DISSERTATION

submitted to the

Combined Faculty of Natural Sciences and Mathematics

of the

Ruprecht–Karls University

Heidelberg

for the degree of

Doctor of Natural Sciences

put forward by

M.Sc. Felix Zahn

born in

Mannheim, Baden-Württemberg

Heidelberg, 2020

Energy-Efficient Interconnection Networks for High-Performance Computing

Advisor: Professor Dr. Holger Fröning

Date of oral exam: .....

Abstract

In recent years, energy has become one of the most important factors for designing and operating large scale computing systems. This is particularly true in high-performance computing, where systems often consist of thousands of nodes. Especially after the end of Dennard's scaling, the demand for energy-proportionality in components, where energy is depending linearly on utilization, increases continuously. As the main contributor to the overall power consumption, processors have received the main attention so far. The increasing energy proportionality of processors, however, shifts the focus to other components such as interconnection networks. Their share of the overall power consumption is expected to increase to 20% or more while other components further increase their efficiency in the near future. Hence, it is crucial to improve energy proportionality in interconnection networks likewise to reduce overall power and energy consumption. To facilitate these attempts, this work provides comprehensive studies about energy saving in interconnection networks at different levels.

First, interconnection networks differ fundamentally from other components in their underlying technology. To gain a deeper understanding of these differences and to identify targets for energy savings, this work provides a detailed power analysis of current network hardware.

Furthermore, various applications at different scales are analyzed regarding their communication patterns and locality properties. The findings show that communication makes up only a small fraction of the execution time and networks are actually idling most of the time. Another observation is that point-to-point communication often only occurs within various small subsets of all participants, which indicates that a coordinated mapping could further decrease network traffic.

Based on these studies, three different energy-saving policies are designed, which all differ in their implementation and focus. Then, these policies are

evaluated in an event-based, power-aware network simulator. While two policies that operate completely local at link level, enable significant energy savings of more than 90% in most analyses, the hybrid one does not provide further benefits despite significant additional design effort. Additionally, these studies include network design parameters, such as transition time between different link configurations, as well as the three most common topologies in supercomputing systems.

The final part of this work addresses the interactions of congestion management and energy-saving policies. Although both network management strategies aim for different goals and use opposite approaches, they complement each other and can increase energy efficiency in all studies as well as improve the performance overhead as opposed to plain energy saving.

Zusammenfassung

In den letzten Jahren ist Energie zu einem der wichtigsten Faktoren für Entwurf und Betreiben großer Rechensysteme geworden. Dies gilt insbesondere für das Hochleistungsrechnen, wo Systeme oft aus Tausenden von einzelnen Rechenknoten bestehen. Besonders nach dem Ende des Dennard Scaling steigt der Bedarf an energieproportionalen Komponenten, bei denen die Energie linear von der Auslastung abhängt, kontinuierlich an. Als hauptverantwortliche Komponente für den Gesamtstromverbrauch haben die Prozessoren bisher die größte Aufmerksamkeit erhalten. Die zunehmende Energieproportionalität von Prozessoren verlagert jedoch den Schwerpunkt auf andere Komponenten wie zum Beispiel Verbindungsnetzwerke. Es wird erwartet, dass ihr Anteil am Gesamtstromverbrauch auf 20% oder mehr ansteigt, während andere Komponenten in naher Zukunft ihre Effizienz weiter steigern werden. Daher ist es von entscheidender Bedeutung, die Energieproportionalität in Verbindungsnetzwerken zu verbessern, um den Gesamtleistungs- und Energieverbrauch ebenfalls zu senken. Diese Arbeit trägt zu dieser Aufgabe bei, indem sie Energieeinsparungen in Verbindungsnetzwerken auf verschiedenen Ebenen umfassend analysiert.

Verbindungsnetzwerke unterscheiden sich in ihrer zugrunde liegenden Technologie elementar von anderen Komponenten. Um ein tieferes Verständnis dieser Unterschiede zu gewinnen und Ziele für Energieeinsparungen zu identifizieren, bietet diese Arbeit eine detaillierte Leistungsanalyse der aktuellen Netzwerk-Hardware.

Darüber hinaus werden verschiedene Anwendungen unterschiedlicher Skalierung hinsichtlich ihrer Kommunikationsmuster und Loakalitätseigenschaften analysiert. Die Ergebnisse zeigen, dass die Kommunikation nur einen kleinen Bruchteil der Ausführungszeit ausmacht und die Netzwerke die meiste Zeit tatsächlich ungenutzt bleiben. Eine weitere Beobachtung ist, dass Ende-zu-Ende-Kommunikation oft nur innerhalb verschiedener kleiner Teilmengen

aller Teilnehmer stattfindet, was darauf hindeutet, dass ein maßgeschneidertes Zuordnen von Prozessen auf physikalische Rechenkerne die Netzwerkauslastung weiter verringern könnte.

Auf der Grundlage dieser Studien werden drei verschiedene Energiesparstrategien entworfen, die sich alle in ihrer Umsetzung und ihrem Schwerpunkt unterscheiden. Diese Strategien werden dann in einem ereignisbasierten Netzwerksimulator mit integriertem Energiemodell evaluiert. Während zwei Strategien, die vollständig lokal auf Linkebene arbeiten, in den meisten Analysen signifikante Energieeinsparungen von mehr als 90% ermöglichen, bietet die hybride Strategie trotz erheblichen zusätzlichen Designaufwands keine weiteren Vorteile. Darüber hinaus umfassen diese Studien Netzwerkdesignparameter, wie z.B. die Transitionzeit zwischen verschiedenen Linkkonfigurationen, sowie die drei häufigsten Topologien, die genutzt werden um Supercomputer zu entwerfen.

Abschließend werden im letzten Teil der Arbeit Wechselwirkungen von Congestion Management und Energiesparmaßnahmen thematisiert. Obwohl beide Netzmanagementstrategien unterschiedliche Ziele verfolgen und entgegengesetzte Ansätze verwenden, ergänzen sie sich gegenseitig und können die Energieeffizienz in allen Studien erhöhen und Performanzindikatoren im Gegensatz den reinen Energiesparstrategien verbessern.

Table of contents

CHAPTER 1

Introduction

Historically, the steady increase of computational power was driven by the decreasing size of integrated circuits and rising clock rates, described by Moore's law. This performance increase was facilitated by Dennard's scaling [1], which states that decreasing feature sizes also result in a proportional decrease in power consumption so that the power consumption per area remains constant. Additionally, pipelining and Instruction-Level Parallelism (ILP) ensured further performance scaling within a single Central Processing Unit (CPU). With mitigating clock rate growth in the 2000s, parallelism ensured further performance scaling. At about 2005, multi-core processors began to take over from single-core processors, which means multiple cores were integrated into one die. In order to take advantage of these parallel cores, software engineers followed this trend by shifting to Thread-Level Parallelism (TLP).

While Moore's law remains still valid, Dennard's scaling has come to an end [2]. Hence, the number of transistor devices per chip is still increasing, but power density for chips has reached its limit. As a result, chips have to operate within a strict power budget. This leads to more specialized functional units on a chip, which can efficiently perform narrower tasks. However, not all of these functional units can operate simultaneously to ensure compliance with the power budget. Transistors or areas on the chip that are not used because of power capping are referred to as dark silicon. Consequently, power and energy efficiency are some of the main drivers for the performance of today's computing systems. Figure 1.1 depicts the trends of these different design features over time. It can be clearly

seen that performance steeply rises until 2006. At the end of Dennard's scaling (indicated by the vertical line), the increase of clock rates ends, which finally leads to a flattening of the performance trend.

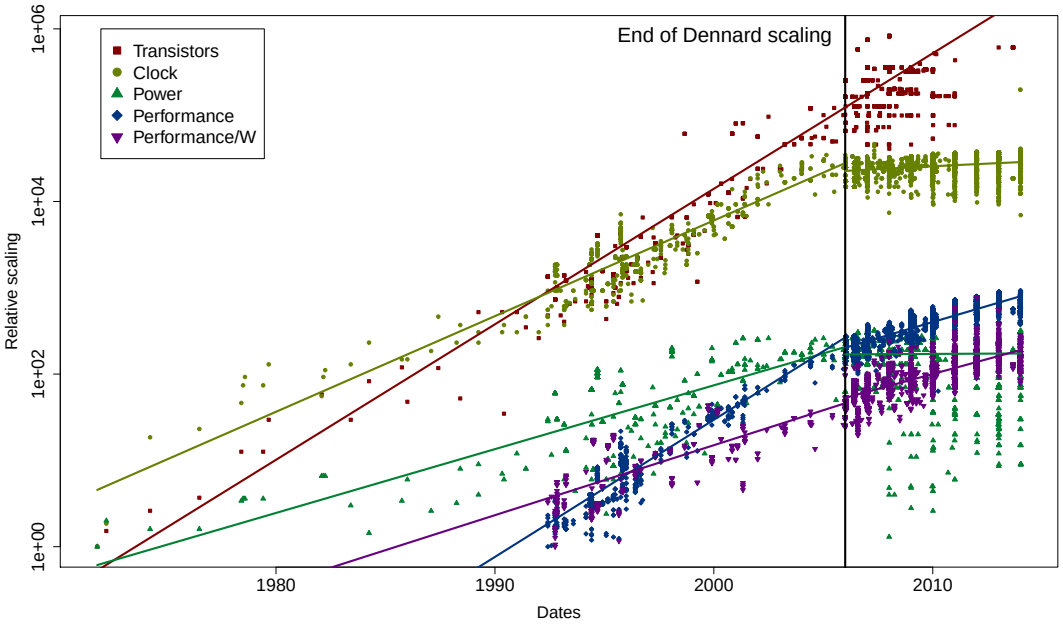


Fig. 1.1 Technology scaling trends for various features.

Beyond specialized on-chip functional units and the increase of the number of cores per die, parallelization also includes the introduction of accelerators. The most common example for such an accelerator is a Graphics Processing Unit (GPU), which are highly parallel vector processors.

The growing number of parallel units in one system raises the demand for communication and data exchange between them. At the node level, different cores or functional units use an on-chip network (communication between parallel cores) or system networks, such as NVLink or Peripheral Component Interconnect Express (PCIe), (communication with accelerating units) to communicate and transfer data. Communication between distinguished nodes, however, is performed on interconnection networks. Furthermore, multiple nodes are clustered together to meet the demand for more computational power by further increasing the parallelism of a system. Hence, the performance requirements for these networks are rather challenging.

High-Performance Computing

High-Performance Computing (HPC) describes systems that are designed for maximal computing performance using cutting edge technology. It takes up a predestined position in the field of computational science and is mostly performed on a supercomputer. The prime targets of these systems are often compute-intensive, scientific workloads. These applications cover a wide range of different scientific applications, including weather forecasts, physical simulations, molecular modeling, nuclear research, quantum mechanics, and artificial intelligence.

Great efforts are made to meet the increasing demand for higher computational power of these compute-intensive applications, and promising new hardware and concepts are often tested in these systems. Hence, HPC is playing a pioneering role in the area of computational performance and is possibly affecting almost all other kinds of computing systems. HPC systems were the first to rely extensively on parallelization and were also heavily affected by the end of Dennard's scaling. While further performance scaling is now rather based on additional hardware accelerators, power supply, and heat dissipation remain the main challenges of high-performance computing at both chip- and system-level.

Adding specialized acceleration units affects the system's overall power consumption. Not only do these added components contribute through their respective power consumption but they also increase the need for additional data movements. This trend becomes even more important with the increasing number of compute nodes in HPC systems. Hence, the growing demand for more and more computing power also leads to special requirements on the interconnection network, such as high bandwidth and low latency. The continuing optimization of hardware and network protocols lead to highly specialized networks, which differ from general-purpose networks. For example, networks in HPC systems are commonly lossless. This means, that the network guarantees that no packets are dropped on their way from sender to receiver, which can be exploited to reduce performance overhead in the network protocols significantly.

The next milestone for HPC to be taken in the near future is a system that enables exascale computing (10^{18} FLOP/s). Although computational power can be easily increased by adding more parallel units, the main challenge remains to build such a system at reasonable acquisition and operational costs. The US Department of Energy (DoE) has set the goal to design such a system within

a power budget of 20 MW[1]. Hence, energy-efficiency is one of the main design goals in new HPC systems.

Energy Proportionality

Operating an exascale system with current technologies and a strict power budget becomes even more challenging due to the lack of energy proportional hardware components. The concept of energy proportionality, which was first introduced by Barroso et al. [3], means that the power consumption of each component should be proportional to its utilization. For instance, if a CPU is working half the time and idling for the other half, the effective energy consumption of a perfect energy-proportional CPU would also be halved.

This approach originates from data centers and cloud installations, which usually operate at low utilization but are designed to handle peak loads. Although HPC installations commonly operate at higher utilization, not all components are evenly utilized. While great efforts have been made to increase energy-proportionality in processing units as the main contributors to the overall power consumption, other components rather operate constantly at peak power. As a result, the remaining components increase their impact on the overall power consumption, although their contribution is rather low at Thermal Design Power (TDP). This effects particularly interconnection networks, which are expected to become one of the main power-consuming components as the development of energy-proportional processing units progresses. Multiple analyses show that their share of the overall power consumption will increase in the near future to up to 30% [4], [5]. Furthermore, the 2015 International Technology Roadmap for Semiconductors (ITRS) report predicts that soon data center power consumed by networking and switching will exceed the aggregated power consumed by storage and cooling [6].

The situation is further complicated by the fact that well-established power saving mechanisms from other components cannot be simply adopted due to fundamental design differences. In contrast to most other CMOS-based components, interconnection networks heavily rely on Current Mode Logic (CML), which enables higher frequencies and stronger drivers at a given power budget.

[1] https://science.osti.gov/-/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf, accessed: 2020-02-05

As its name suggests, CML relies on a static current which makes its power consumption robust against frequency scaling. To gain further insights about the power consumption of interconnection networks and which energy-saving mechanisms are suitable are analyzed in Chapter 3. These energy-saving capabilities are also affected by the applications running on these systems. Generally, HPC applications tend to utilize system resources much more in opposed to common cloud workloads. However, our studies in Chapter 4 show that HPC workloads are also suitable for energy savings, particularly in interconnection networks. These analyses demonstrate that there is no perfect overlap of computation and data movements in a huge variance of different HPC and exascale proxy applications [7], [8].

Although few related works aim to tackle these problems, they rather focus on optimizations of particular settings. This includes a given interconnect technology/provider, such as Energy-Efficient Ethernet (EEE) [9] or Mellanox' InfiniBand [10], or single topologies, in particular fat trees [11]. None of these works study the interactions with other network management strategies, for example, congestion management.

To draw a comprehensive picture of energy-saving in the interconnection network, this work follows a technology-independent approach, which is evaluated in a variety of different scenarios. In Chapter 6, three simple energy-saving policies are introduced, which reflect different trade-offs between performance and aggressiveness in energy-saving [12], [13]. To evaluate these policies, they are tested in a cycle-accurate simulator in different settings, including multiple topologies, applications, and scales. While using a simulator enables technology-independent testing of policies, the parameter set is derived from different kinds of real-world network technologies and is inside the scope of what is technically feasible. Overall, all configurations show high capabilities for energy saving and the introduced policies are able to reduce link energy by 92.7% on average by only increasing the execution time by an average of 5.7%. However, there are also few configurations that seem not to be suitable for energy saving. Although energy can still be saved their execution time is significantly increased.

Building on top of these first studies, the interaction of the power-saving policies with congestion management is investigated in Section 6.4. The main challenge for a possible synergy is that they are pursuing fundamentally different goals. While energy-saving policies benefit from few traffic flows that highly

utilize links, congestion management aims for evenly split traffic flows over the entire network. Despite the combination of both has not been studied before, they both seem to complement each other and work well together [14].

Contributions

This work makes the following major contributions:

1. *Network power analyses* - a detailed analysis of various components of a modern interconnection network is performed. The different components of a switch fabric are studied at different scales, and the main contributors to the overall power consumption are identified. As a result, sweet spots for energy saving in the design and possible power-saving approaches are identified. The insights of this analysis are elaborated in Chapter 3.

2. *Application analyses to identify energy-saving capabilities* - a large variety of different communication patterns of exascale proxy applications is studied regarding their suitability for energy saving. Especially sparse communication and long computation phases are highly suitable since these patterns cause long idling periods in the network. Furthermore, new metrics are introduced to indicate locality in these patterns, which can be exploited to further reduce network traffic. Details are found in Chapter 4.

3. *Energy-aware discrete event-based simulation of interconnection networks* - an existing OMNeT++-based, cycle-accurate simulator is extended with energy features. These include the power consumption of all major components according to their particular configuration, which enables the introduction of discrete power states. These power states are used for energy measurements as well as comprehensive energy and power state analyses. More details are provided in Chapter 5.

4. *Introduction of energy-saving policies* - based on the previous analyses, different energy-saving policies are introduced. These policies, which focus either on aggressive power saving or better performance, are easy to implement and show promising results regarding their energy-saving capabilities. Thereby, a decentralized approach is used to reduce management overhead. Furthermore, their interference with congestion management,

another traffic flow management system, is investigated. While congestion management is essential for HPC interconnection networks, both techniques follow contrary approaches and present a potential for conflicts. These studies are presented in Chapter 6.

Dissertation Outline

The remainder of this work is structured as follows:

- Chapter 2 provides an overview of the structure of interconnection networks. This includes the hardware architecture of common switches, switching and routing schemes, as well as a brief introduction into message passing.
- Chapter 3 provides a power analysis of different components inside a switch fabric. Based on these insights and considering underlying design technologies, the best approaches for energy savings are identified.
- In Chapter 4, a wide range of HPC applications is analyzed regarding their energy-saving opportunities. The selected applications are based on the DoE's exascale mini-applications, representing a wide range of common workloads and communication patterns.
- Chapter 5 provides an overview of the methodology and tools that are used to determine the energy consumption and savings of different approaches. The majority of the energy studies are based on a network simulator that is capable of replaying Message Passing Interface (MPI) traces or generating synthetic traffic patterns.
- In Chapter 6 three different energy-saving policies are introduced and analyzed regarding their energy-saving potential and impact on the overall performance on exemplary HPC applications. Also, the interaction of these policies with often-used congestion management strategies is studied on synthetic traffic patterns.
- Chapter 7 provides an overview of related works and discusses the findings of the previous chapters. Finally, a brief outlook about future research directions is given.
- Chapter 8 summarizes and concludes this work.

Publications

This section provides the works in context of this dissertation that have been published in international conferences and journals with peer review.

- F. Zahn, P. Yebenes, S. Lammel, P. J. Garcia, and H. Fröning, "Analyzing the energy (dis-) proportionality of scalable interconnection networks," in 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), 2016, pp. 25–32.
- S. Lammel, F. Zahn, and H. Fröning, "Sonar: Automated communication characterization for hpc applications," in High Performance Computing, M. Taufer, B. Mohr, and J. M. Kunkel, Eds., Cham: Springer International Publishing, 2016, pp. 98–114
- F. Zahn, S. Lammel, and H. Fröning, "Early experiences with saving energy in direct interconnection networks," in IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), Feb. 2017, pp. 33–40.
- F. Zahn, A. Schäffer, and H. Fröning, "Evaluating energy-saving strategies on torus, k-ary n-tree, and dragonfly," in IEEE 4th International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), Feb. 2018, pp. 16–23.
- F. Zahn, S. Lammel, and H. Fröning, "On link width scaling for energy-proportional direct interconnection networks," Concurrency and Computation: Practice and Experience, vol. 31, no. 2, e4439, 2019.
- F. Zahn, P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, and H. Fröning, "Effects of congestion management on energy saving techniques in interconnection networks," in International Workshop of High-Perfomance Interconnection Networks in the Exascale and Big-Data Era (HiPNEB), Feb. 2019, pp. 9–16.
- F. Zahn and H. Fröning, "On network locality in mpi-based hpc applications," in 49th International Conference on Parallel Processing - ICPP (ICPP '20), Edmonton, AB, Canada: ACM, New York, NY, USA, Aug. 2020, p. 11.

CHAPTER 2

Background: Interconnection Networks

With the increasing size of parallel systems, the importance of networks increases likewise. Thereby, networks are used for various connection purposes, ranging from small distance connections between functional units up to long distances, spanning the world wide web. This wide range of networks can be clustered into four major groups of networks [15]:

1. Network on Chip (NoC): They are used to connect multiple functional units of a System on Chip (SoC), such as processor cores, caches, or register files. Although their connection distances are limited to the order of centimeters, they provide multiple advantages over traditional buses, such as increased energy efficiency, better scaling, or support for asynchronous clock domains.

2. System Area Networks (SAN): This type of network is usually used to connect multiple processors or processor to memory in multicomputer systems. These networks can include several thousands of such devices and are usually used in the context of data centers or supercomputers. Since these systems have to meet ambitious performance requirements and computation tended to be a bottleneck, customized hardware is often used in the context of these systems [16], such as the Infiniband standard [17] or Intel's Omnipath [18].

3. Local Area Network (LAN): LANs are designed to cover small geographical areas, reaching from buildings over few adjacent buildings up to campus

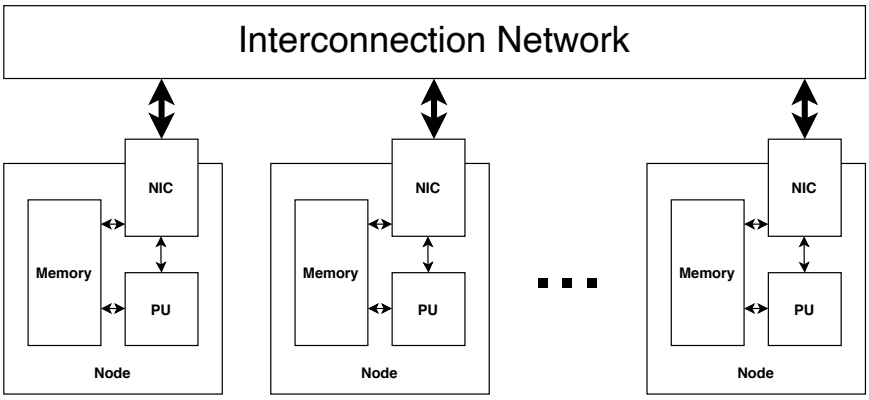


Fig. 2.1 Design scheme for parallel SAN systems.

areas [19]. Most commonly, Ethernet is used to connect autonomous computing systems in this type of network [15].

4. Wide Area Network (WAN): WANs are distributed over large geographical areas, which scale up to thousands of kilometers, and can include many millions of computers. Because of the large distances and amount of transferred data, today's WANs are usually composed of fast optical connections. The Internet WAN is probably the most prominent representative of this network class.

There are no sharp lines between these types of networks and some examples are overlapping two types. However, interconnection networks in HPC systems are generally located in the area of SANs.

Figure 2.1 depicts the schematic set-up a parallel HPC system. These systems consist of a variety of nodes that are equipped with one or more processing and memory units. Additionally, every node has an interface, which is called Network Interface Controller (NIC), that enables communication via the interconnection network with other nodes. Although Uniform Memory Architecture (UMA) shared-memory systems, in which the network connects several processing units with external memory, are also representatives of SANs, they do not scale well and are limited to rather small systems. Therefore, these studies focus on large-scale, distributed shared-memory systems equipped with specialized custom interconnection networks.

2.1 Switch Level Architecture

Transferring data on the network requires interactions of multiple protocols, software stacks, and different hardware components. To enable a better understanding of these processes the Open Systems Interconnection (OSI) project developed a model that describes networks as a series of layers. These layers ease the understanding of processes in the network and responsibilities of particular units or protocols. The focus on this section is on different designs and processes in the two lowest layers, the link and physical one.

2.1.1 Data Transmission

When data is exchanged between network clients (processors or memory), they communicate in units of messages [20]. These messages are created in the application layer (layer 7) and contain additional metadata, such as memory addresses and receiver information, and the actual data that a client wants to transfer. These messages do not limit the amount of data that can be sent and can have an arbitrary length. Since the physical interface demands for a uniform format, the messages are further processed in the NIC. Regarding the OSI model, this takes place in the transport layer (layer 4).

Here, the messages are partitioned into packets. These packets start with a specified header format, which contains information about the sender, receiver, length, and other data that is required by the routing unit, followed by a payload with the actual data or parts of it. Packets end with a tail, that includes an error-detection check, such as Cyclic Redundancy Check (CRC). Although payload length in today's interconnection networks can have a volatile length, their maximum is often limited to a few megabytes. If the message length exceeds the maximum payload length, the message is split up into the resulting number of packets.

In layer 3, the network layer, addresses are handled, routing information is added, and generally routing decisions are made. The lowest layers, (link and physical layer) are responsible for the actual data transmission and error detection and recovery [19]. In this layer, packets are split again into smaller parts for performance reasons. First, they are divided into Flow Control Unit (FLIT) frames. These units are determined by the flow control and buffer structures. In

the physical layer, FLITs can further be sliced into Physcial Units (PHIT) which corresponds to the physical width of a link [21]. However, data transmission in the context of this work is observed at packet granularity, since it focuses on the network layer. At the receiving side, each message is reversely processed in the opposite order of all layers.

2.1.2 Network Interface

The NIC is the host interface that connects a node to the interconnection network to source and sinks packets. One the host side, the NIC is directly connected to the internal bus of the host node and on the network side either to a switch or another NIC. Since the NIC has to handle this two-way traffic, its particular architecture is highly dependent on the network and the host [22].

Essentially, the NIC contains an embedded processor to format packets, which includes splitting up messages and creating a header with all necessary routing and control information, and perform end-to-end error checks [23]. Furthermore, it may be equipped with substantial input and output buffer compared to switches.

