Multi-Processors by the Numbers: Mathematical Foundations of Spaceflight Grid Computing [1]

Laurence E. LaForge [2]
Larry.LaForge@The-Right-Stuff.com

James W. G. Turner [2]
James.Turner@The-Right-Stuff.com

The Right Stuff of Tahoe, Incorporated
The Right Place, 3341 Adler Court
Reno, NV 89503-1263 USA

+1.775.322.5186

Abstract— The mathematics of connectivity elucidates predictive insights about multi-processor architectures. Space missions, can (and should) benefit at a level where avionics designers have the most leverage. Combining classical results with fresh research unveils new methods for maximizing fault tolerance and throughput, and for minimizing latency and cost. We illustrate and explain how to exploit *objective functions* corresponding to *feasible regions of design and operation*. The payoff: platforms that optimize grid computing among spaceborn processors. The grid may span hundreds – or even thousands – of nodes. To put a face on the mathematics of connectivity, we cite key software and hardware enablers for grids of spaceborn processors. Two such technologies stand perched on the brink of operational readiness: i) tunable multi-processor topologies, and ii) the vertical cavity surface emitting laser (VCSEL). Other enablers, such as iii) multi-processor partitioning of flight software, await the fruits of efforts by investigators.

Reaching out to non-specialists with a background in one or more quantitative disciplines, this is a position paper consolidating a host of relevant results.[3] We also include an informal, example-and-diagram tutorial on multi-processor feasible regions of design and operation. Elaborating our exposition in greater detail, two companion contributions appear in these conference proceedings:

– Written for engineers, computer scientists, or applied mathematicians with a rigorous background in dependable computing, "Spaceflight Multi-Processors with Fault Tolerance and Connectivity Tuned from Sparse to Dense" crystallizes the case for software that actively embodies the mathematics of connectivity. Exemplifying fresh results for the latter, [34] introduces a new, efficient algorithm for recognizing and labeling Hamming topologies.

– Technologists and engineers may also be interested in "Vertical Cavity Surface Emitting Lasers for Spaceflight Multi-Processors" [35], our broadly-scoped report on VCSELs as enablers for tunable architectures.

Table of Contents

1. From Multi-Processors to the Mathematics of Connectivity (and Back)

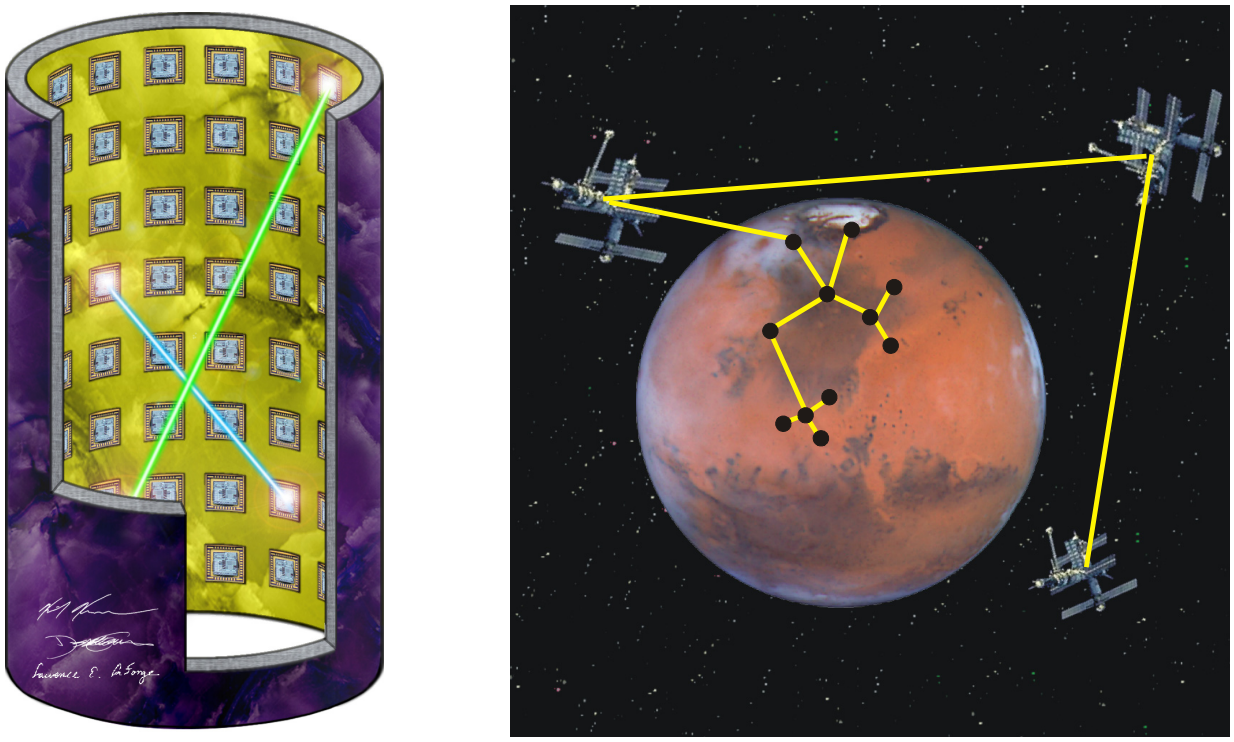
Advances in *commodity* circuit and communication technologies can – and should – enable breakthroughs for fault tolerant, autonomous, evolvable spacecraft avionics. Especially over the past decade:

a) *Technologists* from Alkalai [1] to Zorian [12] have repeatedly underscored the case for *bottom-up* changes in how we approach the design and operation of electronics systems in general, spacecraft avionics in particular.

b) *System architects* have resounded a *top-down* vision, typically invoking mission benefits. Such voices, in fact, frequently reflect technology-driven (albeit high-level) views of what is both achievable and desirable [4], [5], [17], [23].

An examination of the literature (*e.g.*, [2], [25]) reveals that both *bottom-up* and *top-down* are often weighed, or at least co-mingled, within the same exposition. It is nevertheless useful to contrast stock-in-trade viewpoints of (a) technologists with those of (b) system architects. Doing so offers the benefit of a trade-based forum for approaches to spacecraft avionics that achieve cost-effective fault tolerance, autonomy, and evolvability.

1. 0-7803-9546-8/06/$20.00 © 2006 IEEE. In *Proceedings, 2006 IEEE Aerospace Conference*. Big Sky, Montana, March 4-11, 2006.
2. Sponsored by the United States Missile Defense Agency (contract HQ0006-05-C-0028).
Conclusions reached and opinions expressed are those of the authors, and do not necessarily represent or reflect the views of MDA.
3. Session 7.11: Advanced Spacecraft and Mission Concepts. Final version 3, paper #1315 (19 pages).



a) b)

Figure 1: *Spaceflight grid computing* by the numbers, photonics enablers. a) Multi-processor with fault tolerance, throughput, and connectivity tuned from sparse to dense. (Client: NASA Institute for Advanced Concepts [27]). b) Multiple vehicles, processors coupled more loosely than in (a). (Client: U.S. Missile Defense Agency [34]).

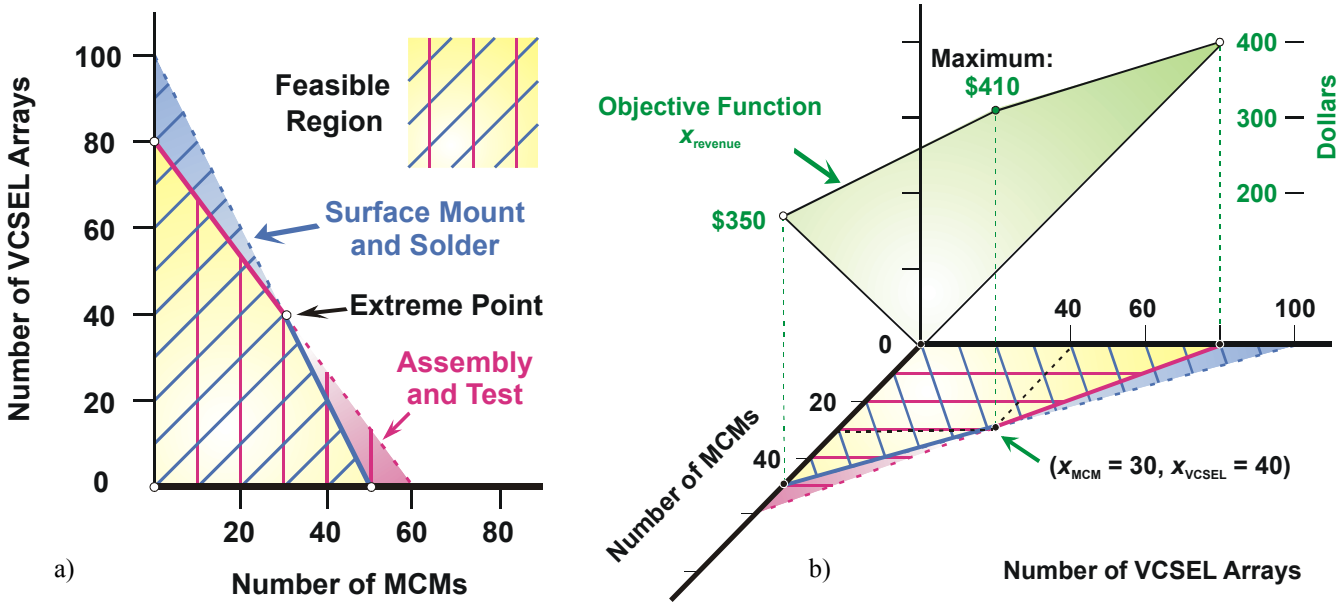


Figure 2: *Manufacturing decisions* by the numbers. Linear feasible region and objective, example (2), page 9. a) Geometers tend to interpret LP feasible regions as the intersection of *half-spaces*. In our case with example (2), four inequalities (4) through (7) in two independent variables prescribe a convex *polygon*, here shown in yellow, with red and blue cross-hatches. b) If our system can indeed be modeled as an LP problem then the objective function depends linearly on the variables. For the example in question, a geometer would likely interpret expression (3) as a *hyperplane* embedded in three-dimensional space. Shaded in green: objective hyperplane slice constituting the *functional image* of the feasible region. In the general case of d independent variables, the corresponding feasible region is a d-dimensional *polytope*, and the overall LP problem spans $d + 1$ dimensions. When $d > 3$, such a polytope and its objective function can be difficult to visualize, and to find a solution we resort to systematic theorems, algorithms, and computer software ([10] Chap. 4; [41]; [45] Chap. 6; [46]; [47] Chap. 7).

Consider, for example, a recent call by the United States Missile Defense Agency (MDA) for fault tolerant space-flight multi-processors that support *grid computing* [53]:

> *… To space systems designers, the performance of leading edge computing systems are often inaccessible, since these systems do not meet the requirements for reliable, failure-free life in a hostile (*i.e.*, radiation) environment. Currently, the most powerful rad-hard processor operates at 250 MIPS. In contrast, desktop computers are available that greatly outpace this capability. To compensate in part for this problem, adaptive computing approaches offer the possibility to blend reconfigurable computing resources into general purpose processors. The resulting processors can accelerate special types of computation (for example, finite impulse response filtering) by orders of magnitude. If such processors could be chained together in different network topologies, a rad-hard adaptive grid network would result, compensated in performance by built-in reconfigurable resources that can be tailored for mission-specific needs.*
>
> *For growth options, a distributed parallel processing approach is needed, since even the most powerful single processor has a limited performance level. New computational schemes, referred to as 'grid computing', appeal to a painlessly scalable network in which computing is increased in a manner analogous to the terrestrial power grid.*
>
> *This feature, in conjunction with other conventional fault tolerance approaches, would result in a flexible, robust computing architecture. Therefore, innovative solutions to distributed spacecraft processing are sought … that combine radiation tolerance with adaptive computing in a network-centric scheme …*

At heart, the preceding suggests dual-pronged integration of of *objectives* and *constraints*, which may in turn be profiled as

a) *Bottom-up*: rad-hard circuits, with attendant designs, manufacturing processes, and analyses.
b) *Top-down*: parallel grid computing.

In either case (a) or (b), moreover, we can combine approaches which might saliently be categorized as

i) *Broad*. *E.g.*, what commercial off-the-shelf solutions (COTS) are available to meet the goals of, or loosen constraints on, rad-hard net-centric grid computing?
ii) *Deep*. *E.g.*, what theoretical or practical problems can we solve, analytically or experimentally, that will enable rad-hard net-centric grid computing?

In a top-down (b), broad (i) fashion, this paper addresses many of the challenges posed by the MDA call for space-flight net-centric grid computing. Complementing [34]'s top-down (b) yet deep (ii) treatment, as well as the bottom-up (a) yet broad (i) exposition of [35], we provide an overview of our work, unfold how feasible regions govern multi-processor design and operation, and elaborate the grid computing application mentioned in the *Abstract*.

