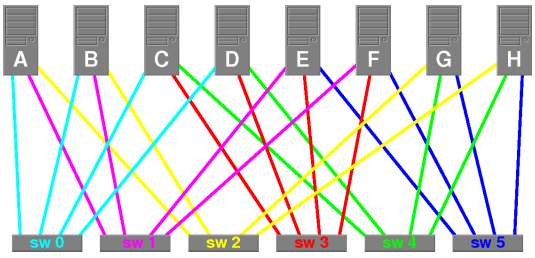
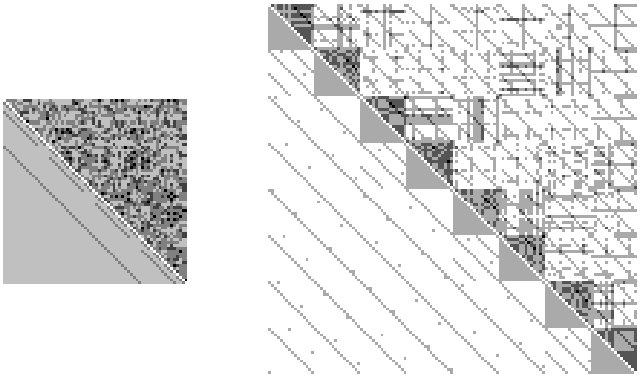
Net Work

INTERCONNECTION NETWORKS, sometimes called SYSTEM AREA NETWORKS (SANs), play a critical role in all types of parallel computers – be they clusters spanning many racks or multiple cores on the same chip. Although commodity hardware and straightforward topologies are sometimes effective, communications within parallel programs tend to have specific properties that allow a well-engineered network to dramatically outperform the obvious alternatives.

Point-To-Point Communication. By far the most commonly discussed type of communication is that in which each node is sending a message to one other node. For parallel systems, the bandwidth of an individual wire is generally less important than the bandwidth available if all processors are communicating simultaneously – the bisection bandwidth. Latency also becomes critical; it might be possible to hide some communication delays by performing unrelated computations while waiting, but latency sets the fundamental limit on the smallest grain size for which speedup can be obtained.

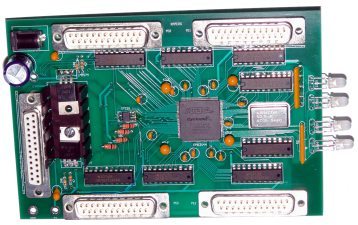


FLAT NEIGHBORHOOD NETWORKS (FNNs), which we invented in 2000, provide single-switch latency and better bisection bandwidth than a comparable FAT TREE by using multiple network interfaces per node. As shown above for 4-port switches connecting 8 nodes, an FNN connects nodes to switches such that each node pair has at least one switch in common. The best wiring pattern usually is *asymmetric*; the design is *evolved* from random wiring patterns using a genetic algorithm (online at `Aggregate.Org`).



FNN designs are summarized by square maps in which the darkness of each point indicates how many single-switch paths exist between the corresponding pair of nodes; the lower left triangle is the minimum design requirement, the upper right is what the FNN design actually provides. The smaller map is for KLAT2, which connected 66 nodes using 31-port switches. The larger map is for KASY0, connecting 128 nodes using 24-port switches. It is a SPARSE FNN (SFNN), ensuring single-switch latency *only for node pairs that are expected to communicate*. Surprisingly, very few parallel programs really have every node talk to every other; usually, each node talks to at most $O(log(N))$ other nodes. This is even true using personalized all-to-all as MPI typically implements it. Thus, SFNNs can provide single-switch latency for all critical communications in a typical application suite using cheap, narrow, switches for systems having 16,000 or more nodes! Both FNNs and SFNNs are now fully compatible with ETHERNET, IP, and most other commonly available network technologies and protocols.

Aggregate Functions. In parallel computing systems, it is very common that the global state of a computation must be sampled – which is not an efficient operation when synthesized as "collective communications" using point-to-point network hardware. In 1994, we invented AGGREGATE FUNCTION NETWORKS (AFNs) as an extension of the fast barrier synchronization hardware we had developed earlier. An AFN doesn't route messages; rather, an AFN is really a simple parallel computer dedicated to computing functions of global state. A typical aggregate function communication is implemented by each processor placing its data and an opcode in its dedicated interface to the AFN and then reading the AFN-computed result back. Thus, a barrier synchronization is the "aggregate NOP" and operations like reliable multicast, put/get, reductions, scans, voting, and parallel signaling (eurekas) all can be implemented in *essentially constant time independent of the number of nodes*.



Using simple custom hardware (the 2006 KAPERS AFN is shown above), most basic aggregate operations are accomplished with just $3\mu s$ total latency. We have placed various AFN hardware designs and support software in the public domain. Cluster AFNs can be implemented for less than $25/node.

This document should be cited as:

```
@techreport{sc06net,
author={Henry Dietz and William Dieter},
title={Net Work},
institution={University of Kentucky},
address={http://aggregate.org/WHITE/sc06net.pdf},
month={Nov}, year={2006}}
```

Aggregate.Org UNBRIDLED COMPUTING

SC06