2.1.3 Links

Links or channels are physical connections between two network entities. They are either composed of one or more electrical wires or optical fiber and connectors at both ends [23]. Although a wide range of different standards and designs exist, they all share the same purpose of transmitting analog signals from a sender to the receiver, where the original digital data stream is obtained. Links can be either uni- or bidirectional. In HPC systems bidirectional links are widely used and, therefore, all links are assumed to be bidirectional in the context of this work.

Links can vary in length, clocking scheme, and width [23]. Electrical links are capable of transmitting data over distances of up to about 100m [24], but since the RC delay in wires increases square of the length, longer cable length demand for stronger, more power-consuming drivers [23]. Additionally, losses due to skin effect, dielectric losses, etc. in these cable increase with cable length, whereas losses in optical fibers are rather small. However, the optical cable requires connectors to transform the digital electrical data stream to an optical

signal. Clocking in the interconnection link can either be synchronous and asynchronous. While asynchronous links obtain a simpler design and interface, they result in lower frequencies and produce significant power overhead to achieve specific bandwidths [25]. Therefore, asynchronous links are not suitable for HPC systems. Synchronous links, however, require constant clock recovery, for example via Phase-Locked Loop (PLL) and Clock and Data Recovery (CDR). As a consequence, synchronous links are sending idle pattern when no data is transmitted to ensure word alignment. The width of links is determined by the number of parallel wires or lanes inside one link. Although increasing the number of lanes is a sophisticated way to improve bandwidth at a given frequency, only limited scaling is possible. The two main restrictions that limit the link width are crosstalk inside the cable and a more important pin count limitation due to the spatial extent of the backplane [15].

This limit of link width also drives the need for serialization and deserialization (ser/des) technology at both ends of the links. On the transmitting side (TX), the parallel data stream that arrives at the NIC is usually wider than the link. Hence, this data stream needs to be serialized according to the link width. Since transmission on the interconnection network aims to at least provide the same bandwidth as internal data paths, the transmitting frequency has to be higher by the same factor as the ratio between the bus and link widths. On the receiving side (RX), this faster serial analog data stream is checked for correctness and deserialized back to a parallel data stream.

Operating data streams at different frequencies also requires buffers to store data that cannot be processed immediately. These buffers can be designed in different ways. Typically, links are equipped with either input or output buffers, or both. Output buffered links have the advantage of head of line blocking prevention, which is explained more detailed in chapter 6, since all packets obtain the same status. However, all incoming packets have to be directly processed and stored in the respective output buffer. This requires an internal speedup of switches of the same factor as there are parallel lanes inside one link. Therefore, this very expensive solution is rarely implemented in lossless networks [15]. In the following, the part of the NIC or switch, which contains the ser/des and buffers for one link, is referred to as linkport.

2.1.4 Switches

Switches are network entities with a set of input ports, output ports, and an internal crossbar that connects every input to every output port [23]. Additionally, a switch is equipped with control units, such as an arbiter or routing unit. Figure 2.2 depicts the scheme of an exemplary switch, introducing all relevant components. On the left side, there are input ports (RX), including input buffers, and on the right side the output ports (TX) along with output buffers. As previously mentioned, the actual buffer structure may include input or output buffer, or a combination of both, depending on the network implementation. Note that for bidirectional links the input and output ports are located in the same physical linkport. For most switches applies that the number of input ports equals the number of output ports and this number is referred to as the switch's radix or degree. The core of the switch is the crossbar and the corresponding control units. A $n \times m$ crossbar connects n inputs to m outputs without intermediate stages [20]. By definition, each output must be connected to at most one input port, which has to be ensured by the control unit, namely arbiter, and routing unit. The former schedules the order in which packets are processed and forwarded. The latter makes the routing decision for each packet, for example, based on routing tables, and configures the crossbar in a way that the input port of the current packet connects to the right output port. When the packet has passed the crossbar, the arbiter selects the next packet and the routing unit configures the crossbar accordingly to the new routing information. The complexity of these control units depends on the routing and scheduling algorithm [23].

Although crossbars are non-blocking, they are not suitable for larger networks due to their poor scalability [20]. Not only do costs increase as n^2 for an $n \times n$ crossbar but also the pin count limitation and scheduling problems become prohibitive. These scaling issues affect the performance of high-radix switches so that low diameter switched provide lower latencies, fewer contention, and higher throughput at the same bisection bandwidth [26]. Therefore, modern high-radix switches are internally composed of multiple smaller crossbars that are connected in a hierarchical pattern, which improves latency significantly and reduces area by almost half [27].

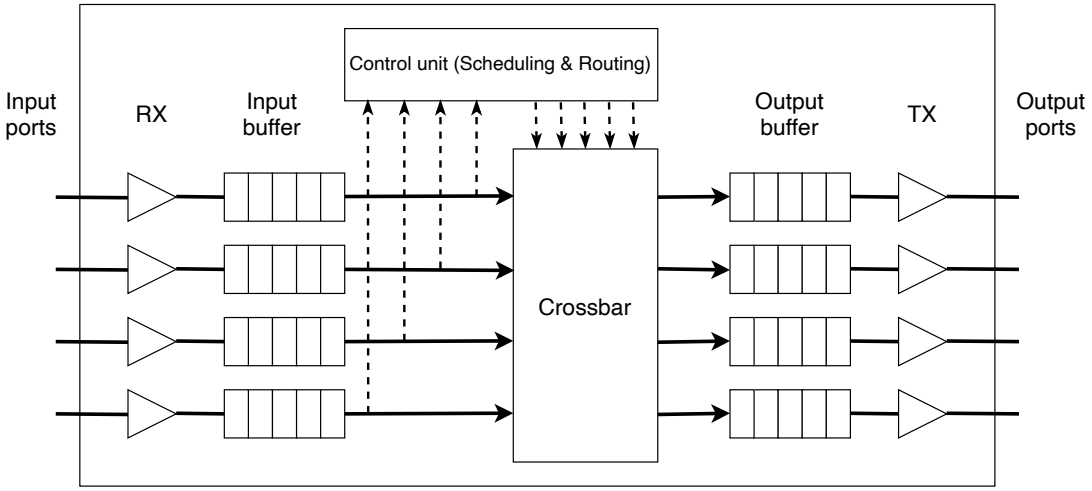


Fig. 2.2 Switch blueprint with essential components.

2.1.5 Message Switching

The last major domain of the link and data layer, and already interfering with the network layer, is message or packet switching. This includes the switching techniques that are implemented in the network and different flow control mechanisms [16]. However, this does not include routing decisions or network paths but message processing inside the switch or the network in general.

Message Switching Techniques

The message switching policy controls how packets are routed inside a switch, e.g. the procedure that happens between the arrival of a packet at the input port and the departure of the same packet at the output port. The time between these two events is referred to as routing delay and is mainly driven by the time it takes to process routing information and set up the internal logic [16]. Depending on various factors, such as application types, vendors, or timing constraints, multiple different message switching techniques exist.

The first and one of the simplest message switching techniques is circuit switching [20]. In this approach, two nodes establish a permanent physical link between them, which is reserved exclusively for communication of this node pair. In the beginning, the source node sends a header packet to the destination, which establishes the path along with all switches in between. Once this header packet arrives at the destination node, this node acknowledges the source node that the path is set up. After the path is established all data can be transmitted directly

to the destination node and it sinks there. At the end of the data stream, a footer indicates the end of the message, and the path is released along the way. Circuit switching is especially sufficient for application with sparse communication and long messages. Note that networks that rely on circuit switching require almost no buffers since they only have to store other header packets, and no flow control. However, this is not feasible for most systems, since these established paths prohibit other network flows that are sharing the same resources. Additionally, long set up times cause a significant overhead for smaller messages. The impact of this overhead decreases with increasing message size.

Packet switching [16] refines the idea of circuit switching. Instead of sending all data at once, messages are split up into packets of a pre-defined maximum length. Each packet is equipped with its header and can be transmitted independently. In the simplest version, each packet is stored entirely in each intermediate network node or switch, before it is forwarded to the next node (store and forward). This technique is much more suitable for frequent smaller messages since only segments (e.g. single links) are occupied by a single packet, which enables more simultaneous traffic flows in the networks. But this also increases the overhead. The amount of data that is transferred on the network increases, since each packet got its header instead of one header per message. Furthermore, the routing delay might increase since each packet is routed individually, and also the demand for buffers increases, since packets have to be stored entirely after each hop.

In order to tackle some of these disadvantages, virtual cut-through switching [16] was introduced. This technique is also based on packet switching but represents an alternative to the store and forward approach. In the store and forward approach, a packet is stored completely before it is routed to the next node. That implies that the header, which arrives first, is first evaluated when the tail of the packet is stored in the buffer. Virtual cut-through switching aims to minimize this delay. In this technique, the header is evaluated instantly when it arrives and, consequently, routing decisions and configurations can be made simultaneously the arrival of the residual packet. Hence, the header might leave the switch, before the packet is stored completely, and the remaining parts are just routed the same way. This effective pipelining of packets does not only reduce the routing delay inside each switch but also dispense the need for output buffers since packets can be routed right away. If there is heavy traffic on the network and packets have to wait before the can be forwarded to the next node,

the header is blocked and the packet drops back to the packet switching case, where the packet is stored completely in the input buffer.

The last message switching technique in the context of this work is wormhole switching [16]. This technique is similar to virtual cut-through switching but aims to reduce buffer requirements. Packet switching and virtual cut-through switching require large buffer sizes to temporarily store packets in the worst case when the network is highly utilized. Therefore, the wormhole switching technique further splits packets up in smaller pieces (FLIT). The size of a FLIT is determined by the flow control and buffers are usually dimensioned to store a few FLITs simultaneously. This scheme basically increases the number of pipeline stages by potentially spreading a single packet over multiple switches and buffers, respectively. While on low traffic virtual cut-through and wormhole switching are very similar, they behave significantly differently on higher loads. The smaller transfer units decrease the probability of blocking other traffic flows. Thus, wormhole switching significantly reduces average message latency and buffer requirements [16].

Flow Control

The flow control mechanism manages the available resources in a network, such as buffer capacities, bandwidth, or control states [20]. To avoid high latencies caused by dropped packets, HPC interconnection networks are usually lossless. This means, that packets are not dropped by the network and the receiver does not have to acknowledge the reception of the packet. Especially in lossless HPC networks, a good flow-control is crucial to avoid the losses of packets due to overflowing buffers on the one hand and ensuring efficient utilization of available network resources on the other hand.

The most basic flow control mechanisms are bufferless [20]. This means that packets are not stored if a certain resource is not available but either routed differently or simply dropped. While this is easy to implement and they are not sufficient for lossless networks. The majority of flow control schemes are buffered and rely on some version of a request/acknowledgment protocol.

The simplest implantation of buffered flow control is a link-level handshake [23]. Here, the source sends a request to the destination, when a FLIT is ready to be sent. Then, the destination acknowledges the receipt of this flit and the sender can request a link for the next FLIT. More sophisticated flow control

schemes aim to reduce synchronization overhead. The most commonly used approaches are Xon/Xoff (Stop&Go) and credit-based flow controls [15]. The former assumes low buffer levels at the receiver and sends packets whenever they reach the linkport. When the buffers on the receiving side obtain high occupancy levels, the receiver notifies the sender with a stop message and a go message, respectively, when the buffer occupancy has decreased. These notifications are usually either transmitted on additional control wires or encoded in control packets [15]. In a credit-based flow control, the sender obtains a credit count that represents the number of FLITs that can be stored in the destination's input buffer [20]. Then, the flow control is throttling network traffic injection when the buffer on the receiving side fill up [15]. To keep track of the buffer levels at the receiving side, the credit counter is reduced every time a FLIT is sent. If the counter reaches zero, this means the input buffer on the receiving side is full and the sender stalls until new credits are available. On the receiving side, the destination sends a credit back, when a FLIT is further processed and its buffer space is freed.

Both techniques are used in modern interconnection networks, depending on the purpose, vendor, etc. Credit-based flow controls usually cause more overhead than a Xon/Xoff, since every processed FLIT causes a credit message, where the latter only needs notifications at high buffer occupancy levels. However, Xon/Xoff requires more than the double buffer size compared to credit-based systems because they need enough buffer size to prevent overflows before the stop notification arrives at the sender side [15]. Since the sender stops sending when it runs out of credits, overflows are impossible with credit-based flow controls. Generally, both techniques can only provide full link bandwidth, when the buffers are well-dimensioned for the distance between the source and destination [15].

2.2 System Level Network Design

Besides the technical features of switches and channels, which are part of layers one and two in the OSI model, the second main domain corresponds to layer three, the network layer. This includes the physical layout, e.g. the pattern in which single compute nodes are connected, and routing of the interconnection network.

2.2.1 Topologies

The topology describes the layout in which compute nodes are connected. Formally, they are described by a graph $G(N, C)$ with a set of nodes (compute nodes or switches) N and a set of links that connects graph nodes [16]. Multiple features are used to characterize topologies [16], including:

- Node degree: it is a measure for direct topology, which describes the number of directly connected neighbor nodes.
- Diameter: is defined by the maximum shortest path in the network.
- Regularity: indicates whether all nodes have the same degree or not.
- Symmetry: describes if the network looks the same from every node's perspective.
- Bisection bandwidth: The minimum bandwidth that is achieved when dividing the network into two halves.

The layout also impacts significantly a network's performance in terms of latency and bandwidth as well as costs in terms of hardware expenses, cooling, and spatial design.

While there are plenty of different topologies for general-purpose networks, HPC systems consist of thousands of nodes and require additionally good scalability. In general, three different classes of topologies are distinguished. In direct topologies, each NIC has an integrated switch [20]. This limits the switch radix due to spatial reasons but reduces latency for network accesses. The group of indirect topologies, in which nodes are always connected through external switching devices, is further divided into two classes: hierarchical and non-hierarchical indirect topologies. Hierarchical networks are composed of discrete layers, where each distinct layer is designed with individual functions to accomplish its purpose. A non-hierarchical indirect topology can often be described as Multistage Interconnection Networks (MIN). In order to overcome the crossbar scaling problem, MINs are used by organizing smaller switches in multiple stages in a way that connectivity between every source/destination pair is ensured [15].

To cover all of these classes, one representative each is used in the following studies.

3D Torus

In the context of network topologies, tori exist in multiple dimensionalities. A torus refers to a grid, which is equipped with wraparound links at the edge of every dimension. Hence, a torus forms a ring in every dimension, which halves the diameter by two compared to a regular grid. In off-chip networks, the most common one is the 3D torus, which corresponds to an enhanced mesh and is usually designed as direct topology. Figure 2.3 depicts a 2x2x2 torus with the characteristic wrap-around links in every dimension.

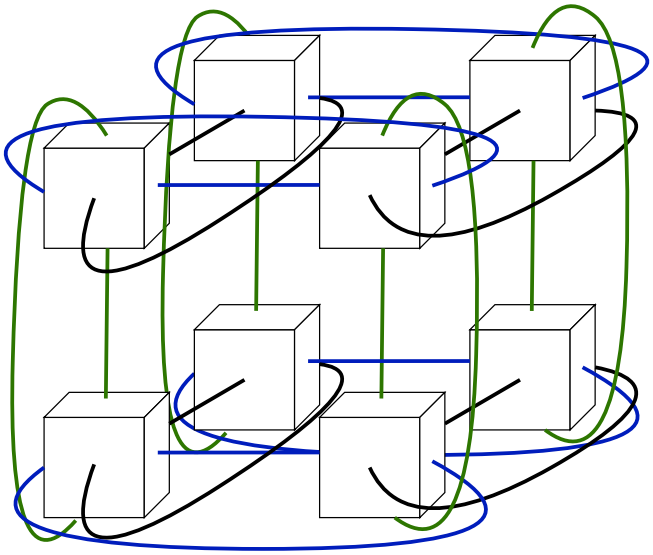


Fig. 2.3 Example 2x2x2 3D torus topology.

This regular and symmetric topology provides multiple advantages. First, as in all direct topologies, is the integration of the switch inside the NIC reduces the latency. This layout is only feasible because each node is only connected to its small set of neighbors. The switch radix that can be integrated into the NIC is limited by the dimensions of the slot in the backplane. However, this modular design enables good scalability and easily allows for expansions with additional nodes without the need to fully reconfigure the entire system. In terms of hardware costs, the neighbor-based connection pattern allows the torus to get along with a relatively large amount of short and cheaper electrical cables up to a certain scale. On the downside, this design results in a fast increasing diameter and the comparatively low bisection bandwidth. Table 2.1 compares the basic aspects of the other topologies used in context if this work.

	3D Torus	Fat Tree	Dragonfly
Diameter	$x+y+z$	$2(\log_{k/2} N)$	3 or 5
Regularity	+	+	+
Symmetry	+	-	-
# of switches	N	$N/k \times (2\log_{k/2} N - 1)$	N/p
# of links	$3N$	$N(\log_{k/2} N)$	$N(\frac{a+h}{2p} + 1)$

Table 2.1 Properties of the studied topologies. N indicates the total number of end nodes, k the switch radix, and x, y, z, the number of nodes per dimension in the torus. The parameter $a, p, andh$ are design parameters of the Dragonfly and are described further in the respective section.

Fat Tree

The next studied topology is tree-based and, therefore, a representative of non-hierarchical indirect topologies. For a small set of end nodes, it is sufficient to connect these nodes through only one switch. However, as the number of nodes increases, the crossbar scaling problem prohibits a further switch radix scaling. MINs provide a solution to this problem. They consist of multiple smaller switches that are connected in specific patterns to emulate larger radices. Generally, tree-based graphs are suitable patterns for MINs, and form a subset of these. However, some designs are more suitable than others. For instance, a regular binary tree provides very poor bisection bandwidth since the root becomes the bottleneck for most of the traffic.

A tree-based topology that aims to tackle this disadvantage is the fat tree, which provides the same bisection bandwidth at every stage [28]. To achieve this, the branches become thicker at the same scale as there are leaves connected to a switch. An exemplary configuration of eight nodes is illustrated in Figure 2.4. However, this is only a schematic depiction, as fixed switch radices require a more advanced connection pattern.

Formally, to remain a constant bandwidth at all stages, the switch ports (k) for each vertex at stage i grow by k^i, except for the root stage, which is only equipped with downward links. This is usually accomplished by using k^{i-1} switches per vertex. Therefore, each stage consists of $2N/k$ switches, which results in $2N/k \times (\log_{k/2} N)$ total switches (minus N/k switches at the root stage) [15]. In particular, this configuration is a special instance of a clos network.

Besides regularity and good bisection bandwidth, fat trees can be scaled

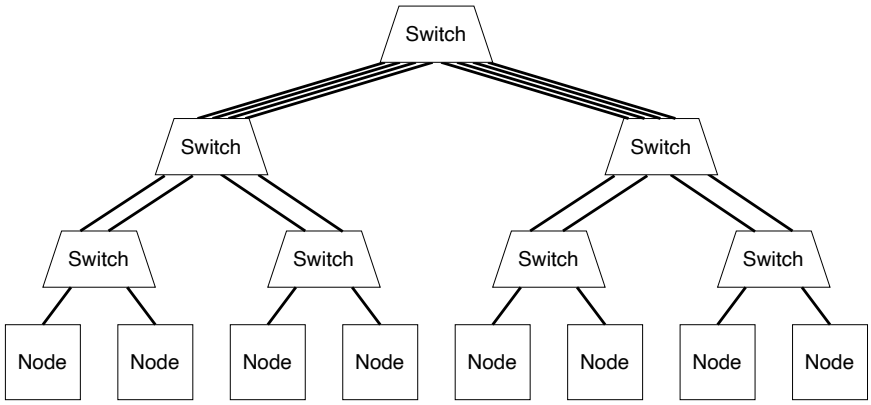


Fig. 2.4 Example k-ary n-tree topology.

simply by adding additional stages. However, it is almost impossible to extend an existing configuration due to the complex connection pattern between single switches and stages. Therefore fat tree systems are not as adaptable to new requirements as for e.g. tori networks. Furthermore, larger configurations demand an increasing amount of more expensive optical cables, as they are used to connect stages close to the root. Overall the fat tree is widely used in HPC system for its beneficial properties. The top 3 of the fastest supercomputers in 2019 are equipped with fat tree-based interconnection networks, namely Summit, Sierra, and Sunway TaihuLight.

Dragonfly

The last class of hierarchical indirect topologies is represented by the dragonfly [29]. The design goal of this rather new topology was to minimize the number of optical channels in order to reduce hardware costs. Besides, this also results in an overall low diameter. Like all hierarchical networks, the dragonfly is composed of different levels. Sets of compute nodes are organized to groups, with originally one group located in one cabinet. The compact spatial distances enable the usage of short and cheap electrical channels. These groups are then connected at the global level using long optical links between different cabinets. An exemplary dragonfly structure is depicted in Figure 2.5. The particular design is then determined by the three core parameters p (the number of compute nodes connected to each switch), a (the number of switches per group), and h (the number of global links in each switch). The remaining design then results from these parameters, such as the total number of groups ($g = ah + 1$) or compute

nodes ($N = ap(ah + 1)$). Kim et al. do not provide a strict rule on how to select these parameters particularly. However, they recommend to follow the rule $a \geq 2h$ and $2p \geq 2h$ to ensure a balanced network.

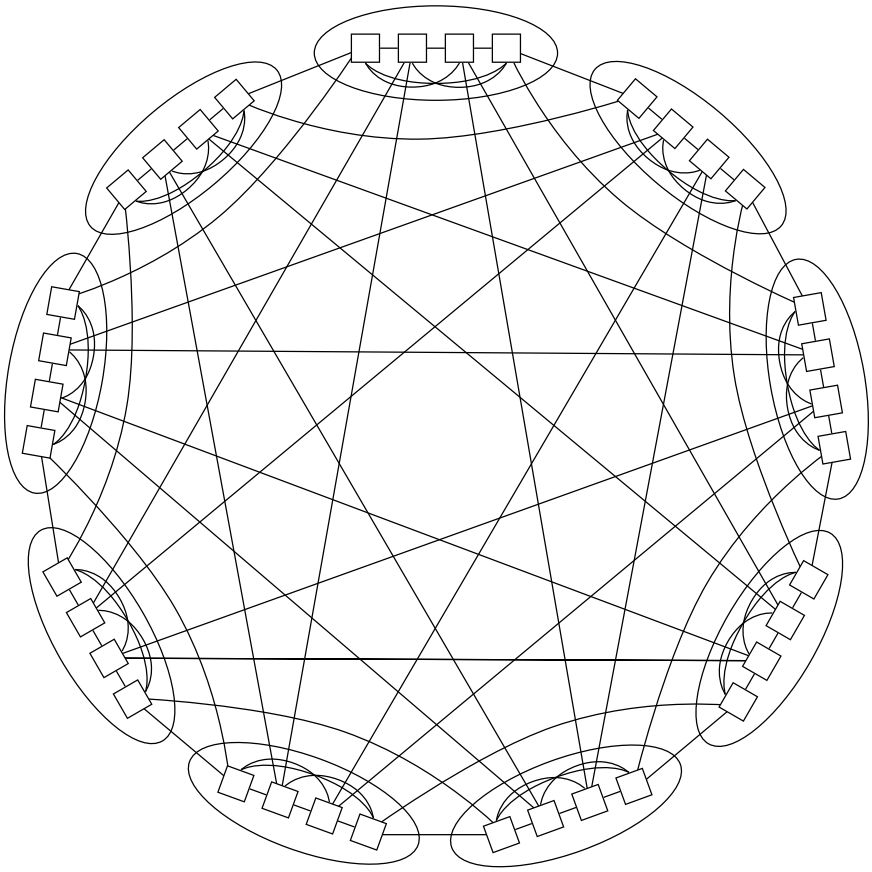


Fig. 2.5 Example of a dragonlfy topology with nine groups.

Applying this rule results in a maximum ratio from global to local links of 1/3. Additionally, the dragonfly uses generally fewer channels than the other investigated topologies. This comparable low number of links in addition to a remaining amount of network traffic rises the risk of congestions. To prevent these, special requirements must be met by the routing in order to balance the traffic evenly over the network. Furthermore, depending on the number of groups and the number of global links, it is not ensured that all groups are fully interconnected. While the original paper does not provide any information about this layout, various schemes for global links exist. Palm tree is one commonly used scheme and is also applied in the context of this work. In this pattern the nth global link of a group connects to the nth group, the $(n+1)$th link to ne $(n+1)$th group, etc. Formally, port i with $i \in (0, ..., s-2)$ of switch j with

$j \in (0, ..., s-1)$ is connected to group $(i+j+1) \mod s$, where s is the total number of switches [30].

Derived from this topology, there exist other topologies that aim to tackle particular limits of the dragonfly, such as slimfly that aims to ensure better resiliency than the dragonfly [29] and dragonfly+, which increases scalability [31]. However, the focus of this work remains on the original dragonfly topology.

2.2.2 Routing

While protocols in the data link layer define the procedure of data exchange between two distinguished nodes connected by one link and topologies define the physical layout of the interconnection network, the routing provides the formal description of the route a message is traversing along in the network. In particular, the routing algorithm reduces the set of possible paths to a limited set of legal paths [23], whereby legal paths must not be necessarily the shortest path between two nodes. Usually, routing algorithms are tailored to particular topologies or classes of topologies. They can exploit certain topology properties to provide different guarantees or to make certain performance trade-offs [23]. For example, some commonly used routing algorithms depart from the shortest path in order to achieve a load balancing over the network and reduce the probability of occurring congestions. Although there exists a large variance in routing algorithms, the ones used in the context of this work are introduced here.

Generally, there are two major classes of routing algorithms: deterministic and adaptive routing. The former are statically computed and there is one deterministic path between every source and destination pair. This allows the network to initialize the routing during set up time and store the calculated results in a fast Look Up Table (LUT). Therefore, there are no further routing computations during execution time necessary. The second advantage of one predefined path between source and destination is the guaranteed packet ordering since all buffer structures are First In First Out (FIFO)s.

