A Scalable Ethernet Clos-Switch

Norbert Eicker

John von Neumann-Institute for Computing

Research Centre Jülich

Technisches Seminar – Desy Zeuthen

9.5.2006

Outline

- Motivation
- Clos-Switches
- Ethernet Crossbar Switches – and their problems
  - Broadcasts
  - Spanning Trees
  - VLANs
  - Multiple Spanning Trees
  - Crossbar Routing
- Implementation within ALiCEnext
- Results
- Summary / Outlook on further work in Jülich

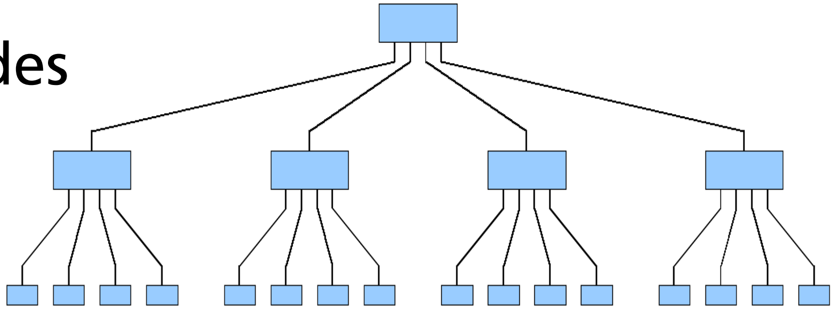
Ideal Network

- Low latency
  - Runtime of short messages
  - Most critical parameter for tightly coupled problems
- High bandwidth
  - Determines runtime of long messages
- Flat topology
  - Two arbitrary nodes can "see" each other
  - Equal distance between two arbitrary nodes
- No contention
  - No bottleneck within the network even for arbitrary patterns
  - Full bi-sectional bandwidth

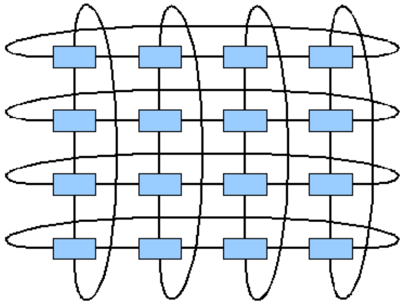
$\longrightarrow$ True Crossbar Switch

Real Networks

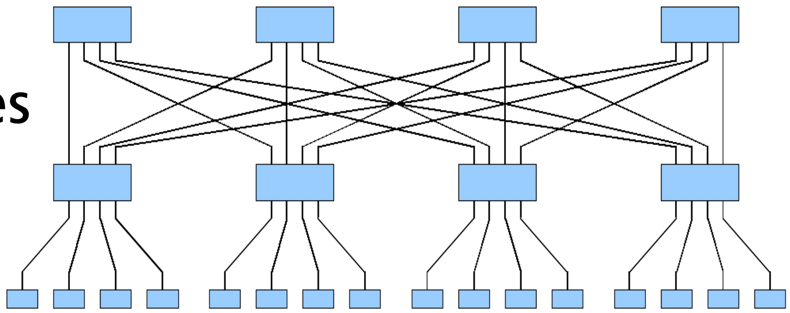
- Fat Tree
  - Communication between arbitrary nodes
  - Varying distances
  - Bottlenecks



- Torus / Mesh
  - Only nearest neighbor communication
  - Cut through routing possible
  - Scalable for adequate applications



- Clos / ω-Network
  - Communication between arbitrary nodes
  - Varying distances
  - No contention – at least in principle

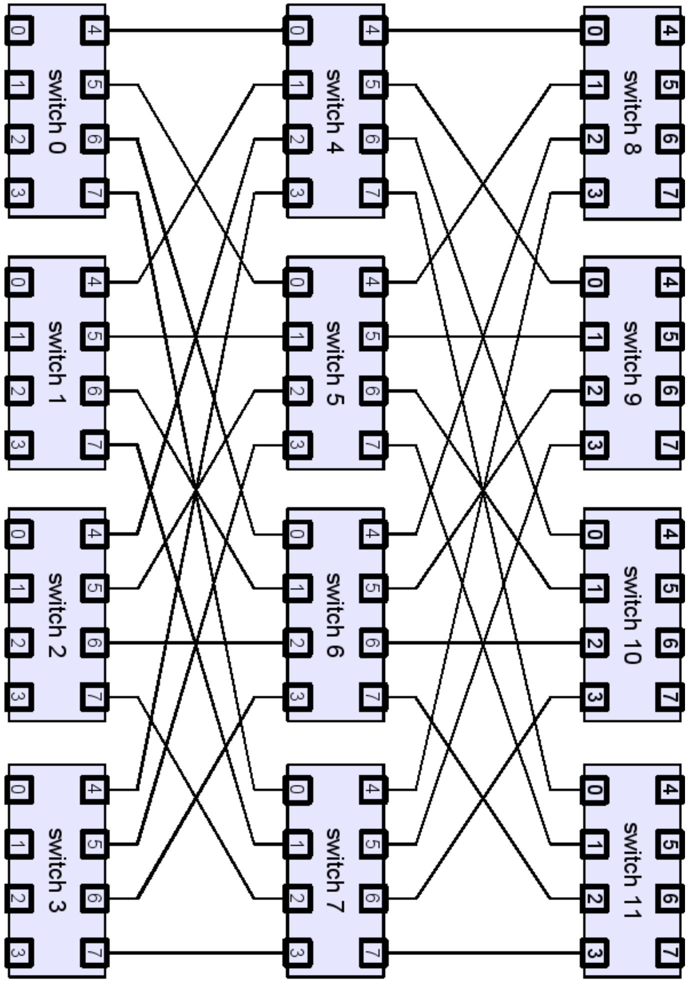


Clos Switches

- Charles Clos in 1953
- Telephony networks
- Redundant
- Scalable
- Full bi-sectional bandwidth

