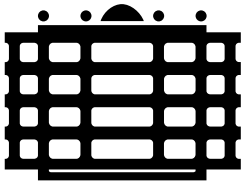

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-06-06

Low Overhead Ethernet Communication for Open MPI on Linux Clusters

Torsten Hoefler · Mirko Reinhardt · Torsten Mehlan ·
Frank Mietke · Wolfgang Rehm

Juli 2006

Chemnitzer Informatik-Berichte

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz

Low Overhead Ethernet Communication for Open MPI on Linux Clusters

Torsten Hoefler Mirko Reinhardt Torsten Mehlan Frank Mietke
Wolfgang Rehm

Technical University of Chemnitz,
Department of Computer Science,
09117 Chemnitz, GERMANY
{htor,rmir,tome,mief,rehm}@cs.tu-chemnitz.de

July 20, 2006

Abstract

This paper describes the basic concepts of our solution to improve the performance of Ethernet Communication on a Linux Cluster environment by introducing Reliable Low Latency Ethernet Sockets. We show that about 25% of the socket latency can be saved by using our simplified protocol. Especially, we put emphasis on demonstrating that this performance benefit is able to speed up the MPI level communication. Therefore we have developed a new BTL component for Open MPI, an open source MPI-2 implementation which offers with its Modular Component Architecture a nearly ideal environment to implement our changes. Microbenchmarks of MPI collective and Point-to-Point operations were performed. We see a performance improvement of 8% to 16% for LU and SP implementations of the NAS parallel benchmark suite which spends a significant amount of time in the MPI. Practical application tests with Abinit, an electronic structure calculation program, show that the runtime of be nearly halved on a 4 node system. Thus we show evidence that our new Ethernet communication protocol is able to increase the speedup of parallel applications considerably.

1 Introduction

The recent Top 500 List, released in November 2005, states that the Ethernet Network [9] is still a widely used interconnect architecture. The largest share of nearly 50% of all listed systems are connected with Gigabit Ethernet. Even though this network was not designed

as a High Performance Computing (HPC) interconnect, positive factors like availability, robustness and the price make it to a reasonable alternative to expensive high performance networks like InfiniBand [13].

Most Ethernet systems (including cluster computers) use the TCP/IP [15, 14] protocol suite that was designed for Wide Area Networks (WAN) for data transmission. The vast majority of Ethernet connected cluster systems are not WANs but rather concentrated in a Local Area Network (LAN). In LANs, the flexibility as well as some special features of TCP/IP are not needed and introduce an additional overhead. TCP/IP offers routing, fragmentation and reassembly, congestion control, port multiplexing, in order delivery and reliability to the user. Features like routing or a WAN optimized congestion control are simply not necessary for Ethernet only networks.

An interesting development in the Top 500 List is the spreading of Linux based systems. The percentage of Linux based systems increased dramatically over the last years and reached today's level of more than 75%. It is not surprising that many projects aim at optimizing message passing systems for Ethernet based Linux systems. The following Section lists some of these projects.

1.1 Related Work

Although several implementations that try to reduce latency and increase bandwidth for Ethernet systems are available today, many of this systems are not actively developed anymore. U-Net [16] was intended to provide user-level access to the network hardware. This approach limited the applicability to several network adapters because driver changes were necessary. The industry standard VIA with its implementation M-VIA [4] superseded U-Net. However, the limitation to specific Network Interface Cards (NICs), the neccessity of adapting the NIC drivers, and the requirement to follow each kernel version limit its availability to older kernels. Bobnet [3] aimed at providing zero copy mechanisms for message passing but a fully MPI-2.0 compliant library is not available. Gamma [2], which supports only a small number of network interfaces, used the active message approach to eliminate the use of data copies and EMP [12] modified a single NIC driver to achieve zero copy message passing. Commercial approaches proposed by SCALI and Par-Tec are not taken into consideration because their design is not open.

Each of these projects is limited to specific combinations of NIC and Linux kernel version. Most of these approaches include changes at the device driver layer in Linux. This makes it hard to follow kernel or driver updates and to perform the installation. Due to this problems, most of these projects are not active anymore.