The latter class, by contrast, makes it routing decisions during execution time. The ability to adjust routes dynamically to current conditions can provide significant performance advantages, especially in congested situations. For example, if there is a hot spot in the network, where many traffic flows combine, the routing can adjust to this situation and redirect single paths. However,

this comes at the price of additional computing resources in the routing unit and higher complexity to observe certain guarantees, such as packet order or livelock freedom. Hereby, livelocks refer to the situation in which paths that have an unbounded number of allowed non-minimal hops from packet sources, for instance, may result in packets never reaching their destinations [15]. Particularly adaptive routing algorithms are susceptible to livelocks since different switching instances in the network can make contrary decisions which cause a packet to swing between continuously.

Deadlock

A network anomaly that commonly needs to be prevented by the routing is the deadlock. A deadlock occurs, when some packets cannot advance towards their destination because the buffers requested by them are full [16]. If multiple packets block and request mutual resources, they or locked forever. In particular, a deadlock arises, when the following necessary conditions are met [19]:

- **Mutual exclusion:** At least one resource must be held in a non-shareable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released. For instance, a packet is stored in the buffer is holding the memory space exclusively.
- **Hold and wait:** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by another process. For example, if the packet in the buffer is ready to be sent, but the flow control has no credits.
- **No preemption:** Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it after that process has completed its task. The output buffer in which the packets were stored is only released when the packet was transmitted successfully.
- **Circular wait:** A set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes must exist such that P_0 is waiting for resource held by P_1, P_1 is waiting for a resource held by P_2, ..., P_{n-1} is waiting for a resource held by P_n and P_n is waiting for a resource held by P_0.

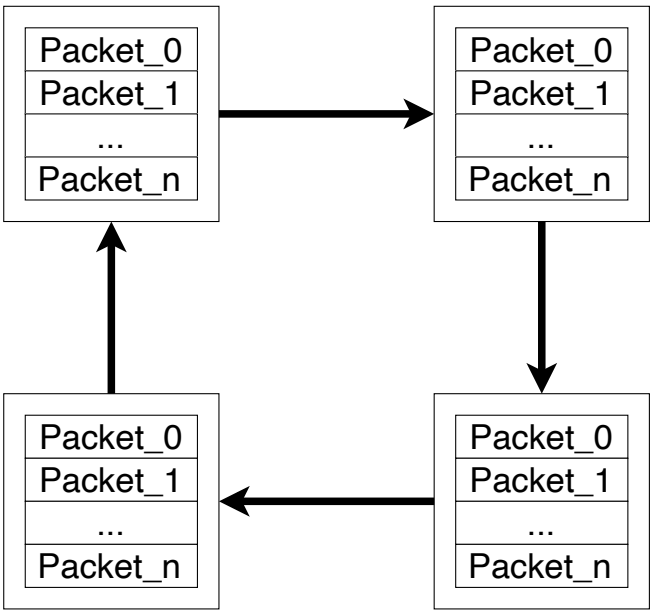


Fig. 2.6 Schematical Illustration of a deadlock. Each buffer is completely filled and requesting to send packet to the next node.

Figure 2.6 depicts a deadlock it the network schematically. In this four-node ring, each buffer is filled with packets and the packets are addressed to the next node. Since the buffers are full, the flow control does not allow to send any packet and the whole network stays in this state forever.

Commonly, there are two main strategies to handle deadlocks: deadlock avoidance and recovery. In the former approach, the routing algorithm avoids deadlocks by breaking one of the four conditions. Although all conditions are evenly effective to avoid deadlocks, usually the routing algorithm selects paths in a way that they cannot form a cyclic dependency [15]. The latter approach starts from the premise that deadlocks are only exceptions and occur rarely, and there is no prevention necessary. If a deadlock situation arises, it is then identified and resolved by the routing algorithm. Two common approaches to resolve a deadlock situation are dropping certain packets, that are causing the deadlock or redirecting them to special deadlock recovery resources [15].

Dimension Order Routing

Dimension order routing is a deterministic routing algorithm that is most commonly used in mesh and tori networks. The procedure of this routing algorithm is to route packets by crossing dimensions in a particular order, nullifying the offset

in one dimension before routing to the next one [16], as depicted in Figure 2.7.

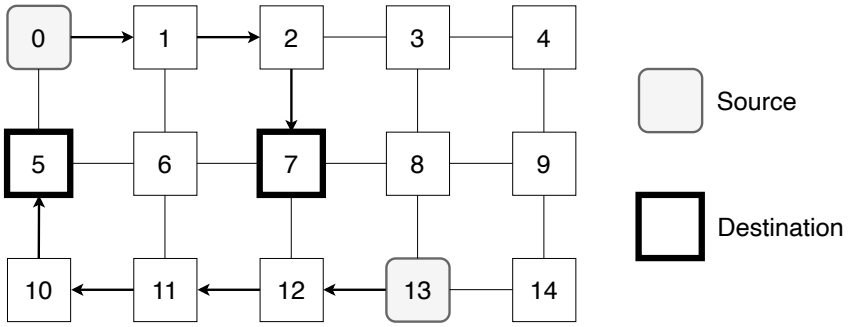


Fig. 2.7 Schematic illustration of dimension order routing.

The order in which the dimensions are run through is usually indicated by a short extension of the algorithm name, such as xyz-dimension order routing. This algorithm is easy to implement and results in a shortest-path routing, which also prevents livelocks. However, there are also some downsides. While dimension order routing guarantees deadlock-freedom for mesh topologies, this does not hold true for tori networks, since their wrap-around links cause a ring in each dimension. One simple way of deadlock prevention in dimension order routing is the introduction of Virtual Channel (VC)s [32]. The usage of multiple virtual channels on the same physical channel brakes up the circular wait condition and guarantees deadlock freedom. Another problem of dimension order routing is load balancing. One major goal of a routing algorithm is to split up the traffic evenly over the network. The dimension ordering approach of this algorithm, however, causes an imbalanced load between single dimensions. In particular, the majority of the traffic traverse to the first dimension and decreases at every following dimension.

Up-Down Routing

This routing algorithm is commonly used in tree-based topologies, such as the fat tree. This shortest-path routing algorithm routes a packet in an upward direction until the source and destination share the same parent node or switch. Then the packet is routed downward until it reaches its destination[23]. Since there is only one turn in this routing pattern no circular dependencies exists here, which ensures this routing algorithm to be deadlock-free and since no path-diversity exists also livelock free. However, this algorithm provides poor load balancing properties, due to the limited path diversity.

DESTRO

Deterministic Destination and Stage-based Routing (DESTRO) aims to improve load balancing and further exploit special features of fat-tree topologies [33]. Although the algorithm tries to balance the link utilization over the entire network, it also determines one distinguished out of multiple valid paths for each source/destination pair. To achieve this, packets are routed adaptively or randomly in the upward direction until they reach the root state and then follow a deterministic path along with the descending links to the destination leave node. While DESTRO does not ensure shortest-path routing, it results in an evenly balanced network and still provides guaranteed deadlock and livelock freedom.

Minimal Routing Dragonfly

The minimal or shortest-path routing [29] varies from two hops if the source and destination node are connected to the same router, to a maximum of five hops. When a sender S connected to switch S_S in the group G_S sends a message to destination D connected to switch S_D in the group G_D, the following scenarios are possible:

1. If $S_S = S_D$, both S and D are connected to the same switch, and packets have to traverse two hops.

2. If $S_S \neq S_D$ and $G_S = G_D$, both are in the same group but not connected to the same switch, which results in a distance of three hops.

3. $S_S \neq S_D$ and $G_S \neq G_D$ the number of hops depends on whether both switches or groups are connected by a global link which results in three and four hops, respectively. If both groups are not connected with a global link, packets have to traverse through a third switch S_A (five hops).

While this algorithm works well for load-balanced traffic patterns, performance worsens significantly for scattered, interfering traffic flows.

UGAL Routing

To improve performance for non-uniform traffic, the Valiant's algorithm[34] can be applied to the dragonfly topology. Load balancing is achieved by selecting

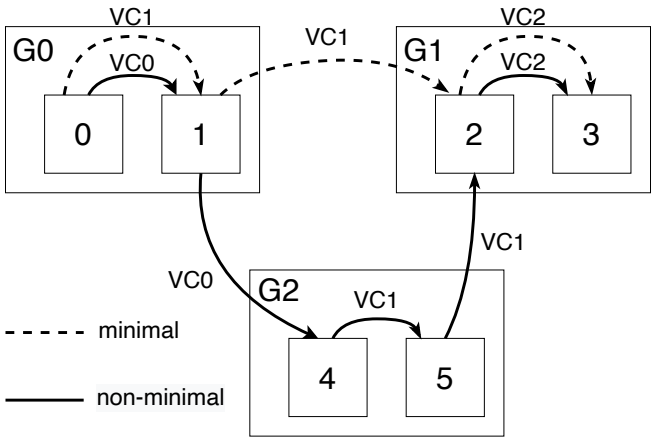


Fig. 2.8 Dragonfly routing algorithms and VC selection.

another group randomly for a packet's first hop, before it is routed to destination regularly [29]. An example of this non-minimal routing is illustrated by the red route in Figure 2.8.

Universal Globally Adaptive Load-balancing (UGAL) [35] dynamically chooses between the minimal routing and Valiant's algorithm, depending on the network delay. This delay is calculated for every packet by taking queue length and hop count into account. In an ideal version of this routing algorithm, the buffer levels for all global links are considered for the routing decision (UGAL-G). While this perfect knowledge of the network results in a completely balanced network, it can hardly be implemented, since all routers have to exchange their buffer levels permanently. A more feasible approach is UGAL-L, which only considers information provided by the local router.

Generally, both algorithms do not guarantee deadlock freedom. As in dimension order routing, VCs are used to avoid circular dependencies [32]. For the minimal routing, two VCs and three VCs for Valiant's algorithm are sufficient to prevent deadlocks [29]. Thereby, the VC changes every time the packets were transmitted on a global link.

2.3 Message Passing Interface

At the application layer, MPI is an abstract interface for transmitting data and synchronization purposes in distributed memory systems [22]. Especially in parallel HPC applications, it is today's de-facto communication standard.

When an application starts, multiple simultaneously running processes (ranks)

are initialized, which are communicating via MPI messages. These ranks can either be located at one node or spread over multiple nodes in a cluster. While the former allows to communicate via shared memory, the latter uses communication through interconnection networks. At the application level, however, MPI does not distinguish between both types.

Generally, there are two different types of messages: Point-to-Point (P2P) and collectives. The data exchange between two ranks in a point-to-point fashion is represented by an `MPI_Send()` and an `MPI_Recv()` call, respectively, that contains detailed information about the transmitting data, such as size (i.e. data type and number of elements), source and destination memory addresses, and a communicator. The communicator, hereby, describes the set of eligible ranks that take part in the communication. Additionally, P2P communication can blocking or non-blocking. The latter is especially useful to overlap computation and communication. In particular, on the receive-side and `MPI_Irecv()` is used to register memory for the incoming message. The rank can continue executing its code until the incoming data gets essential. This moment is indicated by an `MPI_Wait()` or `MPI_Waitall()` call. If the data has already arrived, the rank moves on, if not, the rank is blocked until the data has arrived.

In addition to point-to-point communication, MPI provides also collective operations for communication between a group of ranks. These collective operations are often used for synchronization purposes (`MPI_Barrier()`) , but also allow for shared data processing (e.g. `MPI_Allreduce()` or `MPI_Gather()`) or data distribution (`MPI_Alltoall()` or `MPI_Bcast()`). In the context of collective operations, the participating ranks are determined by the communicator. Generally, there is a variety of specialized collective operations that can speed up the execution and help to simplify the programming of multicomputer [16]. In order to gain further speed-ups, various vendors of HPC interconnection networks provide special hardware support for some collective operations. In particular, tree-based multicast and broadcast support are widely implemented, while many-to-one and many-to-many communication are difficult to implement since multiple senders my start at different times [16].

CHAPTER 3

Energy Proportionality in Interconnection Networks

The idea of energy proportional systems was first introduced by Barroso et al. [3] and then applied to interconnection networks later by Abts et al. [4]. It means that a system component, or the system itself, consumes only the same share of power as it is utilized. For example, if a particular component is idling half the time, the power consumption should also decrease during this idle period and the effective energy consumption should also be halved. Following recent trends, compute-intensive applications are shifted increasingly into cloud installations. Since these systems are used by many users with many diverse workloads, their actual utilization can vary a lot. Although it is rather a corner case, these systems are provisioned to handle bursts of heavy load. However, this leads to idling components in the average case with a mediocre utilization. Traditionally, the power consumption of interconnection networks tended to be rather negligible compared to other components, such as CPUs or GPUs. In order to reduce operating costs in periods of low utilization, such heavy power-consuming components became increasingly energy-proportional, which raised the relative share to the overall energy consumption of all remaining components. Accordingly, Abts et al. conclude that in the near future energy consumption of interconnection networks could amount up to 50% for data centers. Therefore, interconnection networks are mandatory to investigate to increase the energy proportionality of large-scale systems and reduce operating costs.

3.1 Power Consumtion

In order to reduce energy consumption and increase energy proportionality, it is essential to gain an accurate understanding of all power consuming units. A detailed power analysis was performed on the EXTOLL Tourmalet switch. This switch is produced using a Taiwan Semiconductor Manufacturing Company (TSMC) 65nm process, and is representative of a high-performance direct network switch. It includes a core router, six links towards the network, and a 16-bit wide PCIe G3 interface. Figure 3.1 depicts the internal structure of the Tourmalet switch [36].

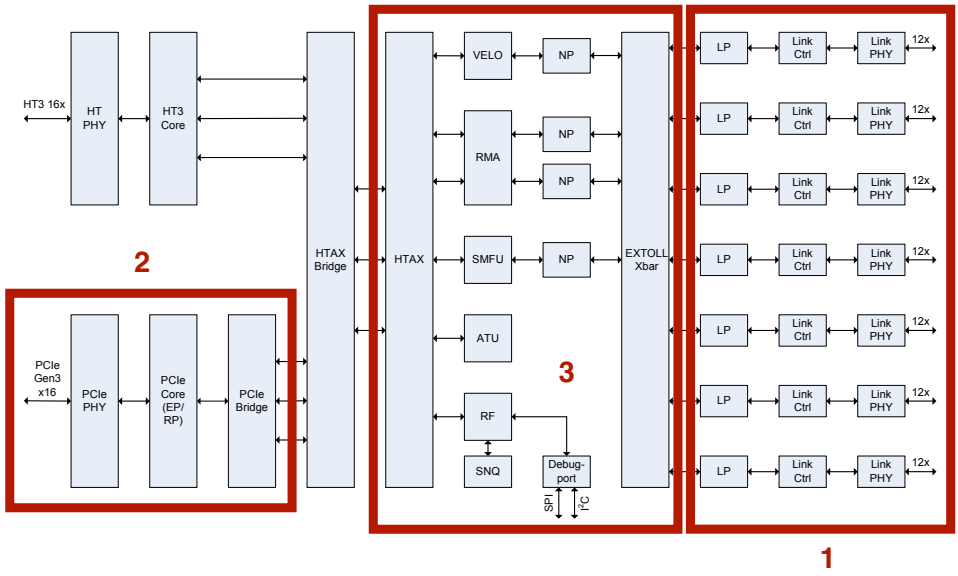


Fig. 3.1 EXTOLL Tourmalet block diagram. [36].

This rather complex scheme can be divided into three major parts, illustrated with red boxes in Figure 3.1. The links, which include all components that are part of the data exchange between multiple NICs, including linkports, link controller, and buffer, are the first major part (1). The second one (2) is the PCIe interface, which is the host interface and provides access to the host system. Last, all other functional units, such as crossbar, Remote Memory Access (RMA) unit, routing unit, etc., form the core logic (3). In this power analysis, all measurements are performed with the integrated functionality of an LTM4600 power supply

controller[1]. The results at top level granularity are shown in Table 3.1.

	Core	PCIe	Links (6)
absolute share	4.3 W	4.6 W	19.2 W
relative share	15%	16%	68%

Table 3.1 Power share of different functional components.

Transmitting components, such as the PCIe interface and links, are the main driver for the network's power consumption. Hence, links and especially serialization technology can be identified as a sweet spot for power saving. However, these components are also crucial for the network's performance. A feasible approach for power saving in these parts has the difficult task of balancing between these two concerns. This is also aggravated by fundamental design differences between serialization technology and most other components in computing systems. Therefore, a detailed understanding of their architecture and design process is necessary to gain more insights about their power consumption and energy-saving capabilities. Two fundamentally different design processes play an important role here, which are Complementary Metal-Oxide-Semiconductor (CMOS) and CML.

3.1.1 CMOS

The first and most commonly used technology to design Integrated Circuit (IC) chips is CMOS [37], [38]. The CMOS circuits are composed of N-Type Metal-Oxide-Semiconductor (NMOS) and P-Type Metal-Oxide-Semiconductor (PMOS) logic. This combination enables high noise robustness as well as low static power consumption. Generally, CMOS power consumption can be divided into a static part or constant leakage power and a dynamic part, which dissipates when current transitions.

$$P_{total} = P_{static} + P_{dynamic} \tag{3.1}$$

Figure 3.2 depicts the design of an exemplary CMOS inverter. As shown here, the output in CMOS circuits connects to one of the power supplies [39]. Hence, the voltage oscillates between the two extremes, which results in large charge

[1]https://www.analog.com/media/en/technical-documentation/data-sheets/4600fd.pdf, accessed: 2019-12-14

and discharge times. These times depend on the capacity respectively transistor size and are a limiting factor for switching frequencies. In particular, if a gate has a capacitance of C_L, the capacitor is charged to V_{DD} and then discharged to V_{SS} at each cycle and, therefore, the dissipated energy for both processes corresponds to:

$$E = \frac{C_L(V_{DD} - V_{SS})^2}{2} \tag{3.2}$$

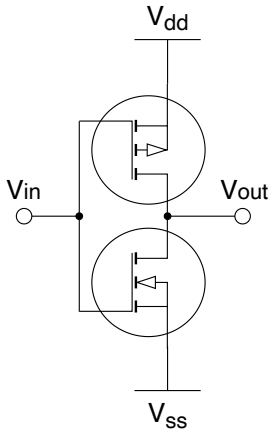


Fig. 3.2 Exmplary logic of a CMOS inverter.

Given an operating frequency f and the switching activity a (probability of a charge and discharge process per cycle), the resulting dynamic power dissipation share is:

$$P_{dynamic} = aC_L(V_{DD} - V_{SS})^2 f \tag{3.3}$$

Since the dynamic part is dominating the overall power dissipation, CMOS power consumption is approximately proportional to frequency. Hence, frequency scaling does apply well as a power-saving mechanism for this technology. Based on this assumption, Dennard et al. [1] observed that voltage and current scale linearly with the dimensions of a transistor, which means the power density remains constant. As a result, the voltage per transistor decreases, and transistors can operate at higher frequencies within the same power budget. This observation was the main drive for raising clock frequencies in the past. However, this rule does not consider leakage power, which becomes an increasing issue with shrinking feature size, and classic Dennard scaling ended at a feature size of about 130nm [40].

3.1.2 CML

Another approach to design logic gates is CML. It consists of a pull-up network, a pull-down network that implements the logic function, and a constant current source. An exemplary CML buffer/inverter is depicted in Figure 3.3. Depending on the value of V_{IN} current flows either through the left or the right branch of the network and the signal level on that side gets pulled down while the opposite side gets pulled up by the resistor, thus creating a differential output with opposite levels. Depending on the location of OUT and $\overline{OUT}$ the circuit can work as either a buffer or an inverter.

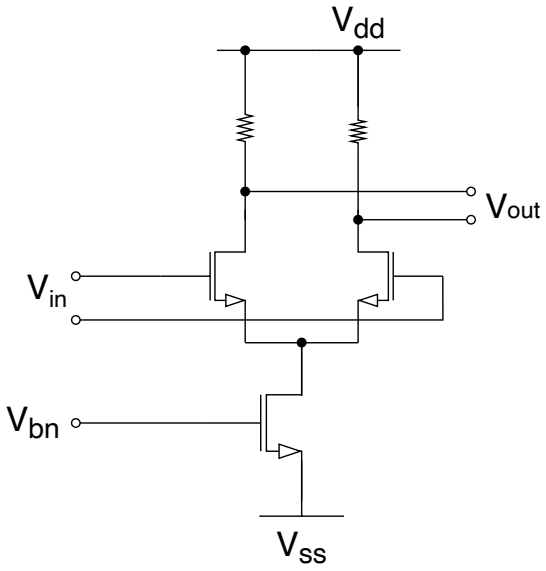


Fig. 3.3 Exmplary logic of a CML inverter/buffer.

Its design has several advantages in terms of noise immunity and emitted interference that makes CML preferable for transmission of signals over longer distances. While the differential signaling suppresses common-mode interference the lower output voltage swing also reduces capacitive coupling to other signals [38]. The power consumption of CML is mostly static and only depends on the voltage and the drawn current:

$$p_{statis} = V_{DD} I_{SS} \tag{3.4}$$

Figure 3.4 [39] summarizes the power consumption of both technologies. While CMOS shows a proportional dependency between current and frequency, CML operates at almost static current, which increases only slightly at higher

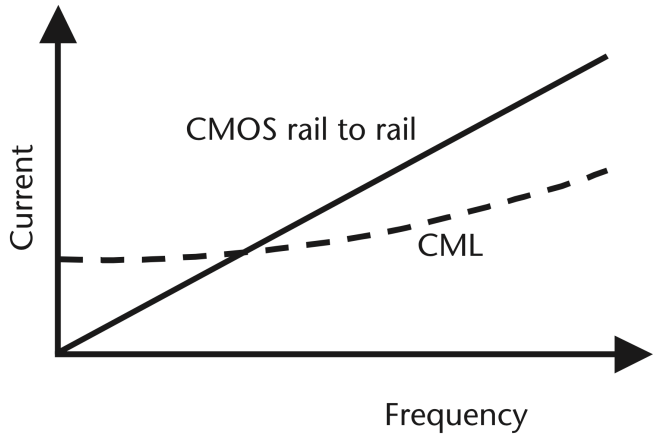


Fig. 3.4 Current/frequency relation for CML and CMOS [39].

frequencies. This means that a CMOS implementation is usually more efficient in terms of power in areas that do not require high frequencies. Since CMOS power consumption is also dependent on the toggle rate of the signals it is well suited for digital implementation where the switching probability is far from 100%. CML, on the contrary, is better suited for high-speed applications such as SerDes circuits. Its power consumption is not only independent of the operating frequency, but it is also not affected by high toggle rates that are typical for serializers which aim for an even distribution of ones and zeros during transmission.

3.2 Switch Core Power

The switch core includes multiple functional units, such as arbiter, routing unit, and the crossbar. Although the transmitting units are dominating power consumption in the previous example, this ratio can vary in other configurations. While the power consumption of linkports and periphery simply scale linearly with the radix, this section takes a closer look at the more complex power scaling of the core logic.

Scaling switch radices affects the complexity of certain functional units inside the switch core. Hence, there are no basic models available to evaluate the detailed impact of certain design parameters on overall power consumption. In order to perform these evaluations, it is necessary to implement a design for these

components. Prototyping of a chip is very time consuming and the implementation in an Field-Programmable Gate Array (FPGA)s or an Application-Specific Integrated Circuit (ASIC)s is costly as well. Therefore, this analysis is based on a pre-existing design of the High Throughput Advanced X-Bar (HTAX) chip [41], which is parameterized in order to allow an easy way of synthesizing different combinations in the design space. While the HTAX design does not provide exceeding functionality, it contains all main components of a common switch core.

The core logic is mainly designed using digital standard cells and, therefore, consists of CMOS technology. Early power estimation is already possible during synthesis with Register-Transfer Level (RTL) code as input, e.g. a Verilog design. At this stage, the numbers only give a first indication and mainly drive the cost function of the optimization step during synthesis. More accurate power numbers can be derived during physical implementation. The synthesis tool has transformed the RTL into a Gate-Level Netlist (GTL) that consists of standard logic cells for the specific technology. The power consumption for these cells has been accurately modeled by the cell library vendor. Although later steps in the design flow will add additional cells into the design (e.g. during clock-tree-insertion, timing optimization), the initial power estimation during the floorplan stage is a reasonable indication of the final power consumption and is also used for power-grid dimensioning. While feeding simulation data into the power simulation will give more accurate results because it simulates normal operating conditions another common approach is using switching probabilities and signal propagation through the design. This usually overestimates the power consumption but provides a good upper bound for power budgeting.

To gain a detailed understanding of how different design aspects affect the power consumption, frequency, and radix scaling are analyzed in the following.

3.2.1 Frequency Scaling

The first study investigates the impact of frequency scaling on power consumption. Scaling up the operating frequency enables not only a latency reduction when processing and forwarding data, but can also be essential to meet certain timing restrictions, e.g. ensuring signal integrity, when scaling switch sizes. Figure 3.5 depicts the power consumption of the HTAX core of a four-port switch operating

at five different frequencies from 400 MHz up to one GHz.

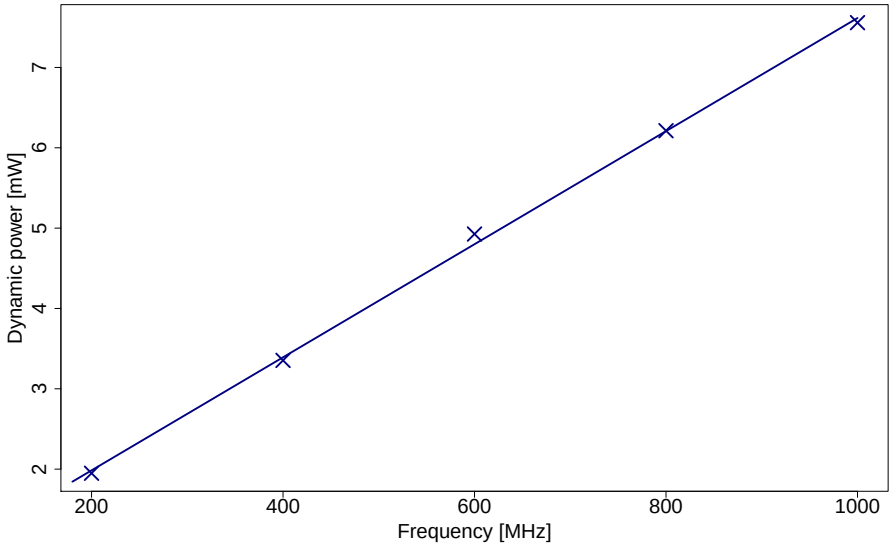


Fig. 3.5 Power consumption of 4-port HTAX core operating at different frequencies.

