal for these architectures is to enable scalable virtual organizations that can provide a set of well-defined services.

Key to the performance of these systems is the particular choice of network topology and its associated bandwidth. To assess the scalability of network bandwidth, this chapter draws on performance bounding techniques described in (Gunther 2005a, Chap. 5). We shall apply those same techniques to the performance analysis of a particular P2P network called *Gnutella* (commencing in Sect. 9.3) since the pros and cons of its capacity have been so well documented. First, we review some of the distinctions between GRIDs and P2P networked computer systems.

9.2 GRIDs vs. P2P

P2P networks and GRIDs share the common focus of harnessing resources across multiple administrative domains. Therefore, they may be distinguished in the following way:

GRID: Supports a variety of applications with a focus on providing infrastructure with quality of service to moderate-sized, homogeneous, and partially trusted communities (Foster 2005). Grid toolkits provide secure services for submitting batch jobs or executing interactive applications. The network architecture tends to be more centralized, hierarchical, and static.

P2P: Support intermittent participation in vertically integrated applications for much larger communities of untrusted, anonymous individuals. P2P systems provide protocols for sharing and exchanging data among nodes. The network architecture tends to be more decentralized, and dynamics require resource discovery.

It should be kept in mind that both these technologies are very much in the process of evolving and have by no means reached a final form. Because they are likely to become more ubiquitous, it is important for our purpose as GCaP planners to understand something about them.

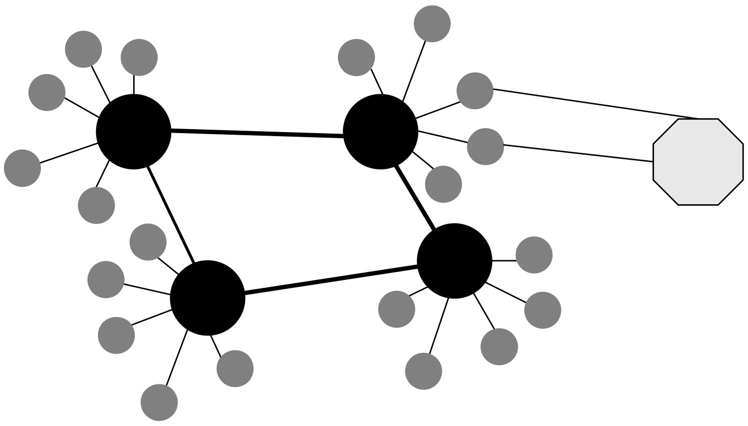

Fig. 9.1. The Skype network showing the network connectivity between its three main entities: supernodes (*black circles*), ordinary nodes (*gray circles*), and a login server (*gray octagon*)

GRID computing is focused on scientific and engineering applications and attempts to provide diverse resources that interoperate. The concept behind the GRID is analogous to the electrical power grid. It is available on demand and it does not matter where you are. When you throw the switch, you expect the light to come on. Consequently, GRIDs should be built from standard interfaces and protocols, and the Open Grid Services Architecture (OGSA) provides *The Globus Toolkit* as an implementation of such a standard based on Web services and technologies. OGSA is a product of the GRID community at large, and it has a major focal point in the Global Grid Forum (GGF). Members of the Globus Alliance have made significant contributions to the development of OGSA. The interested reader can find more information about goals, toolkits, and implementations at the OGSA website `www.globus.org/ogsa`.

These technologies are not mutually exclusive. The P2P model could help to ensure GRID scalability. Architects could employ P2P technologies and techniques to implement decentralized GRID systems in order to avoid or alleviate performance bottlenecks. A recent example of this approach is *GRIDnut* (Talia and Trunfio 2004) based on Clip2, the original Gnutella protocol specification `www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, to which we now turn our attention.

9.3 Analysis of Gnutella

The Gnutella network is a class of open-source virtual networks known as *peer-to-peer* networks. Compared to the more ubiquitous client–server distributed architectures, every P2P node (or *servant*) can act as both a client and a server. Many client-server applications, e.g., commercial databases, have multiple clients accessing a centralized server (see Gunther 2005a, Chap. 9). Conversely, P2P network applications are usually completely decentralized.