To reinforce the MDA solicitation quoted in the lefthand column of this page, consider a recent request for proposals issued by the United States Navy [54]; in particular, a call for W Band avionics based on

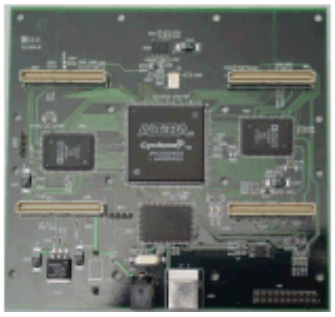
> *A complete mathematical foundation … for end-to-end operation that provides predictable, stable performance. The performance characteristics of the whole network must be guaranteed and stable within the constraints of temporal accuracy, stability and high utilization. Graceful degradation is required and must be based on a mathematical foundation … that emphasizes testability and predictability.*

Just how are we to construe "graceful degradation" in the excerpt above? Indeed, what *could* this catch phrase reasonably mean, in a sense of being i) intuitively understandable; ii) amenable to rigor; iii) experimentally verifiable; and (not least) iv) *useful*? Along similar lines, just how might we ingenuously interpret MDA's call for "processors chained together in different network topologies"? This paper addresses these questions by way of broad-stroke narrative, accented with examples and diagrams.

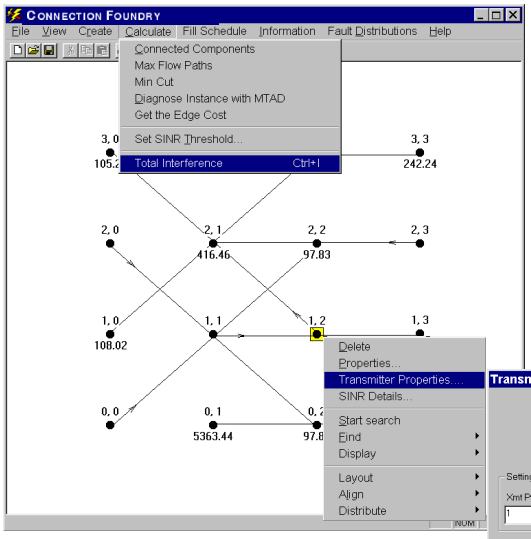
To preview, we *can* (and should) *synthesize* topologies of processors chained together in a fashion that satisfies the MDA call for grid computing. Where processor-to-processor links are relatively unchanging (think: *wires*) we can synthesize these topologies at *design time*. Where processors – along with the links over which they communicate – come and go (think: *wireless*), then we can embed into nodes algorithms for (re-)synthesizing topologies. With a bit of care, we can forge these algorithms to execute in a distributed fashion, thus effecting a self-organizing, net-centric grid.

As shown in Figure 9, we can as well synthesize topologies that meet the Navy's requirement for graceful degradation, serving up latency that is provably minimum, even in the presence of faults. To this end, we would benefit from an *automated* knowledge catalog of theorems and algorithms for tuning multi-processor topologies. Recognizing the inter-disciplinary nature of this knowledge, we will henceforth refer to it as the *mathematics of connectivity*. For example, and as Figures 7, 12, and 13 illustrate, the mathematics of connectivity tells us how to optimally trade redundancy versus fault tolerance. Continuing the nascent work of [32], our companion paper [34] explains how the mathematics of connectivity is at once in need of analytic solutions to pressing problems, yet ripe with solutions for practical application.

We will have attained a new level in the state-of-the-art when software can comprehensively and reliably apply the mathematics of connectivity to the synthesis of tunable topologies. Figure 1 of [34] illustrates how such topologies may be synthesized at *design time*, in a fashion reminiscent of the DrawCraft project at CalTech's Laboratory for Spacecraft and Mission Design [3]. Alternatively, the topologies may be synthesized in *realtime*, via algorithms distilled from design software, and embedded in the nodes of a multi-processor. As Figure 1 illustrates, processors may be coupled by way of VCSELs [35]. Or, and as Figures 3 and 4 depict, processors may comprise a MANET, with nodes somewhat more loosely coupled via radio-frequency (RF) channels.



a) Designed for the GNU Radio, the Universal Software Radio Peripheral (USRP), piggybacks a frequency-specific daughterboard. As an experimental platform for distributed, parallel media access control (MAC) algorithms, the Linux-based GNU Radio is more flexible than software defined radios (SDRs) of large companies, at less than 1/20th the cost.



Antenna beamform, directional gain:

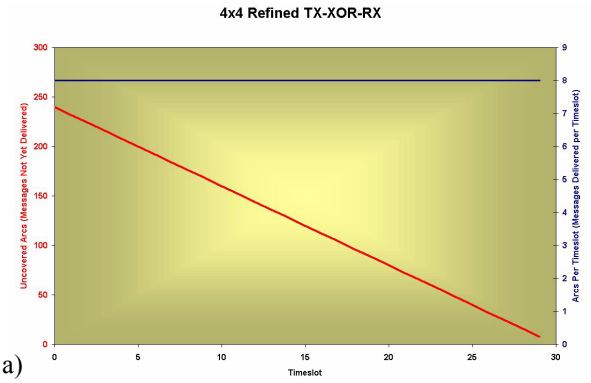
$$\frac{G(\theta)}{(G_0 = 60)} = \begin{cases} e^{(-G_0 \sin^2\theta)/4} & \text{if } |\theta| < 90° \\ 0.001 & \text{otherwise} \end{cases}$$

b) Validation of initial timeslot of a perfect, RF-feasible 16-node schedule, constant channel bit rate. Signal to interference or noise (SINR) no greater than 10. Friis transmissions governed by

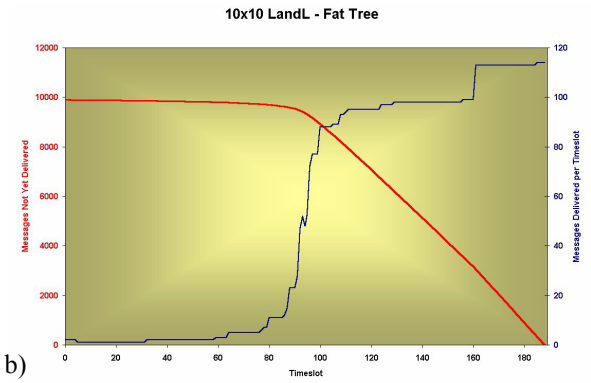
$$\frac{W_r}{W_t} = \frac{G_r G_t}{(4\pi d/\lambda)^2}$$

Receive, transmit gain: G_r, G_t
Receive, transmit power: W_r, W_t
d/λ: separation, in wavelengths

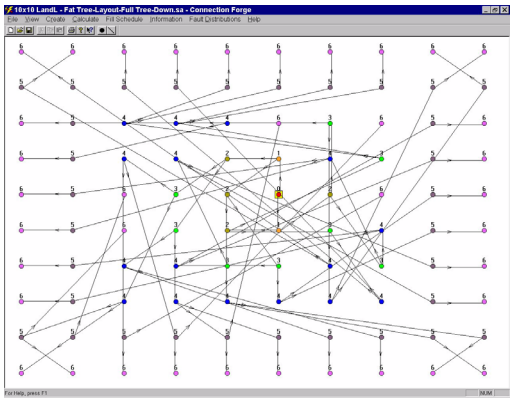
Figure 3: *Wireless grid computing* by the numbers. a) GNU Radio experimental platform. b) Connection Foundry™ software maximizes throughput by combining directed antennas with graph-theoretic factorization of cliques ([22]; [34] p. 8)



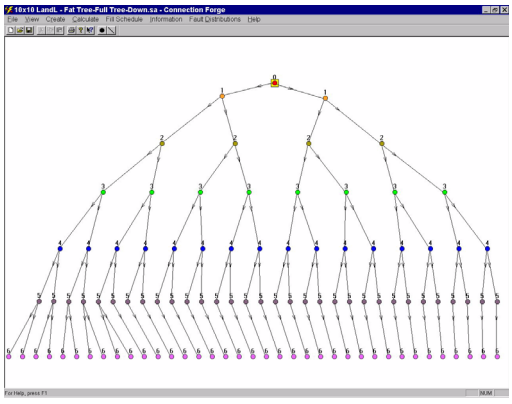
a)



b)



c)



d)

Figure 4: *Wireless throughput* by the numbers, Transmission Hypercube Challenge Problem winners (*cf* this paper, p. 7).
a) Perfect, RF-feasible, 30-timeslot schedule for 4 × 4 grid vignetted in Figure 3, mentioned on page 8 of [34], detailed in [22]. MNYD (*messages not yet delivered*, in **red**), MDTS (*messages delivered per timeslot*, in **blue**)
b) 10 × 10 winner, 189 RF-feasible timeslots. Novel combination of FAT-trees, power adjustment, and Shannon's Law.
c) Messages routed from FAT-tree root toward leaf nodes, 95 final timeslots of (b), aggregated on a single screen.
d) Same topology as (c), rearranged to explicate FAT-tree.

Topology vs. Adjacency — This paper surrenders to the popularly entrenched *mis*-use of *topology* to mean "which nodes are connected to each other" (as opposed to the more proper, graph-theoretic "adjacency"). Tables 1 and 2 of [34] explain this, as well as other nuances of nomenclature.

2. Design and Operation: Feasible Regions, Objectives, and Constraints

2.1. Grid Computing: Loosely Coupled, Nonshared Memory

To *quantify* foundations of spaceflight grid computing, it pays to *qualify* how processors are coupled. Processors are *tightly* (or *closely*) coupled if they share the same *physical memory*, *loosely* coupled if they access or exchange information via a *communication channel*, which we will take as a conduit through which information flows ([14] p. 121). Alas, these attractively simple definitions are also oversimplistic: distinctions among the abundant varieties of computer memories and communication channels are not always clearcut.

For example, tightly coupled processors can share instruction streams, data streams, or both instructions and data ([51] p. 619). The processors could be *asymmetric*, as with, say, the 11/782 and 8800 models in the VAX computer family ([18] Chap. 27), made by Digital Equipment Corporation (DEC, subsequently acquired by Compaq, which then merged with Hewlett-Packard). The presence of *very* tightly coupled multiple processors might be invisible to the programmer, as is the case with throughput-enhancing multi-*cores* [42], which may share cache memory that is both fastest, and closest to the instruction decode unit. With its TMS320C6201 model, Texas Instruments pioneered a multi-core digital signal processor (DSP), whose eight *micro-machine* processors are dedicated to maximizing the rate at which very long instruction words (VLIW) are retired [50].

Shifting from cache-coupled micro-machines toward the other end of the memory hierarchy ([51] p. 134), DEC's VAXclusters couple processors at the level of programmer-visible file sharing. Machines configured into the same VAXcluster *quorum* also communicate directly over a channel, hardware-enabled by a so-called *star-coupler*. Incidentally, DEC's legacy use of *quorum* to designate an ensemble of computers which cooperate (hence communicate) coincides with Moore and Shannon's application of the identical terminology in 1956 [40]. Continuing this tradition, we accord *quorum* preference over *swarm* or *mesh*, and on par with *grid*.

The initial billing of VAXclusters as tightly coupled [19] blurred considerably when, in 1986, DEC introduced *local area network* (LAN) VAXclusters [6], feasibility of which hinged largely on ethernet channels. Today, LAN-based file sharing is commonplace, even with, say, heterogeneous mixes of machines running Linux and Microsoft Windows operating systems. The advent of pervasive, high-speed wired and wireless channels has given rise to the sharing of files over virtual, private networks (VPNs). Contemporary organizations rely on VPNs to consolidate their daily business despite physically distributed locations.

The preceding examples spur us to wonder: just *where*, in the gamut from micro-machines to VPNs, does the coupling of processors cross from tight to loose? For our answer we turn to the theory and practice of dependable computing: *a processor is loosely coupled if its faulty behavior can be isolated from the remainder of the quorum by healthy processors disabling or severing point-to-point-channels*. Our meaning of loose coupling thus implies *nonshared memory*.

Table 1 of the mathematically-oriented companion [34] to this paper spells out rigorous prescriptions for terms such as *fault*, *fault model*, *fault tolerance*, and *quorum*. In keeping with our informal emphasis, we will adopt complementary terminology generally attributed to LaPrie [36], and subsequently elaborated by a multitude of researchers:

a) A system (think: *component*, *node*, *processor*, or even *software*) *fails* when the service it provides differs from the service is was designed to provide.
b) A system is in *error* when its state is other than that intended by the design. Errors lead to failures, but not all errors manifest as failures.
c) A *fault* is a condition which may lead to an error. Faults which have not (yet) caused an error are deemed *latent*. Espousing an alternative, Murphy and Hayes [42] instead define a fault as the difference between a system that is failing, and one which is healthy (see next item (d)).
d) A system that has not failed is *healthy*. However, (and this is a quirk of the literature) we frequently model as *healthy* a system which is either fault-free *or* whose faults are not yet manifested. The former rarely (if ever) exists. Perhaps surprisingly, such cavalier use of *healthy* and *faulty* is often both useful and devoid of confusion.