1.2 Main Goals

The main project goals are to present a viable alternative to TCP/IP which achieves the highest performance with the simplest and most portable solution. The huge number of different Ethernet manufacturers and Chips requires a hardware-independent solution, the steady development and change of the Linux kernel enforces the smallest possible interface to the kernel, and the ease of use is ensured by the avoidance of kernel patches and the use of a kernel module.

The following section describes our Low Latency Ethernet Sockets as alternative to the traditional TCP/IP stack in detail and provides microbenchmarks of the current implementation. Section 3 describes the incorporation into the Open MPI framework; it is followed by a detailed performance comparison of the MPI implementation in Section 4. The last section concludes the research and points out further directions.

2 Reliable Low Latency Ethernet Sockets

We use a kernel module which registers the new protocol family `PF_ENET` to the kernel. It connects through the internal socket and the Virtual File System (VFS) interface to the application and through the network device deriver interface to all ethernet drivers. The general architecture is depicted in Figure 1. The kernel module works with every Ethernet NIC supported by Linux. The new protocol family offers two protocols to the socket layer, the Ethernet Datagram Protocol (EDP) and the Ethernet Streaming Protocol (ESP). EDP is like UDP that is an unreliable datagram protocol which is only used to 'ping' other hosts (for details refer to [11]). ESP offers port multiplexing, ensures reliable in order transmission, and provides segmentation and reassembly. These features are implemented in kernel space to keep the overhead low because only a single system call per message is needed. Normal Ethernet addresses are used to identify endpoints because the addressing scheme in a flat LAN network does not need to be hierarchical. All these features lead to a simplified protocol which requires only a small header compared to TCP/IP. The header hosts only necessary information for LAN communication and is compared in Figure 2 to TCP/IP. EDP uses only 13 bytes of additional header information per Ethernet Packet where TCP/IP uses 20 or 40 bytes.

The state machine of the sockets to implement channel semantics is similar to the TCP mechanism and thus not described here. The data reliability is ensured with a simple acknowledgement protocol. Each packet is marked with a sequence number and buffered at the sender to enable retransmission. Received packets are acknowledged TCP-like by sending the next expected sequence number back to the originator. This enables the

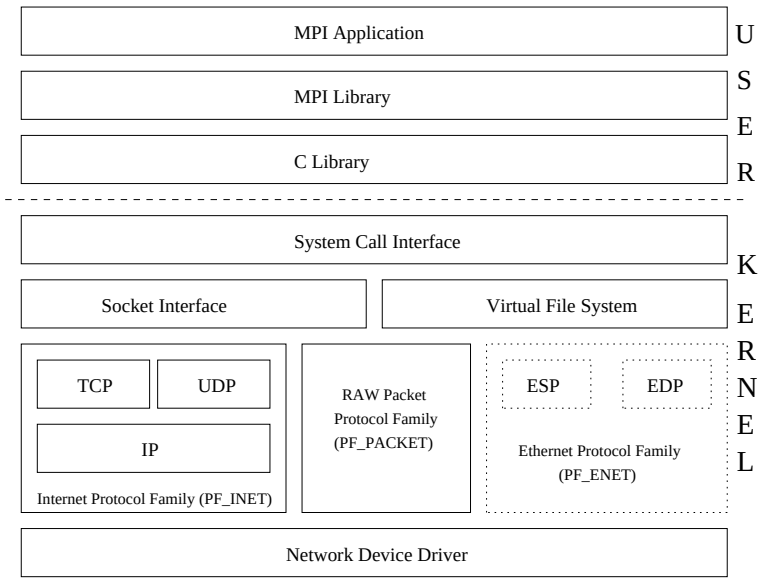

Figure 1: Kernel Architecture Overview

sender to keep track of the last successfully received in-order fragment and to flush all buffered packets with lower sequence numbers. A retransmission occurs if a single packet (identified by sequence number) cannot be deleted in a certain time period. The protocol allows backpacking of the acknowledgement number to normal packets sent by the user. It tries to wait a specified time before sending the explicit acknowledgement to enhance the possibility of backpacking. An explicit acknowledgement is sent if this time goes by without a message transmission to the target host.

