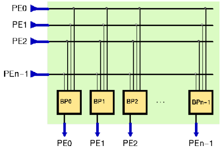
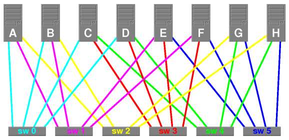
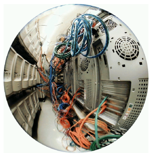
Performance-Engineered Computer Networks

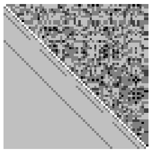
INTERCONNECTION NETWORKS, sometimes called SYSTEM AREA NETWORKS (SANs), play a critical role in all types of parallel computers – be they clusters spanning many racks or many cores on the same chip. A well-engineered network can dramatically outperform the obvious alternatives. At **`Aggregate.Org`**, we have created *public domain* technologies and tools for performance-engineered networks.



AGGREGATE FUNCTION NETWORKS (AFNs). An AFN does not route data between machines, but is a parallel function unit that collects global state information and computes functions sampling that state. In 1994, we built the world's first Linux PC cluster supercomputer to test our PAPERS AFN hardware (PURDUE'S ADAPTER FOR PARALLEL EXECUTION AND RAPID SYNCHRONIZATION), and using secure OS-bypass I/O, it delivered 3µs *barrier synchronization*. Other AFN operations include *broadcast* & *multicast*, *putget*, *reductions*, *scans*, searches (e.g., *first* or *count*), *voting* & *scheduling*, and parallel signaling (*eurekas*).



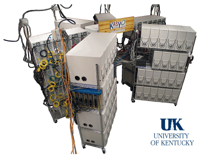


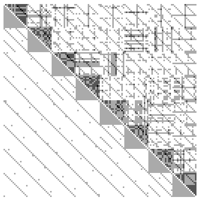


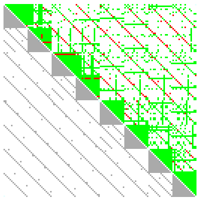
FLAT NEIGHBORHOOD NETWORKS (FNNs). FNNs provide single-switch latency, and better bisection bandwidth than a fat tree with comparable hardware complexity, by using multiple network interfaces per node. As shown above for 4-port switches connecting 8 nodes, an FNN connects nodes to switches such that 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 (GA).

FNN design problems and solution quality 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 photo and map above are for the world's first FNN, KLAT2 (KENTUCKY LINUX ATHLON TESTBED 2), built in 2000.

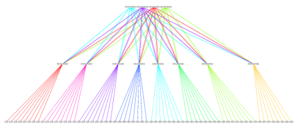
SPARSE and FRACTIONAL FLAT NEIGHBORHOOD NETWORKS (SFNNs and FFNNs). Surprisingly, very few parallel programs depend on every node talking to every other; usually, each node talks to at most $O(log(N))$ other nodes. This is true even using personalized all-to-all as MPI typically implements it.







SFNNs ensure single-switch latency *only for all node pairs that are expected to communicate*, thus requiring shockingly few cheap, narrow, switches even for systems having many thousands of nodes. The first SFNN was KASY0 (KENTUCKY ASYMMETRIC ZERO) in 2003 (above). FFNNs flip priorities, finding the best coverage possible with a fixed network cost. For example, using far less hardware than KASY0, the above map shows coverage of an FFNN in **green** – only the **red** spots deliver poorer latency.



```
[s00 s01 s05 s08 n06 n08 n09 n10 n11 n12
 n13 n14 n15 n32 n33 n34 n35 n36 n37 n38
 n39 n50 n61 n63 n n]
[s02 s04 s07 s10 n00 n01 n02 n03 n04 n05
 n07 n24 n25 n26 n27 n28 n29 n30 n31 n48
 n49 n51 n52 n53 n54 n55]
[s03 s06 s09 s11 n16 n17 n18 n19 n20 n21
 n22 n23 n40 n41 n42 n43 n44 n45 n46 n47
 n56 n57 n58 n59 n60 n62]
```

KENTUCKY'S NETWORK IMPLEMENTATION TOPOLOGY TOOL (KNITT). Knitting converts 1D material, such as yarn *or network cable*, into a 2D or 3D structure. KNITT uses genetic algorithm (GA) evolutionary search technology to create the physical placement structure that will most efficiently implement the wiring of a given logical interconnection pattern. Above is partitioning of a fat tree with 64 nodes and 16-port switches into 3 racks; only 26 of the 96 cables cross racks instead of the expected 40.

NETWIRES and NODESCAPE. As networks become more complex, visualizing networked systems becomes more important. NETWIRES draws static diagrams of networks, like the fat tree above. NODESCAPE colors a photographic image of a parallel computer according to properties sampled from the running system, such as node temperature (below) or load average – which is very useful for KNITT-scrambled physical node placements.

nak

This document should be cited as:

```
@techreport{sc15pecn,
author={Henry Dietz},
title={Performance-Engineered Computer Networks},
institution={University of Kentucky},
address={http://aggregate.org/WHITE/sc15pecn.pdf},
month={Nov}, year={2015}}
```

SC15 Austin, TX hpc transforms.

Aggregate.Org UNBRIDLED COMPUTING