As Equation 3.3 suggests, the power consumption scales linearly with the operating frequency as the dynamic power share is proportional to the frequency. Whereas the static or leakage power is almost independent of the frequency and forms a static offset. While lower frequencies indicate a decreased power consumption, higher frequencies might be necessary to meet timing criteria. However, the absolute amount of power reduction equals only 5 mW, which corresponds to a frequency reduction of 60%, which suggests that the switch core only contributes negligibly to the overall power consumption and is not suitable for significant energy savings.

3.2.2 Radix Scaling

The second and more significant part is radix scaling. The power analyses in section 3.1 show, that transmission technology dominates the power consumption of a NIC. However, the used sample hardware has a comparatively low radix of only six linkports. Since an increase in the number of linkports also increases the complexity of the core logic significantly, this ratio of power consumptions changes, too.

Note that larger radices have a considerable impact on the complexity of the core logic. In particular, this increases the effort to ensure signal integrity. The most common obstacle is a negative timing slack, which means the difference between actual and required arrival time results in a negative value. All solutions, including increasing driver strengths or physical size of the gate, and introducing additional pipeline stages, result in increasing floor space or additional power consumption [42]. Therefore, high-radix switches are usually internally composed in a hierarchical layout of multiple smaller switches. While all samples in this study are based on a single crossbar, the three largest configurations, e.g. 72, 96, and 124 ports, produce a negative timing slack, which results in a further increase of power consumption for the actual functional layout. Figure 3.6 depicts the power consumption for different radix configurations in the 28nm HTAX design. To determine further scaling, a quadratic (red) and cubical (blue) model are applied Based on statistical parameters, a cubical model fits best to describe power consumption at different scales due to a $R^2 = 0.9997$ and a p-value of $7.65 * 10^{-11}$.

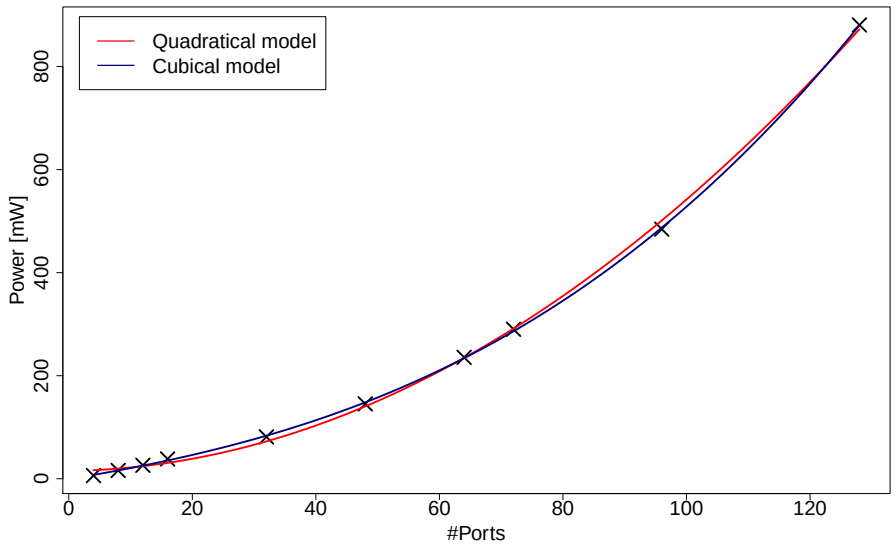


Fig. 3.6 Power consumption of switch radices at scale.

To gain a better understanding of this power consumption Figure 3.7 depicts the static leakage power (red) and the dynamic share of the power consumption (blue). Again, for both power components, the cubical model fits best, which is plotted here. Although both are increasing cubical with more linkports,

the dynamic exceeds the static share by almost a factor of 20 in absolute measures. However, even in the largest configuration of 124 ports, the total power consumption of the core logic is less than a watt. Even considering the additional effort that has to be made to address the negative slack in the static timing analysis when implementing this design, the increase in power is within the same magnitude as the original design.

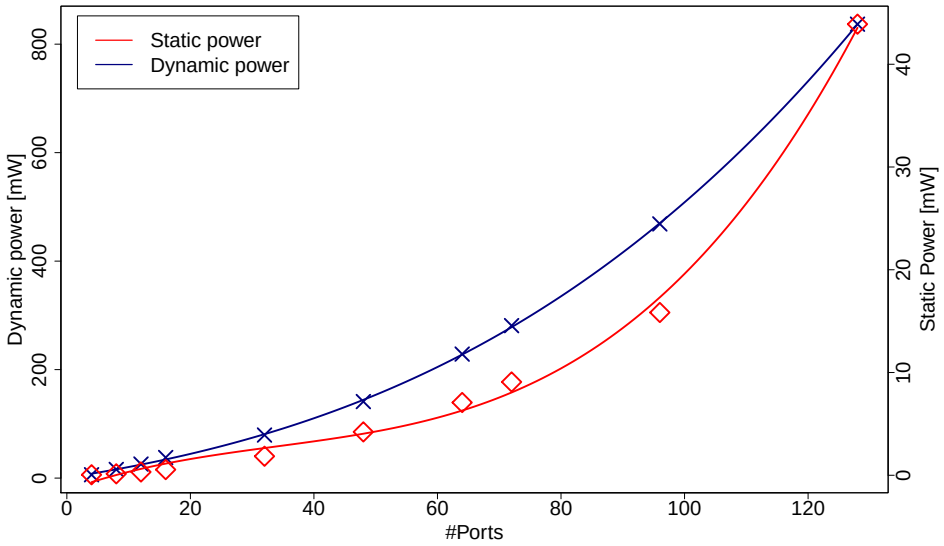


Fig. 3.7 Switch core power for different radices divided in a static and a dynamic part.

In summary, although switch radix scaling has an exponential impact on the power consumption, the absolute power consumption of the core logic remains significantly smaller than the power consumed by the serialization logic in the linkports. Due to design obstacles, a further radix scaling would potentially be driven by hierarchical designs that keep the same port/core ratio than further scaling a single-core logic. Hence, linkports are the main contributors to a NIC's power consumption and are key for energy savings.

3.3 Link Power

As the analysis in section 3.1 describes, are links the main contributor to the overall power consumption of a NIC and, therefore, provide the highest potential for energy savings. However, they are mostly designed in CML technology, which

is why frequently used energy savings mechanisms from other CMOS components may not apply for them.

3.3.1 Design

Although there are a variety of different interconnection technologies available, the basic design of a link applies to all of them. A design scheme of a link from the investigated hardware is presented in Figure 3.8.

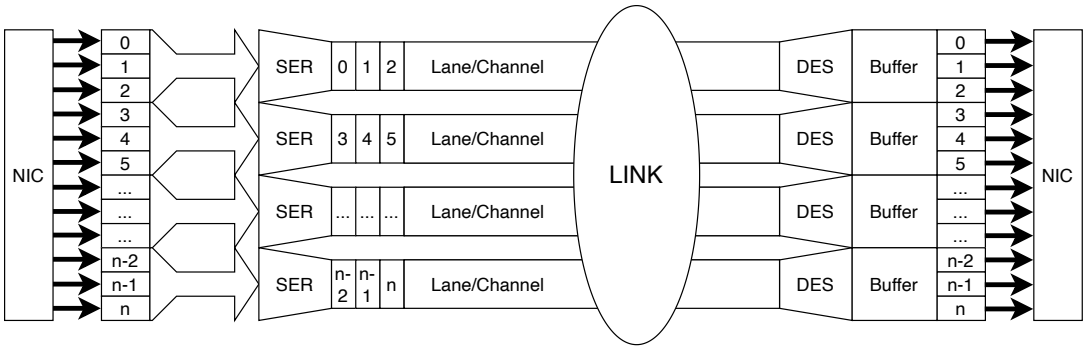


Fig. 3.8 Design example of interconnection link.

The left side shows the transmitting and the right side the receiving parts of a link cross-section. At the most left part, there is a parallel data stream coming in from the NIC, which is split according to the number of parallel lanes or channels inside one link. For each lane, this (still parallel) data stream is then processed by the serializer to a serial data stream at a higher frequency. Although the sender and receiver operate at the same frequency, it is not assured that there is no phase shift between their clocks. Therefore, the PLL or Delay-Locked Loop (DLL) at both sides align these phases and create a uniform clock domain for both sides. After being serialized, the data stream is then transferred on the lanes or channel to the receiving side, where the data is checked for correctness, synchronized again, and stored at the input buffer.

3.3.2 Power Scaling

As the largest contributor to the overall power consumption, links are the best starting point for reducing energy consumption. Due to CML, links differ fundamentally from other components, such as processors. Therefore, energy-saving approaches for these components do not necessarily apply for interconnection

networks. Based on the experimental setting in described in Section 3.1, the most effective approach is analyzed.

Without fundamental changes in the hardware design of NICs, energy savings are based on a trade-off between power and performance. Since components in a computing system are usually never fully utilized the entire time, the performance can be adjusted to the current requirements and this performance reduction can then be translated to reduced power consumption. For interconnection networks, there are two different approaches to reduce performance in terms of bandwidth, which enable power saving: frequency and link width scaling. The effects of both on power consumption are depicted in Figure 3.9. The values at the x-axis illustrate the number of active lanes, different curves indicate different operating frequencies and the y-axis shows the respective power consumption for each configuration. The links in this EXTOLL design consists of twelve parallel lanes, but they are grouped into three quads á four lanes. This pooling allows to reduce overhead in terms of space and power due to the replication of all units at lane granularity. Although these lanes work independently, they are only manageable at quad granularity since they share the same PLL for instance. However, these limitations only apply for this analysis and do not pose a general technical constraint.

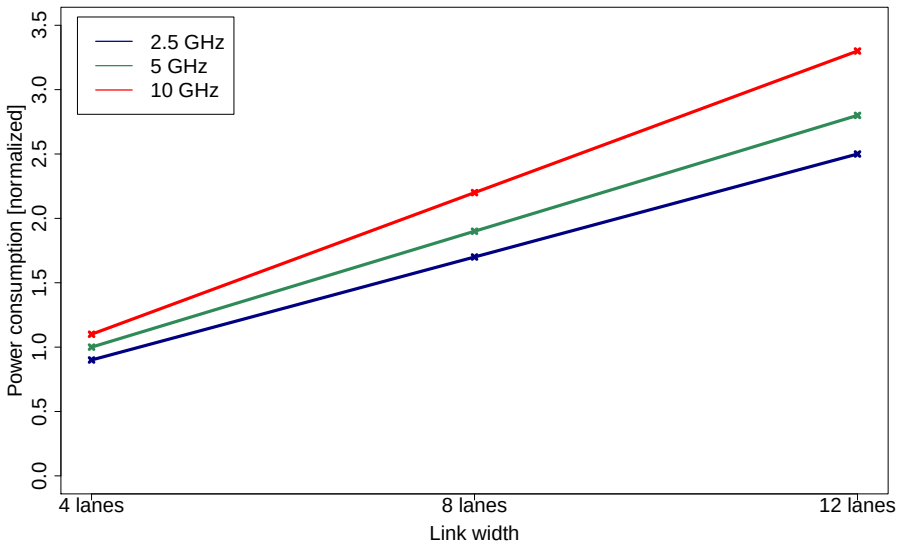


Fig. 3.9 Effects of frequency and link width scaling on power consumption.

Frequency Scaling

The first approach, frequency scaling, which is also often used in other CMOS-based components, appears to be less effective. The amount of saved power is indicated by the difference between the lines at one particular spot on the x-axis. The savings are expectedly small and vary from 18% to 24% per link when operating at a quarter of the base bandwidth. While the underlying CML technology operates mostly frequency-independent, the minor power changes are based on the residual CMOS parts.

Link Width Scaling

The more effective approach is link width scaling, indicated by different positions at the x-axis. The idea is to adjust bandwidth to the current utilization and, thereby, power consumption, by switching on and off single parallel lanes inside one link. In this clock-gating technique, switched off lanes are removed from the clock tree and do not consume power while being in this state. This enables power savings of 64% to 66% per link when reducing bandwidth to one third.

Transition Time

However, all bandwidth changes impact the performance because of their set up time. The only change that is performed almost immediately is switching off lanes. Every other adjustment to the bandwidth causes the link to be retrained. This training process is required to ensure word alignment and PLL and DLL locking. PLL and DLL are necessary to provide a high frequency, low jitter reference clock for the transmitting and receiving side. The training is performed by sending training patterns (predefined sequences of characters) and, therefore, links are not able to transfer any data during this process. The time it takes for a link to finish the training and adjust to a new bandwidth, is referred to as transition time and potentially harms the performance of an application. This is not only costly in terms of performance, but a higher execution time also increases energy consumption and can undo all savings by reduced power consumption.

3.4 Optical Links

The previous studies take only electrical links and networks into consideration. Recently, a transition from electrical to optical networks is a common trend. This trend will also affect the power consumption of interconnection networks. Optical interconnects provide multiple advantages over traditional electrical interconnects, such as lower power consumption, higher bandwidth, and wider distances.

However, this is only true for full optical networks. While optical cables are already widely used in HPC to bridge longer distances between multiple racks, for instance, there are no commercial optical switches available yet. Optical switches are difficult to realize because the optical signal, i.e. light, must be stored in a buffer-like structure until the header is evaluated and the switching decision has been made [43].

3.4.1 Overview

In today's photonic technologies, all required network components can be fabricated as part of a chip using CMOS technology, which not only results in a small device footprint but also in better energy efficiency than common network technologies [43]. The most common approaches use multi-wavelength laser sources. This laser source is filtered by its wavelength using modulators, such as microrings [44] or Mach-Zehnder modulators [45]. Using multiple rings enables separating multiple wavelengths from the light source and transmitting them on the same optical fiber [46], [47], similar to multiple lanes inside one electrical cable.

At the receiving side, there is a wavelength-selective filter, e.g. such as presented by Cheung et al. [48], to split the signal again and guide the single wavelengths to the respective detector [49]. Transmitting multiple signals on the same physical fiber not only reduces hardware costs but also eliminates the pin-limitation problems of electrical chips and cables [50].

3.4.2 Limitations

While there are multiple benefits from photonic interconnection networks, they also provide numerous technical challenges. The most important one is the

missing ability to buffer messages. Currently available network designs require routing decisions at each switch. The interpretation of the header and the switching the internal logic takes at least a few cycles in which the optical packet needs to be stored. Therefore, buffering packets and messages is crucial for switches. However, this requires either conversion into electrical signals or a new storage medium since light, unlike current, is constantly moving. Congestions and interference of different packets inside one switch are especially challenging because of variations in the time it takes to resolve them. Although there are multiple approaches for fully optical switches introduced in academia, e.g. [51]–[53], they all face technical challenges and none of them has gained acceptance in industrial implementations yet. Hence, currently it is not sure which technique will prevail.

The lack of optical switches makes it necessary to follow hybrid approaches and transform the optical signal into an electrical one on the receiving side and vice versa at the transmitting side. Instead of decreasing power consumption by eliminating power-intensive serialization hardware, the power consumption increases, due to additional optical/electrical converter at the link level.

Optical interconnection networks are becoming increasingly important for future networks. However, they also benefit from improving electrical links, since these networks are building on top of electrical logic. Therefore, optical links are not separately considered in the context of this work.

CHAPTER 4

Application Analyses

Network utilization is highly dependent on the communication pattern of the executed application. While CPUs and memory are utilized more frequently, interconnection networks are accessed in a rather sparse pattern. As shown in the previous analysis, idle times in the network are most suitable for link width scaling, particularly when they are long enough to compensate transition times during link reconfiguration. Hence, analyzing the characteristics of communication patterns is key to a deeper understanding of the way interconnects are used and how to exploit this usage in order to reduce energy consumption.

These studies can be performed at various detail levels, where greater details result in higher complexity. However, the more measurements are included, the more complex are not only the analyses but also the measuring itself. For instance, it is not sufficient to record traces and measure CPU utilization at the same time, since measuring the one affects also the other. While clock rates and utilization are measured via performance counter on system-level, traces are usually recorded through code instrumentation. This instrumentation causes a runtime overhead and affects also the CPU utilization. In order to gain valid results, metrics that interfere with each other have to be recorded in distinguished runs, which not only increases the complexity for recording but also can blur the results, if effects such as scheduling decisions or third party effects between the certain executions appear.

In the first part of this chapter, the SONAR tool is introduced. It is used for common evaluations of network traffic as well as verbosity which additionally

relies on processor performance counter. These analyses are only performed on a small scale due to the measurement complexity of these performance counter. Furthermore, public available network traces at different scales are analyzed for their locality properties and energy-saving potential in the latter part.

4.1 SONAR

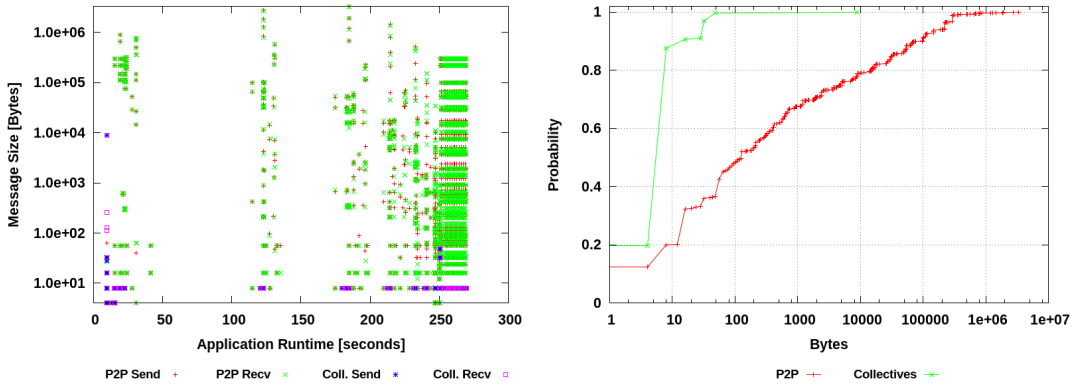
Simple Offline Network AnalyzeR (SONAR) [54] is a tool, which is able to derive network performance metrics from communication traces and performance counter of HPC applications offline. The tool was designed in the context of this work to evaluate HPC applications for their energy-saving potential and to be able to easily implement possible extensions with new metrics.

4.1.1 Metrics

An important step to develop an energy-saving approach for interconnection networks is to understand how applications that run on these systems utilize the network. Likely the most important factor for these studies is an application's communication pattern. Typically, communication patterns are characterized by abstract metrics, such as density plots, communication calls, and volume [55]–[57]. Beyond that, there are few other approaches for communication patterns characterization. Chodnekar et. al. [58] introduce message generation frequency, spatial distribution of messages, and message length. This statistical regression analysis works only for workloads that perform distinct communication and computation phases. Kim et al. [59] analyze communication event locality, message size locality, and message destination locality. However, these metrics are oblivious to variations in system configuration or problem size.

For the SONAR tool the following metrics are implemented to provide useful qualitative and quantitative insight into the communication behavior of HPC applications.

Network Activity Map: This metric visualizes all point-to-point and collective messages by size and the temporal occurrence in a graph. Each network event is represented by a data-point in the plot, which can be recorded per rank or application. Different types of communication are illustrated in different



(a) Network activity map (b) Message Size Distribution Map

Fig. 4.1 Examples of the visual metrics SONAR derives from an application trace.

colors. Figure 4.1a depicts the network activity map of an exemplary workload (AMG2013).

MPI idle time: This metric focuses rather on the idle time periods than on the total idle time and SONAR determines the minimum, maximum and average periods a rank does not spend in a MPI routine. These values are indicated by the white spots in the message activity map in Figure 4.1a.

Message Distribution: SONAR uses a Cumulative Distribution Function (CDF) graph to indicate the message sizes that occur in a trace. The resulting graph depicts the probability of a message having a particular size or smaller. Figure 4.1b shows the CDF graph of the exemplary AMG2013 workload.

Message Rate: An application's message rate indicates how many messages are sent by one node in a given time interval. This metric can be interpreted as a single-value approximation of the network activity map.

Verbosity: This metric is not part of the common traffic pattern characteristics and was introduced by Rumley et al. [60]. A workload's verbosity indicates the ratio of the amount of data, sent over the interconnect, to the work that is computed. Where the work is quantified by the number of Floating Point Operations (FLOP) issued, as many HPC applications are depending on floating-point arithmetic. The FLOPs can be derived from CPU hardware performance

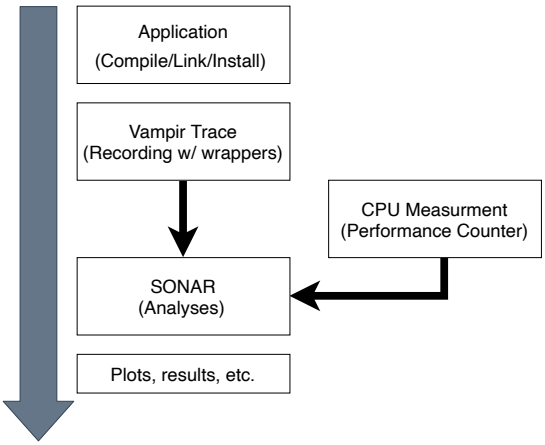


Fig. 4.2 Schematic workflow of acquiring metrics with SONAR.

counters. However, this requires independent trace recording and performance measurements since both methods interfere with each other.

All the described metrics are MPI based, which means that no conclusions about actual link or switch utilization can be drawn. However, they provide a good overview and allow to find possible sweet spots for further studies.

4.1.2 Concept

The development of SONAR is motivated by the need to comprehensively analyze previous gathered application data and the ability to easily implement new metrics if needed. Depending on the number of nodes, the complexity of the underlying problem, the communication characteristics, and other factors, these analyses become very complex and time-consuming. Hence, an efficient structure is key for the design of such a tool.

Figure 4.2 depicts the abstract concept of a SONAR analyses and the steps that are performed to acquire metrics from an application. In the first step, traces and other performance data are recorded.

The Tuning and Analysis Utilities (TAU) and the VampirTrace frameworks are evaluated for this tool. Both provide the possibility to run existing binaries without any instrumentation, for the price of a reduced amount of information. Since all information regarding MPI and communication are retained when running an existing application with run-wrappers, this method is sufficient for SONAR. However, the VampirTrace framework provides advantages in data compression and storage of collective communication, and, therefore, is selected

for SONAR. Furthermore, the Open Trace Format (OTF) library is used for storage and internal processing.

Once a trace has been obtained with VampirTrace, it contains the enter and leave timestamp of each MPI call, as well as additional information regarding each event, such as message length, type, and communicator. SONAR then uses the OTF library to access and filter the trace file before the data is further processed and examined for the various metrics. For more details, the code is publicly available[1].

Since there are no input data for SONAR publicly available, the traces and processor performance data have to be recorded by the user. Due to limited access to large scale systems, SONAR is evaluated with only a limited set of small scale traces. Because of these constraints and the limited significance of small configurations, the results are not further discussed here. However, more details on SONAR and its results are provided in the work of Lammel et al. [54].

4.2 Locality and Selectivity in Exascale Proxy Miniapps

The amount of input data for SONAR eventually requires to record data on a system since most public available data lack either trace or performance information. To gain deeper insights on network utilization, it is essential to collect data from different types of applications and systems at a different scale. These insights can be exploited to increase energy efficiency without affecting performance by an advanced system design or provide useful insights for energy-saving mechanisms. To draw a comprehensive picture of different communication aspects in HPC, exascale proxy applications, for which a sizeable amount of traces at different scales are publicly available, are classified. Communication patterns are the most important characteristic as provided by the SONAR tool. While density plots are suitable for their visualization, they do not provide a quantitative description and are less useful for objective comparisons. To address this issue and to provide potential input data for network models, this work proposes locality and selectivity as further metrics to describe communication patterns [8]. Locality is particularly useful to gain insights about actual network

[1] https://github.com/UniHD-CEG/sonar

utilization since not only the communication volume but also network distances can be taken into account. Equivalent to velocity and the traveled distance, this allows to determine the time the data is moving on the network. Although locality studies already exist, do these studies focus rather on memory locality than on network locality aspects [61]–[64].

4.2.1 Metrics

Locality in communication patterns can be applied at the application level and refers to the distance between distinguished ranks, as well as at the system level, which indicated the distance in the network in terms of actual network hops. However, both can be derived directly from MPI traces by emulating system configurations in simple models. These configurations include different topologies and a study on multi-core scaling. Besides common MPI metrics, the proxy applications are studied for the following metrics:

Locality: Locality represents the distance of communication at the MPI level and rather describes a virtual distance between two distinguished MPI ranks than an actual spatial distance between nodes. This new introduced metric can formally be described as: given a rank sourcing the communication $rank_{source}$, a rank sinking the communication $rank_{dest}$, and based on a linearization of the different rank IDs according to their numerical ID, the distance between two ranks and the respective locality are defined as

$$dist = |rank_{source} - rank_{dest}| \tag{4.1}$$

$$locality = \frac{1}{dist} \tag{4.2}$$

As Equation 4.2 shows, a higher locality describes lower distances and vice versa, where a distance of one (i.e. communication with direct neighbors) results in a locality of 100%. To minimize distortions, locality is defined as the minimal distance in which 90% of the overall traffic is communicated. This abstract metric neglects network layer effect, such as topology and or mapping, and can be evaluated based on traces, with no replay necessary.