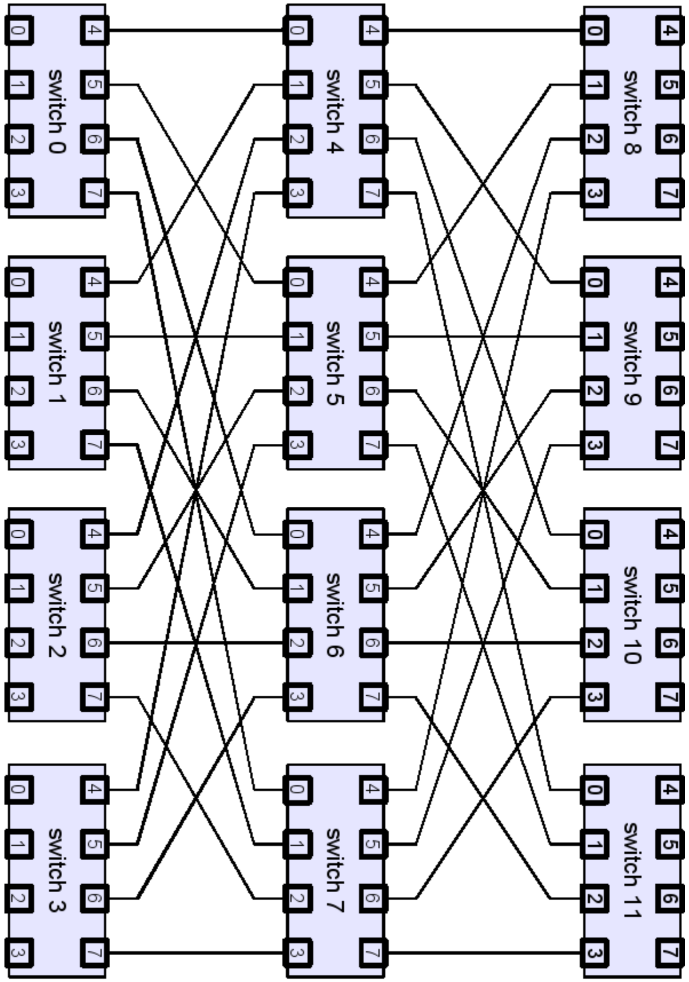
  (at least in principle)

- “Standard architecture” for commercial high-performance networks
  - Myrinet, Quadrics, InfiniBand



Contention

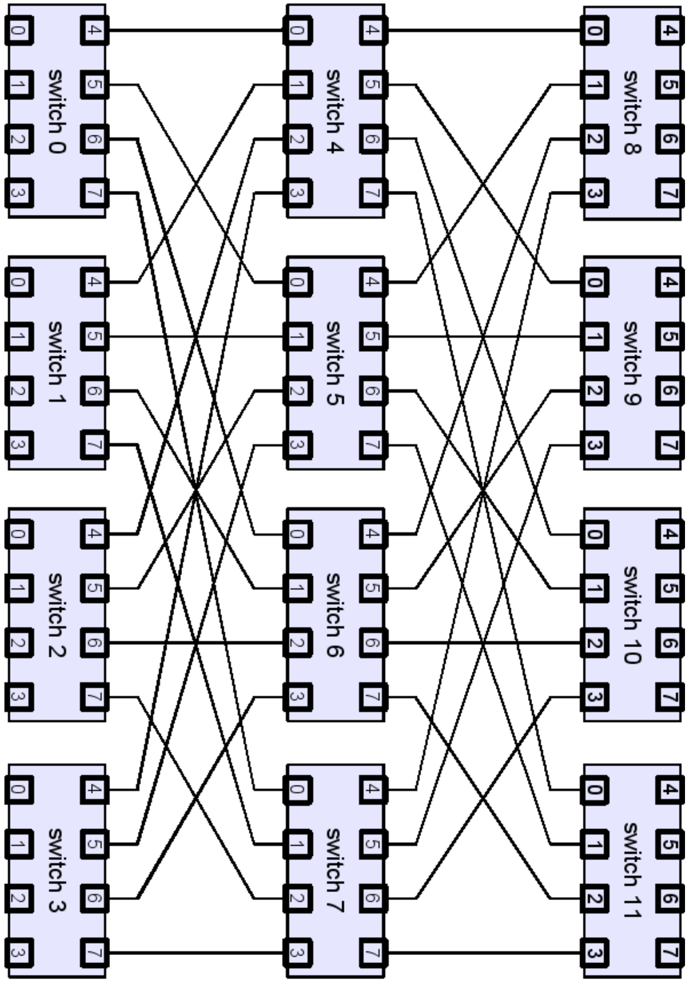
- Clos-Switches not collision / contention free
  - Typical systems use static source- or destination routing
  - Easy to construct colliding patterns
- Possible ways out:
  - More switches in middle layer
    - doubling the number prohibits contention
  - Traffic dependent routing
    - hard to implement
  - Random patterns
    - might reduce probability



