Distributed Resource Collectives: P2P and Computational Grids

David DiNucci, PhD

dave@elepar.com www.elepar.com

Overview

Big Picture

- **Acknowledging Connectivity/Metacomputing**
- **Historically Formative Technologies**
- **Distinguishing features (P2P & Grid & Hybrid)**

Tech issues

Focus on Distributed Computing

- **Existing approaches**
- **Elepar's approach ("compupackets", CDS, SC, PICA)**

The Players

- **Users/Groups**
- **Companies/Projects**

Acknowledging Realities of Connectivity

Topology: **Users highly-connected, no longer star topology of centralized mainframe or server accessed with dumb terminals or clients**

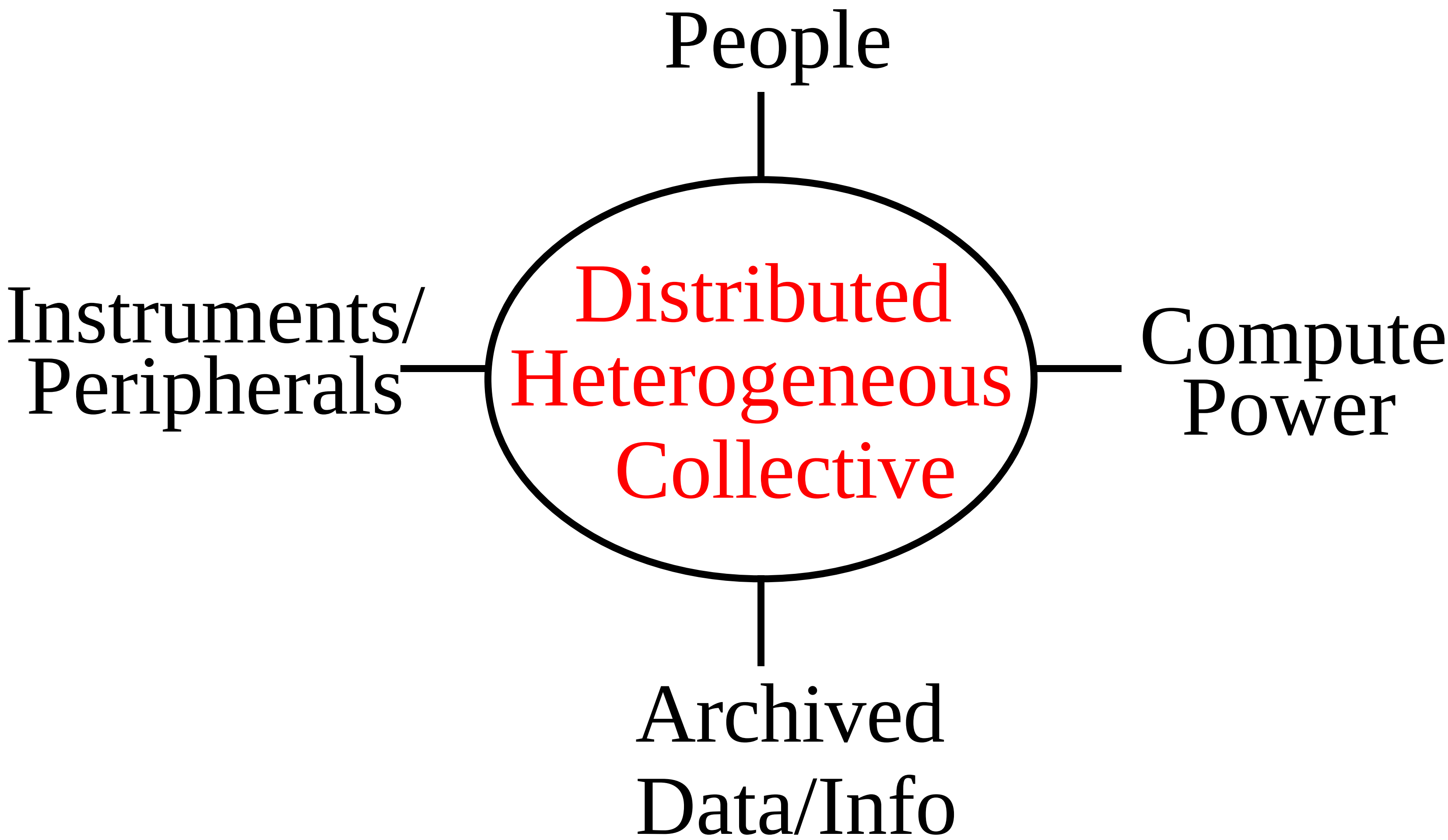
Power: **"Clients" are often nearly as powerful as the "servers" (more powerful in aggregate)—or as simple as cell phones**

Link directionality: **Bidirectional, not restricted to just "push" (like email) or just "pull" (like web)**

Storage locality: **Memories and disks of network nodes effectively form distributed storage hierarchy**

Metcalfe's law: **Value of network ~ (# users)^2**

A Grand Vision: "Metacomputing"



Connections: Some History

82 85 88 89 90 91 92 93 94 95 96 97 98 99 00 01

cosmic cube iPSC1 nCUBE beowulf **Dist. Computing Arch.**

Dist. Computing APIs (sc) pvm3 (cds) elepar
pvm1 pvm2 mpi-1 mpi-2 avika

Grid i-way ipg@NASA ibm
legion paci@NSF dtf
GRID book
* * * gf1/2/3/4/5 ggf1/2

WWW/Peer-to-Peer mosaic java icq napster (2.8B)
ipbackbone netscape jini gnutella jxta
<-usenet wais, archie wiki xml soap p2pwg * *

Cycle Stealing/“P2P computing” seti@home
cancer@intel
condor primenet...entropia
distributed.net

IBM Grid Announcement: Aug 2, 2001

NSF TeraGrid Announcement: Aug 9

News – August 9, 2001

NSF PR 01-67
Media contact: Tom Garritano (703) 292-8070 tgarrita@nsf.gov
Program contact: Bob Borchers (703) 292-8970 rborcher@nsf.gov

Distributed Terascale Facility to Commence with $53 Million NSF Award

High-performance computing system will come on-line in mid-2002

The world?s first multi-site supercomputing system -- Distributed Terascale Facility (DTF) -- will be built and operated with $53-million from the National Science Foundation (NSF). The DTF will perform 11.6-trillion calculations per second and store more than 450-trillion bytes of data, with a comprehensive infrastructure called the ?TeraGrid? to link computers, visualization systems and data at four sites through a 40-billion bits-per-second optical network.