Also note that only point-to-point communication is taken into account since global collective communication is distributed evenly over the network. However,

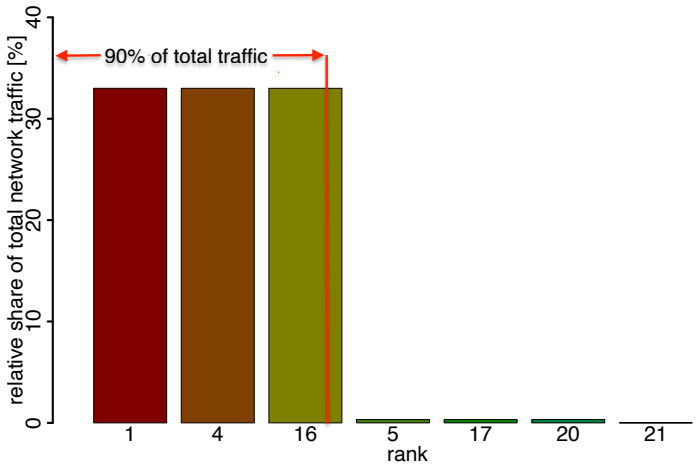


Fig. 4.3 Illustration of selectivity metric. For an exemplary rank (LULESH, rank 0), the communication volume (y-axis) to every other communication partner (x-axis) is shown.

when including more diverse collective patterns and non-global communicators, this type of communication should also be included.

Selectivity: Selectivity is the second newly introduced metric that is completely derived from the application layer and considers only point-to-point communication. Contrary to the distance of locality, selectivity describes the number of communication partners one particular rank is communicating with independent of their location in the network.

To calculate the selectivity of a given source rank, all destination ranks are determined and sorted by data volume exchanged between this pair of ranks. As locality, selectivity has a threshold of 90% of a rank's total communication volume to describes how many destination ranks participate in the communication set. This threshold is introduced because many workloads establish at least little communication across all ranks, but there are significantly fewer ranks that contribute to the majority of overall communication.

Figure 4.3 depicts the schematic illustration of the selectivity for an exemplary rank. The y-axis indicates the communication volume that is sent to particular ranks and the x-axis shows the corresponding receiving ranks.

(Average) Packet Hops: Considering locality at the network layer instead of the application layer provides additional insights about the effects of network parameters, such as mapping, topologies, or routing, on communication patterns. This work defines the network locality as the distance in number of hops between the source rank and the destination rank. Non-temporal models are used for different topologies, including a shortest-path routing algorithm each, to calculate these distances. This means that the entire metric is based on these models and no replay in a simulator is needed. However, these plain models come at the price of non-temporal data as opposed to simulations. This means that there is no information about traffic flows or interaction between messages can be derived. Although these models are simplified, the derived metrics are accurate and the results are not affected by these abstractions.

Another function of the network layer is dividing messages in packets that can be processed by NICs. Hence, MPI messages from the traces are split in the according number of packets, with a maximum payload size of 4kB. [2]. Afterward, the number of hops a particular packet has to traverse is calculated by the respective minimal routing algorithm. Due to the non-temporal character of this model and resulting in the abstinence of congestions and load balancing, shortest-path routing for all topologies is selected to provide fairness and emphasize the impact of a particular topology. Formally, packet hops are described as:

$$packethops = \sum_{p \in packets} \#hops(p) \tag{4.3}$$

It is easy to see, that the number of packet hops is highly dependent on the injection rate of a given application, respectively its communication intensity. Therefore, it is more appropriate to investigate the average number of hops one packet has to be transmitted in order to gain locality insights:

$$\overline{\text{hops}} = \frac{packethops}{\#packets} \tag{4.4}$$

This metric is particularly helpful to compare different topology/application combinations. Furthermore, it can be used to determine how good a particular topology, mapping or routing algorithm exploits locality in the communication

[2] According to Infiniband standard: https://www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf, accessed: 2020-02-16

pattern, where a lower value indicates a better locality.

Network Utilization/Energy: From the number of packet hops another metric can be derived, the network utilization. The network utilization indicates the share of execution time in which links are busy sending, respectively the compliment share represents the idle time of the network. Another use case is an estimation of the probability of congestions, including different increments in hierarchical topologies. However, without exact timing information, this can only be an indicator and no absolute measure for congestions.

Furthermore, to provide fairness for all topologies, only links and switches are considered that are transmitting data, when the number of ranks does not match the number of nodes. This is necessary since not all topologies can be configured in every size.

Network utilization is formally defined as:

$$utilization = \frac{datavolume}{BW \cdot t_{execution} \cdot \#links} \tag{4.5}$$

Where the data volume can be calculated as:

$$datavolume = \sum_{p \in packets} \#hops(p) \cdot size(p) \tag{4.6}$$

A bandwidth of $BW = 12GB/s$ is assumed, as this is also used in all further studies in this work. The number of links is a topology property and can easily be calculated for each topology by the product of diameter and number of nodes. As a result, the torus has three links per node, assuming switches are integrated into the NIC, and the fat-tree has a total number of links of $\#nodes \cdot \#stages$ (only half the links for the last stage). The balanced dragonfly is configured with the same amount of global links as there are nodes are connected to each router and twice as many routers per group. This results in 3.5 to 3.8 links per node in the configurations used for these analyses.

While the non-temporal character of these analyses prevents performance or energy-saving predictions, it enables estimations for the upper bound for energy-saving possibilities. As in Equation 4.7, the total network is the product of a link's power consumption, the execution time, and the number of links.

$$E_{total} = P_{link} \cdot t_{execution} \cdot \#links \tag{4.7}$$

The upper bound, however, represents energy saving in an ideal network. This means, that whenever data is injected into the network, it is transferred at full network speed, which provides also the highest energy efficiency in terms of joule per byte, and links are consuming no power when idling. Additionally, this assumes that links can adjust their link width immediately, which represents a transition time of 0. The actual usage of the network is determined the network utilization, which means that the minimal network energy can be calculated as:

$$E_{min} = P_{link} \cdot t_{execution} \cdot \#links \cdot utilization \tag{4.8}$$

$$E_{min} = E_{total} \cdot utilization \tag{4.9}$$

Although this metric does not reveal results of any real energy-saving mechanisms, it is an accurate measure for the minimum energy that is required by the network and represents the upper bound for network energy savings.

4.2.2 Methodology

This section provides an overview of the analyzed applications and the hardware model that is required to determine the metrics that focus on the network layer. These simple models enable a fast and in context of these metrics accurate evaluation without the need of complex and protracted simulations.

4.2.2.1 Applications

Exascale proxy mini-applications aim to represent the most common kinds of workloads that are expected to be run on exascale systems. This collection provided by the DoE includes climate models, fluid dynamics, molecular dynamic simulation, or other compute-intensive scientific applications. The goal of this selection is to provide information and guidance for designing hardware or even systems that are expected to run these applications. One benefit of this broad selection is the variety of different communication patterns that are used on future HPC systems. A large online repository of proxy mini-application traces at different scales and problem size are provided by the Sandia National Laboratories [3].

[3] https://portal.nersc.gov/project/CAL/doe-miniapps.htm, accessed: 2019-11-20

Although this repository provides a huge amount of different traces, several obstacles exclude particular applications or scales for these analyses. The main issues are custom communicators, such as particularly custom-created carts of ranks (i.e. by using MPI_Cart_create) or changes in the communicator size (MPI_Cart_sub). These changes in the communicator during execution time can affect identifier and it can not be ensured that ranks have still the same identifier as they had before the communicator shift. The link of a particular node to its identifier is crucial for consistent results in all studied metrics. Hence, traces with these types of collectives are not considered in the context of this work.

The selection of traces that is used for the following analyses is shown in Table 4.1. Table 4.1 also provides a brief overview of the fundamental MPI characteristics, such as communication volume (Vol.) of collective and point-to-point communication, total execution time, and the respective throughput (Vol./t). Applications that are marked with a star (*) make use of MPI Derived Data Types (DDT). The size of DDT is set in the code and needs to be stored as additional information in the traces. Because the used traces do not provide this information, the size of the used data type is set to one byte. Although MPI DDTs appear to have bigger data sizes, this enables scaling results easily to the actual size if needed.

4.2.3 Hardware Parameters

The number of hops refers to the distance a certain message has to travel from its source to destination. This distance can be determined with a formal definition of a particular network topology as well as a routing algorithm. Three different topologies, each with a minimal routing algorithm, are assumed to be representative. The first one is a 3D torus, that is equipped with six-port NICs, which are directly integrated into nodes. Furthermore, a fat-tree that is composed of 48-port switches and the dragonfly topology are selected. Configurations of the latter are selected accordingly to the load balancing suggestions by J.Kim et al. [29], with $a = 2h = 2p$.

While all trace can be mapped on the torus correspondingly to their size, configurations for fat-tree and dragonfly cannot always be chosen accordingly to the number of ranks. To provide fairness in the topology comparisons, only

Application	Ranks	Time [s]	Vol. [MB]	P2P [%]	Coll. [%]	Vol./t [MB/s]
AMG	8	0.03	3.0	100.0	0.00	116.3
	27	0.16	13.6	100.0	0.00	86.98
	216	0.30	136.9	100.0	0.00	461.5
	1728	2.92	1208	100.0	0.00	413.7
AMR Miniapp	64	12.93	3106	99.66	0.34	240.3
	1728	42.69	96969	99.45	0.55	2271
BigFFT (Medium)	9	0.18	299.2	0.00	100.0	1659
	100	0.50	3169	0.00	100.0	6340
	1024	1.89	32064	0.00	100.0	17003
Boxlib CNS large (*)	64	572.19	9292	100.0	0.00	16.24
	256	169.05	15227	100.0	0.00	90.08
	256	150.92	15227	100.0	0.00	100.9
	1024	67.54	34131	100.0	0.00	505.4
Boxlib MultiGrid C	64	231.42	23742	99.94	0.06	102.6
	256	62.01	44535	99.95	0.05	718.2
	256	60.28	44535	99.95	0.05	738.8
	1024	20.88	75181	99.94	0.06	3600.9
CESAR MOCFE (*)	64	0.38	19.0	5.01	94.99	50.3
	256	1.10	81.6	5.51	94.49	74.11
	1024	3.95	686.2	6.96	93.04	173.9
CESAR Nekbone (*)	64	11.83	5307	100.0	0.00	448.8
	256	3.17	1272	50.66	49.34	401.8
	1024	5.15	13232	99.98	0.02	2568.8
Crystal Router	10	0.14	133.8	100.0	0.00	930.3
	100	0.71	3439.9	100.0	0.00	4854
	1000	1.28	115521	100.0	0.00	90491
EXMATEX CMC 2D Multinode	64	842.80	16.0	0.00	100.0	0.0190
	256	208.44	16.1	0.00	100.0	0.077
	1024	58.85	16.4	0.00	100.0	0.279
EXMATEX LULESH	64	54.14	3585	100.0	0.00	66.23
	64	44.03	3585	100.0	0.00	81.43
	512	50.24	33548	100.0	0.00	667.8
FillBoundary	125	2.32	10209	100.0	0.00	4393
	1000	5.26	92323	100.0	0.00	17549
MiniFE	18	59.70	1615	100.0	0.00	27.06
	144	61.06	16586	99.99	0.01	271.63
	1152	84.75	147264	99.96	0.04	1737.7
MultiGrid_C	125	0.77	374	100.0	0.00	4889.0
	1000	3.57	2973	100.0	0.00	832.83
PARTISN (*)	168	2.2E+6	42123	99.96	0.04	0.02
SNAP (*)	168	1.2E+6	128561	100.0	0.00	0.11

Table 4.1 Overview of MPI-based exascale proxy applications.

actually used links are considered in this model. For all topologies exist a variety of different routing algorithms that differ significantly in their complexity. However, since traffic flows and load balancing are not considered here, shortest path routing is used for all topologies in order to emphasize their particular locality features. An overview of all different configurations is shown in Table 4.2 (*rad* being radix, *st* being number of stages).

	Torus		fat-tree		Dragonfly	
Size	(x,y,z)	Nodes	(rad, st)	Nodes	(a,h,p)	Nodes
8	(2,2,2)	8	(48,1)	48	(4,2,2)	72
9	(3,2,2)	12	(48,1)	48	(4,2,2)	72
10	(3,2,2)	12	(48,1)	48	(4,2,2)	72
18	(3,3,2)	18	(48,1)	48	(4,2,2)	72
27	(3,3,3)	27	(48,1)	48	(4,2,2)	72
64	(4,4,4)	64	(48,2)	576	(4,2,2)	72
100	(5,5,4)	100	(48,2)	576	(6,3,3)	342
125	(5,5,5)	125	(48,2)	576	(6,3,3)	342
144	(6,6,4)	144	(48,2)	576	(6,3,3)	342
168	(7,6,4)	168	(48,2)	576	(6,3,3)	342
216	(6,6,6)	216	(48,2)	576	(6,3,3)	342
256	(8,8,4)	256	(48,2)	576	(6,3,3)	342
512	(8,8,8)	512	(48,2)	576	(8,4,4)	1056
1000	(10,10,10)	1000	(48,3)	13824	(8,4,4)	1056
1024	(16,8,8)	1024	(48,3)	13824	(8,4,4)	1056
1152	(12,12,8)	1152	(48,3)	13824	(10,5,5)	2550
1728	(12,12,12)	1728	(48,3)	13824	(10,5,5)	2550

Table 4.2 Configurations for different topologies at scale.

As shown in Table 4.2, about one quarter of the investigated applications heavily rely on collective communication. The execution of collective operations in HPC systems can differ between network technologies or vendors, due to custom implementations, such as special broad- and multicast support. Since this work follows a technology-independent approach, non of these customized solutions is implemented.

The model in this work implements a simpler and robust approach: collectives are translated to point-to-point messages, which then map the pattern of the particular operation. For example, an all-to-all call is performed by all ranks sending a p2p message to every other rank. In particular, there are no hardware-implemented tree structure or similar, which allow distributing data

more efficiently. Although this pattern does not represent today's hardware, it ensures that the network is maximally utilized to give a stable estimate for utilization and network energy. Additionally, due to the non-temporal character of these studies, the overhead does not affect performance, since interactions between messages, e.g. congestions, are not considered here.

4.2.4 Results

Locality can be investigated at different levels. Thus, the studied metrics differ also in their level of abstraction. The newly introduced metrics are analyzed in the application layer, which allows to extract information directly from the traces. The remaining metrics are located in the network layer and demand for additional network information, which are here provided by a simple, non-temporal network model. In addition to the introduced metrics, the network layer analyses contain another brief study about multi-core effects on network traffic. Both categories are presented in the following. More details about the trace format and the analyses are provided in Chapter chapter 5.

4.2.4.1 Application Layer

Rank locality and selectivity studies can be directly performed on traces without any additional information needed. Also, this first analysis focuses only on point-to-point communication. Because only global communicators are considered, collectives can be assumed as a constant bias on the network, while communication variations in distance and number of partners occur from point-to-point messages.

An overview of all applications and their rank distance and selectivity is shown in Table 4.3. Klenk and Fröning [65] introduced the peers metric, which describes the peak number of peer ranks any rank is addressing via point-to-point communication in a given application, and found that the number of peers is often significantly smaller than the total amount of ranks. This metric is particularly interesting in comparison to the selectivity, which excludes ranks that only contribute in a minor way to the overall communication. However, even the relatively small number of peers for most workloads indicates point-to-point communication only happens between subset of ranks instead of being distributed over all ranks.

Workload	Ranks	Peers	Rank distance (90%)	Selecitivity (90%)
AMG	8	7	3.7	2.8
	27	26	8.7	4.2
	216	127	35.8	5.2
	1728	293	143.8	5.6
AMR_Miniapp	64	39	27.1	8.3
	1728	490	348.3	13.0
Boxlib CNS large (*)	64	63	35.1	5.7
	256	255	109.2	5.4
	256	255	109.2	5.4
	1024	1023	661.5	20.8
Boxlib MultiGrid C	64	26	27.1	4.4
	256	26	54.3	4.4
	256	26	54.3	4.4
	1024	26	109.1	4.9
CESAR MOCFE (*)	64	12	51.3	8.9
	256	20	195.3	14.0
	1024	20	771.8	13.3
CESAR Nekbone (*)	64	27	15.8	4.8
	256	15	28.4	5.4
	1024	36	127.9	10.2
Crystal Router	10	4	6.4	3.0
	100	8	44.3	5.8
	1000	11	334.3	8.9
EXMATEX LULESH	64	26	15.7	4.5
	64	26	15.7	4.5
	512	26	63.7	5.0
FillBoundary	125	26	42.3	4.8
	1000	26	219.1	5.3
MiniFE	18	8	7.4	3.4
	144	22	31.5	4.6
	1152	22	91.8	5.1
MultiGrid_C	125	22	59.7	5.5
	1000	22	392.0	5.4
PARTISN (*)	168	167	13.8	3.4
SNAP (*)	168	48	139.1	9.8

Table 4.3 Workload characteristics in different application-layer metrics.

Rank Locality

Rank locality and distance indicate the average communication distance between ranks weighted by the transmitted data volume. The results for rank distances

at all scales are provided in Table 4.3. Rank locality, however, is the reciprocal of the rank distance, which means a higher locality indicates a closer distance. Although there is no obvious correlation between rank distance and the number of ranks or peers, this metric allows to draw conclusions about the communication pattern and problem structure, respectively. For instance, nearest neighbor communication is widely used in scientific applications and provides good locality properties. Assumed this corresponds to communication with a distance of two or less, which remains constant at all scales, this pattern could be identified by a locality of more than 50%. This is also true for scattered messages to other ranks since only the nearest 90% of data volume are considered.

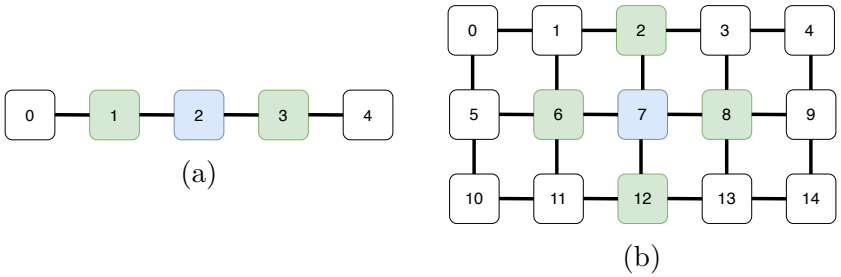


Fig. 4.4 Nearest neighbors (green) of a particular node (blue) for one dimensional problem (a) and two dimensional problem (b).

However, this reflects only a one-dimensional perspective. A spatial distance of one in all other dimensions results in a much low locality. This problem is shown in Figure 4.4, which depicts nearest neighbor schemes in one (Figure 4.4a) and two (Figure 4.4b) dimensions. In the former, the linear numbering matches the spatial setting of neighbors, the latter shows longer distances between neighbor ranks in the y-dimension, for instance from rank seven to twelve. As a result, additional dimensions cause a constant offset that is determined by the number of nodes per dimension. This can be used to evaluate the dimensionality of a given workload by analyzing rank locality in different dimensions.

As shown in Table 4.3, the rank distance increases with the number of ranks for all workloads, which suggests that there is no application with a one-dimensional structure. The evaluations of more dimensions are shown in Table 4.4 for an exemplary selection of applications. CNS and MultiGrid_C show no particular correlation to one of the dimensions, which indicates that these workloads do not have a nearest neighbor communication pattern. This also coincides with the heat map depicted in Figure 4.5a. One example of an application with a

two-dimensional character is PARTISN, which has a 2D rank locality of 100%. The same pattern is also shown in Figure 4.5b. Last, LULESH and AMG have a three-dimensional structure, as indicated by Figure 4.5c and Figure 4.5d and quantified by a rank locality of 100%.

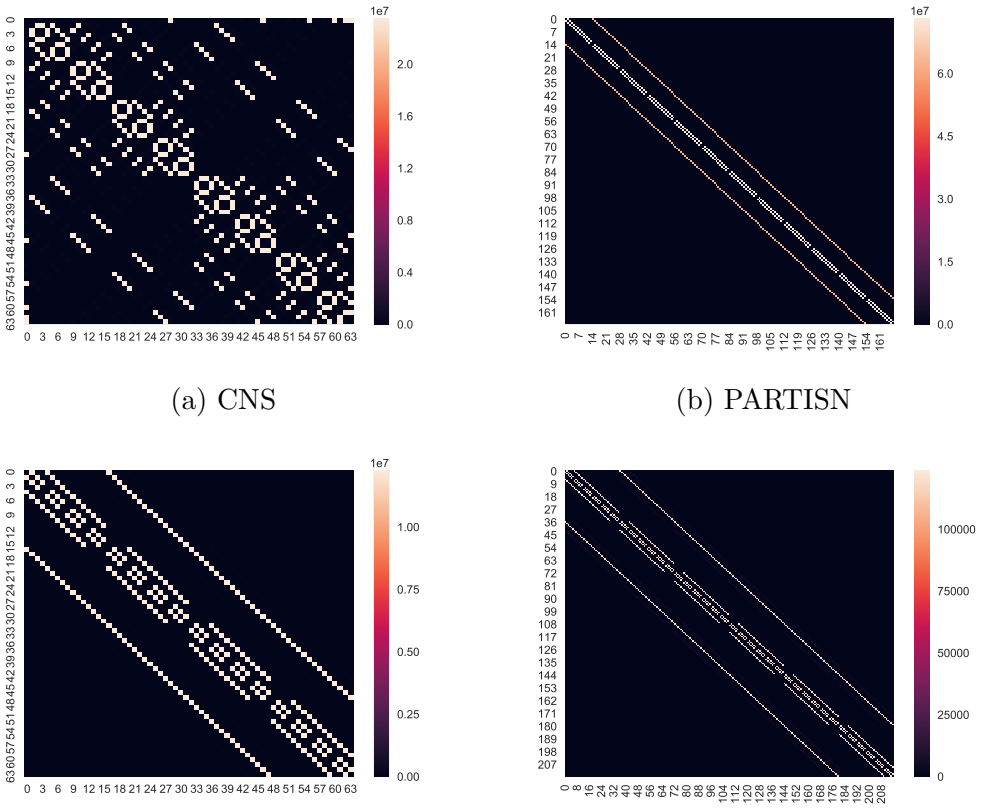
		Rank Locality		
Workload	Ranks	1D	2D	3D
AMG	216	3%	17%	100%
	1728	1%	8%	100%
Boxlib CNS large	64	3%	13%	21%
	256	1%	8%	13%
	1024	0%	3%	7%
EXMATEX LULESH	64	6%	24%	100%
	512	2%	6%	100%
MultiGrid_C	125	2%	6%	17%
	1000	0%	3%	9%
PARTISN	168	7%	100%	22%

Table 4.4 Exemplary workloads for different dimensionalities in rank locality.

Selectivity

The selectivity metric indicates the number of ranks that make up the largest part (i.e. 90%) of the overall point-to-point communication of a particular rank. This metric neglects positions and ranks are sorted by their exchanged data volume. Figure 4.6 depicts the trends for each application at one exemplary size. The x-axis indicates the number of ranks and the y-axis the relative amount of communication volume. In particular, selectivity equals the intersection in which a curve cuts the 90% traffic share. Although the number of ranks varies between 18 and 168, in every application 90% of the communication is exchanged between only small subsets of ten or even fewer ranks. Looking at larger scales that are shown in Table 4.3 only five configurations (AMR (1728 ranks), CNS (1024 ranks), MOCFE (256 and 1024 ranks), and Nekbone (1024 ranks)) exceed this threshold. However, even the largest configuration of 1728 ranks has a selectivity of only 13.

Additionally, the overall trend indicates that selectivity slightly increases with the number of ranks but is also slowing down, indicating a saturation. An example of this scaling trend is depicted in Figure 4.7. The curves of the



(a) CNS

(b) PARTISN

(c) LULESH

(d) AMG

Fig. 4.5 Communication patterns in heat maps (a lighter color indicates more communication in Bytes).

four AMG application's sample sizes are shifting to the right with an increasing number of ranks. Although the slope varies for other applications, the general trend remains the same. There are only three workloads (CNS, MOCFE, and MultiGrid_C) that slightly decrease selectivity when increasing the number of ranks, which seems to be a variation caused by different partitioning than a particular trend.

Furthermore, all applications can be divided into two classes: The first one, including Boxlib MultiGrid C, Crystal Router, LULESH, and Multigrid_C, have a constant ratio between their *selectivity* and number of peers at all scale. The remaining applications do not show any correlation between selectivity and number of ranks or peers.