2.1 Microbenchmarks

We conducted several benchmarks on two Cluster Systems. Cluster 1 (C1) consists of 4 1.4 GHz dual Athlon MP nodes with Syskonnect SK-98xx V2.0 Gigabit Ethernet NICs and Cluster 2 (C2) of 4 2.4 GHz dual Xeon nodes with Intel Corporation 82544GC Gigabit Ethernet NICs. Figure 3 shows the average single packet latencies (left image, measured without pipelining effects) and the socket blocking times (right image) compared between TCP/IP and ESP. The socket blocking time is the time that a send call to the socket needs to return. This is only a rough estimation of the CPU overhead of a network protocol. We see a latency decrease on both systems of nearly 25% for small messages and the send call blocking time could be reduced by up to 50% (see Figure 3).

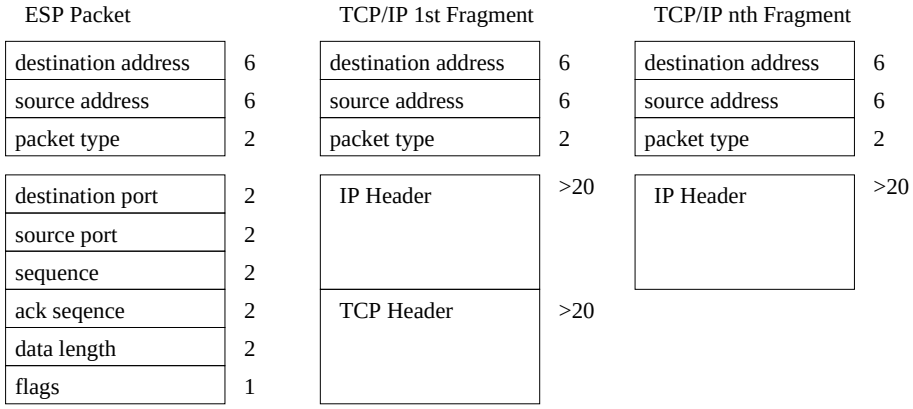

Figure 2: ESP and TCP/IP Header Comparison

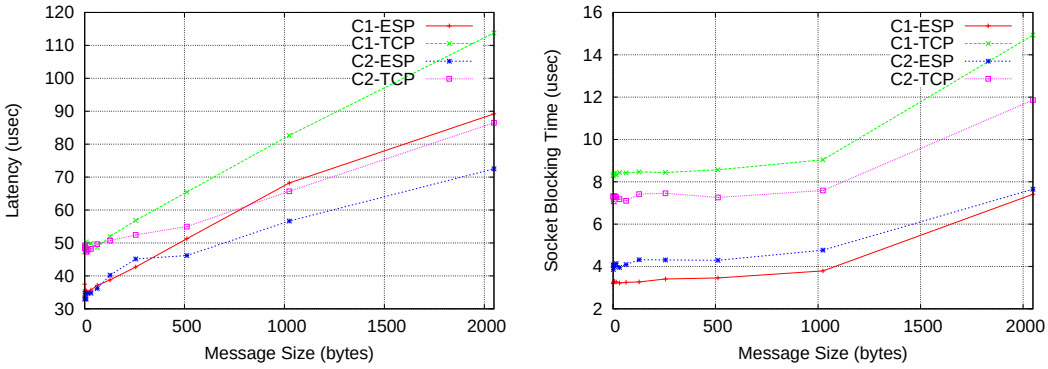

Figure 3: Single Packet Latencies and Socket blocking Times for TCP/IP and ESP

3 Implementation in Open MPI

Open MPI [5] is an open source MPI-2 implementation which offers with its Modular Component Architecture (MCA) a nearly ideal environment to implement our approach. Open MPI hosts several frameworks which define distinct tasks. A component can be implemented to perform these tasks and a runtime instance of a component is called module. Some frameworks can have many active modules at any given time (e.g. the device layer may support many interfaces simultaneously). Important parts of the architecture of Open MPI are depicted in Figure 4. We analyze the performance of two commonly used frameworks, the collective framework (COLL) to perform a collective operations and the Point-to-Point Management Layer (PML) to perform point to point communication. Both layers need to communicate data by accessing the lowest layer, the communication