Definitions (a) through (d) are both causal and recursive. However, that the extent to which (a) is *practical* depends on

i) How well we have *operationally specified* what our system is *supposed* to do
ii) Our ability to effectively check that our system in fact does what it is supposed to do (*verification* [44])
iii) The extent to which our specifications pinpoint what we *really* want the system to do (*validation* [44])

Verifiable theories of fault tolerance (including that advocated in this paper) tend to be founded on *localization* of errors and faults. However, point (i) above suggests that *specification* faults may root *nonlocal* errors. This is especially so with *software*, which bears much of the brunt of the complexities of dynamic systems, *e.g.*, onboard flight control. Herein alternative definition (c), independently advanced by LaForge and Nikora for instrumenting *Deep Space One* flight software [27], could prove fruitful. Requirements 13 and 16 of [23], along with Section 1 of [35], elaborate issues related to software fault tolerance and multi-processors.

As Figure 1 suggests, it is in above-described context of loosely coupled multi-processors that *fault tolerance* serves as our foundation for quantifying grid computing, spaceflight grid computing in particular. Loosely coupled multi-processors set the stage for optimizing fault tolerance – plus other objectives – as Sections 2.2 and 2.3 explain.

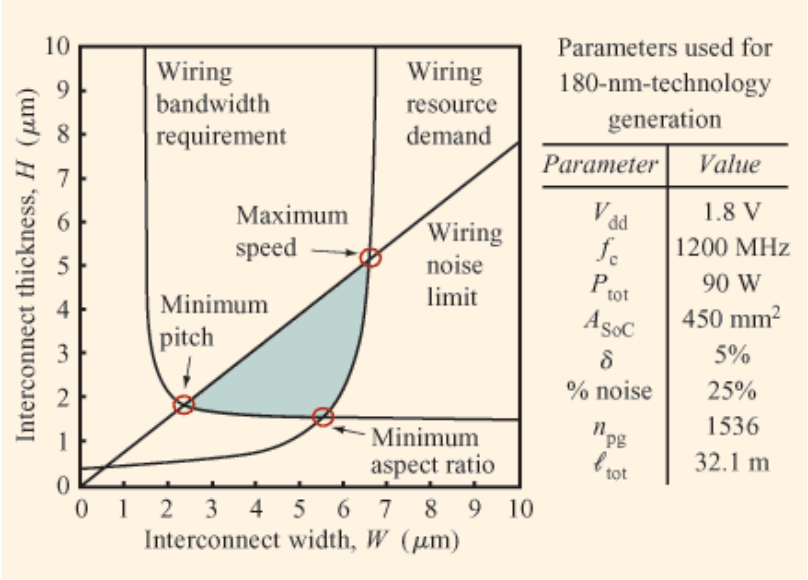


Figure 5: *Microelectronics* by the numbers. Convex (but nonlinear) feasible region of design. Shown in blue-gray: the intersection of three constraints governing wire geometries in very large scale integrated circuits (VLSI). For more than two decades, J. D. Meindl has spearheaded work such as this [38]. The present paper espouses analogous feasible regions for the design and operation of multi-processor grid computing. Reprinted with permission from [55] © 2000 IEEE.

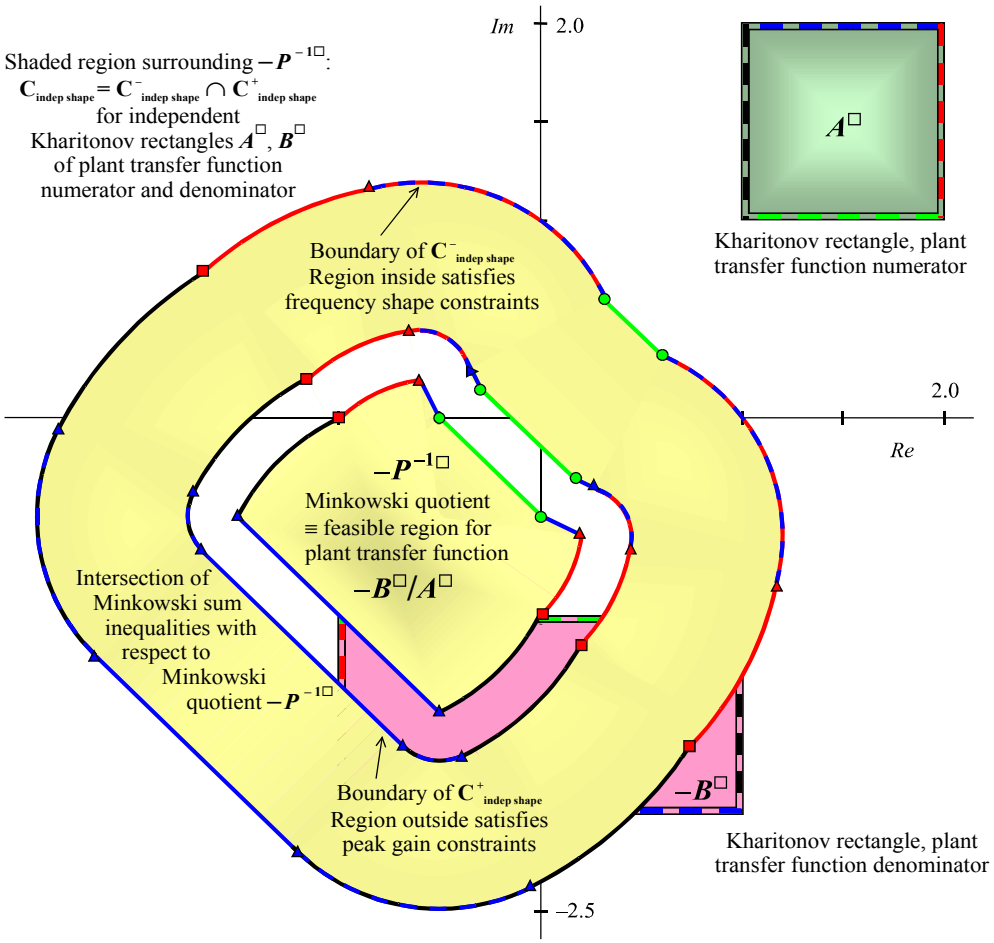


Figure 6: *Robust feedback control* by the numbers. Non-convex, nonlinear feasible region of design and operation. Outer sleeve in yellow: intersection of three constraints governing compensators, robust to uncertainties in transfer function parameters. The complete sleeve is three-dimensional, and extends above and beneath the slice shown here for a particular frequency [13].

2.2. Feasible Regions of Design and Operation Facilitate Models That Are as Simple as Possible, But No Simpler

Section 1 of [35] recounts how and why multi-processor projects to date fall elusively short of *commodity status*, on par with *uni*-processor successes, such as Intel's Pentium family. Continuing from Section 2.1 of this paper, we advance *commodity* net-centric, multi-processors, *tuned* to

maximize

i) tolerance to faults (including radiation-induced faults)

ii) throughput (say, aggregate channel capacity)

minimize

iii) latency (measured, say, by topology diameter)

iv) cost (including power, mass, and dollars)

Fault tolerance (i) is by no means our only figure of merit, but it is our most fundamental one. Fault tolerance also serves as a springboard for illustrating *feasible regions of design and operation*. To get the ball rolling, let us list some examples of what we do *not* mean by *multi-processors by the numbers*:

a) Maximizing instruction throughput by sizing memory hierarchy layers (*e.g.,* [51] Sec. 4.1), such as tape, magnetic or solid-state disks, and high-speed cache

b) Maximizing aggregate processor throughput by allocating instructions to micro-machine cores, via (say) combinations of parallelization and pipelining (*e.g.*, [50])

Topics such as (a) and (b) above are instead the proper province of analysis of systems comprising a single processor, or processors which are tightly coupled. Further, we are *not* writing about …

c) … how to implement spaceflight grid computing with manufacturing-level fidelity. Without question a serious effort to realize the possibilities predicted by the mathematics of connectivity hinges on details, such as coordinating English versus metric units between engineering teams (*cf.* [23] Req 14). That said, and recalling the DrawCraft project mentioned at the bottom of page 3, designing and operating multi-processors by the numbers not only dovetails with software that embodies theorems and algorithms from the mathematics of connectivity, but is *very* compatible with detailed computer simulation.

Point (c) accentuates the difference between multi-processor by the numbers, and prevailing design habits. The latter, we submit, overemphasize complex calculations and voluminous, repeated simulations – at the expense of analytic reasoning. Section 1.1 of [34] underscores this distinction, which we take to the opportunity to amplify here.

Objectives versus Constraints — In 2004, the Air Force Research Laboratory (AFRL) challenged five contractors, including The Right Stuff of Tahoe, to devise a comprehensive approach for software defined radios (SDRs) that conserve RF spectrum. Page 8 of [34] explicates the corresponding *Transmission Hypercube Barebones Problem*, requirements for which entail balancing mobile *ad hoc* network (MANET) *constraints* against *objectives* [22]. Figures 3 and 4 provide a glimpse of how this challenge exposes what we mean by *multi-processors by the numbers*.

For example, we may pick fault tolerance as our *objective* to maximize, with limits on any combination of (ii), (iii), and through (iv) in the lefthand column of this page serving as *constraints* on the feasible region. More generally, our objective may be any one of (i) through (iv), or perhaps even a *function* of (i) through (iv). To illustrate this approach to optimization, let us briefly review its most popular rendition.

Fly-By Tour of Linear Programming — To mathematicians, a mapping is *linear* if it is *additive* and *scalar multiplicative*. For our purposes, an optimization problem is *linear* if its constraints and objective are flat; we deliberately forego a rigorous definition of *flat*. Optimization on feasible regions with flat sides continues to be billed by its historical moniker: *linear programming*, *LP* for short [41]. *Linear programming* refers to a *mathematical* formulation; it is not to be confused with the programming of *computers*, even though practical solution to contemporary LP problems is effected by programming algorithms on computers. In 1945, Nobel Laureate G. J. Stigler [15] presented what is often heralded as the LP progenitor, *the Diet Problem* [47], [52]:

Minimize the annual cost of one person's diet, subject to constraints on minimum caloric intake and nutrients (1)

Expression (1) falls short of explicating the cost and nutritional value of candidate foods, as well as the minimum dietary requirements for nutrients. Stigler distilled these particulars into nine linear inequalities in 77 unknowns. In his biographical sketch of Stigler [15], M. Friedman – himself a Nobel Laureate in Economics – writes:

> *"The Cost of Subsistence" [Stigler's 1945 article] … starts,* "Elaborate investigations have been made of the adequacy of diets at various income levels, and a considerable number of 'low-cost,' 'moderate,' and 'expensive' diets have been recommended to consumers. Yet, so far as I know, no one has determined the minimum cost of obtaining the amounts of calories, proteins, minerals, and vitamins which these studies accept as adequate or optimum." *George then set himself to determine the minimum cost diet, in the process producing one of the earliest formulations of a linear programming problem in economics, for which he found an approximate solution, explaining that "*there does not appear to be any direct method of finding the minimum of a linear function subject to linear constraints." *Two years later [in 1947] George Dantzig provided such a direct method,* the simplex method*, now widely used in many economic and industrial applications.*

In 1947, J. Laderman's ten-person team at the National Bureau of Standards expended 120 man-hours with desk calculators, applying G. B. Dantzig's simplex algorithm to the Diet Problem. The result: Stigler's *ad hoc* approximate minimum of $39.93 (in 1945 dollars) was only 24 cents more than the true minimum of $39.69 [37].

To get a solid feel for feasible regions and objective functions, let us set up and solve a simple LP problem. Rather than engage the nine inequalities and 77 unknowns of Stigler's seminal Diet Problem, our example spans five inequalities in two unknowns. As Figure 2 illustrates, our simplified treatment makes it relatively easy to visualize the geometry of the feasible region.

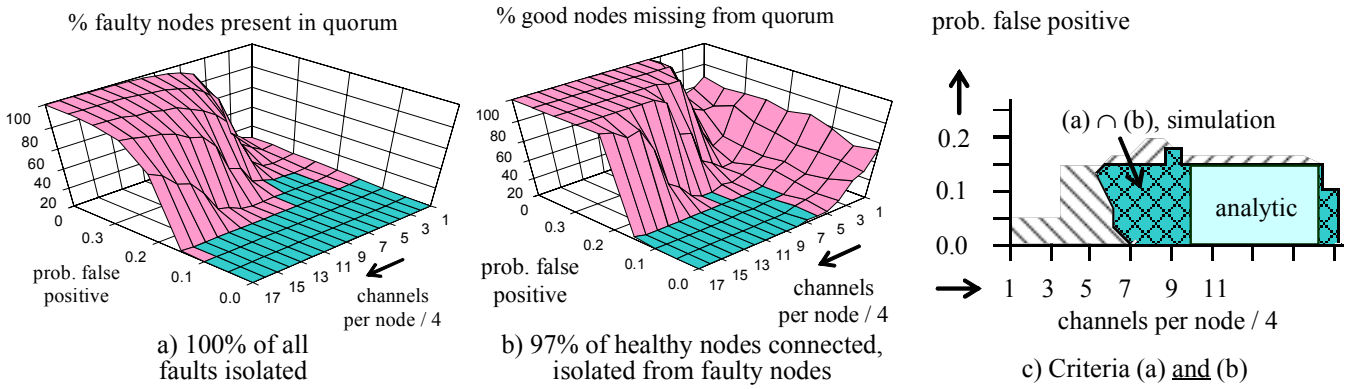


Figure 7: *Probabilistic* feasible (and forbidden) regions of multi-processor *diagnosability* and *connectedness*. a) and b): simulations of from 21 to 357 nodes refine and expand the analytic region conservatively predicted by [30], depicted in (c). Here, faulty nodes are distributed with coin flipping probability 75%. The expected fraction of healthy nodes is 25%, with 22% of all nodes healthy and connected into the same quorum. These results also hold *almost surely* ([34] pp. 7, 9, 19). The number of channels per node measures cost. The false positive probability measures the quality of localized tests.