Finding applications that can make efficient use of P2P is the current gating factor for their widespread adoption. So far, P2P networks have been employed for such applications as the Napster (`www.napster.com`) music file-sharing service and the `SETI@Home` project (`setiathome.berkeley.edu`), although both those implementations rely on a significant centralized server component.

The initial release of Gnutella in 2000 led to the perception that the intrinsic architecture may not be capable of scaling to meet the sharing demands of millions of anticipated[1] users. Similar concerns about scalability have arisen in the context of hypergrowth traffic impinging on popular e-commerce Web sites (see Chap. 8). Based on measurements of popular queries, it was proposed that Gnutella scaling problems could be ameliorated through the implementation of appropriate caching strategies. Other measurements indicated that

[1] In 2001, the size of the Napster network was 160,000 simultaneous users, down from a peak of 1.6 million reported by Webnoize in February, 2001.

there were more readers than writers involved in file sharing. They suggested that a propensity for reading could lead to higher than expected load on the P2P network, thereby degrading its performance as well as increasing its vulnerability to fragmentation.

A mathematical analysis by Ritter (2002) (one of the original developers of Napster) presented a detailed numerical argument demonstrating that the Gnutella network could not scale to the capacity of its competitor, [2] the Napster network. Essentially, that model showed that the Gnutella network is severely bandwidth-limited long before the P2P population reaches a million peers. In each of these previous studies, the conclusions have overlooked the intrinsic bandwidth limits of the underlying topology in the Gnutella network: a Cayley tree (Rains and Sloane 1999) (see Sect. 9.4 for the definition).

Trees are known to have lower aggregate bandwidth than higher dimensional topologies, e.g., hypercubes and hypertori. Studies of interconnection topologies in the literature have tended to focus on hardware implementations (see, e.g., Culler et al. 1996; Buyya 1999), which are generally limited by the cost of the chips and wires to a few thousand nodes. P2P networks, on the other hand, are intended to support from hundreds of thousands to millions of simultaneous peers, and since they are implemented in software, hyper-topologies are relatively unfettered [3] by the economics of hardware.

In this chapter, we analyze the scalability of several alternative topologies and compare their throughput up to 2–3 million peers. The virtual hypercube and the virtual hypertorus offer near-linear scalable bandwidth subject to the number of peer TCP/IP connections that can be simultaneously kept open. We adopt the abbreviation *hypernet* for these alternative topologies. The assumptions about the distribution of peer activity are similar to those employed by Ritter (2002). This is appropriate since our purpose is to rank the relative performance of these hypernets rather than to predict their absolute performance.

9.4 Tree Topologies

In the subsequent discussion, the P2P network is treated as a graph, i.e., a set nodes or vertices connected by a set of edges or links. The nodes correspond to network peers, and the links to the links to network connections.

Because the tree structure of the Gnutella network has been such a hidden determinant underlying the conclusions drawn in previous scalability studies, we commence our performance comparisons by distinguishing clearly among

[2] At the height of the media attention, Napster's legal problems drove some 50,000 users per day over to Gnutella such that peers connected by 56 Kbps phone lines caused the P2P network to fragment into disconnected "islands" of about 200 peers.

[3] As the `SETI@Home` project has demonstrated, 2.8 million desktops (and 10 PetaFLOPS) can be harnessed for free.

the relevant tree topologies. Topologically, all trees are planar and thus have $d = 2$ spatial dimensions.

9.4.1 Binary Tree

The binary tree is familiar in the computing context by virtue of its ubiquity as a parsing and storage data structure. There is a unique root node that is connected only to two sibling nodes, and each of those siblings is connected to another pair of sibling nodes, and so on. At each level h in the tree, there are 2^h nodes. Therefore, the number of nodes grows as a binary exponential number. Because of its relatively sparse nodal density, the binary tree is rarely employed as a bona fide interconnection network.

9.4.2 Rooted Tree

A rooted tree is simply the generalization of a binary tree in which each node (other than the root) has a vertex of degree v. The total number of nodes is the sum of a geometric series:

$$N_{bin}(h) = \frac{v^h - 1}{v - 1} \,. \tag{9.1}$$

9.4.3 Cayley Tree

A Cayley tree (Rains and Sloane 1999) has no root. Recalling the binary tree, what was the root of the parent binary tree now has a link to an another binary subtree of height one less than the parent. All nodes thus become trivalent with $v = 3$ at every level. More generally, for a v-valent tree, the total number of nodes is given by:

$$N_{cay}(h) = 1 + \sum \; v \; (v - 1)^{h-1} \,, \tag{9.2}$$

and therefore is denser than the number of nodes in (9.1).