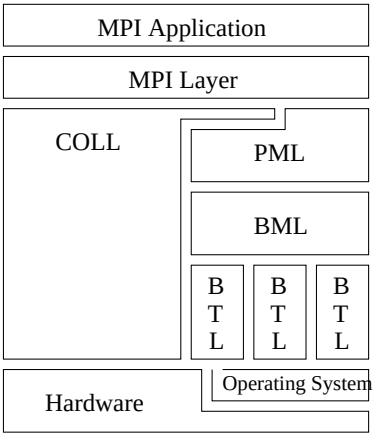

Figure 4: Open MPI Architecture (simplified)

hardware, directly or through the operating system. The coll framework is free to access the hardware directly or to use normal point to point semantics with the PML. The PML administers all low level device drivers which are implemented in a framework called Byte Transport Layer (BTL).

We implemented a BTL component called ETH to support our Low Latency Sockets. The design of our ETH BTL component is identical to the TCP BTL because we are using the same socket semantics. The only differences are that we use a special EDP ping to determine node reachability at the beginning of each run and that we use Ethernet Adresses instead of IP adresses.

4 Performance Comparison

We conducted several benchmarks to compare the performance of our approach with the standatd TCP/IP implementation. We used Microbenchmarks, the LU and SP implementations of the NAS parallel benchmark suite [1] as well as a precisely analyzed quantum mechanical application to show the performance benefits of our new protocol.

4.1 Microbenchmarks

Microbenchmarks of MPI collective and Point-to-Point operations were performed with the Pallas MPI-1 Microbenchmarks [10]. We compared the performance between the TCP and the ETH module. The environment was identical for both runs, the only difference was

"btl=eth" to enforce the ETH BTL or "btl=tcp" to enforce the TCP BTL in the MCA-Parameters. The benchmarked results on cluster 2 for PingPong, Alltoall, Allgather and Allreduce are shown in Figure 5.