The National Science Board (NSB) today approved a three-year NSF award, pending negotiations between NSF and a consortium led by the National Center for Supercomputing Applications (NCSA) in Illinois and the San Diego Supercomputer Center (SDSC) in California, the two leading-edge sites of NSF?s Partnerships for Advanced Computational Infrastructure (PACI). NCSA and SDSC will be joined in the DTF project by Argonne National Laboratory (ANL) in suburban Chicago and the California Institute of Technology (Caltech) in Pasadena.

"The DTF will be a tremendous national resource," said NSF director Rita Colwell. "With this innovative facility, NSF will demonstrate a whole new range of capabilities for computer science and fundamental scientific and engineering research, setting high standards for 21st Century deployment of information technology."

Peer-to-Peer vs. Grids: Different Sociology

	Peer-to-Peer	Grid
People	Sales, marketing, analysts, hobbyists	Researchers, scientists, engineers, designers
Archives	Documents, sales, music, game state	Scientific, design, historical databases
Compute	PCs, PDAs	Parallel servers, supers
Peripherals	GUIs, personal devices, printers	CAD, immersive VR, sensory, robotic devices
Economy	Cheaper (conserve in existing practice)	Bigger (enable new capability / approach)
Motivators	Data access, privacy, autonomy, indep.	Capacity availability, scalability, efficiency
Dynamics	Assemble, disband	Utility, grow & shrink

Hybrids: “Peer-to-Grid”

Tie together the servers and workstations scattered around their enterprise into a collective compute, collaboration, & storage facility

Advantages:

- **Relatively cheap: More effective exploitation of existing resources**
- **Continuous upgrade**

Challenges:

- **Currently very difficult for most applications to utilize this distributed, heterogeneous, dynamic environment effectively**
- **Even behind firewall, need to deal with “it’s my machine”**

Aerospace and biotech companies are using this mode on some easily-distributable problems

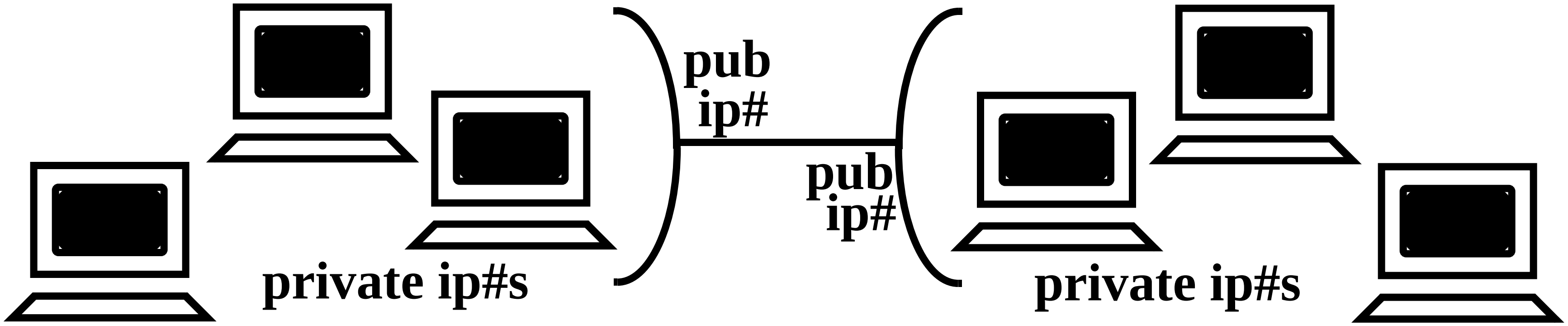
Technical Challenges

- **Dynamic topology -> Resource discovery/reservation/scheduling**
- **Heterogeneous speed/archit. -> Portability/Variable granularity**
- **Local or distributed comm -> Latency tolerance/Low ovhd/QoS**
- **Many decentralized components -> Fault tolerance**
- **Complex & concurrent -> Formal analysis, debugging rules**
- **App still #1 -> Leveraging existing tools, languages, techniques**
- **Utilizing untrusted resources -> Privacy/Security/Anonymity**
- **ROI -> Revenue models, pricing, bidding, accounting**
- **Connectivity -> Firewalls, NATs, dropped lines**
- **Intellectual property -> DRM, digital watermarking, capabilities**
- **Multiple administrative domains -> flexible policies**

And when tech solutions are found, then comes standardization.

P2P Connectivity Challenges/Approaches

- **Network Address Translators (NATs, IP masq) & Firewalls**



- **Brokers**
- **Port 80, Universal Plug-n-Play (UPnP): www.upnp.org**
- **Dialup oversubscribing**
- **Centralized vs. distributed directories, cacheing policies**
- **Collaboration modes (sync, async)**
- **Content delivery networks (CDN), digital rights management**

Distributed Computing Challenges: 4 Ps

Performance: **Wouldn't be using this approach in first place if speed unimportant...so Java sometimes ruled out**

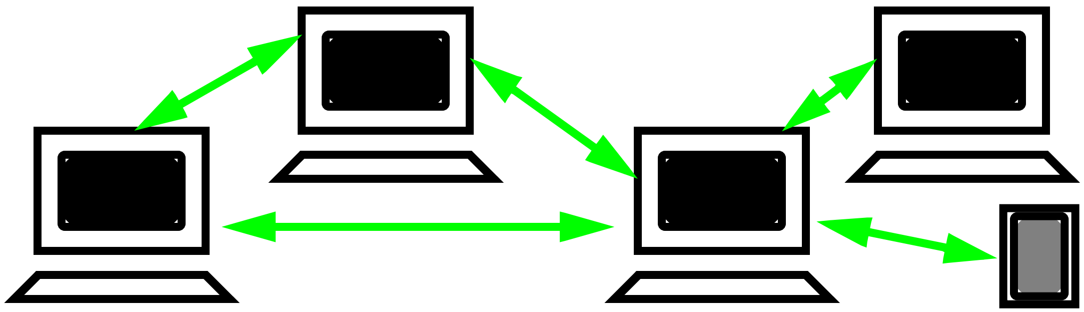
Portability: **Programs must be portable not just among different node architectures, but while concurrently using varying numbers of potentially-faulty heterogeneous nodes running at different speeds with differing topology and connectivity**