Ethernet Clos Switches

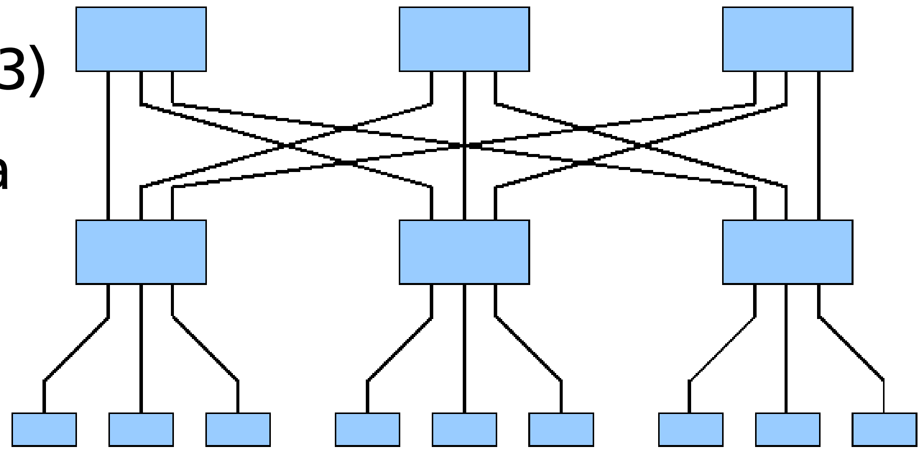
- Motivation:

  Find network for ALiCEnext
  - Limited budget
  - Mix of jobs
- Cascaded Ethernet (Fat Tree)
  - Bandwidth bottlenecks
- Mesh network
  - Only special applications
  - Actually there for QCD
- Ethernet Clos Switch
  - Let's see ...



Ethernet Clos Switches

- Let's start naively
  - Take some nodes (in our case 9)
  - Take some switches (6: 2 layers of 3)
  - Cable them according to Clos' idea
  - See what happening



- First impression: Success
  - All nodes see each other
  - Switches are still accessible
  - Bandwidth test with single pairs give expected numbers
- But

Test with more pairs show unexpected bottlenecks!

Address Request Protocol (ARP)

- Nodes at first only know IP addresses
- Ethernet switches (and NICs) only handle MAC addresses
- Address Request Protocol (ARP) RFC 826
- Ethernet address resolution
  - Requester sends broadcast message:

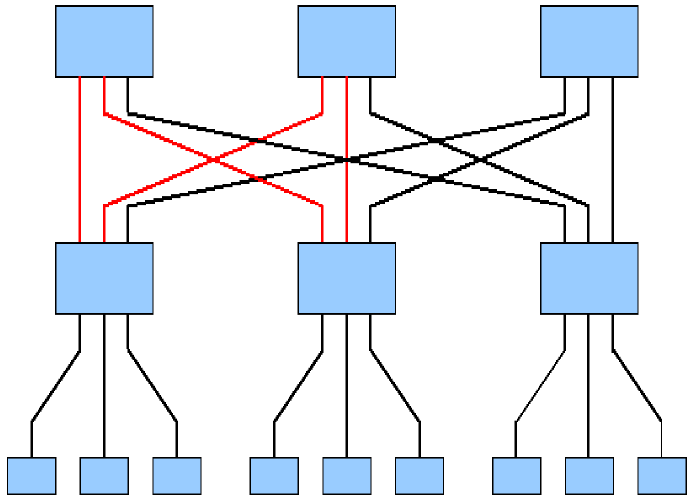
    Who knows IP a.b.c.d ? ARP request
  - Node with correct IP replies via broadcast message:

    My IP / MAC is a.b.c.d / 12-34-56-78-90-ab ARP reply
  - Store known addresses within ARP-table (cache with TTL)
  - Maybe listen to ARP traffic between other nodes

- ARP uses Ethernet Broadcasts

Spanning Trees

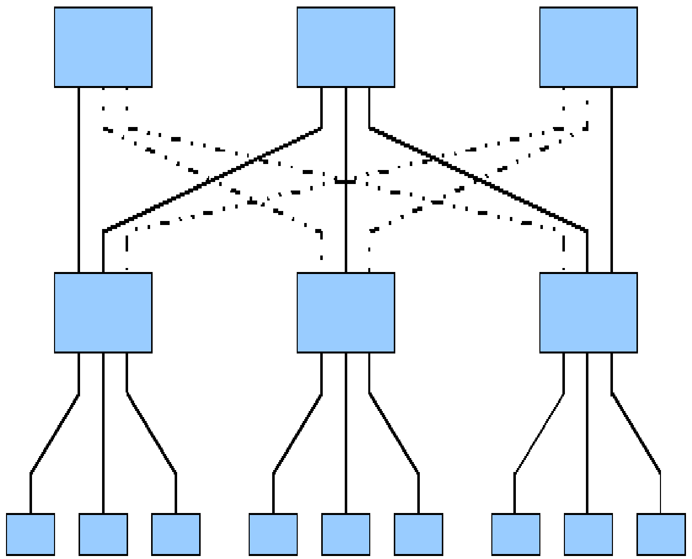
- Ethernet broadcast w/o TTL
- Loops generate packet storms



- Broadcasts inevitable for IP over Ethernet (ARP)

  → There should be packet storms

- Spanning trees (IEEE 802.1D)
- Creates robustness against unwanted loops :-)
- Eliminates all additional connectivity :-(