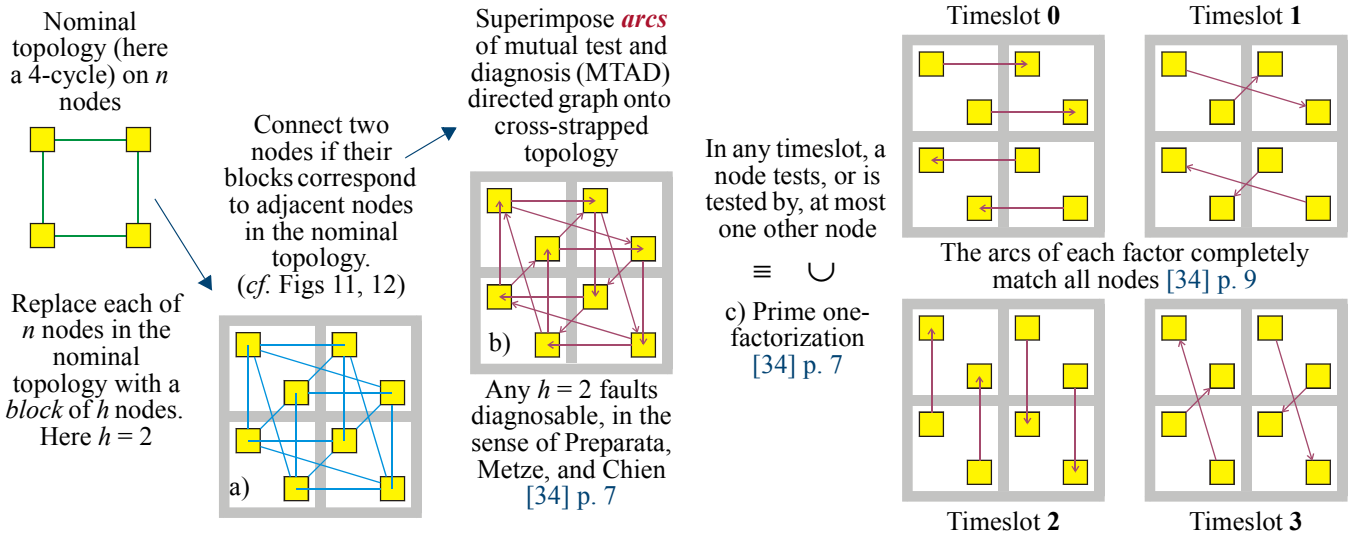


Figure 8: *Cross strapping* by the numbers. How to synthesize a) topology, b) MTAD digraph, and c) optimum test schedule. Also known as *(n, h) local sparing*, the number of nominal nodes equals *n*, with *h* the ratioed node or channel redundancy.

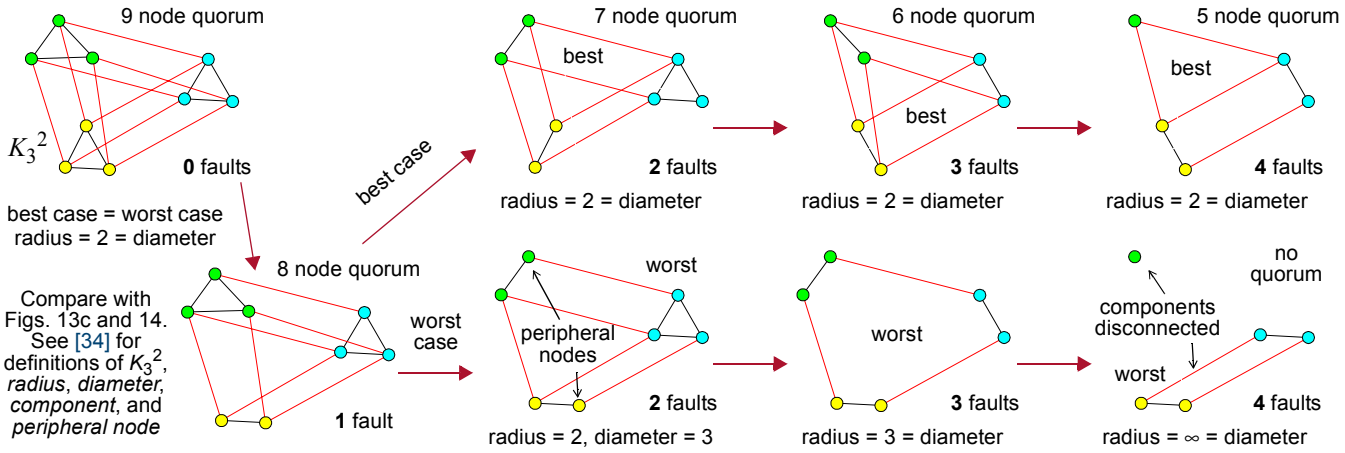


Figure 9: *Graceful degradation* of a complete Hamming topology, the nine-node *two-dimensional ternary K-cube* K_3^2. The mathematics of connectivity brackets the latency with respect to best and worst-case fault patterns, such that all healthy nodes are connected. By actively embodying the underlying theorems and algorithms, software, such as The Right Stuff of Tahoe's Connection Foundry™, optimizes multi-processor topologies at either design time or in operation.

LP Setup — Suppose that we manage an avionics packaging shop which makes and sells vertical cavity surface emitting laser (VCSEL) arrays and multi-chip modules (MCMs). Business is good, and we can readily sell all the parts we can package. Customers in the aerospace supply chain willingly purchase our VCSEL arrays for $7 each, our MCMs at $5 apiece. Our manufacturing line can devote up to 100 hours per week surface mounting and soldering parts, as much as 240 hours per week assembling and testing them. Because our accountant carefully monitors our fabrication efficiency, we know that to package each VCSEL array takes two hours of surface mount and soldering, four hours for assembly and test. On top of one hour of surface mount and soldering, to complete an MCM takes three hours of assembly and test. Subject to these manufacturing constraints, we have the latitude of choosing the quantity of VCSEL arrays and MCM's that we produce each week. Restated as an LP problem:

Subject to manufacturing constraints, what production mix of VCSEL arrays and MCM's maximizes revenue? (2)

Let x_{VCSEL} be the number of VCSEL arrays we package in any given week, and denote by x_{MCM} the number of MCM's we turn out over the same period. Our *revenue objective function* x_{revenue} is therefore just x_{VCSEL}, weighted by the selling price for a VCSEL array, plus x_{MCM}, weighted by the selling price of an MCM. Formulating this relation as a (linear) *objective function*:

$$x_{\text{revenue}} = \$7 \cdot x_{\text{VCSEL}} + \$5 \cdot x_{\text{MCM}} \tag{3}$$

Re-casting our manufacturing *constraints*:

$$(4 \text{ hrs / VCSEL array}) \cdot x_{\text{VCSEL}} + (3 \text{ hrs / MCM}) \cdot x_{\text{MCMs}} \leq 240 \text{ hrs assembly \& test} \tag{4}$$

$$(2 \text{ hrs / VCSEL array}) \cdot x_{\text{VCSEL}} + (1 \text{ hr / MCM}) \cdot x_{\text{MCMs}} \leq 100 \text{ hrs surface mount \& soldering} \tag{5}$$

$$x_{\text{VCSEL}} \geq 0 \tag{6}$$

$$x_{\text{MCMs}} \geq 0 \tag{7}$$

Figure 2a illustrates the bounded, two-dimensional feasible region prescribed by linear inequalities (4) and (5), as well as *nonnegativity constraints* (6) and (7). Mindful of our objective function (3), we see that our LP problem in fact spans *three* dimensions. Figure 2b depicts how (3) geometrically defines a *hyperplane* in a 3-D space, with x_{VCSEL} along one axis, x_{VCSEL} along another, and x_{revenue} along the third. In the precise language of mathematics, the part of x_{revenue} over which we seek a maximum is the *image* of the feasible region with respect to the linear mapping (3).

LP Solution — Techniques abound for solving LP problems whose complexity substantially outstrips that expressed by (3) through (7). Figure 2 lists references which serve as but a starting probe for the voluminous literature on the subject. For example, and invoking Dantzig's simplex method mentioned on page 7 of this paper, we need only check the objective function at the vertices, or *extreme points*, of the feasible region ([45] Chap. 6). There are but four extreme points on the boundary of the feasible region prescribed by (4) through (7), with the maximum value of $x_{\text{revenue}} = \$410$ attained when $x_{\text{MCMs}} = 30$ and $x_{\text{VCSEL}} = 40$. Figure 2b illustrates this result, the unique mix of VCSEL arrays and MCM's that maximizes our weekly revenue.

Benefits of Linear Programming — LP courses are standard fare at business schools, and with good reason. While its demonstrated power has yet to reach a maximum, LP is arguably the most successful weapon in the *by-the-numbers* armamentarium. The advantages of LP emerge with striking clarity when we examine its use by commercial airlines.

Introduced in the mid-1980's, the LP-based Station Manpower Planning System was estimated to have saved United Airlines $6M annually ([47] p. 314). American Airlines' TRIP (Trip Re-evaluation and Improvement Program), a personnel assignment LP optimizer evolved since the 1970's, is claimed to have saved the company $20M per year. As an additional indicator of success, by the 1990's, American had sold TRIP to 10 other airlines and one railroad ([47] p. 297). In the 1990's, Delta Airlines began solving an LP model *on a daily basis*: 40,000 constraints on 60,000 variables reflecting aircraft availability and capacity, arrival and departure schedules, maintenance requirements, and passenger reservations. The objective: minimize operating costs and lost passenger revenue. Industrial engineers peg this model, called *Coldstart*, as saving Delta $100M per year, when compared with business before *Coldstart* ([47] p. 345).

The Curse of Dimensionality — Returning to the 2004 Transmission Hypercube workshop mentioned on page 7, the LP success of United, American, and Delta Airlines may have spurred Engenium Technologies President M. Pascale to propose naïve, if well-intentioned, MANET media access control (MAC), with two properties of note:

a) Ultra-fine fidelity that attempts, in realtime, to adjust tradeoffs among, for example, transmit and receive power, distance, RF path loss, bit error rate, routing choices, and throughput … all *globally* captured and optimized on a single computing platform. For a MANET of even modest proportions, say, 36 nodes, this means setting up and solving, on a single MAC master node, an LP problem comprising *more than 300,000 variables*.

b) *Tree* topology, with LP-solver embedded in the MAC master, also the root of the tree. The MAC master must *re*-run its LP-solver as nodes enter and leave the quorum, or as the distance between nodes changes, or as other environmental conditions (*e.g.,* ambient noise) change. To specialists in both wireless networks and the domain of fault tolerance, such occurrences are known as *reconfiguration events*. Page 12 of [34] provides a starting point for more information about tree topologies.

Properties (a) and (b) epitomize how *not* to go about designing or operating multi-processors by the numbers. Though perhaps *daunted*, we should not be *surprised* at the moribund confusion arising from a model which suppresses the dominance of first-order effects. A 300,000-plus variable linear program on just 36 nodes underscores how according equal import to a maximum number of conceivable variables is not only unscalable, but intractable when the count of nodes is – by grid computing standards – modest. Engenium Technologies never did present a solution (even if suboptimal) to the nominal 10×10 Transmission Hypercube Barebones challenge ([34] p. 8). By contrast, and as Figure 4 details, The Right Stuff of Tahoe's FAT-tree approach outstripped all other Transmission Hypercube companies by 231%.

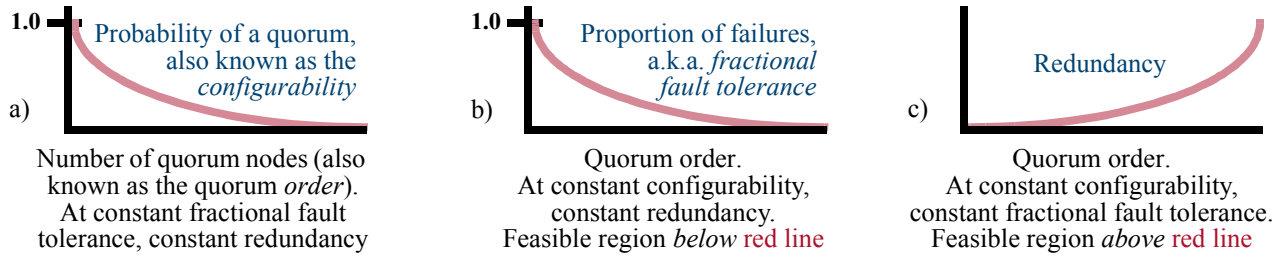


Figure 10: Concepts and trends, feasible regions of multi-processor design and operation, sliced along each of three dimensions.