Programmability: **Need methodologies for developing and verifying programs to manage complexity inherent in these concurrent distributed heterogeneous programs**

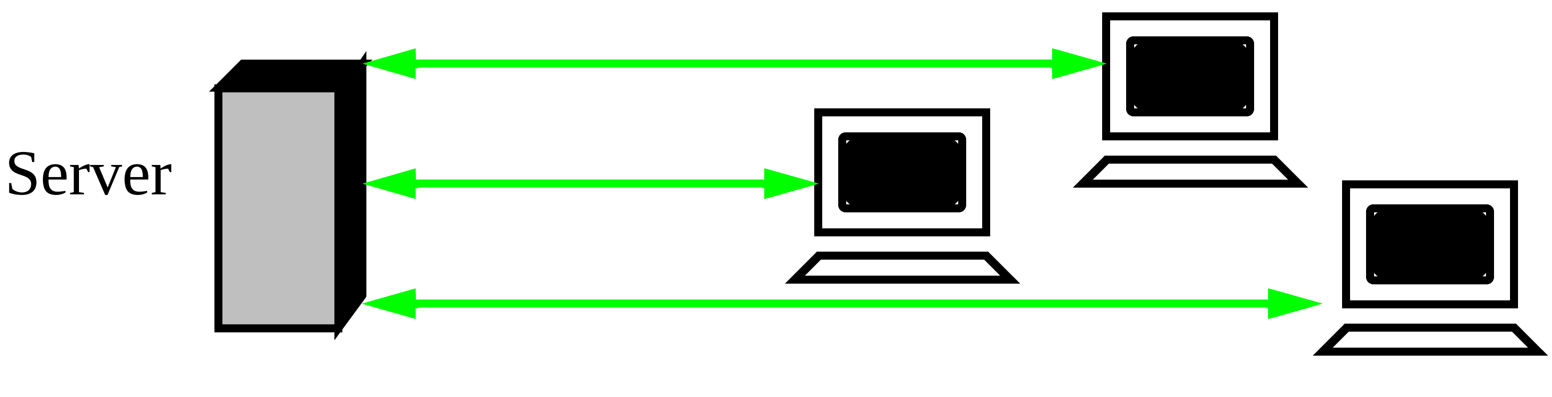
Profitability: **A revenue model that works for app developers, compute providers, and users**

Distributed Computing: Current Approaches

Grids (e.g. Legion, Globus): Explicit decomposition and embedding a la high-perf parallel. Relatively high perf, low prog, low port.

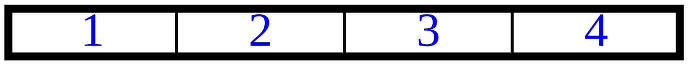


P2P (e.g. Entropia, U.Dev): Client server, SPMD, divide & conquer search (parameter or huge domain) a la SETI@Home.



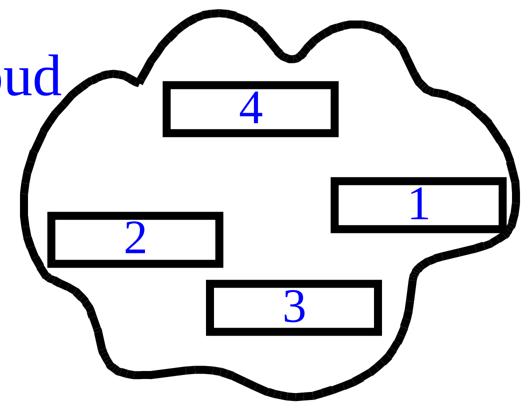
Elepar Approach: CompuPackets (Dataflow)

Traditional Packet Switching

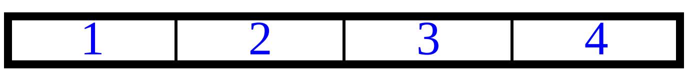


Message is indep packets

BW cloud

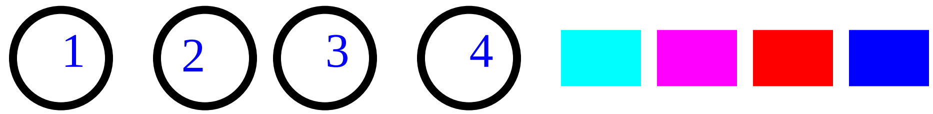


Packets independently use BW to get to destination



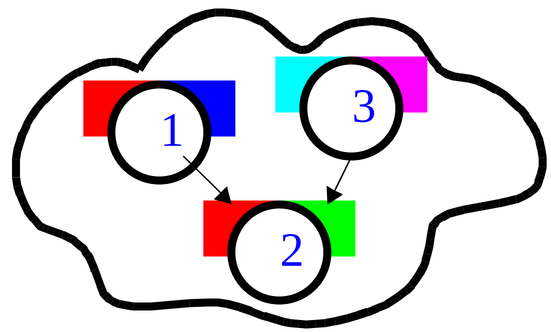
Re-interpret packets as msg

"Compu-Packet Switching"

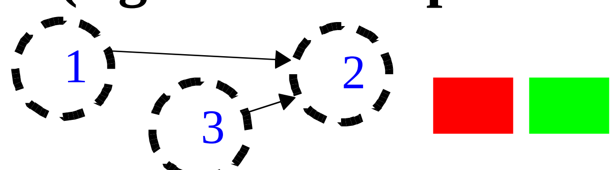


Program is "indep" threads, init data

cycle+BW cloud (for compute+ create)



"Compupacket" = thread+its data. Compupackets independently use cycles to run, create new data (e.g. for new c'packets).