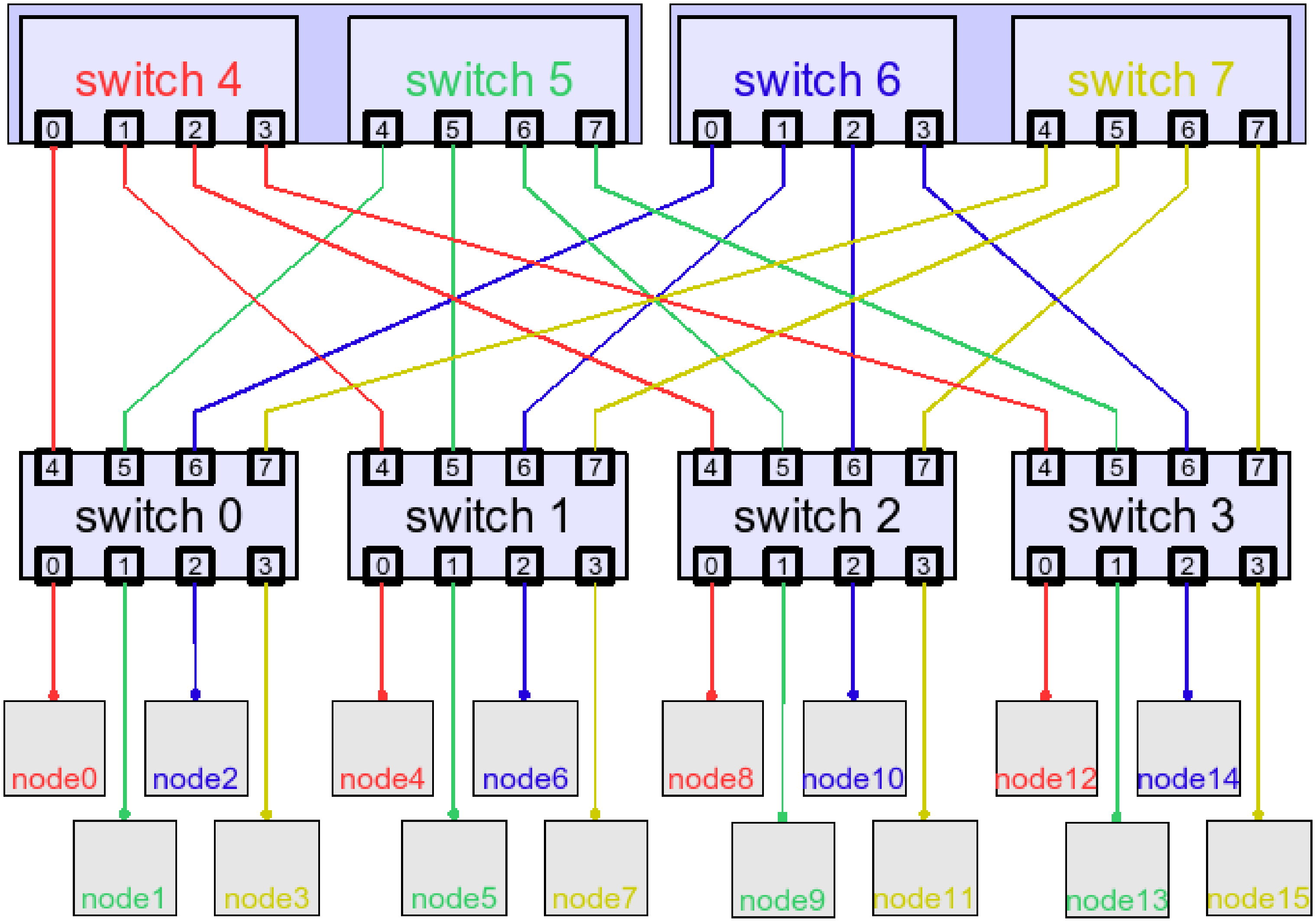
VLAN

- Virtual LAN (VLAN, IEEE 802.1Q, IEEE 802.3ac)
  - Segment physical network into virtually disjunct parts
  - Segments might overlap
- Yet another layer of indirection
  - Wrap Ethernet frames into VLAN container
  - In practice low (almost no) overhead
  - Extremely useful to manage department networks
  - Wide availability in medium sized commodity switches
- Idea:
  - VLANs might be used in order to hide loops
  - Create various VLANs, each forming a spanning tree

VLAN

- Switches support different types of VLAN links:
  - Tagged:
    - Packets delivered are explicitly tagged, i.e. wrapped into VLAN header
    - Inbound traffic expected to contain
  - Untagged
    - VLAN header discarded when packet is delivered
    - Inbound traffic is plain Ethernet – might be tagged automatically
- Our concept:
  - Don't touch the nodes configuration
  - Make Crossbar as transparent as possible
- Node-ports are untagged and belong to all VLANs
- Inbound traffic mapped into VLAN (depending on port #)

Spanning VLANs



Multiple Spanning Trees

- Each single VLAN might contain loop
- Every VLAN needs its own spanning tree
- Multiple spanning trees (MST, IEEE 802.1s)

- On startup / change of topology
  - Throw away old MST configuration
  - Determine network “root” via switches MAC address
  - “root” sends test packets
  - Each switch forwards (updated) test packets on all ports
  - First received packet determines route to root
  - Switch off all alternative routes
  - Do this for all configured VLANs

Multiple Spanning Trees

- We tested the implementation on SMC 8648T
- Works for 3x3 setup
- Test with bigger (24x6 nodes / 6+4 switches) setup shows
  - Not robust enough for our (quite unusual) setup
  - Kind of packets storms totally occupy fabric
  - Switches totally lock up
  - Network becomes useless
  - (Impractical) tricks make it work: Plug one cable after the other
- MST has to be switched off: Be aware of loops!
  - Less tolerant against cabling / configuration errors
  - Robust enough in production (we almost never replug cables)