This is the central formula used in the scalability analysis of Ritter (2002). The network he analyzed is thus a Cayley tree with vertex degree v corresponding to the number of open network connections per servant. Ritter analyzed valences in the range $v = 4 \ldots 8$; the former value being the default setting in the original Gnutella release, and the latter more closely resembling the number of peers claimed for the contemporaneous Napster network.

9.5 Hypernet Topologies

An alternative to bandwidth-limited trees is a topology with higher dimensionality. We examine the performance attributes of two hypernets in particular: the binary hypercube and the hypertorus, each in d dimensions.

9.5.1 Hypercube

In a Boolean or binary hypercube each node forms the vertex of a d-dimensional cube. The number of nodes is simply 2^d, and the degree of each vertex v is equal to the dimensionality d of the network. Hence, each node can be enumerated or addressed using a base-2 (binary) d-digit number. Moreover, since neighboring nodes differ in address by only 1 digit, sending a message on the hypercube becomes a simple matter of shifting successive bits as the binary address passes each node between source and destination.

In $d = 3$ dimensions the hypercube is simply a cube. Each vertex has degree $v = 3$, so there are $2^3 = 8$ nodes. A 4-dimensional hypercube, can be visualized as spatially translating a 3-cube such that the locus of its four vertices trace out the additional connections.

9.5.2 Hypertorus

A d-dimensional hypertorus is a d-dimensional grid with each node connected to a ring of nodes in each of the d orthogonal dimensions. The hypertorus reduces to the binary hypercube when there are only two nodes in each ring. The simplest visualization is, once again, in three dimensions. A two-dimensional grid is first wrapped about one axis such the edges join to form a tube. The tube is wrapped about the orthogonal axis to form a ring such that the open ends of the tube become joined. The result is a 3-torus, otherwise known as a donut.

All of these topologies fall into a class known as single stage networks, and they are relatively easy to implement in software. The more exotic topologies, such as cube-connected cycles, butterflies, and other multistage networks, are not considered here because they are likely to be more difficult to implement.

9.6 Capacity Metrics

9.6.1 Network Diameter

The notion of a network diameter is analogous to the diameter for a circle. There, it is the maximum chordal length between two points on the circumference. For a network, it is the maximum number of communication links that must be traversed to send a message to any node along the shortest path. It represents a lower bound on the latency to propagate messages throughout the entire network. In 1997 the Web was estimated to comprise more than half a million sites (Gray 1996). By 2001, it was estimated (OCLC 2004) to have grown to 3.1 million publicly accessible sites.

The diameter of the Web has been estimated to be about 20 hops. If the Web is modeled as a Cayley tree, its height would be half the diameter, i.e., $h = \delta/2 = 10$ hops (Table 9.1). A vertex degree of 5 (connections per node) would contain just under half a million nodes, while a vertex degree of 6 would contain nearly 3 million (2,929,687) nodes.

Table 9.1. Network diameter

Topology	δ
Tree	$2h$
Hypercube	d
Torus	$dN^{1/d}/4$

9.6.2 Total Nodes

Next, we determine the total number of peer nodes in the P2P network. For a binary tree:

$$N(h) = \sum_{k=1}^{h} 2^{k-1} . \tag{9.3}$$

For a d-dimensional binary hypercube the number of nodes is 2^d.

9.6.3 Path Length

The path length is the maximal distance between a leaf node and the root. For a tree, it is half the diameter. The path length corresponds the peer horizon used by (Ritter 2002) in his analysis. A better measure of network latency is the average number of hops H, which we shall define shortly.

9.6.4 Internal Path Length

The internal path length is the total number of paths between all nodes. For a binary tree of depth h, the total number of paths is:

$$P(h) = \sum_{k=1}^{h} k \; N(k) . \tag{9.4}$$

9.6.5 Average Hop Distance

Since the network diameter is a maximal distance, it tends to overestimate message latency. A better measure is the average number of hops between source and destination. This quantity is found by dividing the internal path length in (9.4) by the total number of nodes in (9.3)

$$H = \frac{P}{N} . \tag{9.5}$$

It corresponds to the average number of network hops traversed by a P2P query.