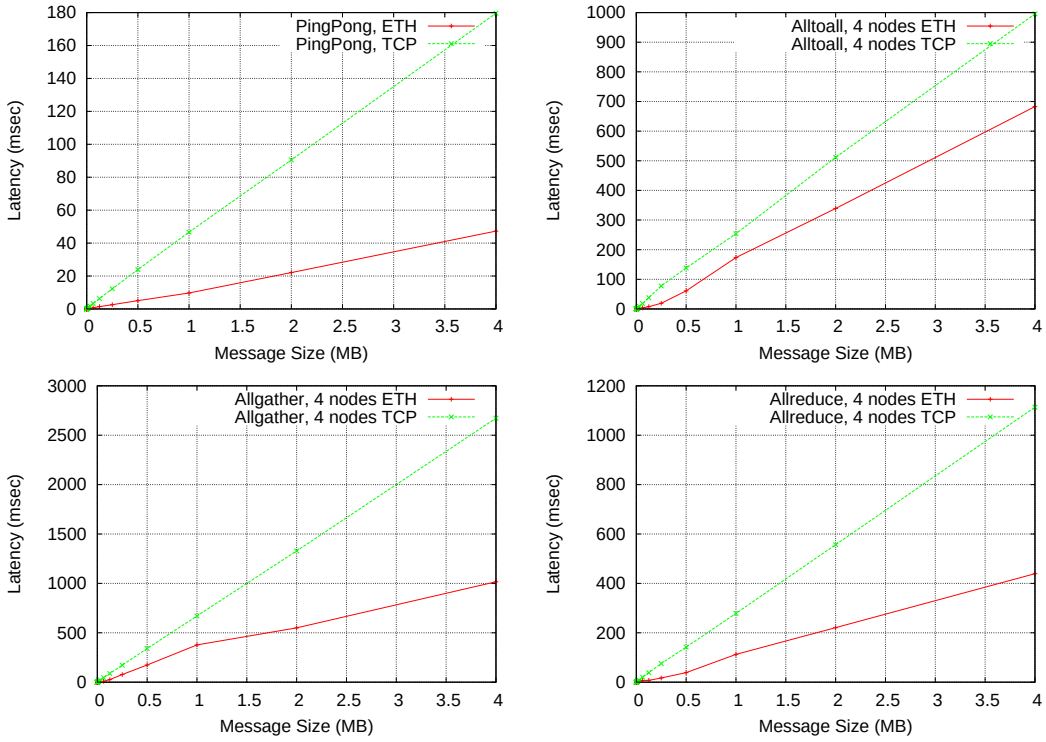

Figure 5: Pallas MPI-1 Benchmark Results

4.2 Compute Kernel Benchmarks

We used the LU and SP implementations of the NAS parallel benchmark suite as a compute kernel which spends a significant amount of time in the MPI library. LU performs a LU decomposition and sends a large number of very small messages while SP solves three sets of uncoupled systems of equations. We see a performance improvement of 8% (TCP: 1312.24 Mop/s vs. ETH: 1432.46 Mop/s) for LU and a performance improvement of 16% for SP (TCP: 606.45 Mop/s vs. ETH: 723.88 Mop/s) on 4 nodes.

4.3 Application Benchmarks

We chose the electronic structure calculation program Abinit to perform benchmarks with our new protocol. Abinit [6] has been well analyzed with regards to its parallel running time [8]. We use the new parallelization scheme introduced and analyzed in detail in [7] which performs a conjugate gradient based minimization and a 3D-FFT in parallel. The main communication operations are MPI_Allreduce of single double values (16 byte) to perform dot-products and MPI_Alltoall for the 3D-FFT. Both operations were significantly enhanced by our ETH implementation (see Figure 5). Figure 6 shows the runtime and scaling results for the calculation of a small SiN system with 14 atoms on cluster 2. The parallel runtime of Abinit could nearly be halved on a four node system.

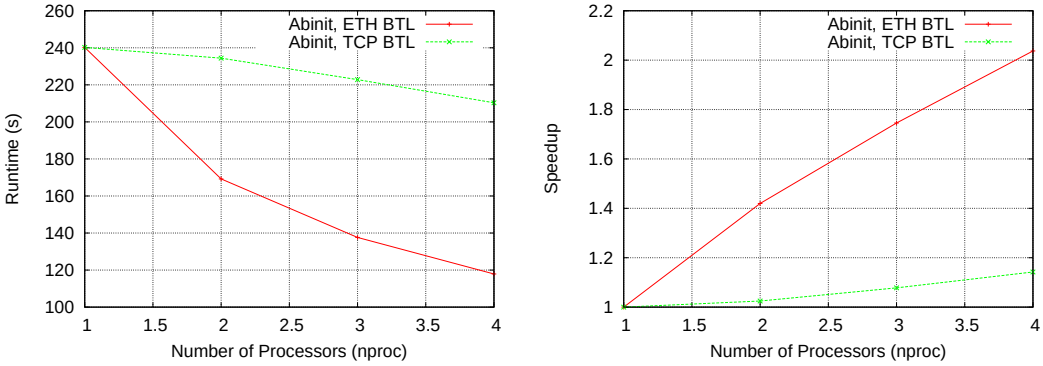

Figure 6: Runtime and Speedup of Abinit for a 14 Atom SiN system

5 Conclusion and Future Work

We show that it is possible to enhance the performance of Ethernet networks in HPC systems by simplifying the communication protocol. We show a device independent Linux kernel module to implement a reliable Ethernet communication subsystem with the socket interface. A BTL component for Open MPI enables MPI programs to use this accelerated Ethernet subsystem. The raw latency benefit is about 25% in comparison to TCP and the CPU overhead can be decresed by nearly 50%. MPI level Microbenchmarks show that the MPI point to point and collective performance can be significantly enhanced. Application tests with Abinit showed the positive influence to practical applications. The runtime of Abinit could be nearly halved on a 4 node system. Future work includes the addition of a proper flow control algorithm as we see a slight performance degradation for larger messages.