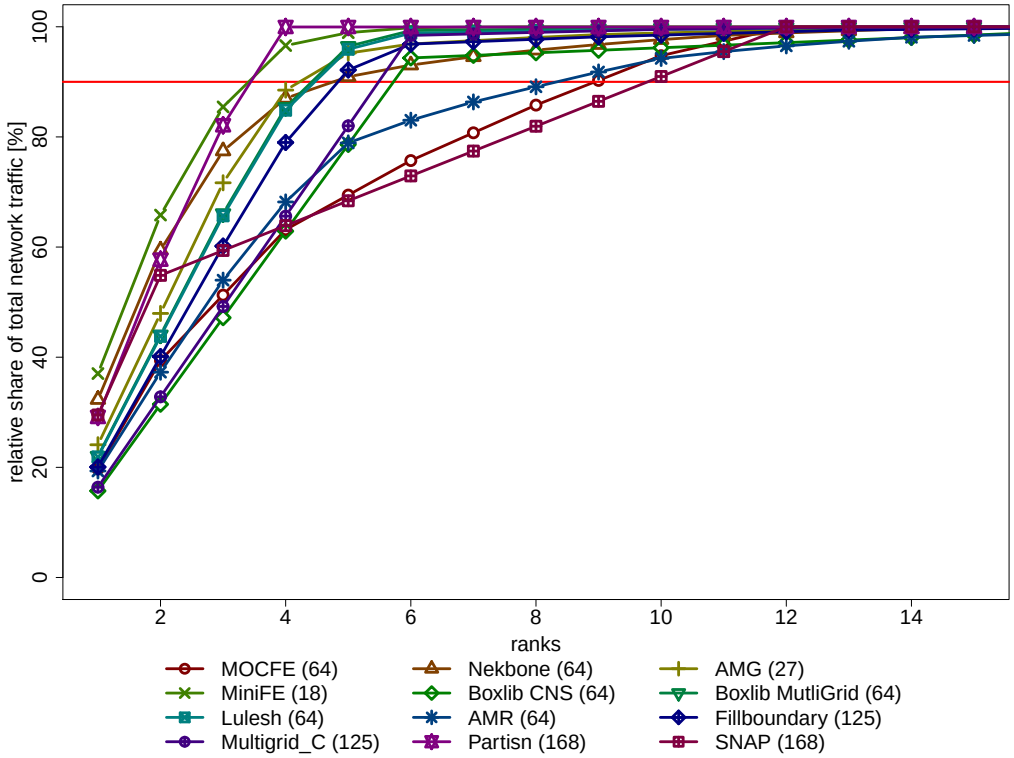


Fig. 4.6 Selectivity trends for all workloads.

4.2.4.2 Network Layer

After looking at the application layer, network layer effects, including the impact of different hardware configurations on locality, are analyzed in the following. However, this requires a framework of a particular network that includes network properties, such as mapping and topology, which are used as input for a non-temporal model. This model provides precise results about the distances in terms of hops and traffic volume in the network. This input is particularly useful during a system's design stage since these metrics can be directly translated into minimal network energy consumption or minimal packet latency. Here, traffic includes both point-to-point and collective messages.

Multi-Core Effects

The first study focuses on the impact of multi-core systems on network traffic. In particular, this study investigates how the traffic scales with different ratios of cores to network endpoints. This study is topology-independent and indicates the traffic volume that is transmitted on the network, independent of the distance

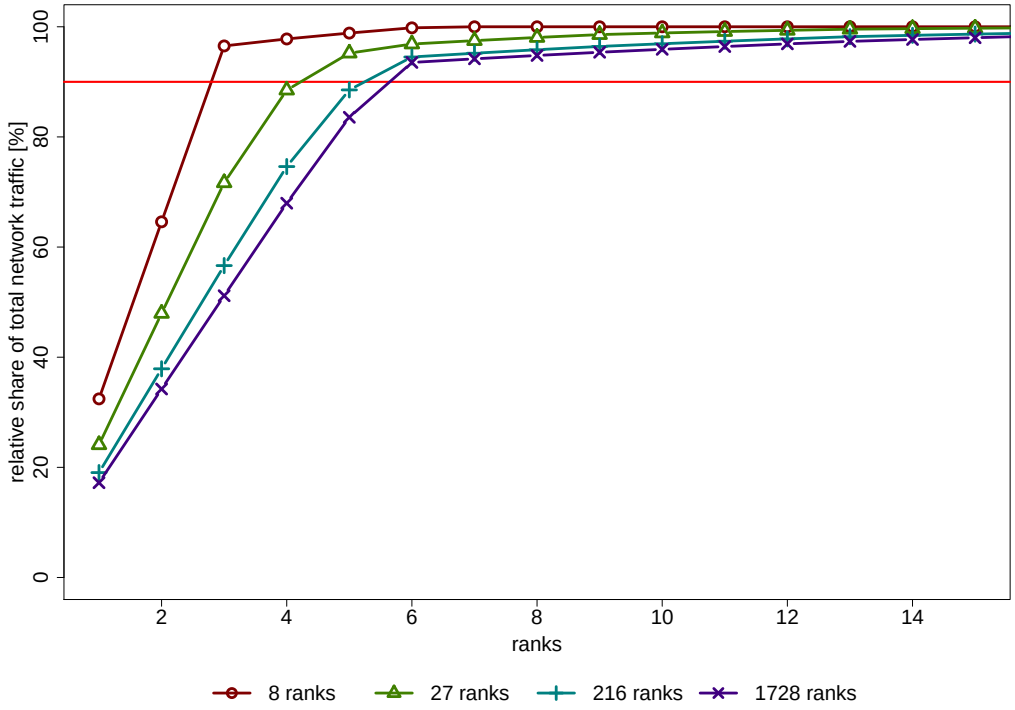


Fig. 4.7 Scalability of selectivity (example: AMG).

between source and destination. The number of cores per socket increases continuously, which allows to execute more ranks on the same node and, thereby, reducing the amount of data that has to be exchanged on the interconnection network.

The scaling trends are shown in Figure 4.8. To sustain network traffic and avoid sophisticating scaling effects, only applications that have a configuration of at least 512 ranks are considered here. These ranks are mapped consecutively to one node, according to the number of cores. The x-axis indicates the number of cores per socket, where one core executes one rank. The y-axis shows the amount of inter-node traffic relative to the one-rank-per-node configuration.

The general trend for all applications shows a varying degree of network traffic decrease between one to eight cores per socket and then reaches a plateau. Note that the variations in the course of the scaling are caused by mapping effects. As shown in the selectivity studies, there are subsets of ranks that heavily communicate with each other. A particular configuration can locate this subset into one node or splits it up into multiple nodes, which causes some fluctuation.

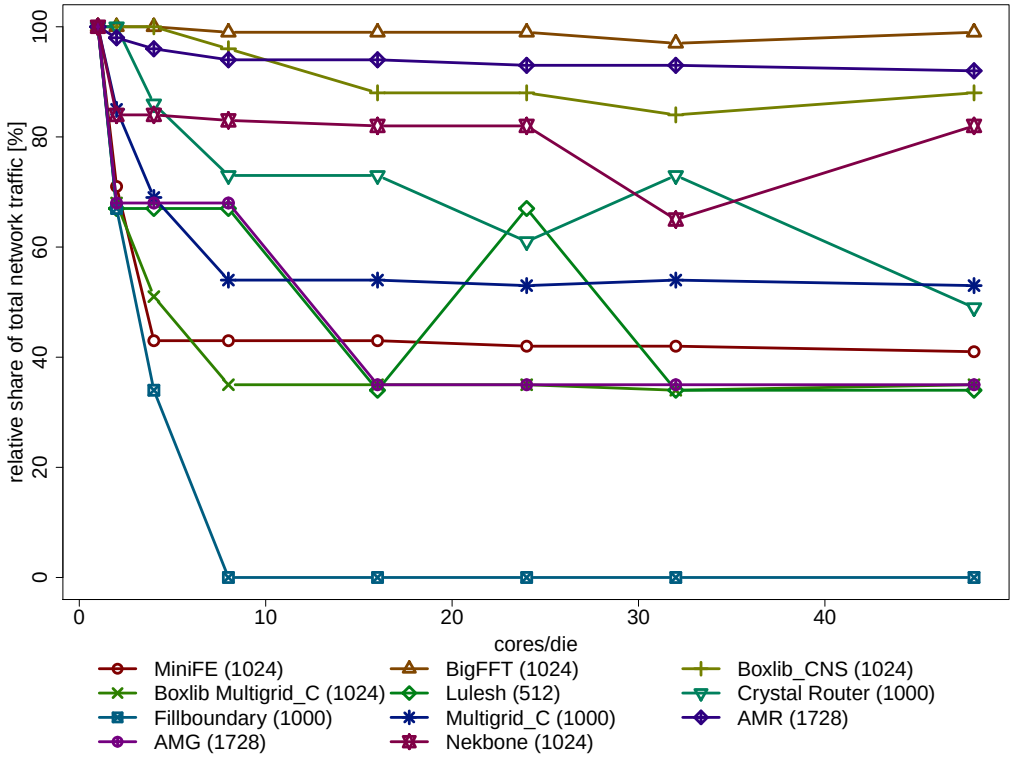


Fig. 4.8 Network traffic for different cores/socket configurations.

It is rather surprising that all applications reach saturation of inter-node traffic at 8-16 cores per socket. Although the level of saturation differs among them, this seems to be the optimum for minimizing network traffic and there are no evident improvements when further scaling, as long as the application's scale is much larger than the number of cores. Notably, the Fillboundery application reaches its saturation close to zero inter-node traffic, .

Topological Locality

Topological locality refers to actual distances in the network in terms of hops. The following studies focus on how topologies affect this network locality. To provide more realistic results for network overhead, large messages are split up into packets with a maximum payload of 4kB, and each packet in the network is expanded with a 48B header. Although the header size is small compared to the maximum payload size, its impact on the overall traffic can sum up especially if an application uses many small messages. Besides the total number of packet hops, the average number of hops ($\overline{\text{hops}}$) is provided to ensure comparability

		3D Torus		fat-tree		Dragonfly	
Workload	Ranks	Packet Hops	$\overline{\text{hops}}$	Packet Hops	$\overline{\text{hops}}$	Packet Hops	$\overline{\text{hops}}$
AMG	8	4.2E+03	1.57	5.7E+03	2.00	8E+03	2.83
	27	2.9E+04	1.74	3.5E+04	2.00	7E+04	4.01
	216	5.5E+05	2.36	8.2E+05	3.41	1E+06	4.14
	1728	6.0E+06	2.62	8.5E+06	3.62	1E+07	4.28
AMR_Miniapp	64	5.9E+06	2.93	6.6E+06	3.20	9E+06	4.19
	1728	8.9E+09	8.97	4.9E+09	4.86	5E+09	4.74
BigFFT (Medium)	9	1.0E+06	1.56	1.2E+06	1.78	2E+06	2.91
	100	7.7E+07	3.40	2.7E+08	3.52	3E+08	4.36
	1024	6.4E+10	8.00	3.5E+10	4.35	4E+10	4.69
Boxlib CNS large (*)	64	5.7E+06	2.99	6.5E+06	3.23	9E+06	4.23
	256	1.5E+07	4.93	1.2E+07	3.75	2E+07	4.49
	256	1.5E+07	4.93	1.2E+07	3.75	2E+07	4.49
	1024	1.1E+08	7.97	6.4E+07	4.35	7E+07	4.68
Boxlib MultiGrid C	64	2.6E+07	2.92	3.0E+07	3.19	4E+07	4.19
	256	3.9E+08	4.94	3.0E+08	3.76	4E+08	4.50
	256	3.9E+08	4.94	3.0E+08	3.76	4E+08	4.50
	1024	8.9E+09	7.96	4.9E+09	4.33	5E+09	4.67
CESAR MOCFE (*)	64	2.4E+06	2.96	2.7E+06	3.28	3E+06	4.24
	256	6.2E+07	4.96	4.7E+07	3.80	6E+07	4.53
	1024	3.2E+09	7.98	1.7E+09	4.36	2E+09	4.69
CESAR Nekbone (*)	64	4.0E+07	2.92	4.6E+07	3.25	6E+07	4.24
	256	1.2E+09	4.99	9.0E+08	3.80	1E+09	4.53
	1024	2.5E+10	7.96	1.4E+10	4.35	1E+10	4.69
Crystal Router	10	2.4E+05	1.74	2.7E+05	2.00	4E+05	3.18
	100	1.4E+06	2.41	7.4E+06	2.76	1E+07	3.61
	1000	2.8E+08	4.69	1.9E+08	3.26	2E+08	3.82
EXMATEX CMC 2D Multinode	64	7.9E+05	3.00	8.4E+05	3.28	1E+06	4.25
	256	5.2E+06	5.00	4.0E+06	3.81	5E+06	4.54
	1024	3.4E+07	8.00	2.0E+07	4.36	2E+07	4.69
EXMATEX LULESH	64	2.3E+06	2.70	3.8E+06	3.17	5E+06	4.18
	64	2.3E+06	2.70	3.8E+06	3.17	5E+06	4.18
	512	1.7E+08	5.80	1.3E+08	3.88	2E+08	4.60
FillBoundary	125	6.6E+06	3.27	6.9E+06	3.32	9E+06	4.13
	1000	9.9E+07	7.13	6.6E+07	4.15	8E+07	4.55
MiniFE	18	8.9E+05	1.82	1.1E+06	1.90	2E+06	3.69
	144	4.5E+07	3.97	4.2E+07	3.62	5E+07	4.40
	1152	4.6E+09	7.98	2.6E+09	4.47	3E+09	4.71
MultiGrid_C	125	1.2E+06	3.52	1.3E+06	3.57	2E+06	4.33
	1000	1.0E+08	7.43	6.0E+07	4.31	7E+07	4.66
PARTISN (*)	168	8.0E+07	2.70	1.0E+08	3.04	1E+08	3.88
SNAP (*)	168	1.6E+08	3.85	1.5E+08	3.74	2E+08	4.41

Table 4.5 Network locality aspects in torus, fat-tree, and dragonfly.

between applications. The results for all configurations are provided in Table 4.5.

3D Torus

Due to its high modularity, the 3D torus can be configured to every problem size in this study. For small problem sizes (<256 ranks) the torus shows better locality in terms of a lower number of average hops than the other topologies. The only exception is the SNAP application for which the fat-tree provides the best locality. In larger configurations, however, the $\overline{\text{hops}}$ in the torus scale stronger with the number of ranks than in both other topologies. This is caused by the also stronger scaling diameter of a torus. The only application to which this does not apply is AMG. Applications that have a nearest-neighbor communication patter, and a rank locality of close to 100% in a 2D or 3D grid respectively, are generally a good fit for the torus, but can also result in a larger $\overline{\text{hops}}$ due to their share of collective communication, which is spread evenly over the entire network.

fat-tree

All fat-trees here are composed of 48-port switches, which is a common radix size to set up large systems with only a few stages. This radix results in only three different configurations that are sufficient to map all configurations onto this topology. Since the ranks are mapped consecutively to the nodes, the unused parts of the fat-tree can be ignored without affecting the results. For all applications, the fat-tree seems to be robust against scaling effects, since there is only a slight increase in $\overline{\text{hops}}$ with an increasing number of ranks. The maximum average hop distance is 4.47 for MiniFE at 1152 ranks. Surprisingly, except for AMR (1728 ranks), the fat-tree provides always a smaller number of packet hops than the low-diameter dragonfly.

Dragonfly

Although four different configurations are sufficient for the dragonfly to cover all applications, the incremental mapping seems to have a higher impact here as opposed to the fat-tree. Since minimal routing is used, the number of hops can vary from two in the best case (destination is connected to the same switch) to five in the worst case (destination is part of another group and the switches

are not directly connected). Hence, these are also the boundaries for the $\overline{\text{hops}}$. Although the dragonfly is known as a low diameter topology, the $\overline{\text{hops}}$ is close to the maximum in most cases. Only for the AMG (8 ranks) workload, the $\overline{\text{hops}}$ remains below three hops on average, since this configuration consists of only one group. This is most likely causes by a relatively small group size, which originates from the load balance rule and low switch radices. The small group size caused a large amount of global traffic and increases the number of hops. In particular, 95% of all messages over all applications use the global inter-group link. However, non-minimal adaptive routing algorithms are commonly used in the dragonfly topology, which further increases the number of hops.

Network Utilization/Energy Impacts

Network utilization is a useful parameter to gain insights about the dimensioning of the network and the probabilities of congestion, respectively, as well as an estimation about the minimal energy demand. A smart mapping or a good fitting topology is most suitable to improve utilization for a particular application. The utilization results are shown in Table 4.6. Surprisingly, there is only one application (BigFFT) that utilizes the network for more than 1%. Looking at topologies, most applications follow a trend in which the fat-tree shows the highest utilization, while especially for a large number of ranks, a torus' utilization exceeds the one of both fat-tree and dragonfly. The trend is correlated to the link/node ratio and the $\overline{\text{hops}}$. While the former remains more or less the same, the latter shows a similar trend as the utilization.

Another important use case for network utilization is an estimation for the upper bound of energy-saving potentials. Although the non-temporal approach of these studies is not suitable to evaluate actual energy-saving mechanisms, network utilization can be directly correlated to the minimum energy consumption, as shown in Equation 4.8. This minimal energy does not only consider the data volume but also takes spatial locality, e.g. number of hops per packet, into account. The complement of this minimal energy equals the upper bound for energy-saving potentials and allows to compare topologies regarding their (theoretical) energy efficiency. However, the utilization and, thereby, the minimal energy is too low to provide useful comparisons, which might also be caused by a sparser communication volume of proxy applications compared to actual HPC applications.

Workload	Ranks	Network utilization [%] 3D Torus	[%] fat-tree	[%] Dragonfly
AMG	8	0.0052	0.0303	0.0116
	27	0.0012	0.0034	0.0034
	216	0.0008	0.0032	0.0021
	1728	0.0001	0.0004	0.0002
AMR_Miniapp	64	0.0034	0.0058	0.0048
	1728	0.0278	0.0229	0.0119
BigFFT (Medium)	9	0.6721	3.0725	1.2943
	100	7.4849	10.5544	7.6985
	1024	47.2317	38.4346	22.1491
Boxlib CNS large (*)	64	0.0002	0.0003	0.0003
	256	0.0004	0.0005	0.0004
	256	0.0005	0.0006	0.0004
	1024	0.0012	0.0010	0.0006
Boxlib MultiGrid C	64	0.0011	0.0020	0.0017
	256	0.0035	0.0045	0.0032
	256	0.0036	0.0046	0.0033
	1024	0.0106	0.0092	0.0054
CESAR MOCFE (*)	64	0.0498	0.0769	0.0605
	256	0.1216	0.1368	0.0895
	1024	0.4495	0.3656	0.2108
CESAR Nekbone (*)	64	0.0027	0.0090	0.0081
	256	0.3447	0.3882	0.2541
	1024	0.0029	0.0057	0.0035
Crystal Router	10	0.0469	0.1938	0.0882
	100	0.0408	0.0637	0.0490
	1000	0.1475	0.1531	0.0959
EXMATEX CMC 2D Multinode	64	2.0E-05	3.0E-05	2.4E-05
	256	0.0001	0.0001	0.0001
	1024	0.0008	0.0007	0.0004
EXMATEX LULESH	64	0.0004	0.0013	0.0011
	64	0.0004	0.0016	0.0013
	512	0.0005	0.0020	0.0012
FillBoundary	125	0.0319	0.0466	0.0351
	1000	0.0245	0.0248	0.0160
MiniFE	18	0.0008	0.0031	0.0015
	144	0.0017	0.0025	0.0017
	1152	0.0039	0.0037	0.0022
MultiGrid_C	125	0.0038	0.0056	0.0041
	1000	0.0013	0.0013	0.0008
PARTISN (*)	168	7.4E-08	1.6E-07	1.2E-07
SNAP (*)	168	4.2E-07	6.2E-07	4.0E-07

Table 4.6 Network utilization for different topologies.

CHAPTER 5

Simulation Tools

This chapter provides a summary of different tools and frameworks that are used to analyze different approaches and perform the evaluation of various power-saving strategies. Hardware design is a costly process regarding both money and time. Hence, it is not feasible to evaluate every new approach in real hardware. As a consequence, the standard procedure is to test new features in simulators to determine their impact. Following this approach, all experiments are performed in the SAURON network simulator [66], which can either simulate synthetic traffic patterns or use recorded application traces.

5.1 Network Simulator

In order to analyze the impact of hardware features without actually implementing them, there are two different possibilities: an analytical model or a simulation. While the former calculates static results depending on its input parameters, simulations are imitating the behavior of systems over time. This time-dependency allows taking snapshots of the system. Afterward, these snapshots can help to fully retrace and understand the processes going on in different parts of the system. However, simulations are usually more costly in terms of computations and time compared to an analytical model. But since a detailed understanding of internal processes is key to derive a model, the focus of this work is on detailed hardware simulations.

Simulations can be performed at different levels of accuracy. The more details implemented in a simulator, the more accurate are the simulated results. But greater accuracy is gained to the expenses of decreased performance since a higher detail-level results in increasing computation time.

In order to evaluate approaches for saving energy in interconnection networks, it is crucial to provide a high level of accuracy because there are not only multiple parameters that can be tuned for better results but these parameter tunings also impact various processes and effects in the network. For example, does a reduced bandwidth in certain links consume less power. But it can also decrease performance which leads to increased energy consumption. Furthermore, a locally decreased bandwidth can cause congestion in the network and, therefore, impact other traffic flows and slow down the entire network.

5.1.1 SAURON Simulator

SAURON (“Simulador de ArqUitecturas de Red en OmNet++”, Spanish for "simulator of network architectures in OMNeT++") is a cycle-accurate, OMNeT++-based network simulator, developed by Pedro Yébenes at the Universidad Castilla-La Mancha in Albacete, Spain [66].

OMNeT++

OMNeT++ is a C++-based simulation framework for modular network architectures. An OMNeT++ simulation model usually consists of multiple modules, that can be either "simple" or "compound". Compound modules are always composed of at least two or more simple modules or, again, compound modules. This allows the developer to introduce various levels of hierarchy. The framework provides certain interfaces for simple modules that allow, for example, communication between different modules via message passing.

To define the structure of a network model, OMNeT++ uses a declarative Network Description (NED) language. Features of different modules are defined as parameters, while connections for message and information exchange between different modules are defined as gates in the respective NED file. The network parameters itself are defined in Initialization (INI)-files. This structure allows running different simulations without re-compiling the simulator code and to define a set of values for each parameter. If one or more parameters are defined

with a set of values, OMNeT++ creates a configuration for every possible combination of these parameters and runs them independently while output data is stored in separate output files. Besides the hardware description in the network layer, a complex simulation model in OMNeT++ also has an application layer. This application layer is used to generate traffic, which is injected into the network. Usually, this can either be synthetically generated traffic patterns or input data from real application traces.

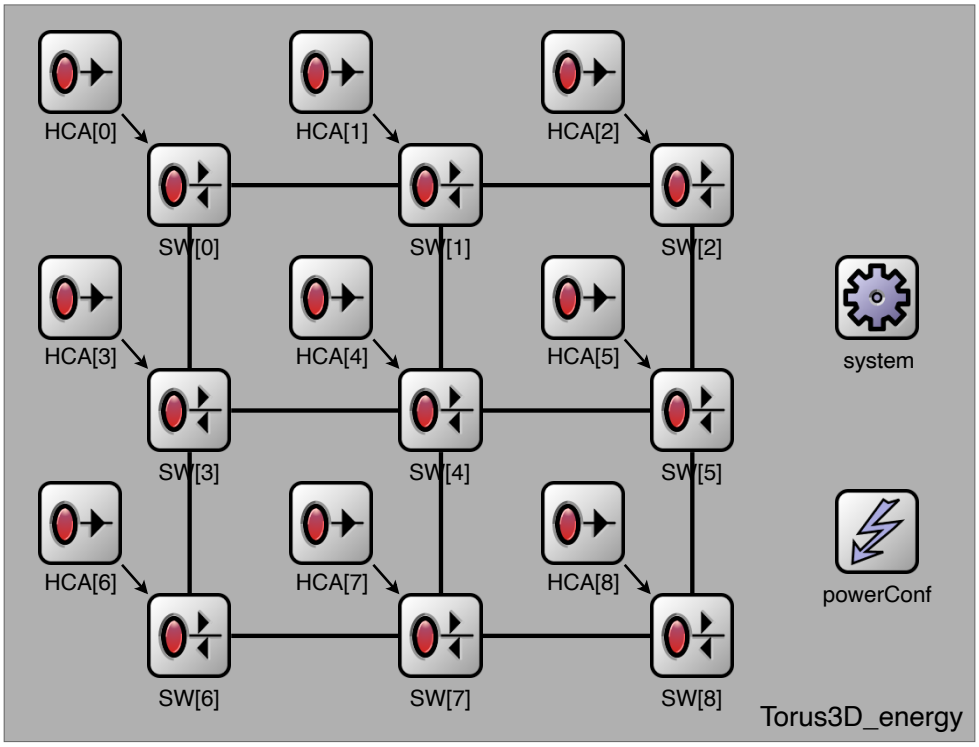


Fig. 5.1 Top-level view of a 3x3 torus system structure.

SAURON

The top part of SAURON's network layer shows the system view of the network and is the core of the network model. It contains all essential components, in particular all instances of links, switches, and Host Channel Adapter (HCA)s. HCAs model the interface between the host node and the network. The network model also shows the connections in the network and, therefore, depicts the underlying network topology, defined in the INI file. An example network layer layout of a 3x3 torus in the OMNeT++ Graphical User Interface (GUI) is shown in Figure 5.1. Each switch is equipped with a HCA to source and sink packet.

This emulates a direct network, where switch and HCA are integrated into the NIC. Furthermore, the topology has a system module, in which are general network parameters defined and statistics recorded. The powerConf module provides all energy and power-related functionality at the system level and also records energy-based statistics. Note that, the OMNeT++-GUI draws the line in the shortest distance, so the wrap-around links are overlapping with other links.

Both the switch and the HCA module are compound models. The design of the HCA module is depicted in Figure 5.2. Packets that are injected into the network by the application layer are inserted into the injection queue. Additionally, if multiple applications are running in the simulator, each application has its queue to avoid interactions between different applications. Once a packet is in the injection queue, the crossbar register and arbiter handle these packets and send them to the port which forwards them into the oppositely connected switch. When packets traverse over the network and reach their final destination, they arrive at the port, then they are forwarded to the sink. There, the packets are deleted and the system is notified that the packet arrived at its destination.