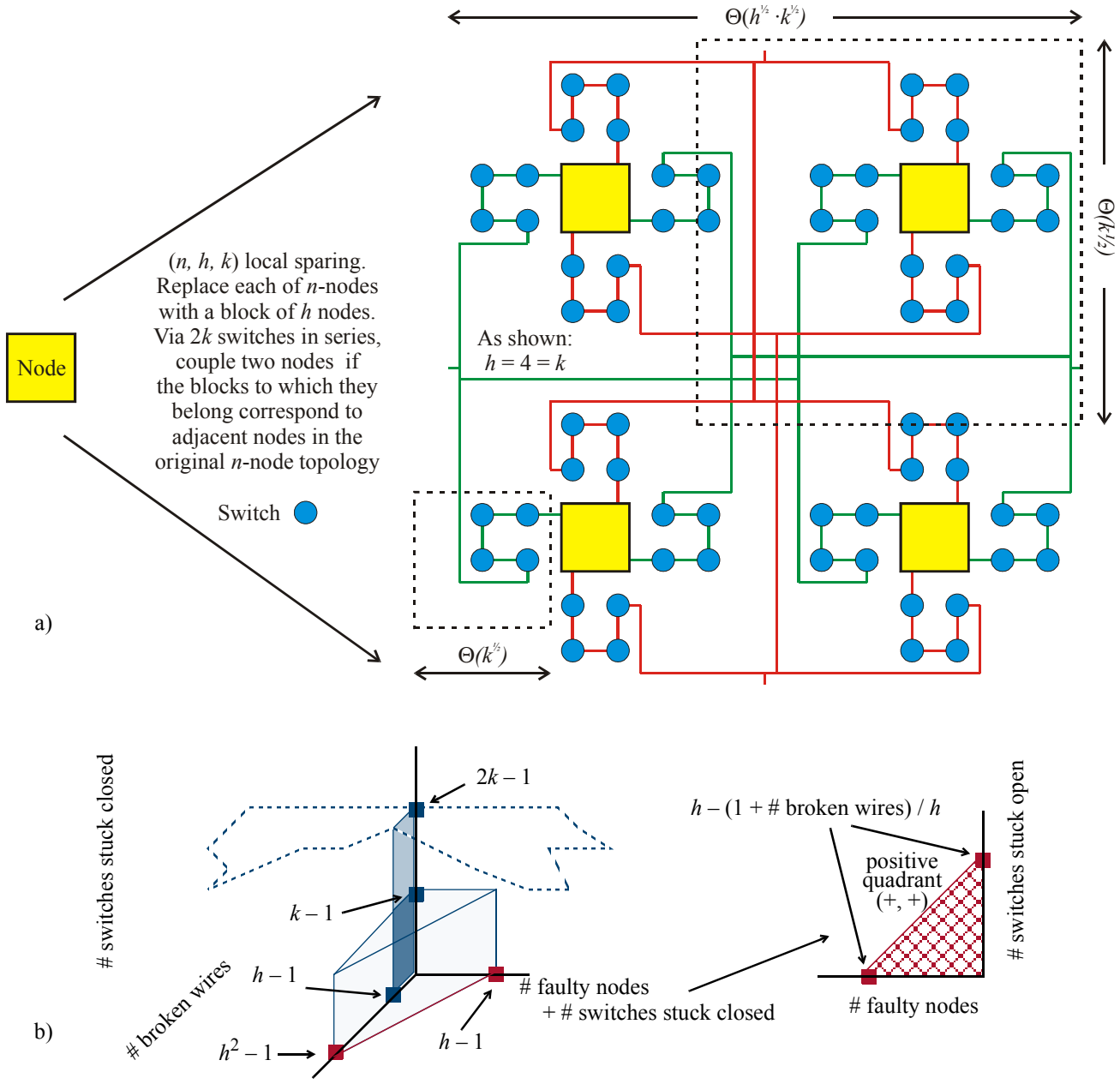


Figure 11: *Cross strapping*, also known as *local sparing*, revisited. a) Switching and routing reflecting chip-level layout, in somewhat finer detail than that depicted by Figure 8. b) Four-variate feasible region of worst-case tolerance (*cf.* Figure 7b) to faulty nodes, broken wires, and switches stuck open and closed. In contrast with the probabilistic cases depicted in Figure 12, the worst case constraints (*i.e.*, configurability = 1, *cf.* Figure 7a) are *linear*, no matter what subset of four fault types we consider. Original results detailed in [29]. See Table 2 of [34] for precise meaning of order-of-magnitude Θ-notation.

Evocative of the 120-man-hours to solve Stigler's Diet Problem, property (a) on page 9 draws attention to the *computational cost* of LP problems with a large number of variables. Dantzig's simplex algorithm, along with other techniques which have been advocated, may execute in time *exponential* in the number of variables. As Figure 2 depicts, the feasible region occupies a volume whose geometric count of dimensions equals the number of variables; such exponential running time is therefore known as *the curse of dimensionality*. This poses difficulties even when we invoke the power of LP software and high-speed computers.

Computer scientists consider an algorithm executing on a single processor to be efficient if it completes in time that is polynomial in the size of its input [9]. The simplex algorithm does *not* meet this theoretical standard, even though practical problems against which it is pitted are frequently sufficiently well-behaved to engender acceptable runtime. It was for some time not known whether there *could* be a polynomial time algorithm to solve LP; this was settled in the affirmative when, in 1984 ([47] p. 289) N. Karmarker introduced his elliptical algorithm. Alas, the elliptical algorithm has yet to attain widespread adoption. The simplex algorithm remains the pervasive LP method of choice for optimizing software, and is even built into Microsoft Excel [39].

Distributed versus Centralized Quorum Management — Properties (a) and (b) of the Engenium proposal suffer the tyranny of centralization. Not only is a tree zero-fault tolerant (killing any non-leaf node breaks the quorum apart), but embedding a mission-critical LP-solver into the root of the tree invites systemwide failure, perhaps by hostile targeting (*cf.* [24] p. 31). Nevertheless, customers to whom we deliver tunable multi-processor topologies should, we submit, enjoy the option of trading zero fault tolerance against other priorities (*e.g.*, a minimum Euclidean spanning tree, or MEST, connects a quorum using least power; *cf.* Figure 16). Such was the case for our winning FAT-tree solution to the Transmission Hypercube Barebones challenge, wherein AFRL valued workload throughput at the expense of fault tolerance or economy of power (Figure 4b, c, and d; with respect to Figures 3 and 4a, note that our constant power solution is not based on a tree; also see [34] Sec. 2.1). Perhaps most telling, neither

a) the complexity (300,000+ variables on 36 nodes)
nor b) the centralized MAC / LP-solver

of the Engenium proposal is warranted, as we proceed to explain.

In lieu of (a), we re-iterate our list of four, first enumerated at the top of page 7: i) fault tolerance; ii) throughput; iii) latency; and iv) cost. To ice the case against (a), and in addition to the curse of dimensionality: multi-processors by the numbers is highly ***non****linear*, even when we restrict feasible regions to our list of four. Figure 14 illustrates how this is so, with plots of instances to which we have applied results detailed in [34]. Therefore, stock-in-trade linear programming approaches (including algorithms for explicating feasible regions as the intersection of half-spaces; [10] Sec. 4.2) are in general ***not*** appropriate for optimizing multi-processor grid computing, spaceflight grid computing in particular.

With respect to property (b), and as remarked on page 3, proper application of the mathematics of connectivity entails converting theorems to algorithms, embodying algorithms as part of design software (*cf.* Figures 3 and 4; [34] Fig. 1), and re-casting the algorithms into a distributed, parallel form [8] that can be embedded into operational nodes (*cf.* GNU Radio, Figure 3a), and which maximizes global benefit based on local actions. The latter is at odds with properties (a) and (b), which, are distinctly *not* amenable to distributed, parallel coordination of (re)configuration events.

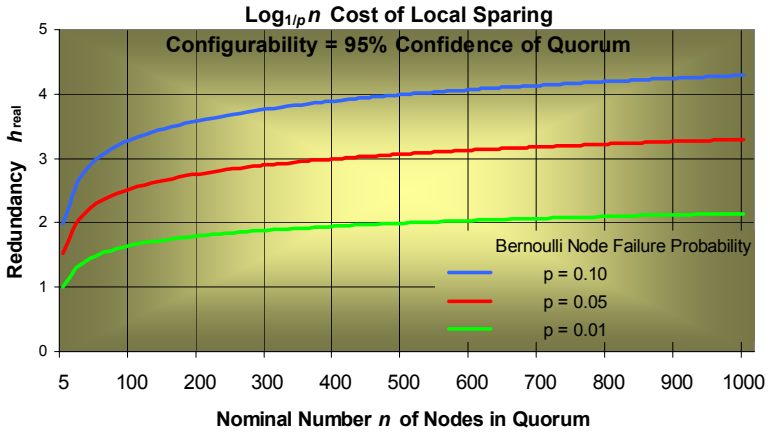
For example, in bringing a quorum to life, how are nodes to decide who is master, and who is not? Generalizing the case of a single node joining a quorum of many nodes, how to merge quorums which come into proximity with each other? Who takes over as MAC master if the current MAC master fails? How is unwanted MAC takeover stymied when the current MAC master is healthy? While the Engenium proposal falls short of addressing these questions, a starting point for answers can be found in Digital Equipment Corporation's legacy of *Vaxclusters* ([19]; also see page 5 of this paper). Bolstering and complementing this foundation: synthesis of topologies – either at design time or during multiprocessor operation – using classical and contemporary theorems and algorithms from the theory of graphs, especially *Hamming graphs* ([20], [34] Sec. 2). Figure 9 illustrates how Hamming topologies promote graceful degradation, as described by the Navy solicitation quoted on page 3.

While we can rely on DEC's Vaxclusters as a springboard for contemporary MANET media access control (MAC), we should *not* mimic DEC's brandishing of a 262-dimensional space of boot-time adjustable parameters for the VMS operating system ([18] Apx C.1.16). The inadvertent but harmful upshot of according ill-specified import to so many parameters was to stunt attempts at rigorously and predictively modelling the salient behavior of the operating system. With the benefit of hindsight, we now know better. If 262 boot-time adjustable parameters are too many for VMS, then 300,000-plus MAC variables are certainly too many for a realtime MANET. Such bloated dimensionality defeats criteria, first advanced on page 3 of this paper, for a multi-processor model that is intuitively understandable, amenable to rigor, experimentally verifiable, and *useful*. Our model should keep things, in the reputed words of A. Einstein, "as simple as possible, but no simpler" [7]. To this end, we proffer the list of four enumerated at the top of page 7.

In summary, kith and kin to properties (a) and (b) run contrary to simple-as-possible, useful predictions of the *salient* behavior of multi-processors. The attendant realtime implementations are both unscalable and intractable, even when the count of nodes is modest. Systems with properties (a) and (b) suffer the tyranny of centralization, with consequential penalties assessed against fault tolerance and throughput. Moreover, an LP formulation neglects the glaring fact that multi-processor optimization is highly ***non****linear*. All this brings us to a central posit of this paper: *as important as it is to accurately and efficiently optimize objective functions with respect to a given feasible region, it is even more important to determine feasible regions of multi-processor design and operation in the first place*.

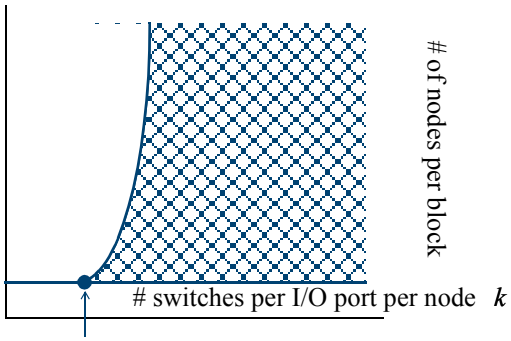
a) For each Bernoulli node failure probability p, the corresponding feasible region of redundancy consists of integer values on or above the respective colored line. The configurability Y is synonymous with the *quorum confidence*.

$$h = \lceil h_{\text{real}} \rceil = \left\lceil \frac{\ln (n) + \ln (1/\ln [1/Y])}{\ln (1/p)} \right\rceil$$



b) Feasible (h, k) redundancy at constant configurability Y, constant bivariate fractional fault tolerance $(p_{\text{node}}, p_{\text{stuck-closed}})$

$$h_{\min} \approx \frac{\ln (n) + \ln (1/\ln [1/Y])}{\ln (1/p_{\text{node}})}$$



$$k_{\min} \approx \frac{\ln (nh_{\min}) + \ln (1/\ln [1/Y])}{\ln (1/p_{\text{stuck-closed}})}$$

c) Feasible (h, k) redundancy at constant configurability Y, constant trivariate fractional fault tolerance $(p_{\text{node}}, p_{\text{stuck-open}}, p_{\text{stuck-closed}})$

Figure 12: Probabilistic feasible regions of redundancy (*cf.* Figure 7c) for *cross strapping*, also known as *local sparing*. a) Univariate ratioed redundancy h, in the presence of faulty nodes, switches incorporated into nodes or external to node, but (*cf.* Figure 11) switch failures negligible. b) Bivariate feasible (h, k) region: faulty nodes with switches (*cf.* Figure 11) stuck closed. The latter pushes the (minimum ratioed) area redundancy from $\Theta(\log n)$, in the case (a) of faulty nodes only, to $\Theta(\log n)^2$. c) Trivariate model. Same as (b), only with switches stuck open *and* stuck closed. The conflicting nature of the two types of switch failures pushes the area redundancy from $\Theta(\log n)^2$, in case (c), to $\Theta(n \log n)$. As detailed in [26], these results quantify how switches that are separate from processors, but which are stuck closed, can constitute the most costly type of fault. Bottom line: we can more readily and predictably optimize grid computing when (*cf.* Section 2.1) nodes are loosely coupled.