Interpret as computation DAG+results

Advantages/Challenges of this Approach

Advantages:

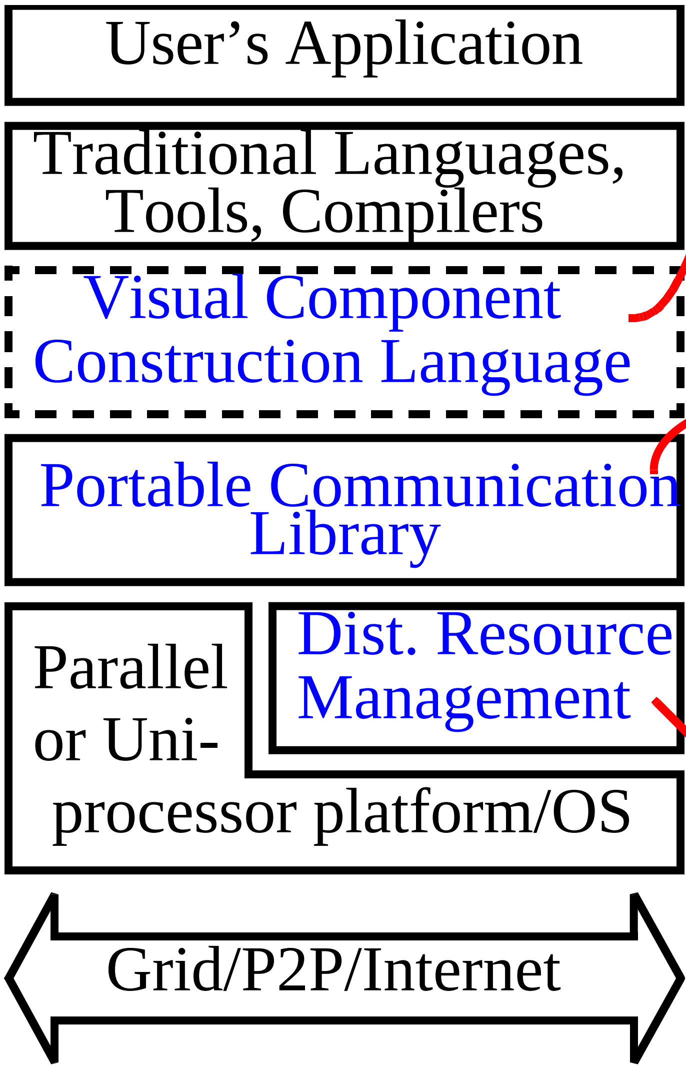
- **Compackets indep, either run or don’t: Nowaiting for each other**
- **Compupackets fill up processors like pebbles into bucket, efficiently using whatever cycles are available**
- **Compupackets are atomic transactions, aiding fault tolerance**
- **Each compupacket is functional, easy to specify & reason about**

Challenges:

 # ready compupackets should be large (> # available processors)

- **Need methods to build programs in this form (w/existing langs)**
- **Binding data to compupacket and initiating it must be low ov’h’d**
- **Link latency must be hidden or avoided when possible**
- **Need strategies for compupacket binding, processor assignment**

Elepar's Three Layers



Software Cabling (SC): Visual OO component coordination methodology for building & analyzing "independent thread" apps from modules implemented in trad'l langs

Cooperative Data Sharing (CDS): Efficient runtime support for portable threads and communication between them, on variety of architectures

People, Instruments, Computers, and Archives (PICA): Rules and protocols for finding, bargaining for, and scheduling distributed, independently-controlled resources of all kinds

From Program to Threads

Writing multi-threaded apps or parallelizing compilers tough, BUT

- **virtually all programs are built from smaller components (i.e. functions, subroutines, methods, etc.)**
- **those components already act like compupackets—i.e. they begin with their input data, run to completion creating results**
- **so, just need to augment component composition, initiation rules**

Software Cabling (SC): Modules

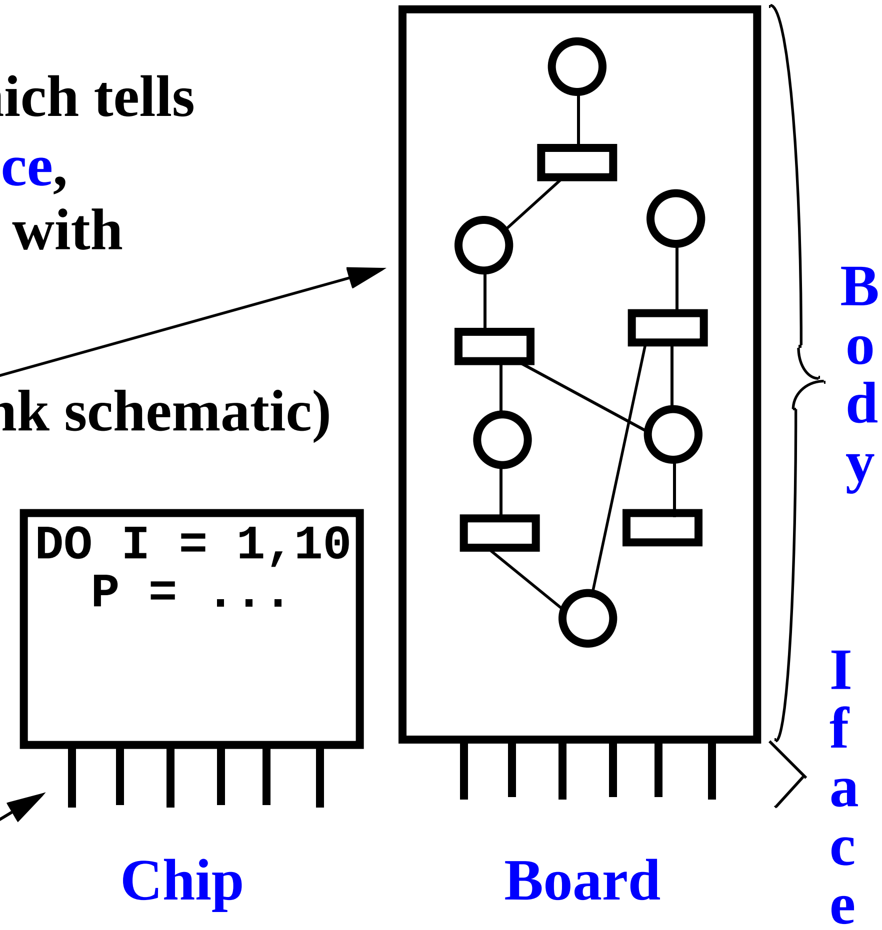
Two kinds of modules:

Both consist of a body, which tells what to do, and an interface, which allows it to interact with its environment

Board body is graphical (think schematic)

Chip body is written in your favorite language (C++, Java...Fortran?)