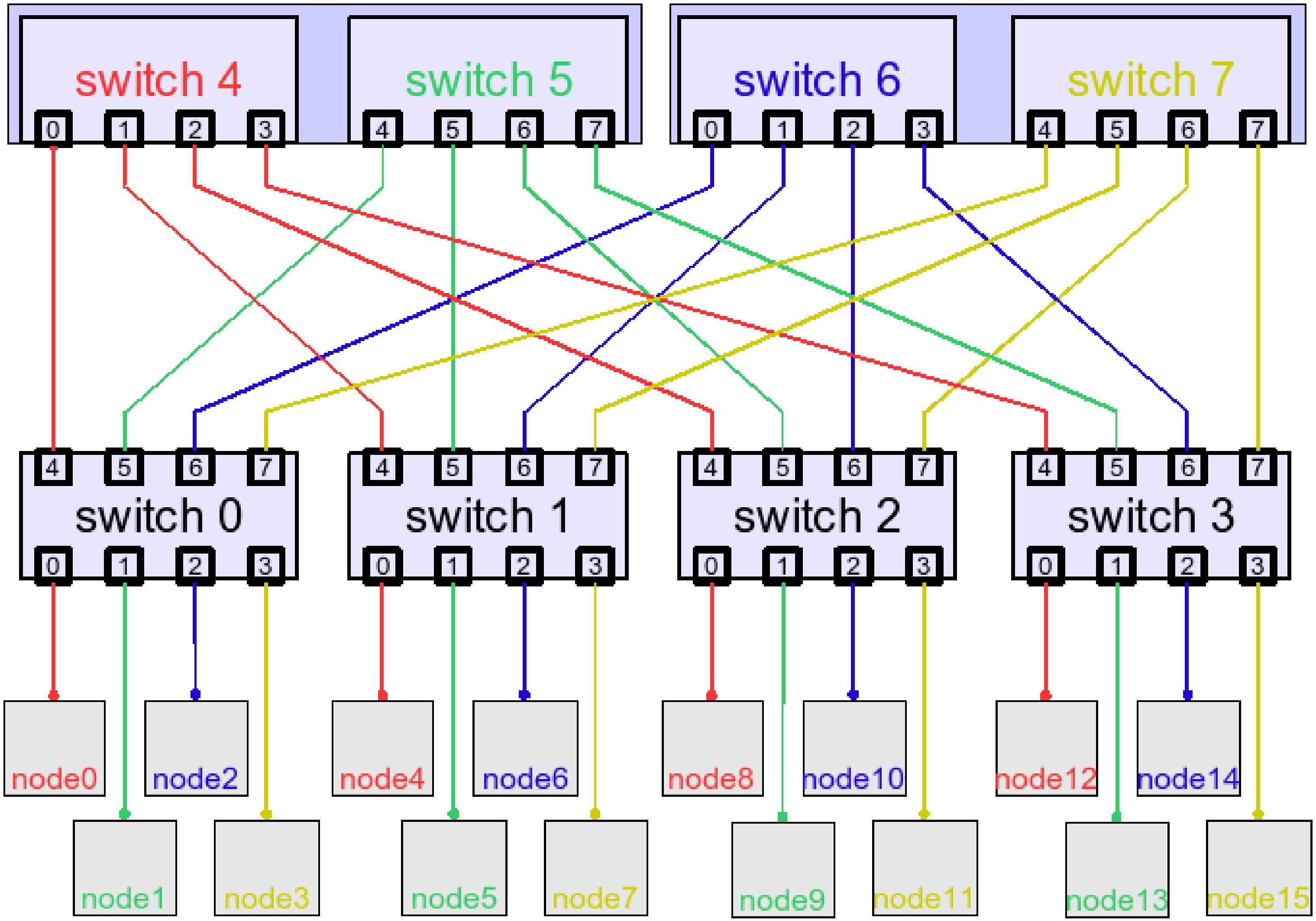
Routing

- Next test: Setup works for some patterns, but:

  Tests for most patterns show unexpected bottlenecks!

- We can understand this
  - Sending node designates employed VLAN
  - Problem: Nodes only visible in one VLAN
  - Learning impossible for L2 switches
  - Unknown destination
    - Switches deliver in broadcast mode
    - Contention by multiple broadcasts
- Explicit routing tables suppress needless traffic
- Up to 12k routing entries per switch
  - 512 nodes x 24 VLANs

Routing



switch 4
switch 5
switch 6
switch 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
4 5 6 7
switch 0
0 1 2 3
4 5 6 7
switch 1
0 1 2 3
4 5 6 7
switch 2
0 1 2 3
4 5 6 7
switch 3
0 1 2 3
node0
node2
node4
node6
node8
node10
node12
node14
node1
node3
node5
node7
node9
node11
node13
node15

Routing

- Next test: Setup works for some patterns, but:

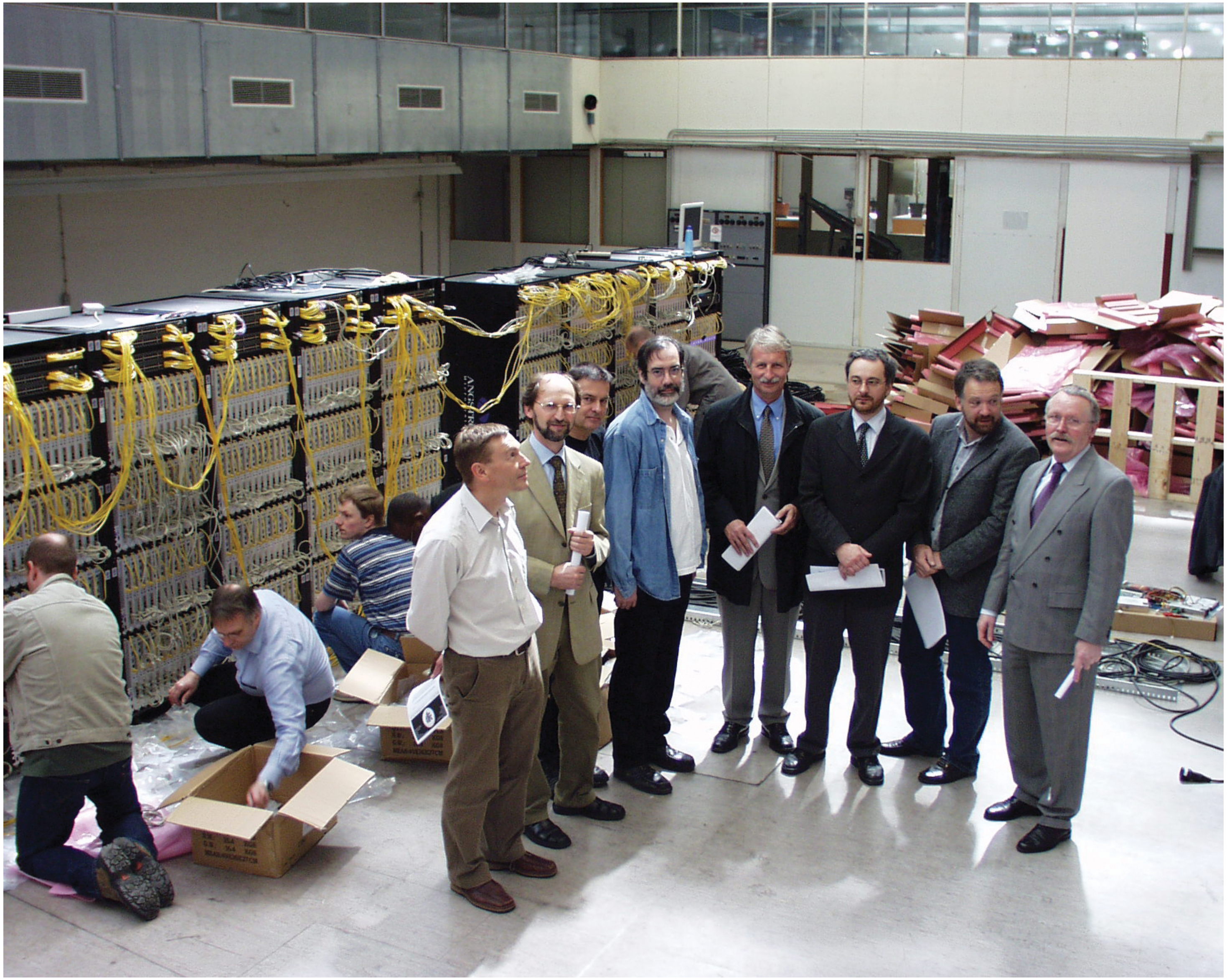
  Tests for most patterns show unexpected bottlenecks!

- We can understand this
  - Sending node designates employed VLAN
  - Problem: Nodes only visible in one VLAN
  - Learning impossible for L2 switches
  - Unknown destination
    - Switches deliver in broadcast mode
    - Contention by multiple broadcasts
- Explicit routing tables suppress needless traffic
- Up to 12k routing entries per switch
  - 512 nodes x 24 VLANs