From Diets to Proteins — With judicious choice of modeling parameters, feasible regions spanning even a handful of dimensions serve to illuminate what man or Nature can build. As an instance of the latter, G. N. Ramachandran spearheaded the formulation and use of plots that distinguish forbidden versus feasible structures of proteins, such as the helices of hair keratin and blood hemoglobin [11].

From Microelectronics, to Robust Control, to Multi-Processor Mutual Test and Diagnosis (MTAD) — Somewhat closer to this paper's topic of multi-processors, Figure 5 depicts but one example from the body of work by integrated circuit pioneer J. D. Meindl. Meindl's feasible regions explicate the range of interactions among increasingly *growing* numbers of integrated *microelectronic* devices, simultaneously constrained by the physics of increasingly *shrinking* geometries. Meindl's contributions have inspired the first author of this paper to discover and report feasible regions (often nonlinear) that elucidate *macro-architectural* limits of loosely-coupled systems (*cf.* Section 2.1). For example, Figure 6 illustrates the output of an algorithm, developed in collaboration with M. S. Fadali [13], that computes feasible regions of robust compensation; this enables the fabrication of actuators that properly maintain control (think: *spacecraft attitude*) despite uncertainties about the values of system parameters. As yet another example, the feasible regions and mechanisms of Figures 7 and 8 illustrate mutual test and diagnosis of faults among the nodes of a multi-processor [33]. Such *macro-architectural* feasible regions fairly sample what we mean by *multi-processors by the numbers*.

There are marked differences between feasible regions governing microelectronics and those which characterize multi-processors. Among other things, distributed, parallel quorum diagnosis and configuration is paramount to our most fundamental multi-processor figure of merit: fault tolerance.

2.3. Emblematic of Feasible Regions of Fault Tolerance: Local Sparing, Also Known at NASA as 'Cross Strapping'

Fault Tolerance Should Not Be (But Often Is) Confused with Redundancy — Rather, *fault tolerance* is a desired quantifiable property whose *cost* can be measured in terms of *redundancy*. As Figure 10 illustrates, succinct models of fault tolerant multi-processors span three dimensions:

a) *Configurability*. The probability that a quorum exists. The exact definition varies with the application. In general, a quorum comprises trustable, healthy processors that can communicate via some (perhaps indirect) path. Often, but not always (*cf.* Figure 7) quorum criteria stipulate that *all* healthy nodes be included. Quorum topology may be highly constrained, as, for example, with VLSI arrays (*cf.* Figures 8 and 11).

b) *Fractional fault tolerance*. The maximum proportion of all nodes that can be faulty, such that, with probability at least the configurability, a quorum exists.

c) *Redundancy*. Resources required to sustain a quorum, up to the rated fault tolerance. For loosely-coupled multi-processors, we measure redundancy in terms of the number of channels, perhaps ratioed by dividing by the number of nodes.

The mathematically-oriented companion [34] to this paper treats (a) through (c) in more detail. To keep things at the level of broad strokes, we continue our narrative description, accented with examples and diagrams.

Example: Four-Variate Feasible Region of Fault Tolerance — Configurability (a) is often held fixed, with fault tolerance or redundancy plotted against the *quorum order*, or number of nodes. For example, the *worst case* corresponds to 100% configurability. Fortifying this notion, Figure 11a spells out switching and routing that *implements* local sparing, as presented in Figure 8. Arguably *under*utilized, local sparing is simple, yet remarkably effective over a broad range of fault tolerant applications. Figure 11b plots four-dimensional feasible regions of worst-case fault tolerance delivered by local sparing. Although illustrated here for the case of two-dimensional arrays, similar results pertain for local sparing of *any* topology [28].

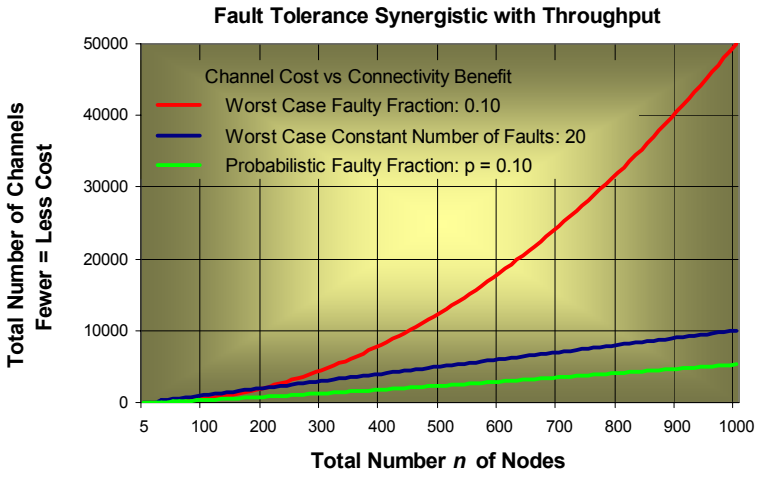
Examples: Probabilistic Feasible Regions of Redundancy — In the worst case, both the fault tolerance (Figure 11b) and redundancy of local sparing are independent of the quorum order n. In consequence, local sparing *guarantees* fault tolerance that is usually much less than what we could achieve. In fact, *no* amount of locally spared redundancy assures worst-case fault tolerance in proportion to the quorum order n. However, the situation brightens substantially when we expand our analysis to probabilistic criteria. Figures 12a through 12c quantify the good news by way of a rich fault model that spans not just faulty nodes, but (in the case of VLSI implementations) switches stuck open and closed.

2.4 Feasible Regions of Design and Operation: Keys to Multi-Processors by the Numbers

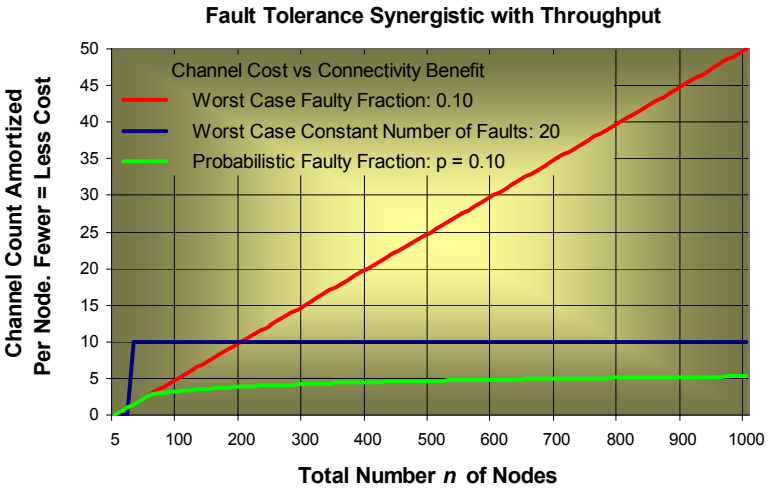
As Figure 1 illustrates, we prefer *loosely coupled nodes* as our foundation for quantifiably predictive grid computing. Sections 2.1 through 2.3 unfold how this simple yet powerful model sets the stage for optimizing i) fault tolerance (*e.g.*, Figures 7, 8, 9, 11b); ii) throughput (*e.g.*, Figures 3 and 4); iii) latency (*e.g.*, Figure 9); and iv) cost (*e.g.*, Figure 12).

When we write *multi-processors by the numbers*, we do *not* mean to suggest certain recognized, venerable pursuits of computer architects, such as sizing memory hierarchies of uni-processors, or applying pipeline techniques with the objective of maximizing uni-processor throughput. Rather, and standing on the shoulders of optimization pioneers, *multi-processors by the numbers* emphasizes insights afforded by *feasible regions* spanning as few dimensions as possible. The prime benefit of such feasible regions: best or near-best topologies, synthesized at either design time, or in realtime, while the multi-processor is in operation. Although these feasible regions *may* be conveniently linear (*e.g.*, Figures 2 and 11b), we should anticipate constraints and objectives which are curved (*e.g.*, Figures 5, 12a), and perhaps even non-convex (*e.g.*, Figures 6, 7, 12b; even 12c, with respect to quorum order n). Moreover, designing and operating multi-processors by the numbers not only dovetails with software that embodies theorems and algorithms from the mathematics of connectivity, but (*cf.* Figure 7) is *very* compatible with detailed computer simulation.

a) *Channel Cost of Fault Tolerance, Connectivity, and Throughput* — Topologies optimizing fault tolerance and throughput enjoy many *disjoint* paths between nodes. The maximum number of disjoint paths between two nodes, say *s* and *t*, is the *s-t connectivity*. The *connectivity* of a topology is the overall minimum *s-t* connectivity. Connectivity and throughput are often identical. The worst-case fault tolerance is one less than the connectivity. Hence, fault tolerance, connectivity, and throughput are synergistic. At right: curves delimiting the bottom of feasible regions of channel cost, three different criteria for quorum configurability ([34] pp. 5, 7).



b) *Amortized Channel Cost* — Feasible regions of (a), replotted to reveal the count of channels *per node*. At $n = 1000$, for example, the green curve tells us that we pay 60% (6 versus 10 channels) for probabilistic tolerance to five times as many faults (100 versus 20) as in the worst case. The probabilistic cost drops even further if we permit wide disparities in the per-node count of channels. Caution: while we can optimize worst-case fault tolerance and throughput within probabilistically well-behaved topologies ([34] Apx A.4), in many circumstances there is no substitute for dense connectivity. We are likely to find our throughput and latency hard-pressed in the face of, for example, channel starvation or intelligent hostilities.



c) *Performability: Fault Tolerance Combined with Latency* — Let the amortized channel cost scale logarithmically, according to the green curve in (b) above ([34] Apx A.4). This enables the minimum connectivity for probabilistically tolerating a constant fraction *p* of faults. Having paid this price, we seek to maximize worst-case fault tolerance and throughput, while simultaneously minimizing latency. The latter is bounded from above by the red curve plotted on the right, and from below by the green. The respective bounds are attained by *Harary-Hayes topologies* (bloated latency) and by *K-cube Hamming topologies* (tight, low latency; [34] Sec. 2.2, Apx A.2). Compare with Figures 9 and 14.

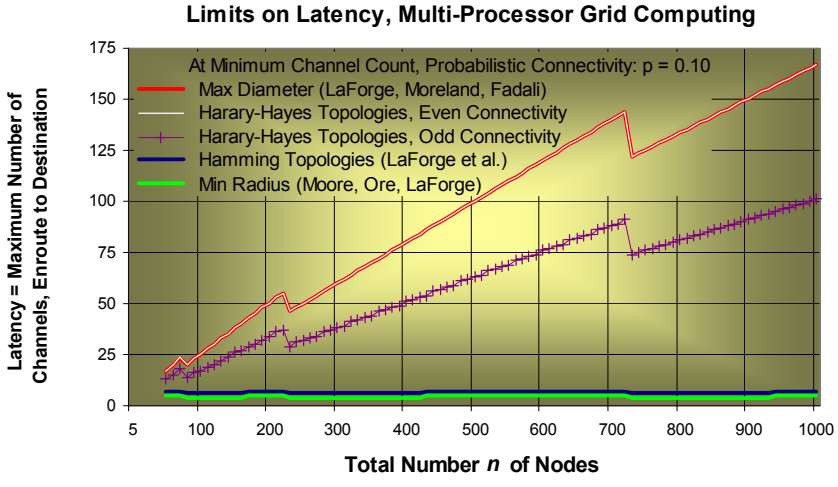


Figure 13: *Grid computing* by the numbers. Limiting curves in (a) through (c) explicate the boundaries of feasible regions governing channel cost, fault tolerance, throughput, latency, and the total number of nodes. Details in the mathematically-oriented companion [34] to this paper.

3. Grid Computing by The Numbers: Software and Hardware Enablers

3.1. From MANET FAT-Trees and Local Sparing to Grids

Section 2's exposition of fault tolerance, throughput, and redundancy serves as segue for *grid computing by the numbers*. As mentioned on page 7, our Transmission Hypercube MANET FAT-tree solution arose in response to a *benchmark problem* that is highly structured, in terms of both geometry and workload (Figures 3 and 4; [34] p. 8). Similarly, a preponderance of challenges for fault tolerant integrated circuits are very constrained, either by physical considerations (*cf.* Figure 5), or by target topology (*e.g.*, arrays, Figures 11 and 12). Grid computing is attractive because of the extent to which it loosens such shackles.