Interface consists of pins which carry data and signals.



SC: Chip Body

A chip's body is a subprogram or function, and the pins in the chip's interface are its arguments.

To change the color of a memory, the chip posts a signal to the pin, using a special statement of roughly the form:

```
post signame to pinname
```

This is the only special statement in your code, and it does not block or otherwise change the local behavior of the code*

***except for optionally making the pin argument inaccessible**

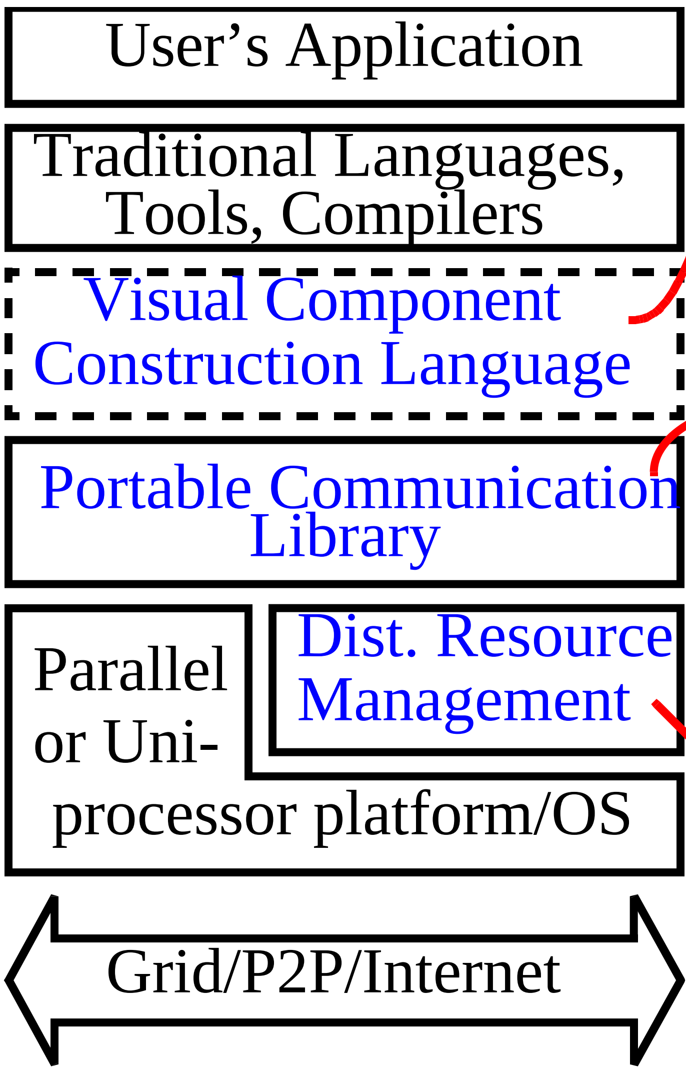
SC As High-Level Design/Spec Language

SC is a very-high-level graphical component composition language which contains the constructs required to manage the complexity and make real-world programming possible: e.g.

- **Modular (OO), template-based program construction**
- **Adapts to most any source language (e.g. Java, Fortran, C, C++)**
- **Formal, functional specifications provide leverage for program verification and powerful debugging techiques & replay**
- **Fault tolerance (because compupackets are atomic transactions!)**
- **Supports distributable arrays, mem allocation, data parallelism**

...making SC an excellent alternative design & specification language for all large-scale mission-critical software development.

Elepar's Three Layers

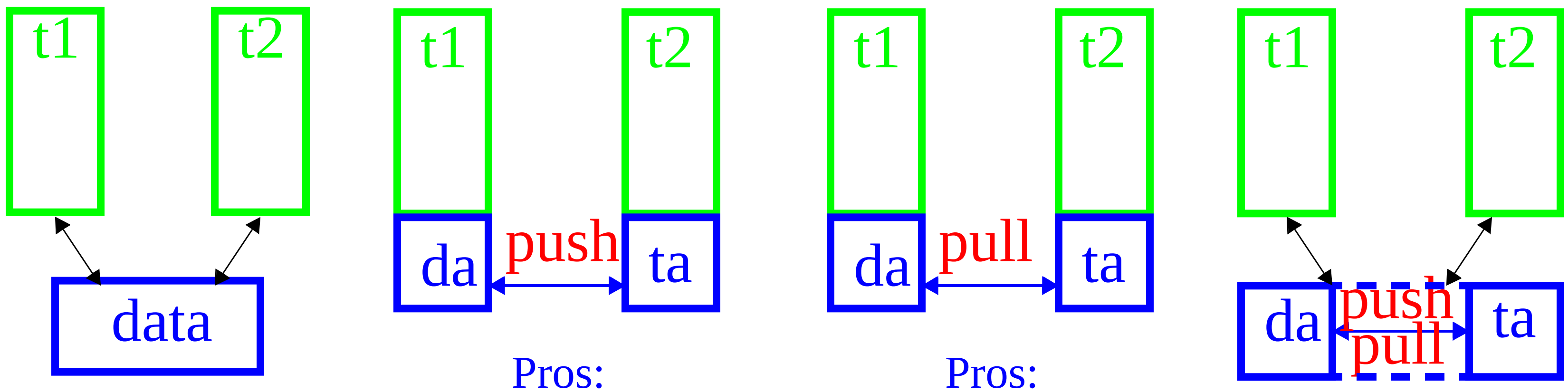


Software Cabling (SC): Visual OO component coordination methodology for building & analyzing "independent thread" apps from modules implemented in trad'l langs

Cooperative Data Sharing (CDS): Efficient runtime support for portable threads and communication between them, on variety of architectures

People, Instruments, Computers, and Archives (PICA): Rules and protocols for finding, bargaining for, and scheduling distributed, independently-controlled resources of all kinds

CDS: Efficient Data/Thread Binding



Trad'l threads

Pros:
Low overhead

Cons:
All data must be held in common or shared memory

Msg passing

Pros:
Latency hiding
Queuing
Data translation

Cons:
Copy overhead always suffered

DSM

Pros:
No extra copy

Cons:
No latency hdg
No queuing
No data trans
Mem mgmt

CDS

Pros:
Latency hiding
Queuing
Data translation
No extra copy
No mem mgmt

CDS (Cooperative Data Sharing) blends the semantics & advantages of message passing and distributed shared memory, includes support for process control, active messages, conversion/marshalling.