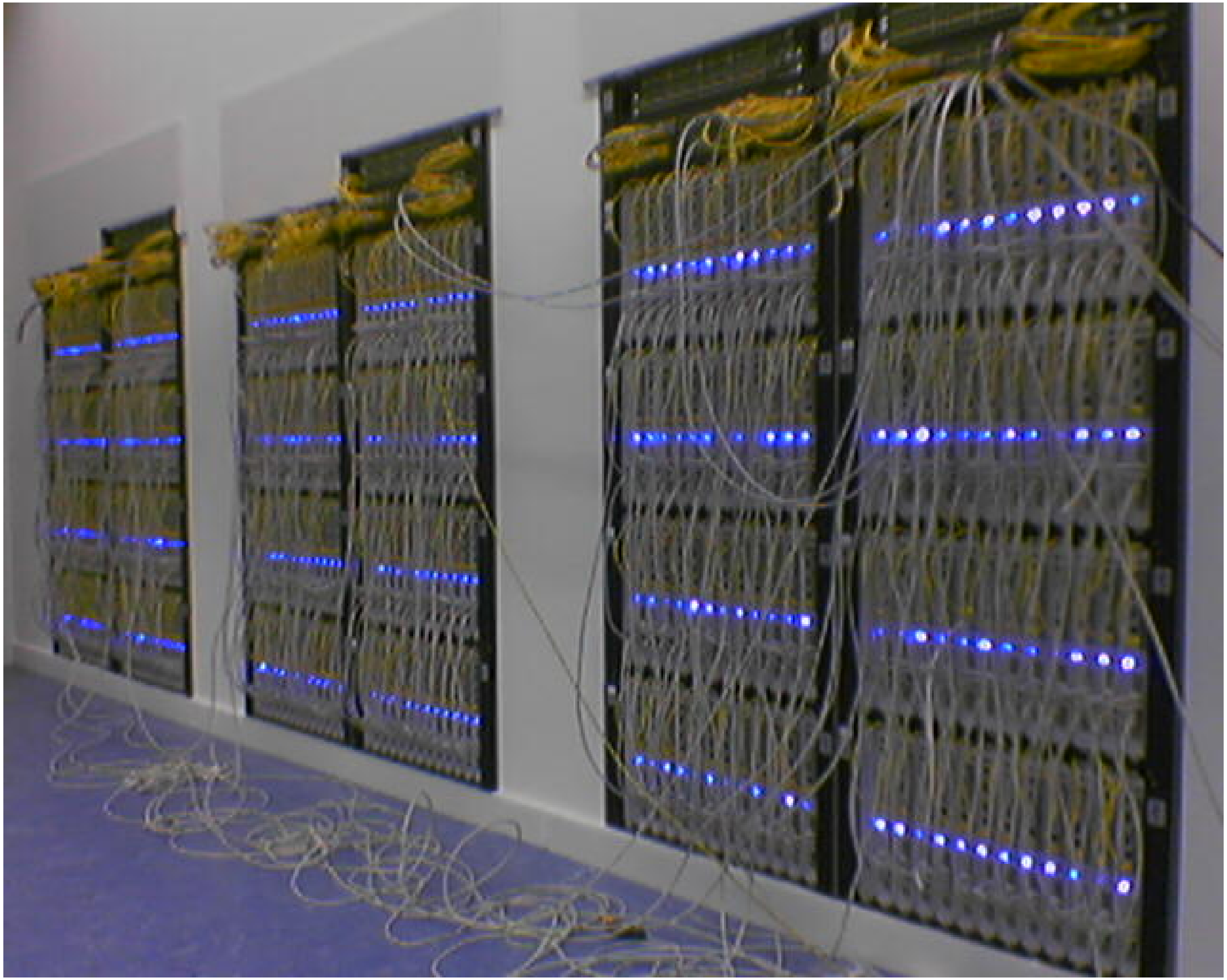
ALiCEnext

- Advanced Linux Cluster Engine - next generation
- Dual Opteron nodes
- Gigabit Ethernet network
- 512 node / 1024 CPUs
- 1 TByte memory
- ~150 TByte harddisk
- ~ 3.6 TFlops Peak performance
- Installation: June 2004

ALiCEnext



ALiCEnext



ALiCEnext Hardware

- Nodes: 512 Tyan Dual Opteron Blades
- Processors: 1.8 Ghz Opteron 244
- Cache: 1 MByte (2nd), 64k/64k (1st)
- Memory: 2 GByte/node → 1 TByte
- Disks: 320 GByte/node → 160 TByte
- Connectivity: Gigabit Ethernet
  - 2D (4x4) Torus Mesh for QCD
  - Ethernet Crossbar
- Power: 140 kW
- Price: 1.5 M€

ALiCEnext – nodes

- Tyan Dual Opteron blades
- Opteron 244 @ 1.8 GHz
- 1 MByte Level 2 Cache
- 2 GByte ECC RAM @ 3.2 GByte/sec
- 2 × 160 GByte IDE hard disk
- 1 × 64bit PCI-X slot @ 100 MHz
- 2 × Gigabit Ethernet on board (Broadcom BCM 5700)
- 1 × Quad Gigabit Ethernet card (Broadcom BCM 5700)

Switches

- SMC 8648T
- Supports:
  - VLAN
  - static routing tables on MAC level (up to 16 k)
  - STA / MST
  - Load/Store config via tftp
  - Status requests via SNMP
- Non-blocking switching architecture
- Aggr. bandwidth: 96 Gbps

- ~ € 2000

Implementation

- Full setup needs 12k entries in routing tables!
- No manual configuration possible

- Configuration & monitoring via set of scripts:
  - collect information from nodes / switches
  - consistency tests
  - generate configuration files
  - deployment into switches
  - status control
  - reconfiguration

- Last but not least: Put 1024 cables into place!

Results

- Results from prototype implementation:
  - 144 nodes
  - 6 level 1 switches
  - 4 level 2 switches
- Test w/ Pallas MPI Benchmark (sendrecv & pingpong)
- Building blocks:
  - 2 nodes back-to-back: 214.3 MB/sec 18.6 μsec
  - 2 nodes with switch: 210.2 MB/sec 21.5 μsec
- Almost no bandwidth loss
- ~3 μsec latency per switch stage
- Low Ethernet latency due to ***ParaStation*** middleware

Results

- Crossbar with 3 stages: Expect ~9 µsec total latency
- No bandwidth loss, even if all nodes communicate
- Actual test:
  - 140 processes / 70 pairs
  - Comm. partners connected to different level 1 switches
  - All traffic through level 2 switches
- Results per pair (worst): 210.4 MB/sec 28.0 µsec
- Average throughput ~5% more
- Total throughput: > 15 GB/sec
- Bi-sectional bandwidth!

Summary

- Ethernet Clos switches are possible
- Doable with components of the shelf
- Full bi-sectional bandwidth

- Total bandwidth of 144 port prototype: > 15 GB/sec
- Observed switch latency of prototype: 9.4 µsec

- Expected bandwidth full system (528 port): > 50 GB/sec
- No change in latency expected

- Cost of ALiCEnext Clos network < €125 / port

- Patent pending (Uni Wuppertal, FZ Jülich, ParTec)