9.6.6 Network Links

This is a measure of the number of physical network links. As revealed in Table 9.2, L scales with the number of physical nodes N for the topologies we are considering.

Table 9.2. Number of network links

Topology	L
Tree	N_{tree}
Hypercube	$dN_{\text{cube}}/2$
Torus	dN_{torus}

9.6.7 Network Demand

The transit frequency across a link f_{link} is a measure of the average query size per link. Under the assumption of uniform message routing, it can be defined as:

$$f_{\text{link}} = \frac{H}{L} \,. \tag{9.6}$$

If the latency across a link is denoted by S_{link}, then the total service demand (Gunther 2005a, Chap. 2) is:

$$D_{\text{link}} = f_{\text{link}} \; S_{\text{link}} \,. \tag{9.7}$$

For simplicity and without loss of generality, we normalize the network demand to unit periods, i.e., $S_{\text{link}} = 1$.

9.6.8 Peer Demand

In a manner similar to the definition for the time spent on a link S_{link}, we define S_{peer} for node latency. Under the assumption of uniform message routing:

$$f_{\text{peers}} = \frac{1}{N} \,, \tag{9.8}$$

and the total peer service demand is:

$$D_{\text{peers}} = \frac{S_{\text{peer}}}{N} \,. \tag{9.9}$$

Again, we normalize the peer demand to unit periods ($S_{\text{peer}} = 1$) in the subsequent discussion.

9.6.9 Bandwidth

It follows from Little's law, $U = XD$ (Gunther 2005a, p. 44) that when any node in the network reaches saturation i.e., $U = 1$, the maximum in the system throughput is determined by:

$$X_{\text{max}} = \frac{1}{Max(D_{\text{peers}}, D_{\text{link1}}, D_{\text{link2}}, ...)} . \tag{9.10}$$

The node with the longest service demand D_{max} is the system bottleneck. The service demand at the bottleneck therefore determines the maximum system throughput. With these metrics defined, we are in a position to compare the asymptotic performance of each of the topologies described in Sects. 9.4 and 9.5.

9.7 Relative Bandwidth

Since we are interested in network scalability up to a few million peers, it is sufficient to base the comparison on the asymptotic network throughput defined in (9.10). In particular, we will rank the above hypernets according to their relative maximal bandwidth,

$$X_{\text{relative}} = X_{\text{max}}(N)/N , \tag{9.11}$$

where N is the number of peers in the horizon (Table 9.3 at the end of this section). $X_{\text{relative}} = 1.0$ corresponds to linear scalability since $X_{\text{max}} = N$ in (9.11).

In several respects our approach is similar to that taken by (Culler et al. 1996) for their LogP model of assessing parallel hardware performance. In both approaches, the respective network topology enters into the performance model via the network demand defined in Sects. 9.7 and 9.9.

9.7.1 Cayley Trees

First, we consider the relative performance of tree topologies. Figure 9.2 shows the normalized bandwidths of a fourth-degree rooted tree, a 4-valent Cayley tree and an 8-valent Cayley tree.

The 4-valent Cayley tree represents the default peer connectivity in the original release of Gnutella. Similarly, the 8-valent Cayley tree corresponds to Ritter's comparison with Napster scalability. The curves in Fig. 9.2 terminate at different peer populations because the population is an integral multiple which is dramatically affected by the vertex degree and the height of the tree.

We see immediately that the 8-valent Cayley tree has the greatest bandwidth up through 2 million peers. The 4-valent Cayley tree has the lowest bandwidth, even lower than the rooted tree. This follows from the fact that at its root the 4-tree has the same connectivity as the 4-Cayley tree, but all its descendants have vertices of 5 degrees. Even for the 8-Cayley, at 2 million peers the bandwidth is less than one quarter of linear scalability.