Comparing CDS Featureset

Features	CDS	DSM	MPI	SOCK	LINDA
Some data can be traded/shared in place (true 0 copy!)	x	x			
Consumer can pull (get) data from passive producer	x	x	2		x
Consumer can prefetch/prepull data to hide latency	x	?	2		
Producer can push (send) data to passive consumer	x		x	x	?
Data can be queued at producer waiting for pull	x		x	x	?
Pushed data can be made to overwrite previous value	x	x			x
Producer can retain access rights to comm'd data	x		2		x
Producer can relinq access rights to comm'd data	x	x	x		x
Dynamic memory allocation for shared memory	x	?			
Consumer can specify timeout for waiting	x	?			
Supports heterogeneous platforms	x		x		
Simplicity (~number of function + macro interfaces)	51	20	!!!	13	5

The CDS Interface

Managing comm heap and contexts/cells

`rgalloc rgmod rgfree rgsize rgrealloc`

`addcntxt delcntxt grwcntxt`

Communication Primitives

`read deq benq enq write zap enqm writem`

`iread ideq ibenq wait waitm ienqm benqm`

Copying and Translation

`copyto copyfm copytofm transtab`

Composite functions (shared mem and msg passing)

`recv bsend recvx send sendx sendm sendxm`

`acqrl acqwl rlsrl rlswl wl2rl`

`irecv ibsend irecvx iacqrl iacqwl`

Process and thread control

`enlist init myinfo hdlr prior`

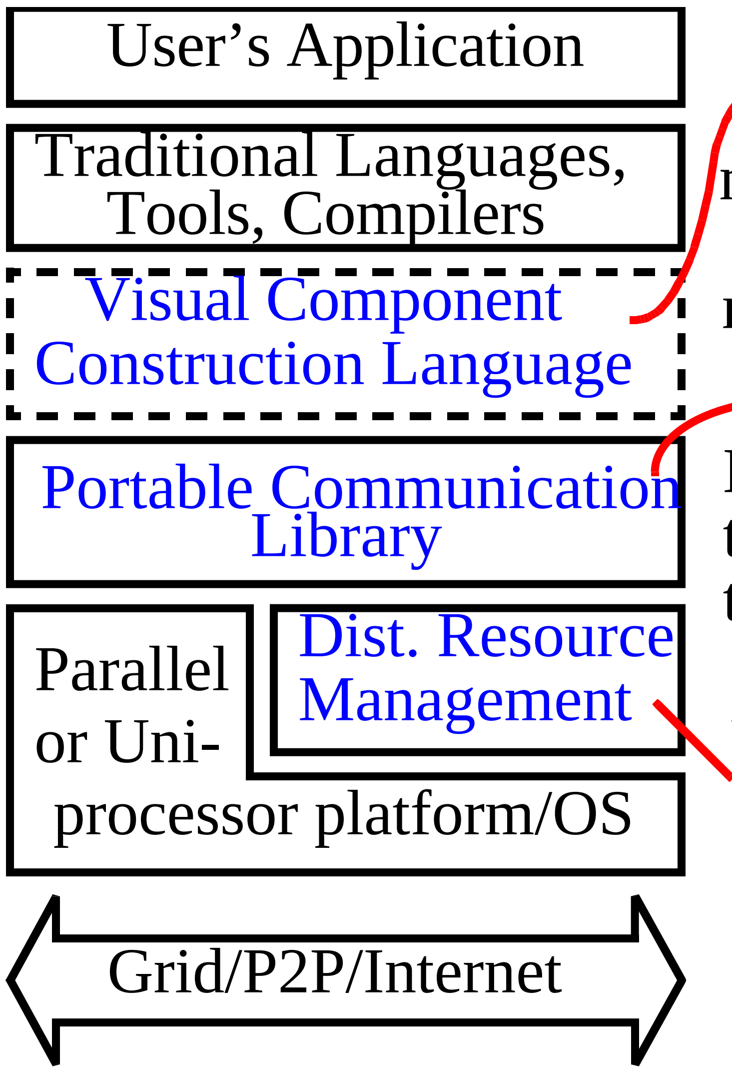
CDS As General-Purpose API

CDS offers very general concurrent programming support, addressing many current challenges in parallel and distributed computing—e.g.

- **Programming heterogeneous architectures (e.g. clusters of SMPs)**
- **Making applications more portable between distributed and/or shared memory and/or uniprocessor architectures**
- **Providing a much simpler programming interface than MPI-2 while offering similar (or greater, in some cases) functionality**
- **Providing a common API capable of leveraging the power of newer transport protocols like VIA and InfiniBand**

CDS is currently at prototype stage within Elepar, built upon SysV shared memory segments, UDP/IP, and custom locking protocols

Elepar's Three Layers



Software Cabling (SC): Visual OO component coordination methodology for building & analyzing "independent thread" apps from modules implemented in trad'l langs

Cooperative Data Sharing (CDS): Efficient runtime support for portable threads and communication between them, on variety of architectures

People, Instruments, Computers, and Archives (PICA): Rules and protocols for finding, bargaining for, and scheduling distributed, independently-controlled resources of all kinds