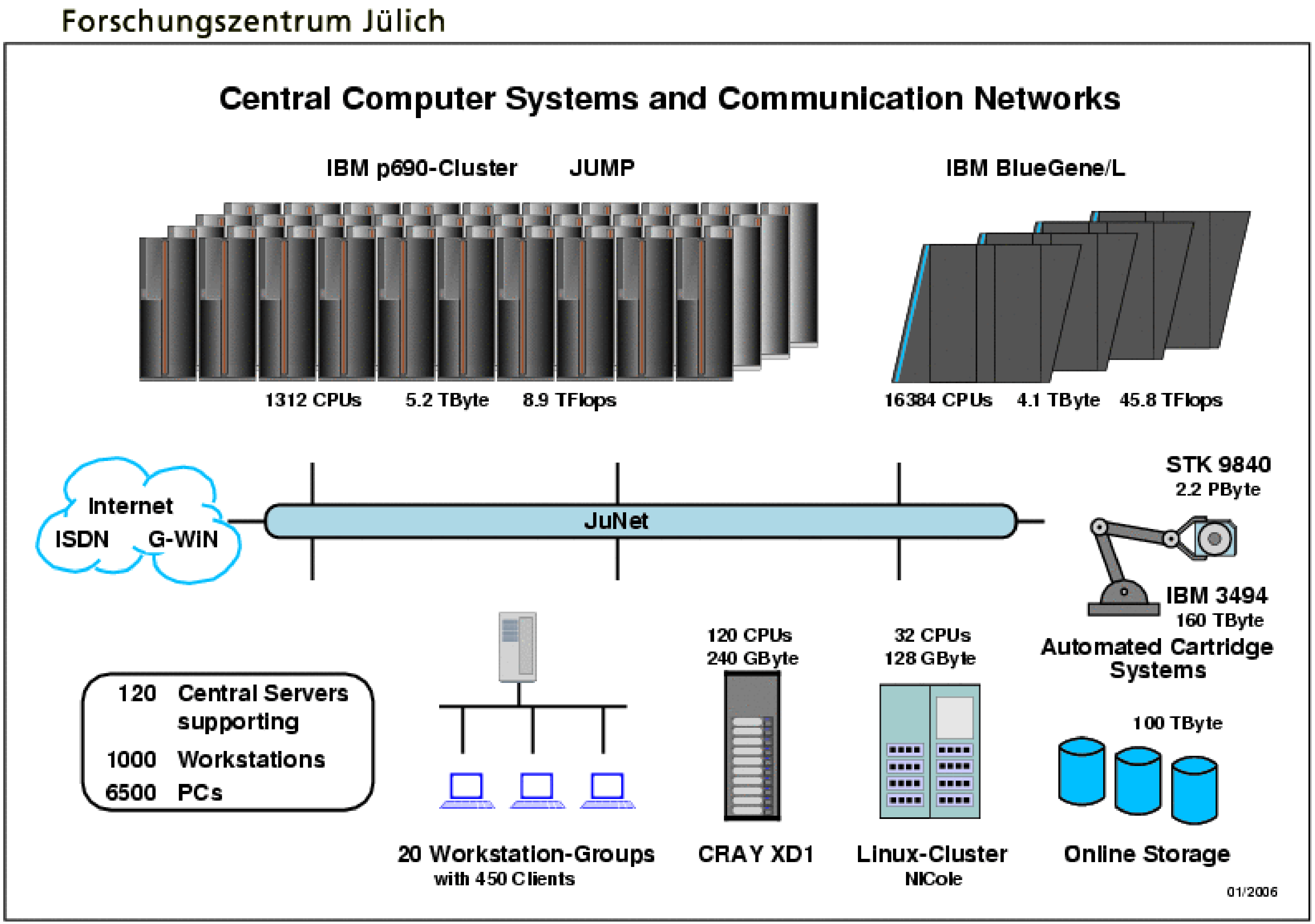
Achieved Features

- Configuration is completely transparent to end nodes
- Nodes see each other
- (Static) traffic shaping possible
- Broadcasts work (and are restricted to a single VLAN)
- ARP-requests work (for nodes)
  - Request send (as broadcast) via the requester's VLAN
  - Answer send (as broadcast) via the responder's VLAN
- Even multicasts work
  - used for installation of nodes via systemimager / flamethrower

Future (Cluster) Plans in Jülich

Project JuLi

Infrastructure Jülich



Infrastructure Jülich

- JUMP: IBM p690, 1312 Power4 @ 1.7GHz,
  41x32 SMP, 8.9 TFlop/s, 5.2 TB Memory,
  Federation Switch: 5.5 µs Latency, 1.4 GB/s, AIX 5.2,
- JUBL: IBM BlueGene 16384 PowerPC440 @ 700MHz,
  8x(2x16)x32, 45.8 TFlop/s, 4.1 TB Memory,
  3D-Torus, µK/Linux, SLES9
- Cray XD1: 120 AMD Opteron 248 2.2 GHz, 12 FPGAs,
  60x2 SMP, 528 Gflop/s, 240 GB Memory,
  RapidArray: fat-tree, 2.2 µs Latency, 2.3 GB/s, SLES9
- NICole: 32 AMD Opteron 250 @ 2.4 GHz,
  16x2 SMP, 153.6 GFlop/s, 128 GB Memory,
  GigE, SuSE 9.3

Major Upgrade End 2007

- Jülich's main idea: Two systems instead of one
  - Capability system
    - Only for few selected groups with very demanding needs
    - Applications have to (be made) fit on the machine
    - Very scalable system
    - Most probably BlueGene like
  - Capacity system
    - Open for all users
    - Less scalable applications
    - Applications should run out of the box (more or less)
    - More general purpose
    - Optimal integration into existing environment
    - Most probably Cluster like

Prototype JuLi

- Philosophy: Build Clusters from "best" components:
  - CPU → Power Platform (PowerPC 970)
  - Interconnect → InfiniPath ("Better InfiniBand")
  - Middleware → ParaStation
- Actual Hardware
  - 56 x IBM JS21 PowerPC Blades
    - 2 x Dual-Core PowerPC 970 (2.5 GHz)
    - 2 GB SASDDR2 SDRAM, 36 GByte SAS HD
  - BladeCenter H Chassis
  - PathScale Infinipath HS-DC
  - Voltaire ISR 9096 InfiniBand Switch
  - DS4100 Storage Server with 14 x 400GB Disks (= 5.6 Tbytes)
- Aspired Delivery July 2006 (right after ISC)

JuLi Plans

- Test of production mode
  - Selected users / applications
  - Stability tests
  - Feasibility of maintenance
- Integration tests
  - Interoperability with JuMP, JuBL
  - How do Clusters fit into our storage environment?
- Test of alternative components
  - Storage
    - GPFS vs. TerraScale vs. Lustre (vs. PVFS ?)
  - Batch system
    - LoadLeveler vs. Torque (vs. SUN GridEngine ?)

Thank you