To reinforce the MDA solicitation quoted on page 3: instead of targeting a narrow workload or application-oriented topology, grid computing seeks to endow an ensemble of processors with a margin of aggregate compute power, in excess of the bare minimum required to accomplish the mission. Either at design time or in operation, *tuning* algorithms [34] synthesize topologies that give healthy nodes the best chance of communicating, in a fashion that indeed gets the job done.

It is beyond our scope to venture how application software can fully exploit a multi-processor grid. Three contributions to parallel processing underscore the importance and difficulty of bringing such as-yet immature work to bear. Chalasani and Thulasiraman develop and analyze parallel algorithms for optimization [8]. Y. Shi [49] establishes the equivalence of two erstwhile different formulae, attributed to Amdahl *resp.* Gustafson, for predicting execution time speedup. More recently, T. G. Robertazzi [48] promotes *divisible load theory*. To this end, our *quorum model* hinges on applications that change, (re)negotiate, or influence objectives or constraints derived from the four criteria enunciated the top of page 7: i) fault tolerance; ii) throughput; iii) latency; and iv) cost. As to *how* to achieve these in practice, Figure 5 suggests the capstone posit of this paper:

As concentrated abundances of photolithographically fabricated transistors have enabled microelectronic integration, concentrated abundances of wireless channels enable grid computing (8)

For spaceborn multi-processors, such as those evoked by Figure 1, *pencil beams of electromagnetic radiation* provide a compelling, low-interference means of concentrating abundances of communication channels. Accordingly, the device-oriented companion [35] to this paper examines possibilities for innovatively leveraging the mathematics of connectivity using a technology whose time has arrived: the VCSEL.

3.2. From Fault Tolerance to Throughput, Latency, and Cost

By no means do the local sparing examples of Section 2.3 and Figures 11 and 12 constitute the final word on feasible regions of fault tolerance. To the contrary, the mathematics of connectivity is just getting underway, and undiscovered delights outnumber known results.

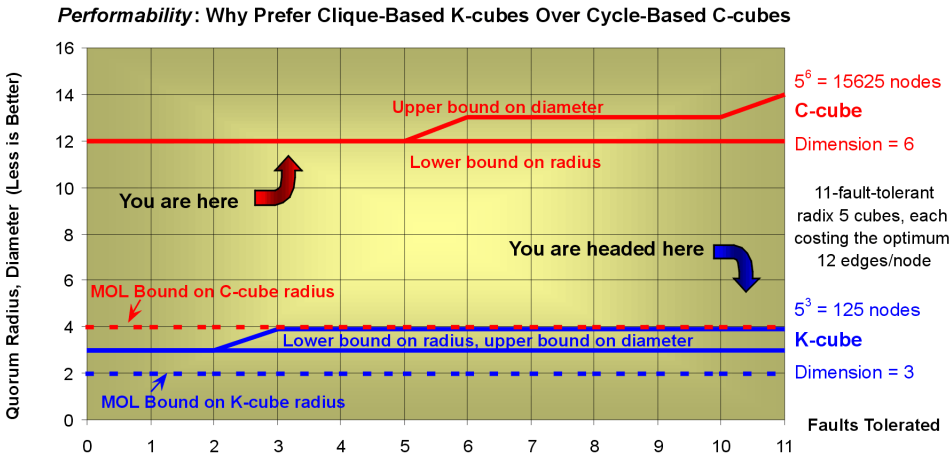
For example, blue and red curves in Figures 13a and b depict feasible regions of scalability, as bounded by the *Harary-Hayes optimum tradeoff* between worst-case fault tolerance and minimum (unweighted) channel cost ([34] Eqn (3)). By contrast, the green curves in Figures 13a and b delineate feasible regions reflecting the minimum channel cost of *probabilistic* fault tolerance ([34] Thm 3). While these results are encouraging, it remains to generalize them to the case of nonuniform channel cost.

Concerning the important (and formidable) objective of minimum latency, Figure 13c crystallizes the tight-to-loose range of topologies sporting identical connectivity and fault tolerance, and with the same number of channels for each of n nodes. It therefore behooves us to intelligently choose topologies, such as clique-based *K-cubes* ([34] Sec. 2.3), which feature many *short* disjoint paths between nodes. This as opposed to topologies, such as those attributed to Harary and Hayes, which contain many *not*-so-short disjoint paths. Of course, exactly what constitutes an intelligent choice depends on our particular objectives and constraints. Customers, mission directors, or military commanders to whom we deliver multi-processor topologies should therefore enjoy the option of selecting from fault tolerance and connectivity *tuned* from sparse to dense [34].

3.3. Performability: From Theorems, to Algorithms, to Software for Multi-Processor Design and Operation

Nabli and Sericola employ *performability* to designate combinations of fault tolerance and performance [43]. If we allow that performance encompasses throughput, latency, and cost, then – and as pointed out on pages 7 and 13 – feasible regions proffer prospects for predicting the performability of multi-processor grids. For example, Figure 13c folds performability bounds for Harary-Hayes topologies (bloated latency) alongside performability bounds for Hamming topologies (K-cubes, with asymptotically best possible latency approaching the Moore Bound), as well as absolute, natural limits (red and green curves) on *any* topology with commensurate fault tolerance, throughput, and connectivity.

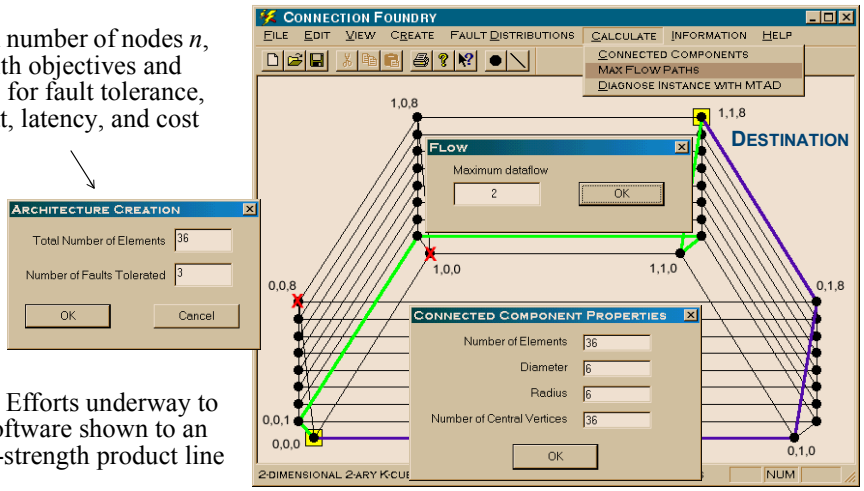
As Figure 9 illustrates, a more thorough treatment would bracket the *performability envelope* of quorums, as we render nodes inoperative, and in quantities from zero on up. At this fine-grained level of fidelity, Figure 14 compares the performability of K-cubes with a general lower limit (the *Moore Bound*) on latency, as well as with more popular (but generally suboptimal) C-cubes. The seven applications described in Sections 1.2 through 2.3 of [34] bolster the case for compiling these results, along with a host of similar such results, as a theorem-and-algorithm catalog to the mathematics of connectivity. Figure 15 illustrates how such a catalog can (and, we submit, should) be embodied by software that *synthesizes* optimum, or near-optimum, topologies. As Figure 16 depicts, this software also serves as a springboard for distributed, parallel algorithms for diagnosis and configuration, in turn ported to the operational nodes of a multi-processor. Consistent with contemporary trends in experimental mathematics [16], we have used the same software as an adjunct for crafting new theorems and algorithms, including the bulk of those presented in the appendices of [34].



See [31] or [34] for details about *K-cubes*, *C-cubes*, *radius*, *diameter*, and the *Moore Bound*

Figure 14: Feasible regions of *performability* illuminate how the *fractional* fault tolerance of Hamming K-cubes bests that of traditional C-cubes. For given fault tolerance, moreover, and at minimum count of channels per node, K-cube latency converges to the best possible Moore Bound, while C-cube latency diverges from the Moore Bound. Compare with Figures 9 and 13c.

Input: Total number of nodes *n*, along with objectives and constraints for fault tolerance, throughput, latency, and cost



Status — Efforts underway to refine software shown to an industrial-strength product line

Design-time precursor to the formulation of distributed, parallel synthesis algorithms, suitable for embedding as part of *operational* media access control (MAC) software in the nodes of a multi-processor (*cf.* Figure 16)

Figure 15: Software enabler for multi-processors by the numbers. Drawing from the mathematics of connectivity, Connection Foundry™ synthesizes optimum topologies for grid computing. (Client: NASA / Jet Propulsion Laboratory [25])

In a distributed, parallel fashion, MANET quorum nodes adaptively reconfigure radio-frequency (RF) channels, *without* recourse to a central MAC master. Shown here: simulated Hamming routing of a packet through a minimum power, minimum Euclidean spanning tree (MEST). The MEST spans a topology known as the *Delaunay triangulation*. Compare with Figures 1b, 3, and 4

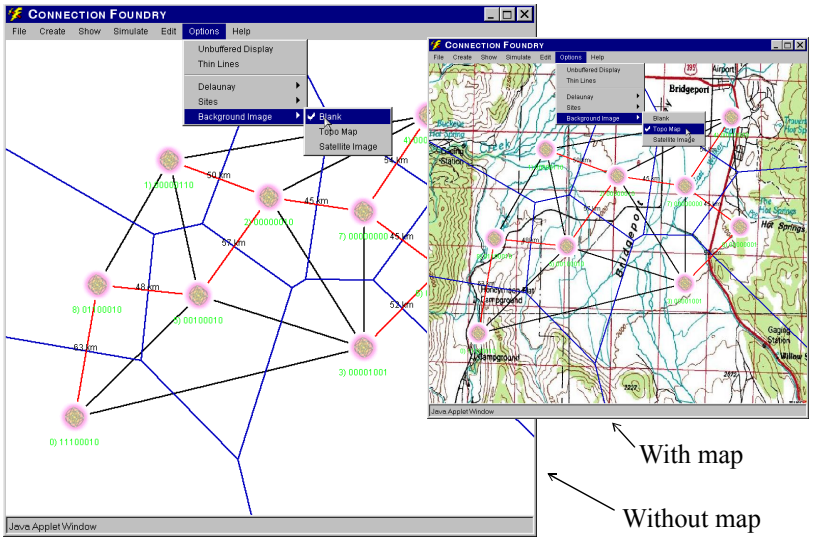


Figure 16: Tunable sparse connectivity, by the numbers. As a foundation of power-conserving MANETs, Connection Foundry™ invokes *Voronoi algorithms* from computational geometry [10]. (Client: Department of Homeland Security [21])

4. Recap, Unfinished Business

It makes compelling sense to cast multi-processor design and operation in terms of feasible regions and objective functions. The precise figures of merit should, we submit, span but a handful of dimensions stemming from combinations of i) fault tolerance; ii) throughput; iii) latency; and iv) cost. Moreover, and as underscored by the seven applications described in Sections 1.2 through 2.3 of [34], we seek topologies that *constructively* achieve optima within feasible regions of design and operation. A key enabler: *software* that *synthesizes* optimal topologies. As a practical test, we would benefit from a multi-processor on, say, 128 nodes enabled by VCSEL channels. Suggested by Figure 1 and described in [35], such a multi-processor would constitute first serious delivery to customers of spaceflight grid computing.

Acknowledgements

A number of collaborators inspired our informal, example-and-diagram tutorial; in particular: Erwin Myrick, of the U.S. Missile Defense Agency; Ross F. Gallo, of The Right Stuff of Tahoe; and Alan R. Lindsey, formerly with the Air Force Research Laboratory and now with Austral Engineering and Software. James D. Meindl, of the Georgia Institute of Technology, graciously endorsed reprint permission by the Institute of Electrical and Electronics Engineers (IEEE) for Figure 5. David Avis, the first author's doctoral thesis advisor, continues to champion feasible regions and techniques for optimization. Thank you, one and all!

References

[1] L. Alkalai and D. Geer, "Space-Qualified 3D Packaging Approach for Deep Space Missions: New Millennium Program, *Deep Space 1* Micro-Electronics Systems Technologies". Viewgraph presentation. Pasadena, CA: Jet Propulsion Laboratory, 7-Oct-1996. Cited p. 1.

[2] L. Alkalai and A. T. Tai, "Long-life Deep-Space Applications". *Computer*. Aug-1998, pp. 37 – 38. Cited p. 1.

[3] S. M. Ardalan, "DrawCraft: A Spacecraft Design Tool for Integrated Concurrent Engineering". *2000 IEEE Aerospace Conference*. Big Sky, Montana: 18-Mar-2000. Online at www.ugcs.caltech.edu/~sardalan/. Cited p. 1.