References

[1] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1991.

[2] Giovanni Chiola and Giuseppe Ciaccio. Gamma: Architecture, programming interface and preliminary benchmarking, 1996.

[3] C. Csanady and P. Wyckoff. Bobnet: Highperformance message passing for commodity networking components, 1998.

[4] D. Cameron and G. Regnier. *The Virtual Interface Architecture*, 2002.

[5] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004.

[6] X. Gonze, G.-M. Rignanese, M. Verstraete, J.-M. Beuken, Y. Pouillon, R. Caracas, F. Jollet, M. Torrent, G. Zerah, M. Mikami, P. Ghosez, M. Veithen, J.-Y. Raty, V. Olevano, F. Bruneval, L. Reining, R. Godby, G. Onida, D.R. Hamann, and D.C. Allan. A brief introduction to the ABINIT software package. *Z. Kristallogr.*, 220:558, 2005.

[7] Torsten Hoefler, Rebecca Janisch, and Wolfgang Rehm. Analyzing the parallel scaling of teter's conjugate gradient based minimization for *ab initio* calculations. In *Submitted to the Europar 2006 Conference*, 2006.

[8] Torsten Hoefler, Rebecca Janisch, and Wolfgang Rehm. A performance analysis of abinit on a cluster system. In Karl Heinz Hoffmann and Arnd Meyer, editors, *Parallel Algorithms and Cluster Computing*. Lecture Notes in Computational Science and Engineering, 2006. accepted to be published.

[9] IEEE Computer Society. *802.3 IEEE Standard for Information technology*, 2002.

[10] Pallas GmbH. Pallas MPI Benchmarks - PMB, Part MPI-1. Technical report, Pallas GmbH, 2000.

[11] Mirko Reinhardt. Optimizing Point-to-Point Ethernet Cluster Communication. Master's thesis, TU-Chemnitz, 2006.

[12] Piyush Shivam, Pete Wyckoff, and Dhabaleswar Panda. Emp: zero-copy os-bypass nic-driven gigabit ethernet message passing. In *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, pages 57–57, New York, NY, USA, 2001. ACM Press.

[13] The InfiniBand Trade Association. *Infiniband Architecture Specification Volume 1, Release 1.2*. InfiniBand Trade Association, 2003.

[14] University of Southern California. *RFC 791: Internet Protocol*, 1981.

[15] University of Southern California. *RFC 793: Transmission Control Protocol*, 1981.

[16] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-net: a user-level network interface for parallel and distributed computing (includes url). In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, pages 40–53, New York, NY, USA, 1995. ACM Press.

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-00-01 J.A. Makowsky und K. Meer, Polynomials of bounded tree-width, Januar 2000

CSR-00-02 Andreas Goerdt, Efficient interpolation for the intuitionistic sequent calculus, Januar 2000, Chemnitz

CSR-01-01 Werner Dilger, Evelyne Keitel, Kultur und Stil in der Informatik?, Januar 2001, Chemnitz

CSR-01-02 Guido Brunnett, Thomas Schaedlich, Marek Vanco, Extending Laszlo's Algorithm to 3D, März 2001, Chemnitz

CSR-01-03 M.Köchel, U.Nieländer, M.Sturm, KASIMIR - object-oriented KAnban SIMulation Imaging Reality, März 2001, Chemnitz

CSR-02-01 Andrea Sieber, Werner Dilger, Theorie und Praxis des Software Engineering, Oktober 2001, Chemnitz

CSR-02-02 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Energy-Efficient Size Approximation of Radio Networks with no Collision Detection, Februar 2002, Chemnitz

CSR-02-03 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Efficient Algorithms for Leader Election in Radio Networks, Februar 2002, Chemnitz