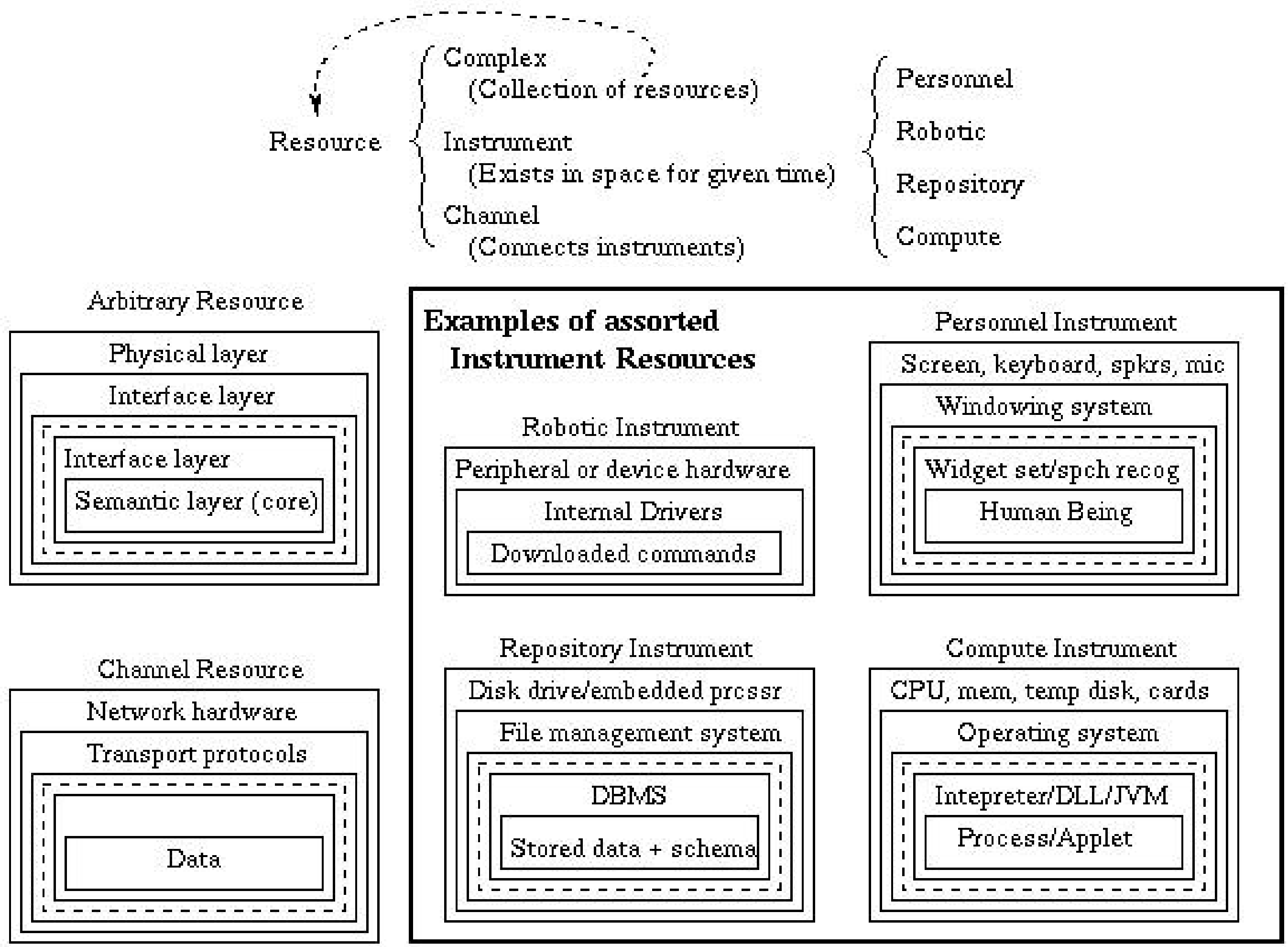
People, Instruments, Computers, Archives

"PICA": Protocols and guidelines for distributed resource discovery, bidding, (co)scheduling and reservation, and usage/relinquishment

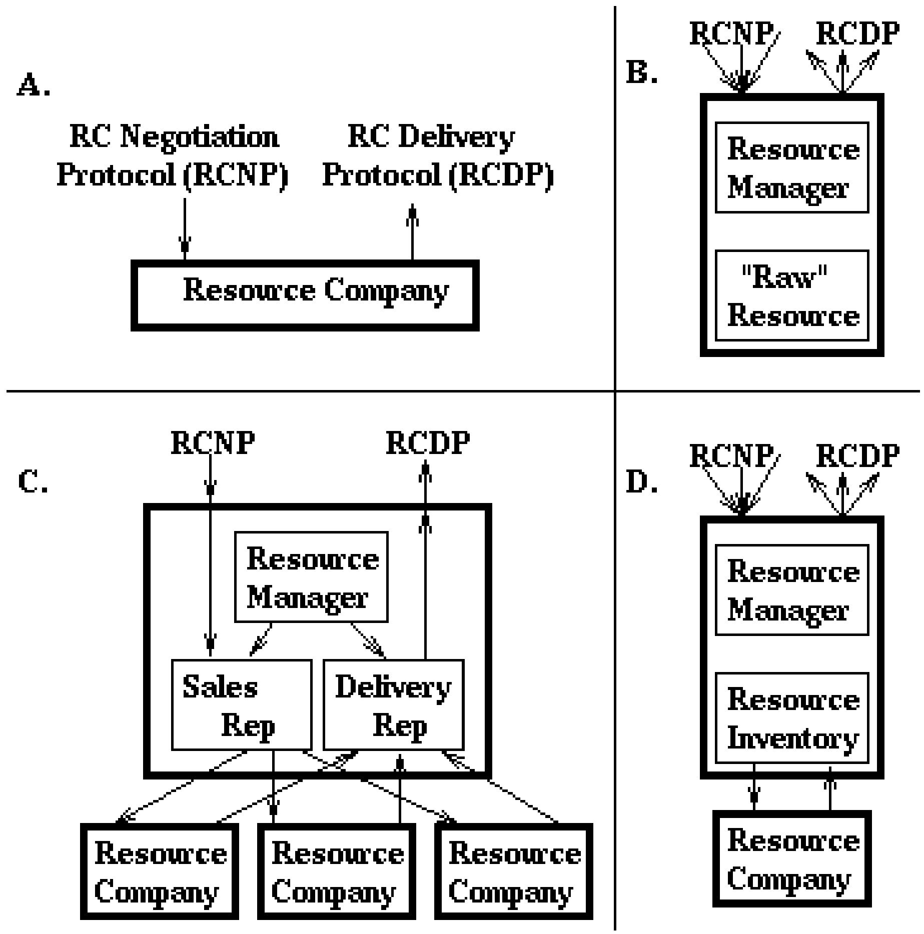
Four principle components:

- Resources: Standard way to specify complex resources
- Resource Companies: Standard way to request, bid for, and/or provide resources
- Resource Keys (i.e. capabilities): How resources are passed from place to place
- Resource Supply Chains: Fan-in/Fan-out of complex resources and payments between suppliers and customers