[4] A. Avizienis, "The Hundred Year Spacecraft". *Proceedings, First NASA/DOD Workshop on Evolvable Hardware*. Pasadena, CA: Jet Propulsion Laboratory, Jul-1999, pp. 233 – 239. Cited p. 1.

[5] A. Avizienis, "Toward Systematic Design of Fault-tolerant Systems". *Computer*. Apr-1997, pp. 51 – 58. Cited p. 1.

[6] C. G. Bell, *Announcing Local Area VAXcluster Systems: a New Dimension in Work Group Computing*. Digital Equipment Corporation archive, 1986. Online at http://research.microsoft.com/~gbell/Digital/timeline/1986-4.htm. Cited p. 5.

[7] Brainy Quote, *Make Everything as Simple as Possible, But Not Simpler*. Albert Einstein Quotes. Online at www.brainyquote.com/. Cited p. 11.

[8] P. R. Chalasani and K. Thulasiraman, "Parallel Computing for Network Optimization: A Cluster-Based Approach for the Dual Transshipment Problem". *Seventh IEEE Symposium on Parallel and Dist'd Processing*. IEEE Computer Society, Oct-1995, pp. 66 – 73. Cited pp. 11, 15.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press. Tenth printing, 1993. Cited p. 11.

[10] M. de Berg, M. van Krevald, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. 2nd edition. Berlin: Springer-Verlag. 2000. Cited p. 1; Figures 2 and 16.

[11] R. E. Dickerson and I Giess, *The Structure and Action of Proteins*. Menlo Park, CA: W. A. Benjamin. 1969. Cited p. 13.

[12] D. Edenfeld, A. B. Kahng, M. Rogers, and Y. Zorian, "2003 Technology Roadmap for Semiconductors". *Computer*. Jan-2004, pp. 47 – 56. Cited p. 1.

[13] M. S. Fadali and L. E. LaForge, "Linear Time Computation of Feasible Regions for Robust Compensators". *International Journal of Robust and Nonlinear Control*. **11**, Jun-2001, pp. 819 – 856. Cited Figure 6, p. 13.

[14] J. M. Feldman and C. T. Retter, *Computer Architecture: a Designer's Text Based on Generic RISC*. New York: McGraw-Hill. 1994. Cited p. 5.

[15] M. Friedman, *George Joseph Stigler: January 17, 1911 – December 1, 1991*. Biographical Memoir, National Academy of Sciences, National Academies Press, 1998. Online at www.nap.edu/html/biomems/. Cited p. 7.

[16] J. Horgan, "The Death of Proof". *Scientific American*. Oct-1993. pp. 92 – 103. Cited p. 15.

[17] D. S. Katz and R. R. Some, "NASA Advances Robotic Exploration". *Computer*. Jan-2003, pp. 52 – 61. Cited p. 1.

[18] L. J. Kenah, R. E. Goldenberg, and S. F. Bate, *VAX/VMS Internals and Data Structures*. 2nd edition. Bedford, MA: Digital Press. 1988. Cited pp. 5, 11.

[19] N. P. Kronenberg, H. M. Levy, and W. D. Strecker, "VAXcluster: a Closely-coupled Distributed System". *ACM Trans. on Computer Systems*. **4** (2), 1986, pp. 130 – 146. Online via http://portal.acm.org/citation.cfm?id=214421. Cited pp. 5, 11.

[20] L. E. LaForge, "Hamming Graphs". *Proc., 2004 IPSI: Internet, Processing, Systems for E-education/E-business, and Interdisciplinaries*. 9-Oct-2004. Cited p. 11.

[21] L. E. LaForge, *Crypto-Secure Remote Terminal Unit for New and Retrofit Supervisory Control and Data Acquisition*. Technical report, U.S. Homeland Security contract NBCHC040088. Reno, NV: The Right Stuff of Tahoe, Incorporated. 18-Oct-2004. Cited Figure 16.

[22] L. E. LaForge, *Clique-Factorized Optimum MANET Throughput Enabled by Directed, Power-Controlled Antennas*. Technical report, U.S. DoD contract FA8750-04-C-0020. Reno, NV: The Right Stuff of Tahoe, Incorporated. 14-Apr-2004 revised 7-Aug-2004. Cited Figure 3, p. 7.

[23] L. E. LaForge, "Self-healing Avionics for Starships". *Proceedings, 2000 IEEE Aerospace Conference*. 18-Mar-2000. Cited pp. 1, 5, 7.

[24] L. E. LaForge, *Architectures and Algorithms for Self-Healing Autonomous Spacecraft*. Phase 1 report, NASA Institute for Advanced Concepts, 9-Jan-2000, revised 28-Feb-2000. Cited Figure 1, 11.

[25] L. E. LaForge, *Fault Tolerant Physical Interconnection of X2000 Computational Avionics*. Pasadena, CA: Jet Propulsion Laboratory, document number JPL D-16485. 28-Aug-1998, revised 18-Oct-1999. Online at http://faculty.erau.edu/laforgel/. Cited p. 1, Figure 16.

[26] L. E. LaForge, "Configuration of Locally Spared Arrays in the Presence of Multiple Fault Types". *IEEE Trans. on Computers*. **48** (4), Apr-1999, pp. 398 – 416. Online at http://faculty.erau.edu/laforgel/. Cited Figure 12.

[27] L. E. LaForge, *NASA / ASEE Fellowship, Control and Autonomy Group, Jet Propulsion Laboratory*. 1997 collaboration with Dr. Allen P. Nikora. Includes overview of work on *Deep Space One*, NASA probe propelled by an ion engine. Online at http://faculty.erau.edu/laforgel/Current-and-Recent-Research/NASA-ASEE/. Cited p. 12.

[28] L. E. LaForge, "What Designers of Wafer Scale Systems Should Know About Local Sparing". *Proc., 1994 IEEE International Conference on Wafer Scale Integration.* R. M. Lea and S. K. Tewksbury, eds. Los Alamitos: IEEE Computer Society Press, 1994, pp. 106 – 131. Cited p. 13.

[29] L. E. LaForge, *Fault Tolerant Arrays.* Ph.D. dissertation. Montreal: McGill University, 1991. Cited Figure 11.

[30] L. E. LaForge, K. Huang, and V. K. Agarwal, "Almost Sure Diagnosis of Almost Every Good Element". *IEEE Trans. Comp*. **43** (3), Mar-1994, pp. 295 – 305. Online at http://faculty.erau.edu/laforgel/. Cited Figure 7.

[31] L. E. LaForge, K. F. Korver, and M. S. Fadali, "What Designers of Bus Structures and Networks Should Know About Hypercubes". *IEEE Transactions on Computers*. **52** (4), Apr-2003, pp. 525 – 544. Online at http://faculty.erau.edu/laforgel/. Cited Figure 14.

[32] L. E. LaForge and K. F. Korver, "Graph-theoretic Fault Tolerance for Spacecraft Bus Avionics". *Proc., 2000 IEEE Aerospace Conference.* 18-Mar-2000. Cited p. 3.

[33] L. E. LaForge and K. F. Korver, "Mutual Test and Diagnosis: Architectures and Algorithms for Spacecraft Avionics". *Proceedings, 2000 IEEE Aerospace Conference.* 18-Mar-2000. Cited p. 13.

[34] L. E. LaForge, J. R. Moreland, and M. S. Fadali, "Spaceflight Multi-Processors with Fault Tolerance and Connectivity Tuned from Sparse to Dense". *Proceedings, 2006 IEEE Aerospace Conference*, 4-Mar-2006. Cited pp. 1, 3, 5, 7, 9, 11, 13, 15; Figures 1, 3, 7, 8, 9, 11, 13.

[35] L. E. LaForge, J. R. Moreland, R. G. Bryan, and M. S. Fadali, "Vertical Cavity Surface Emitting Lasers for Spaceflight Multi-Processors". *Proceedings, 2006 IEEE Aerospace Conf.* 4-Mar-2006. Cited pp. 1, 3, 5, 7, 15, 17.

[36] J-C LaPrie, *Dependable Computing and Fault-Tolerance: Concepts and Terminology*. Research Report 84.035 (1984). Laboratoire d'Automatique et d'Analyse des Systemes, Centre National de la Recherche Scientifique, 7, avenue du Colonel Roche, 31077 Toulouse Cedex, France. Also in *Proc. 15th International Symposium on Fault-Tolerant Computing*. June, 1985, pp. 2 – 11. Cited p. 5.

[37] I. Lustig, *How George Dantzig Solved The Diet Problem*. Interview with G. B. Dantzig, 2002. Online at www.e-optimization.com/directory/trailblazers/dantzig/index.cfm. Cited p. 7.

[38] J. D. Meindl, J. A. Davis, P. Zarkesh-Ha, C. S. Patel, K. P. Martin, and P. A. Kohl, "Interconnect Opportunities for Gigascale Integration". *IBM Journal of Research and Development.* **46** (2/3), Mar/May-2002, pp. 245 – 263. Cited Figure 5.

[39] Microsoft Excel 2003, *About Solver: Algorithm and Methods Used by Solver*. Online help. Cited p. 11.

[40] E. F. Moore and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays, Part I". Early, perhaps first, use of *quorum* on p. 202. *Journal of the Franklin Institute*. **262**, Sep-1956, pp. 191 – 208. Cited p. 5.

[41] F. H. Murphy, "Annotated Bibliography on Linear Programming Models". *Interactive Transactions of Operations Research/ Management Science (ITORMS)*. **1**, 1996. Online at http://itorms.iris.okstate.edu/. Cited Figure 2, p. 7.

[42] B. T. Murray and J. P. Hayes, "Testing IC's: Getting to the Core of the Problem". *Computer*. Nov-1996, pp. 32 – 38. Cited p. 5.

[43] H. Nabli and B. Sericola, "Performability Analysis: a New Algorithm". *IEEE Transactions on Computers*. **45** (4), Apr-1996, pp. 491 – 494. Cited p. 15.

[44] National Aeronautics and Space Administration, *NASA Independent Verification and Validation Facility*. Online at www.ivv.nasa.gov/index.php. Cited p. 5.

[45] B. Noble, *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1969. Cited Figure 2, p. 9.

[46] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall: Englewood Cliffs, NJ. 1982. Cited Figure 2.

[47] B. Render and R. M. Stair, Jr., *Quantitative Analysis for Management*. Upper Saddle River, NJ: Prentice-Hall, 1997. Cited Figure 2, pp. 7, 9.

[48] T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory". *Computer*. **36** (5), May-2003, pp. 63 – 68. Cited p. 15.

[49] Y. Shi, *Reevaluating Amdahl's Law and Gustafson's Law*. Oct-1996. Online at http://joda.cis.temple.edu/~shi/docs/amdahl/amdahl.html. Cited p. 15.

[50] R. Simar, Jr., "DSP Architectures, Algorithms and Code Generation: Fission or Fusion?" *Proc., 1997 IEEE International Conference on Innovative Systems in Silicon.* L. E. LaForge, H. Bolouri, D. Sciutto, and S. K. Tewksbury, eds. Los Alamitos: IEEE Computer Society Press, 1997, pp. 220 – 227. Cited p. 5.

[51] W. Stallings, *Computer Organization and Architecture: Principles of Structure and Function*. 2nd edition. New York: McGraw-Hill. 1993. Cited pp. 5. 7.

[52] G. J. Stigler, "The Cost of Subsistence". *Journal of Farm Economics*. **27**, 1945. pp. 303 – 314. Cited p. 7.

[53] United States Missile Defense Agency, "Adaptable/Reconfigurable Distributed Spacecraft Processing". Department of Defense SBIR Topic MDA04-183. Aug-2004. Online at www.acq.osd.mil/sadbu/sbir/solicitations/sbir044/. Cited p. 3.

[54] United States Navy, "W Band, Real Time Wireless Network for Avionics Applications". Small Business Innovative Research (SBIR), Department of Defense SBIR Solicitation Topic N05-142. 1-Aug-2005. Online at http://www.dodsbir.net/solicitation/sbir053. Cited p. 3.

[55] P. Zarkesh-Ha and J. D. Meindl, "An Integrated Architecture for Global Interconnects in a Gigascale System-on-a-Chip (GSoC)". *IEEE Symposium on VLSI Technology, Digest of Technical Papers*. June 2000, pp. 194 – 195. Cited Figure 5.

BIOGRAPHICAL SKETCHES

Laurence E. LaForge (IEEE Member) is President of the Right Stuff of Tahoe. Previously on faculty with Embry-Riddle Aeronautical University, he twice held NASA/ASEE Fellowships at the Jet Propulsion Laboratory, where he worked on bus fault tolerance and the *Deep Space 1* probe. He has been guest editor for the *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, and program chair for the IEEE International Conference on Innovative Systems in Silicon. His baccalaureate in mathematics is from the Massachusetts Institute of Technology; his PhD from McGill University.

James W. G. Turner Holds a BS from the McGill University. His experience encompasses more than 17 years of developmental, programming, and leadership roles in technology industries. Most recently, he was communications consultant to SSTDigital Communications of Seattle. Until 2001, Mr. Turner was Vice President of Sales and Marketing for LMS Medical Systems.