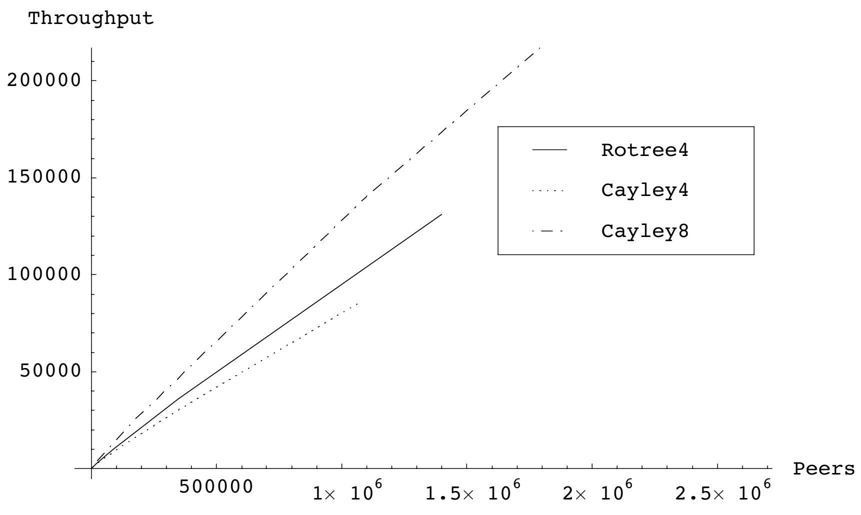

Fig. 9.2. Relative throughput of binary and Cayley trees

9.7.2 Trees and Cubes

We next consider the relative performance of high-degree trees and hypercubes. In particular, Fig. 9.3 shows the normalized bandwidths for an 8-Cayley

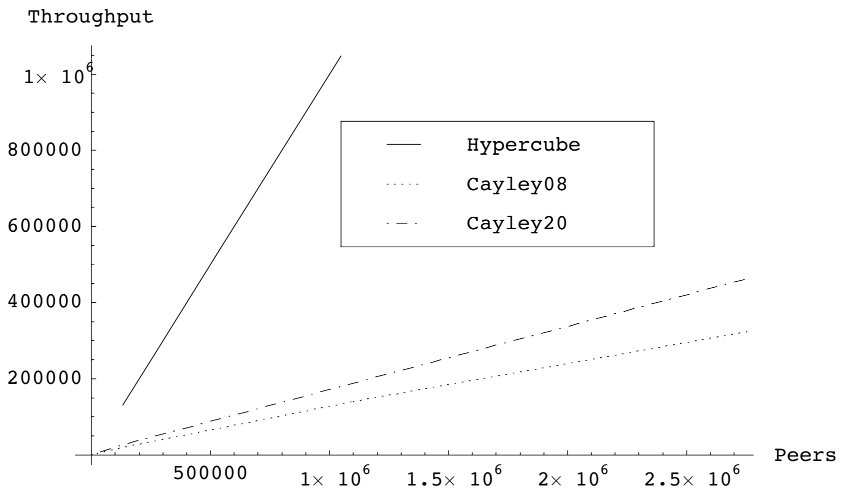

Fig. 9.3. Relative throughput of Cayley trees and hypercubes

(the best throughput of the trees considered in Fig. 9.2), a 20-Cayley, and a binary hypercube. The d-dimensional hypercube clearly exhibits superior scalability.

9.7.3 Cubes and Tori

Of these high-order topologies, the binary hypercube offers linearly scalable bandwidth beyond one million active peers (Fig. 9.4). The ten-dimensional hypertorus has comparable scalability up to one million peers but degrades beyond that point. The three-dimensional hypertorus is also shown for com-