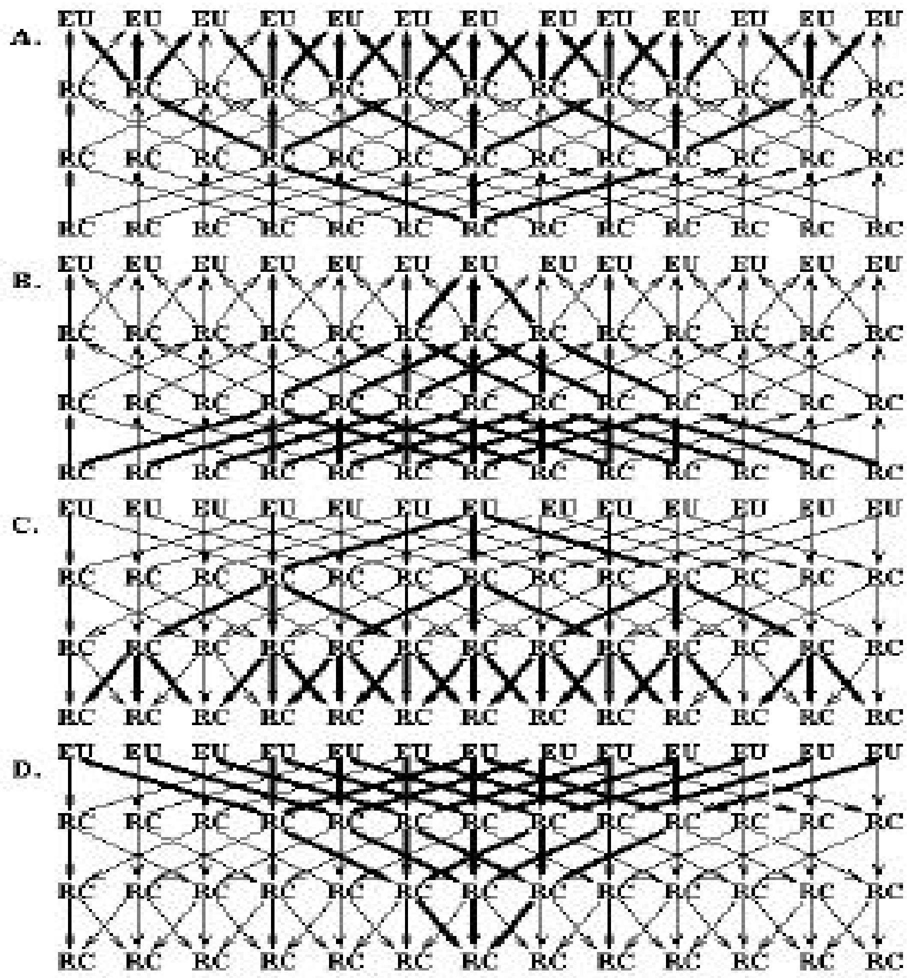
PICA: Resources



PICA: Resource Companies



PICA: Supply Chain



Companies (Profit & Non-)

File Sharing: Freenet, Napster , Gnutella, Publius, Mojo Nation, Buylink, Free Haven, Y aga, Enfi sh, TRUEDISK, CENTERSPAN Scour

Streaming, W ebcast: OpenCola Swarmcast, Allcast, CENTERSPAN c-star , Chaincast

Distributed Computing: Data Synapse, Entropia, SETI@Home, Computepower (Raj), Parabon, Popular Computing, United Devices

IM: AIMster , Jabber

Collaboration: Groove, Endeavor/Magi, Engenia, WorldStreet, Consilient

Distributed Search: Jibe, NextPage, OpenCola Folders

Infrastructure: Elepar , (Sun) JXT A, Avika, Gridworks

Local Involvement

Intel (P2PWG, TeraGrid)

IBM (TeraGrid, petaflops, Linux scalability)

TrueDisk (Data sharing/accessibility)

Centerspan (Content delivery, streaming)

Elepar (Portable parallel/distributed computing tools)

Open Source Development Lab (Linux scalability)

OGI (Multi-modal communication, distributed & heterogeneous database & digital library, InfoPipes QoS)

Driving Users/Groups

Global Grid Forum (www.gridforum.org)

- **NASA Information Power Grid**
- **NSF PACI (NCSA@UIUC + NPACI@UCSD)**
- **Distributed Tera Grid Facility (above+)**
- **GriPhyN, Particle Physics Data Grid (PPDG)**
- **DOE Science Grid & DisCom^2 (Distance and Distributed Computing and Communication)**
- **EuroGrid**
- **German Federal Ministry E&R (BMBF) Uniform Access to Computing Resources (UNICORE)**

New Productivity Initiative (www.newproductivity.org)

- **HP+Compaq, Platform Computing, Cadence, SGI, Blackstone Tech Grp, CLRC, Neolinear, Aurema, Teraport**

Driving Users/Groups (cont'd)

Peer-to-Peer Working Group (www.p2pwg.org)

- **Members include: Avaki, CenterSpan, Consilient, Data Synapse, Endeavors, Engenia, Entropia, Fujitsu PC, Groove, HP, Hitachi, Intel, NTT, OpenCola, O'Reilly, Proksim, Static, United Devices..**

Other information sources:

- **groups.yahoo.com/group/decentralization/**
- **www.openp2p.com (aka www.oreillynet.com/p2p/)**
- **www.peerintelligence.com**
- **www.peertal.com**
- **www.nsf.gov/od/lpa/news/press/01/pr0167.htm**
- **www.nytimes.com/2001/08/02/technology/02BLUE.html**

Summary

- **Tech building for decades, exploding worldwide in last few years**
- **Peer-to-peer & Grids will converge, and challenges are similar, but different focii for now**
- **Elepar is taking a "compupacket" approach to computing**
- **Many organizations like Peer-to-peer WG, Global Grid Forum, companies, working on the multitude of challenges**
- **One of those problems: When is decentralization superior?**
- **Another: What kind of economy can fuel this work?**
- **Cooperative Data Sharing (CDS) powerful enough to use in place of MPI, DSM**
- **Software Cabling (SC) is high-level module construction language a la UML**