CSR-02-04 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Weak Communication in Radio Networks, Februar 2002, Chemnitz

CSR-03-01 Amin Coja-Oghlan, Andreas Goerdt, André Lanka, Frank Schädlich, Certifying Unsatisfiability of Random $2k$-SAT Formulas using Approximation Techniques, Februar 2003, Chemnitz

CSR-03-02 M. Randrianarivony, G. Brunnett, Well behaved mesh generation for parameterized surfaces from IGES files, März 2003, Chemnitz

CSR-03-03 Optimizing MPI Collective Communication by Orthogonal Structures, Matthias Kühnemann, Thomas Rauber, Gudula Rünger, September 2003, Chemnitz

CSR-03-04 Daniel Balkanski, Mario Trams, Wolfgang Rehm, Heterogeneous Computing With MPICH/Madeleine and PACX MPI: a Critical Comparison, Dezember 2003, Chemnitz

CSR-03-05 Frank Mietke, Rene Grabner, Torsten Mehlan, Optimization of Message Passing Libraries - Two Examples, Dezember 2003, Chemnitz

Chemnitzer Informatik-Berichte

CSR-04-01 Karsten Hilbert, Guido Brunnett, A Hybrid LOD Based Rendering Approach for Dynamic Scenes, Januar 2004, Chemnitz

CSR-04-02 Petr Kroha, Ricardo Baeza-Yates, Classification of Stock Exchange News, November 2004, Chemnitz

CSR-04-03 Torsten Hoefler, Torsten Mehlan, Frank Mietke, Wolfgang Rehm, A Survey of Barrier Algorithms for Coarse Grained Supercomputers, Dezember 2004, Chemnitz

CSR-04-04 Torsten Hoefler, Wolfgang Rehm, A Meta Analysis of Gigabit Ethernet over Copper Solutions for Cluster-Networking, Dezember 2004, Chemnitz

CSR-04-05 Christian Siebert, Wolfgang Rehm, One-sided Mutual Exclusion A new Approach to Mutual Exclusion Primitives, Dezember 2004, Chemnitz

CSR-05-01 Daniel Beer, Steffen Höhne, Gudula Rünger, Michael Voigt, Software- und Kriterienkatalog zu RAfEG - Referenzarchitektur für E-Government, Januar 2005, Chemnitz

CSR-05-02 David Brunner, Guido Brunnett, An Extended Concept of Voxel Neighborhoods for Correct Thinning in Mesh Segmentation, März 2005, Chemnitz

CSR-05-03 Wolfgang Rehm (Ed.), Kommunikation in Clusterrechnern und Clusterverbundsystemen, Tagungsband zum 1. Workshop, Dezember 2005, Chemnitz

CSR-05-04 Andreas Goerdt, Higher type recursive program schemes and the nested pushdown automaton, Dezember 2005, Chemnitz

CSR-05-05 Amin Coja-Oghlan, Andreas Goerdt, André Lanka, Spectral Partitioning of Random Graphs with Given Expected Degrees, Dezember 2005, Chemnitz

CSR-06-01 Wassil Dimitrow, Mathias Sporer, Wolfram Hardt, UML basierte Zeitmodellierung für eingebettete Echtzeitsysteme, Februar 2006, Chemnitz

CSR-06-02 Mario Lorenz, Guido Brunnett, Optimized Visualization for Tiled Displays, März 2006, Chemnitz

CSR-06-03 D. Beer, S. Höhne, R. Kunis, G. Rünger, M. Voigt, RAfEG - Eine Open Source basierte Architektur für die Abarbeitung von Verwaltungsprozessen im E-Government, April 2006, Chemnitz

CSR-06-04 Michael Kämpf, Probleme der Tourenbildung, Mai 2006, Chemnitz

CSR-06-06 Torsten Hoefler, Mirko Reinhardt, Torsten Mehlan, Frank Mietke, Wolfgang Rehm, Low Overhead Ethernet Communication for Open MPI on Linux Clusters, Juli 2006, Chemnitz