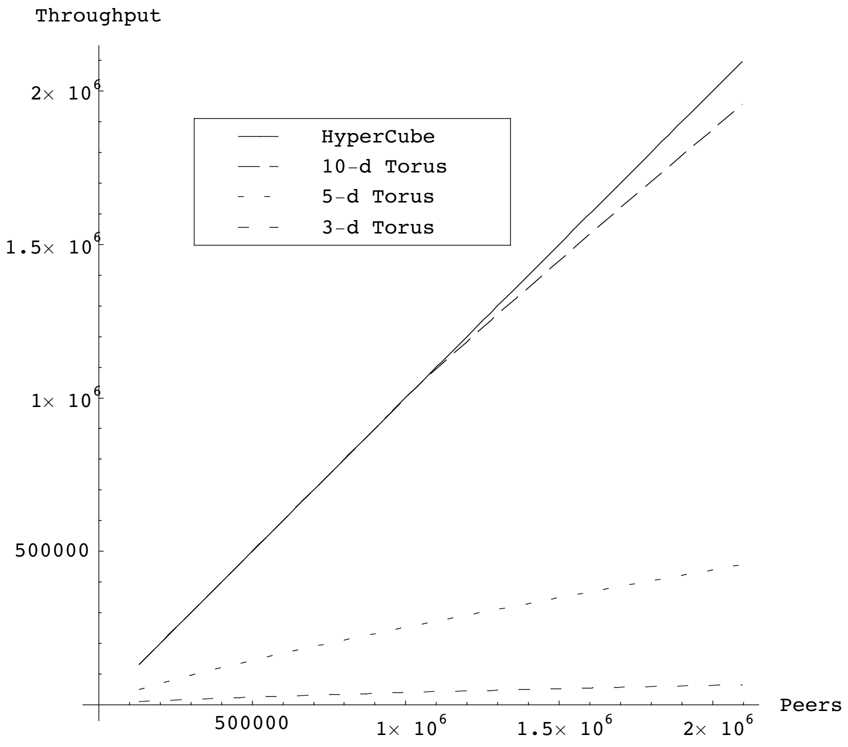

Fig. 9.4. Relative throughput of hypercubes and hypertori

parison since that topology has been used in large-scale hardware implementations up to several hundred nodes per cluster, e.g., HP NonStop s88000 server (formerly the Tandem Himalaya).

9.7.4 Ranked Performance

The main results of our analysis are summarized in Table 9.3, which shows each of the topologies ranked by their relative bandwidth as defined in (9.11). The 20-dimensional hypercube outranks all other contenders on the basis of query throughput. For an horizon containing 2 million peers, each servant must maintain 20 open connections, on average. This is well within the capacity limits of most TCP/IP implementations. The 10-dimensional hypertorus is comparable to the 20-hypercube in bandwidth up to an horizon of 1 million peers but falls off by almost 10% at 2 million peers. The 10-torus is also arguably a more difficult topology to implement.

Table 9.3. Topologies ranked by maximal relative bandwidth

Network topology	Connections per peer	Hops to horizon	Peers $\times 10^6$ in horizon	Relative (%) bandwidth
20-Cube	20	10	2.1	100
10-Torus	20	11	2.1	93
5-Torus	10	23	2.1	22
20-Cayley	20	6	2.8	16
8-Cayley	**8**	**8**	**1.1**	**13**
4-Tree	4	11	1.4	12
3-Torus	6	96	2.1	10
4-Cayley	**4**	**13**	**1.1**	**8**

The 20-valent Cayley tree is included since the number of connections per peer is the same as that for the 20-cube and the 10-torus. An horizon of 6 hops was used for comparison because the peer population is only 144,801 nodes at 5 hops. Similarly for 8-Cayley, a 9-hop horizon would contain 7.7 million peers. These large increments are a direct consequence of the high vertex degree per node.

The 4-Cayley (modeling early Gnutella) and 8-Cayley (modeling the Napster population) show relatively poor scalability at 1 million peers. Even doubling the number of connections per peer produces slightly better than 50% improvement in throughput. This confirms the conclusions reached by Ritter (2002) and, moreover, supports our proposal to consider hypernet topologies.

9.8 Summary

Previous studies of Gnutella scalability have tended to overlook the intrinsic bandwidth limits of the underlying tree topology. The most thorough and accurate of these studies is that presented by Ritter (2002). Unfortunately, his analysis could be accused of straining at a gnat. As a viable candidate for massively scalable bandwidth, our analysis demonstrates that trees are essentially dead wood.

Conversely, by going to higher dimensional virtual networks (and the hypercube in particular) near linear scalability can be achieved for populations on the order of several million peers each with only 20 open connections. According to Sect. 9.6, this level of scalability would already match the number of nodes present in the entire Web.

The dominant constraint for hardware implementations of high-dimensional networks is the cost of the physical wires on the interconnect backplane. Since the hypernets discussed here would be implemented in software, no such constraints would prevent reaching the desired level of scalability. In this sense, hypernets appear to offer good (g)news for Gnutella-like P2P networks.