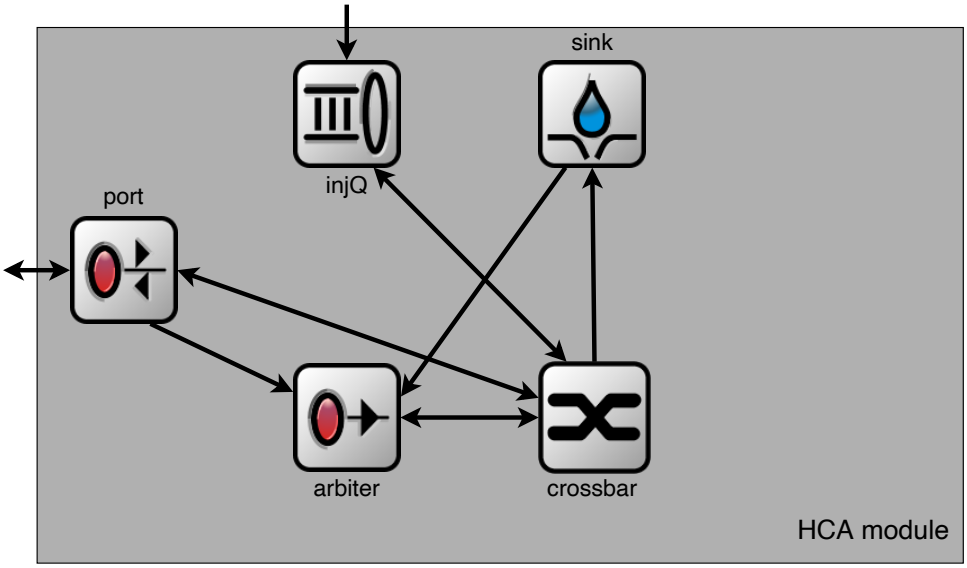


Fig. 5.2 Structure of the HCA interface in SAURON.

The other major compound module is the switch module. Its structure is shown in Figure 5.3. Similar to the HCA module, arriving and outgoing packets are handled in the port modules. Furthermore, this module performs the routing, packets are assigned to their respective Virtual Output Queue (VOQ)

inside the input buffers, and feature a VC-level credit-based flow control. The actual switching logic is modeled in the crossbar register and arbiter. Once a packet has got an assigned output port, it is stored in the crossbar register and wait to be processed. The decision about the order in which packets are sent through the crossbar is made in the arbiter. Additionally, the arbiter also tracks available credits and includes logic to assign packets to VCs and VOQs. The basic switching method used in this simulation model is virtual cut-through switching, which requires only the header information to make all routing decisions. Payload packets are then forwarded the same way. Switches in SAURON are input-queue-based switches which means that the buffers are located only at the input port. Depending on the number of VCs, the buffers are partitioned in the same way, and packets are stored in the buffer according to their VC.

The SAURON simulator provides a variety of different topologies, such as different tori, dragonflies, slim-flies, or k-ary n-trees, each with multiple suitable routing algorithms to choose from. The topology defines the switch radices as well as the rules in which way the switch modules are connected. The setting for each simulation, which also includes the number of nodes, buffer sizes, and message sizes, for instance, are defined in the .ini-file. Another part of SAURON is the system manager, which collects the statistics of all packets in the network to generate performance statistics at the end of the simulation.

Besides the network layer, there is the application layer, which models an application running on the system and contains the following modules: the application manager module and one or more application modules. Further on, these application modules can either be a synthetic application module or a trace-based application module.

Generally, messages are generated inside the application module and forwarded to the application manager for further processing. The application manager is also used for a statistical recording of predefined messages or network properties. The actual instance of the application module depends on the underlying traffic pattern. As the name suggests, the synthetic application generates synthetic traffic which can follow different patterns. These patterns vary from uniform traffic, which utilizes the network evenly to more sophisticated patterns that allow controlling of traffic flows in all areas of the network. On the contrary, the trace application module does not generate its pattern. It uses traces files, which are

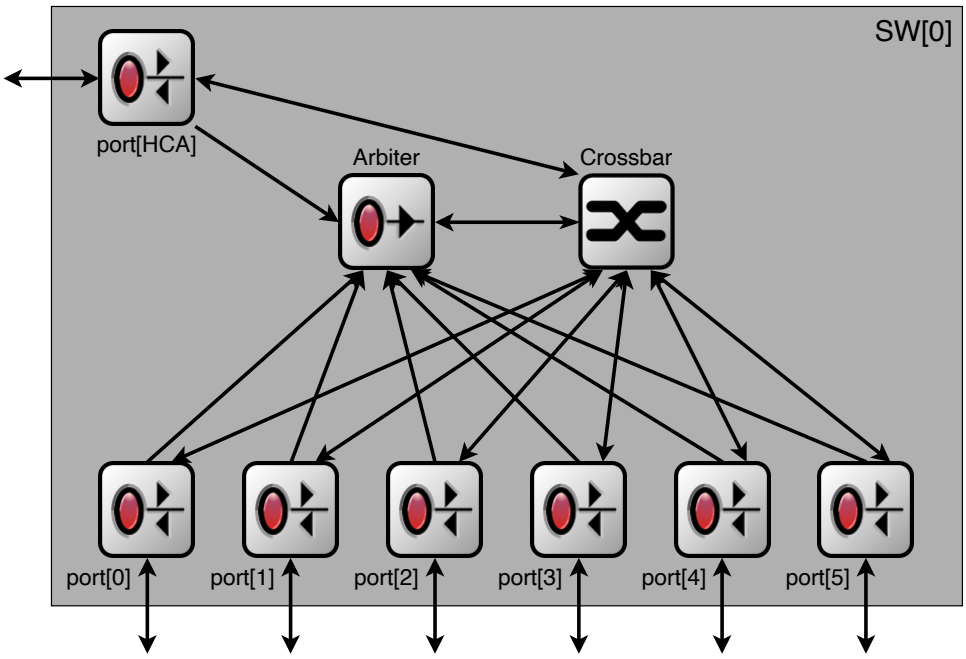


Fig. 5.3 Schematical blueprint of a SAURON switch.

previously recorded from actual applications on real systems, to set up messages for the simulator. This allows testing problems and possible solutions that are tailored to a certain configuration including hardware and a special traffic pattern. It is especially useful when the impact of new features on current systems are evaluated. The SAURON simulator has included an existing TraceLib [67] that uses VEF traces as input data. Once a message is generated in the application module, it is sent to the application manager. There, the message is split into the resulting number of packets and these packets are then forwarded to the injection queue in the HCA module of the respective sending node. Additionally, most of the performance statistics for the entire network are recorded here.

For more details about the introduced modules and their actual implementation, see [68].

5.2 Energy-Aware Simulations

Although SAURON provides a wide range of statistics, these statistics do not include power and energy at all levels. Besides energy-awareness of the simulation tool, the right setting is also key to investigate the potentials and impact of power saving in interconnection networks correctly. The following section provides

information about the implementation of these subjects in the context of this work.

5.2.1 Energy Features

As explained in chapter 3, the power consumption of interconnection networks is mainly driven by serialization technology inside linkports. Therefore, this is the obvious choice for reducing energy consumption. Power saving can either be achieved by frequency or link width scaling. While frequency can be adjusted by the clock multiplier, link width can be scaled by switching on and off parallel lanes inside one link. Therefore, both scaling methods are restricted to discrete values instead of continuous scaling, independent of the applied technique. Based on these discrete values, different power states can be derived. Each power state represents a certain frequency/link width combination and features a resulting bandwidth and a measured energy efficiency. An overview of all possible power states from the analysis in chapter 3 is shown in Table 5.1.

Index	# lanes	Frequency [GHz]	BWs [Gbit/s]	Energy/Data [nJ/bit]
0	0	0	0	0
1	4	2.5	8	688
2	4	5	16	350
3	4	10	32	178
4	8	2.5	16	394
5	8	5	32	203
6	8	10	64	106
7	12	2.5	24	296
8	12	5	48	154
9	12	10	96	82

Table 5.1 Power states implemented in the SAURON simulator.

These power states are implemented into the simulator by adding them as properties of the link module. In order to set the right power state for each link, the system manager module was extended to a power system manager. This power system manager collects all statistics measured in the entire system. Besides pre-existing performance data, the power system manager also takes track of the current power state of every link, the time spent in each power state, which is necessary to calculate the overall energy consumption, and the utilization of each link. The utilization is decisive for choosing the right power

state. Based on these statistics, the link modules can change their power states and adjust them to their current utilization, for example. Although it looks like a centralized management unit, the decisions are made decentralized inside each port and are only based on its statistics. Collecting all information in one module reduces overhead and it allows easily introducing an additional energy management unit, which could be used to adjust power states of certain links based on network information. This could be especially useful to implement advanced central managed global policies, which try to remain connectivity or to cap the maximum latency caused by the energy-saving mechanisms. However, this work focuses on a technology agnostic, decentralized approach.

5.2.2 Traffic Pattern

Another topic that is crucial for reliable results is the traffic pattern used in the simulator. SAURON generally includes two approaches to simulate network traffic: synthetic patterns and application traces.

Synthetic Traffic Pattern

Messages, send in the simulation from one node to another, are generated in the application module. The most simple approach is using synthetic traffic patterns, generated by the synthetic application module. This approach is widely used when simulations aim for certain levels of network utilization, but the actual utilization of a particular link can be neglected. Two common pattern are uniform random traffic and hot spot traffic. Uniform random patterns generate messages with a pseudo-random source and destination and aim to split the total traffic evenly over the network. Contrary to the balanced uniform pattern, hot spot patterns aim to stress particular regions of the network. In this pattern, multiple nodes send messages to one single node to generate a congestion tree. Additionally, all nodes that are not sending to the hot spot node generate a uniform random pattern again. For both patterns, the exact parameters, such as network utilization or the number of nodes participating in the congestion tree can be configured in the simulator. The pseudo-randomness of the source-destination pair ensures that simulations with the same input parameters deliver the same results.

Trace-Based Pattern

Besides the synthetic application module, SAURON includes a trace application module, which generates messages from traces. MPI traces are previously recorded traffic patterns of applications running on actual existing systems. To read traces, SAURON uses a tracelib [67], which processes traces in the VEF format. In this format, traces are self-related, i.e. trace entries contain task-based dependencies. Periods between two sending activities are assumed as computation time in the simulator, which enables the overlapping of computation and communication in simulations.

The most common approach to save energy is exploiting idle times to reduce energy consumption when the resource is not working. In the context of interconnection networks, these idle periods are the duration in which there is no traffic on a certain link. Therefore, the feasibility and quality of energy-saving mechanisms depend heavily on the traffic pattern. Naturally, synthetic traffic patterns are developed by the developer and can be designed in each way. However, evaluations should be objective and show the actual potential for interconnection networks in HPC systems. Thus, actual HPC traces are assumed as the most suitable benchmark to evaluate energy saving potentials.

5.3 MPI Traces

Energy saving in interconnection networks is based on a trade-off between performance and reduced power consumption in situations of underutilization. Therefore, it is mandatory to create a realistic simulation environment that reflects patterns of real systems. MPI traces are the best way to provide an underlying traffic pattern that generates a realistic utilization behavior of network links. Furthermore, traces can be statically analyzed in order to extract more generic and abstract information about different classes of workloads and their pattern.

Because there are many different use cases for communication traces, there are a variety of different trace formats available. They differ a lot in the scope of recorded information and, consequently, in size. Depending on the communication density, even applications with rather short execution times can result in traces of multiple gigabytes or even terabytes, especially at large scale systems. Hence,

different trace formats try to avoid overhead and compress the collected data to fit their purpose.

In the context of this work, two formats are used. The VEF format as input data for the SAURON simulator and the more detailed DUMPI format for detailed trace analysis.

5.3.1 DUMPI Traces

The main purpose of introducing the DUMPI trace format was gathering more detailed information about MPI calls than other available trace formats [1]. The traces are recorded during runtime by linking the DUMPI profiling library. This library replaces all MPI calls with the according to DUMPI routines which store all information of the MPI call in a binary format to reduce trace size [2]. Besides CPU and wall time, DUMPI stores all parameters of the MPI call per rank. The DUMPI library also provides an ASCII converter, which allows for easy further processing and analysis. Listing 5.1 depicts an example MPI_Isend call from the LULESH workload. It was developed as part of the SST/macro simulator and is maintained by the U.S. Sandia National Laboratories [69].

```
1 MPI_Isend entering at walltime 2187576.976228167, cputime
    ↪ 0.410076794 seconds in thread 0.
2 int count=2601
3 MPI_Datatype datatype=14 (MPI_DOUBLE)
4 int dest=4
5 int tag=1024
6 MPI_Comm comm=2 (MPI_COMM_WORLD)
7 MPI_Request request=[10]
8 MPI_Isend returning at walltime 2187576.976234828, cputime
    ↪  0.410083486 seconds in thread 0.
```

Listing 5.1 Example Isend call in the DUMPI Format.

Given the wealth of information provided by the DUMPI traces, this format suits well for meta-analysis. Hence, it was used in the context of this work to

[1] https://github.com/sstsimulator/sst-dumpi/blob/master/docs/traceformat.dox, accessed 2019-10-21

[2] https://portal.nersc.gov/project/CAL/trace.htm, accessed: 2019-11-20

characterize workloads by different features and derive basic underlying traffic patterns from them.

5.3.2 VEF Traces

Contrary to the DUMPI format, which aims for gathering as much information as possible, the VEF format was developed as input data for network simulators and aims to minimize the stored information to reduce overhead. The VEF format does not provide any framework for trace recording, but it can be converted from other common trace formats. The traces are self-related, which means single trace entries do not contain an absolute timestamp but a predecessor event and the relative time. These time intervals are assumed as computation time in simulations and allow to simulate an overlapping of computation and communication [70] [67]. A short sample trace file is shown in listing 5.2 and usually consists of the following parts:

```
1 VEF 4 5 2 3 10 0          // Trace Header
2 C0 0 1 2                  // Communicators
3 G0 C0 0 0 4 0 0 0 -1      // Collective communications
4 G0 C0 0 1 0 4 0 0 -1      // (independent records)
5 G0 C0 0 2 0 4 0 0 -1
6 0 0 1 16 3 300 G0         // Point to point messages
```

Listing 5.2 Example trace file in VEF Format.

Header: The first line contains the trace header that provides general information about the recorded traces in the following order:

```
VEF nTasks nMsgs nCOMM nCollComm nLocalCollComm nRecvDep
```

Listing 5.3 Header in the VEF format.

Which is translated to:

- **VEF:** Initial world which flags the header line.
- **nTasks:** Number of MPI tasks recorded.
- **nMsgs:** Number of traced point-to-point messages.

- **nCOMM:** Number of communicators used by the application.
- **nCollComm:** Number of global collective operations.
- **nLocalCollComm:** Resulting number of local collective messages. (This means the number of global collectives times the number of tasks participating in the communicator.)
- **nRecvDep:** Optimization flag which indicates whether or not dependencies between point-to-point messages exists.

Communicators: Following the header, the next one or more lines contain the different communicators used by the application. The first entry is the communicator id, starting with a "C" and then followed by an integer number. After the ID all ranks participating in the communicator are attached. An example of this structure is presented in listing 5.2 line 2.

Point-to-point message: Next to the Communicator starts the actual trace body with collective and/or point-to-point messages, which are defined in different patterns. The format of P2P trace entries in the VEF format is depicted in listing 5.4.

```
ID src dst length Dep dTime IDdep
```

Listing 5.4 Point-to-point message in the VEF format.

The different fields contain the following information:

- **ID:** Unique ID in order to identify every message in the traces globally.
- **src:** MPI taks that sends the message.
- **dst:** MPI taks that receives the message.
- **length:** Size of the message in bytes. E.g. the product of send count and size of the according data type.
- **dep:**Dependency type.
- **dTime:** Relative time stamp of the message in nanoseconds.
- **IDdep:** ID of previous message, which this messages is depending on.

Collective Operations: Although there is some overlap, collective operations are represented differently, shown in listing 5.5.

```
ID comm op task sendBytes recvBytes Dep dTime IDdep
```

Listing 5.5 Collective message in the VEF format

While some parameters equal the P2P ones, there are various other:

- **ID:** Unique ID. In order to determine it is a collective operation, all collective IDs start with the letter ”G”, followed by an integer.
- **comm:** Global communicator ID in which this collective operation is executed.
- **op:** Collective type.
- **task:** Rank that is performing this collective.
- **sendBytes:** Data volume that is send by this rank in context of this collective operation.
- **recvBytes:** Data volume that is received by this rank.
- **dep:** Dependency type.
- **dTime:** Relative time stamp of the message in nanoseconds.
- **IDdep:** ID of previous message, which this messages is depending on.

Note that, following the example in listing 5.2, every rank participating in the collective operation has one trace entry with the same collective ID. Furthermore, details of these different trace entries and parameters can be found in [70]. Particularly regarding collective operations, there are some limits regarding the supported types and various other rules for valid values for send and receive size, respectively. Other issues, which affect both collectives and P2P messages, are the different dependency types and their repercussions on the trace procedure.

CHAPTER 6

Energy Saving in Interconnection Networks

The studies in the previous chapters have shown that interconnection hardware as well as the applications running on HPC systems have enormous potential for energy savings. As in most components, reduced power consumption is often accompanied by decreased performance. However, interconnection networks differ from most other components in their underlying design. In NICs, the CML-based serialization technology, especially used in linkports, dominates the power consumption. Hence, CML-based parts qualify best for energy optimizations. The most effective approach is to adjust bandwidth to the current utilization by switching between distinguished power states, which represent a particular link width/frequency combination and the resulting power consumption. Furthermore, scaling link-width by turning single lanes on and off has proven to be the more efficient approach compared to scaling clock frequency, since CML's power consumption is current-driven.

The studies in this chapter assume a network, which is based on the data measured in section 3.1; particularly the power states provided in Table 5.1 are considered here. These power states do not only differ in their bandwidth and resulting energy consumption but also their degree of efficiency. Therefore, this chapter addresses multiple topics, including the best granularity of power states to adjust bandwidth most efficiently as well as how to determine which is the best power state for a given situation.

6.1 Approach

Energy as a physical quantity is time and power dependant. Therefore, reduced power consumption can only be effective in combination with reasonable performance. To maintain performance while operating with reduced network bandwidth requires a method which determines the most suitable power state for every individual link in the network. It is crucial to have a management mechanism or unit, which measures network-related data, such as utilization for instance and adjusts power states in every link accordingly.

6.1.1 Energy Saving Management

Generally, there are two fundamental approaches for such a management. The first one is a centralized network/energy management unit, which collects statistics from all essential network components and decides then which are the best settings. Second, in a decentralized approach, every component selects its setting based on its locally available data.

Centralized Power Management

The main advantage of a centralized approach is the wealth of information about the entire network. This allows for a high degree of power-saving while maintaining connectivity and the opportunity to react quickly to changing amounts of network traffic. A network can be configured in a way that all nodes remain connected but not all links are used. For example, in a torus topology could every second link per dimension switched off to reduce power consumption on low utilization periods. This would obtain full connectivity by slightly reduced latency and increased distance. If the load on the remaining links increases, links in a lower power state can be switched to a higher one again. In order to develop the full potential of this approach, the network could use a fully adaptive routing algorithm, since routing paths can change dynamically while a packet is already propagating on the network.

However, there are some disadvantages to this approach. The first and most important one is overhead in both, hardware and network traffic. The hardware overhead is caused by the management unit itself, which has to be attached to the network and to be able to evaluate all network statistics and

calculate the resulting settings. Although this is highly dependent on the actual implementation and complexity of the power saving mechanism, it requires presumably significant additional computing power. The network traffic increases since all statistical data have to be collected from the entire network, which causes a constant overhead of network traffic. This is especially problematic for non-hierarchical, direct topologies, such as tori. The management unit has to be part of the network and these constant traffic streams to one entity cause a hotspot traffic pattern in the network which raises the risk of congestions. Furthermore, an additional load on the network reduces the opportunities for energy saving. Associated with the traffic overhead is also the update rate for network statistics. While a low update rate could help to reduce this overhead, it also increases the time to react to unexpected appearing traffic bursts. But even at high update rates, detecting and quickly resolving hotspots is difficult, especially at larger network distances. The management unit has to detect a change in the network traffic, calculate a solution, and has to update the respective devices.

Decentralized Management

The opposite approach includes multiple decentralized management units that work independently. The granularity of these units can differ from a view in the network, managing certain regions to link or switch granularity. While the last one is the most common approach, the first one represents a hybridized form of centralized and decentralized management.

The high degree of locality is the most significant benefit of this approach. The management units are rather small and can be integrated inside the switch or even the linkports. Although this requires a hardware integration the actual overhead remains low. The management decisions are only are made locally, based on local statistics, which leads to quick adjustments. For example, a link could clutch its bandwidth to its utilization, without taking global effects, such as remaining connectivity into account. Such solutions are easy to implement in existing NIC-hardware with only small overhead. Another benefit is the short reaction time of these units. All statistics are collected and evaluated locally and do not have to propagate over the network. A filling buffer, for instance, can trigger a change of the power state.

Contrary to centralized management, it is not possible to implement an

advanced network administration. This is especially problematic when a packet can take multiple routes and at least one link in each of these routes decides to switch off. This could eliminate connectivity for nodes or even entire regions in the network and cause high latencies since the network has to reconfigure first. An additional problem occurs when multiple links along one path are switched off. While central management could switch all of them simultaneously, in the decentralized case, all these links have to be triggered one by one, which further increases latency.

Although both approaches provide multiple benefits, this work focuses mainly on the decentralized approach. A central management unit can be very useful in certain configurations, but it is highly tailored to the special circumstances, such as network technology, topology, or running applications. Since this work tries to analyze a wide range of configurations in a technology-agnostic approach, power management at the link level is more applicable and fairer for comparing the effect of various parameters. Furthermore, this enables the usage of the same energy-saving policies through all experiments without adjusting them.

6.1.2 Power State Granularity

An important factor for power saving management is the granularity of power states. The concept of power states is well-known and broadly used in most other components. For example canCPUs operate at different power states, which are distinguished as discrete values resulting from Dynamic Frequency Scaling (DFS), Dynamic Voltage Scaling (DVS), or their combination Dynamic Voltage Frequency Scaling (DVFS). Modern chips do not have to operate holistically at certain power states but can apply different power states to different regions or components according to their current utilization. This segmentation is particularly useful to cap the power consumption of some units when others are highly utilized. As a consequence, this allows complex chips to operate within strict power budgets.

Contrary to interconnection networks, changing power states in these units can be achieved very fast. When using DFS, it requires only about 20 cycles. However, switching power states by using DVS can take up to milliseconds [71]. While modern CPUs can select from three or more power states, McLaughlin et.al. [72] conclude that more fine-grained power states in processors could

improve energy efficiency for irregular workloads. Beyond that, there is little research on the benefits of power state granularities.

Although this work is designed in a technology-agnostic way, the designs of the energy-saving policies are based on the power measurements introduced in section 3.1. As Table 5.1 indicates, there are multiple configurations resulting in the same bandwidth. Although all configurations are valid, they differ regarding their energy efficiency indicated in the last column. Since a better efficiency provides more potential for energy saving, only the most efficient one is taken into consideration if there are multiple configurations for the same bandwidth.

Furthermore, the differences in the efficiency of all power states show that today's hardware is not energy-proportional yet. In an ideal energy-proportional network, in which all power states provide the same efficiency, a fine-grained granularity works best. For example, if there is a 50% load on the network it could be more efficient to delay the data by selecting half the bandwidth, than operating half the time at full speed and switching the links off for the other half. Due to transition times, this could result in even longer execution times and, therefore, in higher energy consumption. To utilize these fine granular power states, a deep understanding of the running application, and the ability to predict traffic patterns precisely is necessary. Without this knowledge, it is most efficient to operate in a more coarse-grained way. Furthermore, long transition times in interconnection networks prohibit permanent fine-grained adjustments of the bandwidth. Therefore, the analyses of this work are based on the following three power states shown in Table 6.1.

Index	# lanes	Frequency [GHz]	BW [Gbit/s]	Energy/Data [nJ/bit]
0	0	0	0	0
1	1	10	8	82
2	12	10	96	82

Table 6.1 Actual used power states for the power saving policies.

First, if a link is not used, it can be switched off completely (state 0). Since a switched-off link consumes no power, this is the most efficient power state for this situation. Second, if there is data available to send, the most efficient way in terms of energy consumption per data is to send it as fast as possible. In this case, power state 2 is selected. Third, if the link is idling, but has to stay available, for instance, to ensure connectivity, power state 1 is used. This

power state cannot actually be configured in the switch from which the power states origin from, because lanes can only be configured at a granularity of four. However, these are no technical limitation and the management could easily be expanded to lane granularity. It is important to operate the last remaining lane at the same frequency, even if a smaller frequency would further reduce the power consumption, since changing the operating frequency results in a new link training, which prevents the link from data transmission. Switching off particular lanes does not require the link to perform this training.

6.2 Energy Saving Policies

After determining which power states are effective to use, the network needs to decide which state applies best to a particular situation. In view of the fact that changing power states and especially changing frequency and switching on lanes, stalls the link, frequent transitions between power states have a negative impact on performance. Hence, pondering when and how often power states are changed is key to save the energy within a reasonable performance.

To perform studies about energy savings, three different policies are developed and evaluated. The first two operate at link granularity in which every link decides independently which power state to choose. The third one follows a hybrid approach in which links can also affect other links around them to overlap transition times.

6.2.1 On/Off

The first policy (on/off), is the most simple one and is introduced in [73]. As the name suggests, links are in the high power state (2), when they are active and switched off (power state 0), when they are idling. Switching off only one direction in a bi-directional link, would not provide any benefits in terms of performance, since the backward direction has to transmit credit and acknowledge messages. Therefore, switching the power state at one link includes both directions. In an ideal network, which means power states can be changed immediately without performing the reconfiguration training, this policy would result in minimal energy consumption for a given technology. Provided that, links are switched off if there is no traffic on the network and switched on again if new data need to

be transmitted. This set up can be used to define the maximum power saving potential for a certain configuration. However, since this is just an ideal scenario, actual evaluations have to consider performance decreases due to transition times.

While links are switched off, no data can be transmitted on the link. To minimize potential performance losses, links are switched on immediately when the routing unit decides to route a packet through a link that is in power state 0. A different approach could be to wait until output buffers are filled to a certain threshold level before links are switched on again. This ensures a better utilization but causes a significant increase in tail latencies. Although some systems, such as cloud installations, are suitable for this concept, HPC systems are not among them since low latencies and shorter execution times are more important.

To determine when to switch off a link is more complex. On the one hand, an aggressive approach that switches links off immediately after the output buffer is empty, exploits the maximal idle period. However, this could also affect performance negatively, because the link has to reconfigure every time a new packet arrives. This is particularly harmful if a link is frequently used with only short idle periods between two packets. On the other hand, waiting too long before switching links off reduces the total amount of energy that can be saved. Therefore, a parameter Δt is defined that represents the period a link is idling before it is switched off.

This parameter integrates two goals: to maintain a good performance and to maximize the energy savings. A. Venkatesh et al. [74] introduce an approach to cap the maximum performance loss. While their work focuses on MPI optimizations in the application layer for a network with different power levers, this approach can partly be adapted to the network layer. The authors introduce a parameter ρ for the maximum performance loss or increase of execution time, the system operator can tolerate. For instance, $\rho = 0.1$ would result in a maximum increase in execution time of 10%. To derive a Δt from ρ, the authors define a worst-case scenario, in which a packet arrives immediately every time a link is switched off. Hence, the idling period Δt is followed by the transition time t_t. Therefore, the performance loss can be described as:

$$\frac{t_t}{\Delta t} = \rho \qquad => \qquad \Delta t = \frac{t_t}{\rho} \tag{6.1}$$

Since the transition time t_t is a system property defined by the implemented hardware, the time Δt is inversely proportional to ρ. Figure 6.1, 6.2, and 6.3 depict schematically the operating principle of the on/off policy.

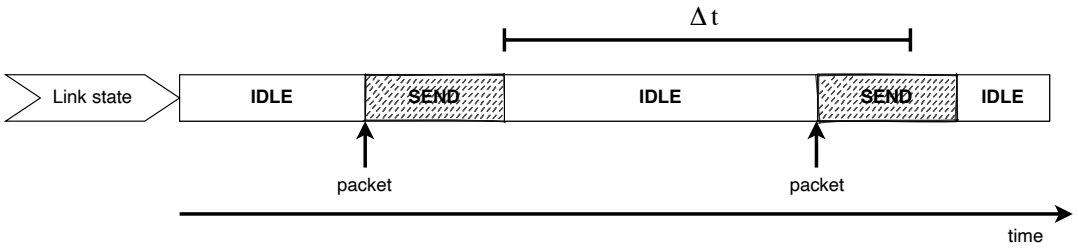


Fig. 6.1 Operating principle of the on/off policy: link is used and stays active.

Figure 6.1 shows the impact of the on/off policy on a moderately used link. The x-axis depicts the temporal progress and the state of an example link is shown inside the bar. In the beginning, the link idles until a packet arrives. After transmitting the packet, the link idles again and starts the internal Δt timer. But before the timer expires, which would put the link to a different power state, a new packet arrives. This packet is directly transmitted and the Δt timer is reset. As long as the link does not idle contiguously for the period Δt, the policy does not affect the link and it provides permanently full bandwidth.

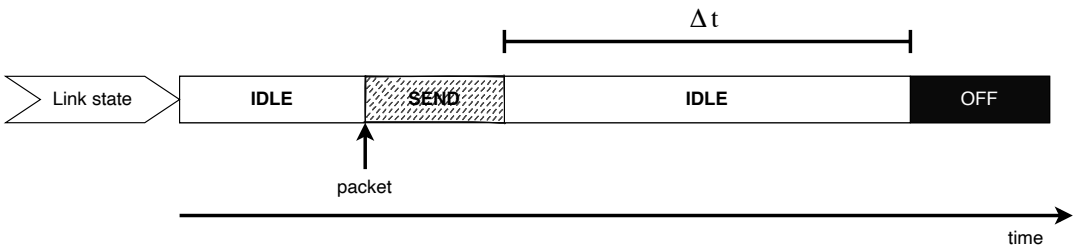


Fig. 6.2 Operating principle of the on/off policy: link is idling and switched off.

In Figure 6.2 the impact of the policy on low utilized links is illustrated. The process starts like the one in Figure 6.1 with an arriving and processed packet. However, this time it is not followed by a second one; the timer expires and the link is triggered to switch to power state 0 to safe power.

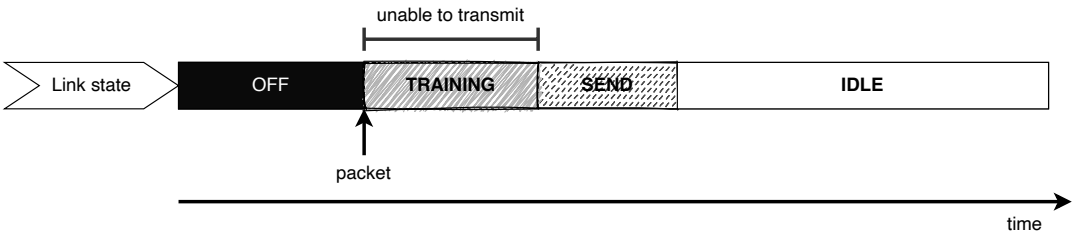


Fig. 6.3 Operating principle of the on/off policy: switched off link is switched on.

Following the switch-off, Figure 6.2 depicts the arrival of a new packet. If the routing unit decides to use a link that is switched off, this triggers the link to change to a transmitting power state. The link performs its training for a period of t_t to ensure word alignment and DLL and PLL locking. Once this training is finished, the packet can be transmitted on this link.

Accumulated Transition Times

Contrary to the original approach introduced in [74], the actual performance loss applied to the network can indeed exceed the previously defined maximum. This is due to the fact that the maximum performance loss is only calculated from a node's perspective. Considering the network's view, packets can have to perform more than one hop on their path from one node to another and the transition times can sum up along this path. This case is depicted in Figure 6.4. Analog to Figure 6.1 - 6.3, the two bars contain the states of two neighbouring links and the temporal progress is illustrated on the x-axis.

At $t = 0$, both links are switched off due to their previous idling periods. When a new packet arrives at node 0, link 1 changes its power state to transmit it. After the transition time, the packet is transmitted on link 1 and arrives at node 1. This arrival triggers link 2 to change into an active power state and starts the same process as before in link 1. As a consequence, the delay in terms of the transition time can accumulate in every hop of a packet's path. To adjust the previously introduced maximum performance loss, the parameter ρ needs to be multiplied by the topology's diameter, which equals the longest path a packet could take, assuming minimal routing.

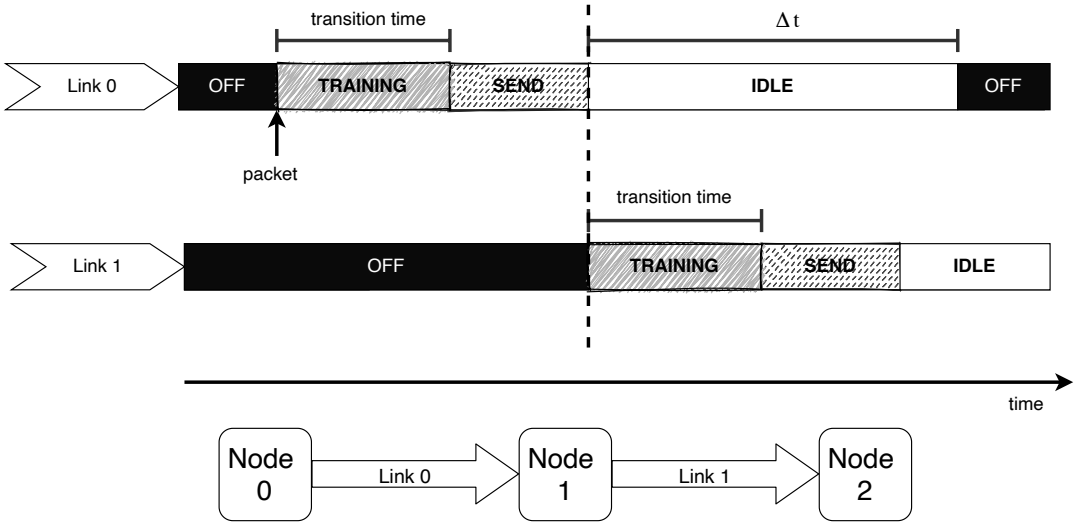


Fig. 6.4 A packet performs multiple hops and observe multiple delays due to transition time.

6.2.2 High/Low

While the on/off policy enables most energy-saving potential, the performance can be significantly harmed, when an application uses many small messages in a coarse-grained pattern. The second policy (high/low) aims to tackle this problem and to ensure reasonable performance. Instead of switching links completely off (power state 0), they are set to the power state with the lowest power consumption that is still able to transmit data (power state 1) in which only one out of twelve parallel lanes remains active. This obtains permanent connectivity at the costs of a decreased maximum energy-saving potential since the low power state produces a higher power overhead for unused links than switching them completely off.

At low utilization, switching to power state 1 follows the same procedure as the previous policy. The timer Δt is calculated the same way and once this timer expires, the link decreases its power state. Regarding switching back to the fast power state (2), this approach differs from the previous one. The procedure of this policy is shown in Figure 6.5 and 6.6.

In the first example in Figure 6.5 the exemplary link starts in the high power state. When a packet arrives **(1)**, it is transmitted with maximum bandwidth **(2)**. Then, the link idles until the Δt timer expires **(3)** and the link is set to the lower power state. When the next packet arrives **(4)**, the link remains in this

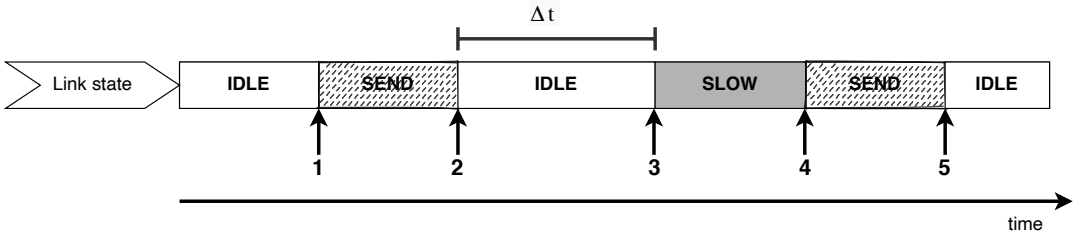


Fig. 6.5 An example of packet forwarding using the high/low policy in the fast and slow power state.

lower power state and transmits the packet at reduced bandwidth which results in a longer sending process **(5)**. However, sending at this lower speed remains faster than switching the link to the higher power state and accepts the delay due to the transition time.

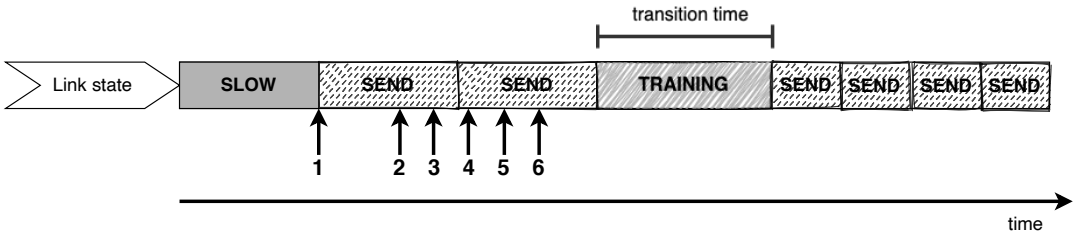


Fig. 6.6 Link at high/low policy switching back to high power state.

This procedure is only beneficial as long as the transmission at lower bandwidth is faster than reconfiguring the link and sending it at a higher bandwidth. The break-even point when a higher power state becomes more advantageous is determined by the buffer fill level and the bandwidths of the slow and fast states, as the sending time is the quotient of the amount of data and the respective bandwidth. Therefore, the threshold buffer fill level can be calculated as:

$$\frac{data}{BW_{low}} > \frac{data}{BW_{high}} + t_t \qquad => \qquad data > \frac{t_t}{\frac{1}{BW_{low}} - \frac{1}{BW_{high}}} \tag{6.2}$$

This situation is illustrated in Figure 6.6. IN the beginning, the link is already in a slow power state because it was idling previously. Alike in Figure 6.5, when the first packet arrives **(1)**, it is sent directly to the destination node. The same procedure applies for the second packet, which arrives while the link is still transmitting the first packet **(2)**. However, while the link is transmitting the second packet, multiple packets arrive at the output queue **(3-6)** and the

buffer fill level begins to rise and exceeds the threshold. Once the second packet has completed the sending process, the link switches to the higher power state performs its training, and can then process the remaining packets at the high bandwidth.

6.2.3 Awake

The last policy leaves the scope of a single linkport and allows to links to affect power states of other links as well. The idea of the policy is to exploit a packet's routing information and to overlap transition times in order to further improve performance compared to the high/low policy. This can be particularly useful for large messages that exceed the threshold size for switching to the high power state.

Analog to the buffer level threshold, there is a message size threshold that triggers an awake message:

$$size > \frac{t_t}{\frac{1}{BW_{low}} - \frac{1}{BW_{fast}}} \tag{6.3}$$

If a message exceeds this size is injected into the network, the first NIC generates an awake message with the same destination. This small message is similar to a credit but has priority over other messages. If it arrives at a linkport that is operating at low bandwidth, the awake message is first transmitted slowly and then the link is triggered to change its power state, by the notification of a large message approaching. In the best case, this reduces the delay due to link reconfiguration from the number of hops to one, because the entire path is put in a high power state before the message traverses along this path. Note that large messages are divided into multiple packets when they are injected into the network layer. However, all packets are routed the same path assuming deterministic routing, which is a necessary condition for this policy.

6.3 Evaluating Policies

The best way to evaluate these energy-saving policies and gain deeper insights about their effects on execution time, energy consumption, and other metrics, is to run them under practical circumstances. Firstly, since implementing experimental

concepts in hardware is prohibitively costly in terms of time and money, a highly detailed simulator is the best approach for first evaluations. Besides an environment that simulates all major process in the network, realistic conditions also include traffic patterns. A common way for network simulation is the usage of synthetic generated traffic, which usually represents some basic patterns such as uniform or hotspot traffic. However, since energy-saving policies exploit idle periods in traffic patterns, the results are directly related to the input patterns of the network traffic. Therefore, using traces of actual HPC applications is key to enable comprehensive comparisons and realistic insights.

6.3.1 Applications

Although the selection of proxy application in Chapter 4 represent a broad cross-section of many different communication patterns, they are less suitable for these analyses since these mini-application are too small and short in terms of execution time for comprehensive studies. Hence, traces of actual HPC applications are more suitable here. Another important factor is the availability of large traces compatible with the simulator. To provide fairness between all applications 512 ranks are selected as the largest configuration in which traces of all applications are available.

The following five applications are selected to draw a diverse picture of the effects of energy-saving policies on the performance and energy consumption under realistic conditions. Table 6.2 depicts their basic parameters as provided for the proxy mini-applications.

Parameter	LULESH	NAMD	Graph500	WRF	HPL
Vol. [MB]	94829	18304	262221	87246	26599
P2P [%]	100	99.99	0	99.89	99.99
Peers	26	511	N/A	4.00	22.00
Locality	63.71	319.28	N/A	15.84	223.11
Selectivity	3.66	23.26	N/A	2.37	7.89
Torus	5.38	2.47	6.00	5.82	3.59
Fattree	3.55	1.84	3.81	3.71	3.53
Dragonfly	4.31	2.27	3.81	4.51	4.35

Table 6.2 Communication characteristics of HPC applications.

LULESH (Livermore Unstructured Explicit Shock Hydrodynamics)

LULESH [75] is the only exascale proxy mini-application provided by the United States DoE that is used to evaluate energy-saving policies. However, these traces differ from the ones that are analyzed in Chapter 4 in their respective problem size. LULESH is a hydrodynamic simulation, which uses a stencil code to calculate the physical forces. The traces used here are generated with a problem size of 100 and 50 iterations, which results in roughly one million elements per rank.

NAMD (Nanoscale Molecular Dynamics program)

NAMD [76] simulates dynamic biomolecular systems by performing n-body particle calculations. The underlying n-body calculations are typically compute-bound and based on all-to-all communication. As shown in Table 6.2, NAMD is performing almost only point-to-point communication, however in an all-to-all pattern, as indicated by the number of peers (511). This occurs due to ring-based communication optimizations, long-range force optimizations, and load balancing of the CHARM++[1] parallel objects runtime. The input molecule for these traces is the Satellite Tobacco Mosaic Virus (STMV), which is consists of about one million atoms and is one of the most frequently used molecules.

Graph500

Graph500[2] is a benchmark with a data-driven communication that performs a breadth-first search (BFS) graph traversal. The resulting communication pattern is entirely based on collective operations, which are dominated by all-to-all communication. This application is also used to benchmark the Graph500 list, which is an alternative to the TOP500 list. These traces are generated with replicated-CSR implementation, an edge factor of 16, and a scale factor of 20.

WRF (Weather Research and Forecasting)

As its name suggests, WRF[3] focuses on atmospheric research, including operational weather forecasting and climate research. It is based on numerical simulations. WRF performs a two-dimensional nearest-neighbor communication

[1] http://charm.cs.illinois.edu/research/charm, accessed: 2020-03-22
[2] https://www.graph500.org/, accessed: 2020-03-22
[3] https://www.mmm.ucar.edu/weather-research-and-forecasting-model, accessed: 2020-03-22

pattern, which coincides with the peer, selectivity, and locality metrics. While four peers are the four nearest neighbors, the locality of 15.8 is caused by the 2D underlying problem, which results in distances of one and 16, depending on the orientation. This benchmark is also often used for comparing various aspects of computing systems, including CPUs, Interconnects, and MPI library performance.

HPL (High-Performance Linpack)

HPL [77] is widely used and most known as a benchmark for the Top500 list and solves a dense $N \times N$ system of linear equations. The problem size and software optimizations allow the user to tune the performance for a given system. The communication pattern of the Linpack benchmark, which consists of frequent small messages and rare large messages to distant ranks [78], is rather unique. Additionally, HPL does not fit well with the dragonfly topology in terms of average network hops, compared to 3D torus and fat-tree. The traces are generated with the following parameters: row-mapping, N = 9984, P = 16, Q = 32, threshold = 16, NBs = 192.

6.3.2 Methodology

The evaluations of the energy-saving policies are performed on a cycle-accurate, OMNeT++-based network simulator, described in Chapter 5. This simulation environment provides high degrees of freedom to configure network architectures. Hence, the particular network configurations for all evaluations are presented in the following.

6.3.2.1 Network Simulation

The goal of these studies is to gain detailed insights about the functionality and effects of the introduced energy-saving policies, as a technology-agnostic approach. These effects are based on a wide range of different parameters and the focus is on a detailed understanding of these parameters. The residual network configurations are based on commonly used state-of-the-art interconnection networks.

Simulation Parameters

Network parameters that have minor relevance for the energy-saving are provided in Table 6.3.

Parameter	Value
Flow control	credit-based
Cable length	5m
Header size	48B
Payload size	2000B
Credit size	64B
Credits	4096
Xbar datarate	200 Gbps

Table 6.3 Simulation parameters.

Notably is the credit-based flow control, which is commonly used in HPC systems and a wide range of other data center networks. With this flow control mechanism, network traffic in bidirectional links is only possible, if both directions are available. Hence, links cannot be switched off unidirectional to preserve this functionality. Also note that although eight MB buffer size is rather large, this buffer size includes all input buffers per switch and not per linkport.

Topologies

To study how different categories of network topologies affect energy saving, one representative of each of the three classes is selected. The 3D Torus represents the class of direct topologies, the k-ary n-tree, a fat-tree version, serves as example for non-hierarchical indirect networks, and the dragonfly topology is a representative for hierarchical indirect networks. The structure of these topologies has to follow certain design rules, which can prevent them to configure particular sizes. To provide fairness in the analyses, the power consumption of unused links and

switches are ignored and not considered in the results. For the simulations, these topologies are configured as:

3D Torus The 512 nodes are split evenly over the three dimensions with eight nodes each. Typically for tori, x-y-z dimension order routing is used.

K-ary n-tree This topology is highly depending on the switch radix and the number of stages. To investigate potential impacts, two different configurations are examined: three stages with a switch radix of 16 resulting in 512 nodes and two stages with a switch radix of 64 resulting in 1024 nodes. Both configurations are using the destro routing algorithm.

Dragonfly The dragonfly topology provides high degrees of freedom to the system designer without providing strict rules on how to organize the groups and global connections. However, Kim et al. [29] provide some guidelines to ensure load balancing for example. Following these rules, the dragonfly here is equipped with 28-port switches and a fully global interconnect, resulting in 756 nodes. UGAL is used a the respective routing algorithm.

6.3.2.2 Comprehensive Energy Saving

Besides the comparisons between the different energy-saving policies, the contrast to other technology-specific energy-saving mechanisms is investigated. For further comparisons with a topology specific energy-saving policy the one introduced by M. Alonso et al. is selected, which is particularly tailored to k-ary n-trees [11], [79]. The policy is further simply referred to as Alonso. Similar to the on/off policy introduced in this work, their policy also switches links on and off depending on the current router utilization. However, the authors introduce a dynamic threshold, which depends on he remaining number of active outgoing links and determines if links are switched on or off. This threshold is continuously updated at runtime. Another difference is the pattern in which links are switched off. While the on/off policy only operates at the link level and does not consider the big picture of the entire network, Alonso's policy defines a subset of links and switches that is needed to ensure connectivity between all nodes. This minimal subtree cannot be switched off and, therefore, reduces the maximum of power that can be saved. To define this subset of k-ary n-tree specific properties are

considered, including the link direction (up or down) and the respective stage. Hence, it is rather difficult to adapt this approach to other topologies, which limits its scope.

To fit the simulator in this evaluation, the policy was slightly adjusted. Instead of switching links completely off, they are set to the lowest power state, analog to the high/low policy. This is necessary due to bidirectional links and the credit-based flow control which was not part of the original work. However, the slow links can only be used for credits and do not contribute to the overall power consumption, in order to be as close as possible to the original work. Although Alonso's policy additionally requires adaptive routing, the adaptive part in the upward direction of the destro routing algorithm is sufficient and is also used here.

6.3.2.3 Energy Saving Parameters

Two major parameters impact the energy-saving policies: transition time and maximum performance loss. Both have a direct impact on the maximum amount of energy that can be saved as well as increases in the execution time. Therefore, the selection of these parameters is key to a comprehensive analysis.

Transition Time

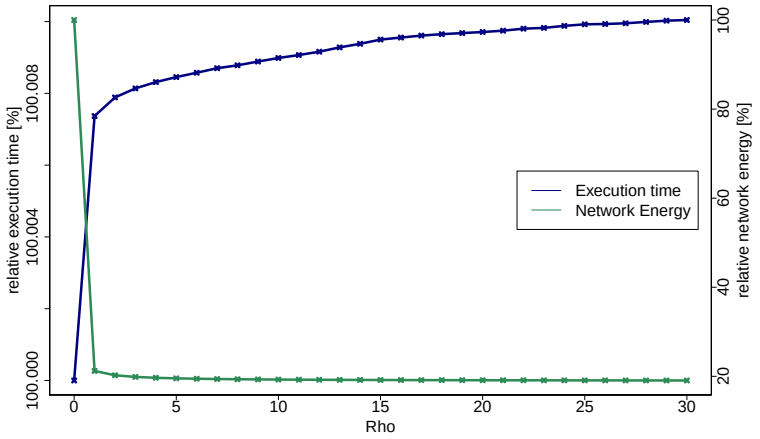
The transition time is a hardware parameter, which determines the time a link needs to reconfigure after changing its frequency or switching on additional lanes. Regarding energy saving, this is the most important network property since the transition times determine how switching power states affect the execution time and, therefore, how aggressive energy saving can be approached within a given performance.

However, since energy-saving policies do not exist yet in today's hardware and reconfigurations during runtime are not common, it is difficult to ascertain realistic numbers. The IEEE 802.3az standard specifies multiple different transition times ranging from single-digit to $182\mu s$ [80]. Dickov et al. [81] use a transition time of $10\mu s$ and Kim et al. [82] $1\mu s$ for their studies. Abts et al. also assume a transition time for $1\mu s$ to $100\mu s$ to be feasible in today's hardware [4]. Hence, the evaluations in this work also assume transition times in this range.

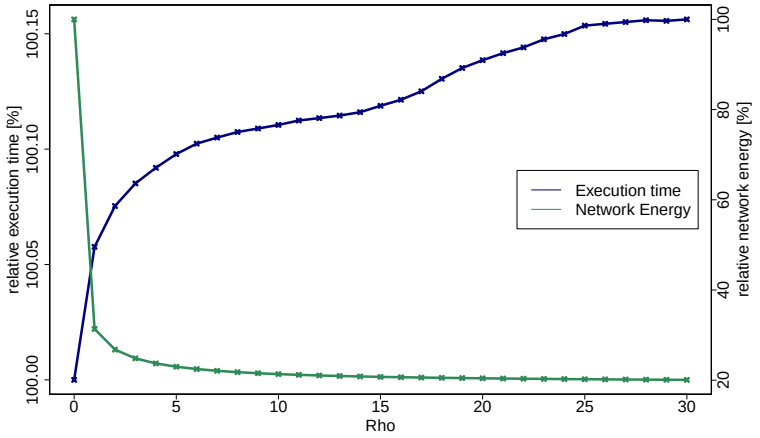
Performance Loss

The other important parameter is the maximum performance loss ρ. This parameter can freely be selected and determines the idle time after which links are moved to a lower power state. Also depending on an interconnect's transition time, ρ can be translated to t_{idle} following Equation 6.1.

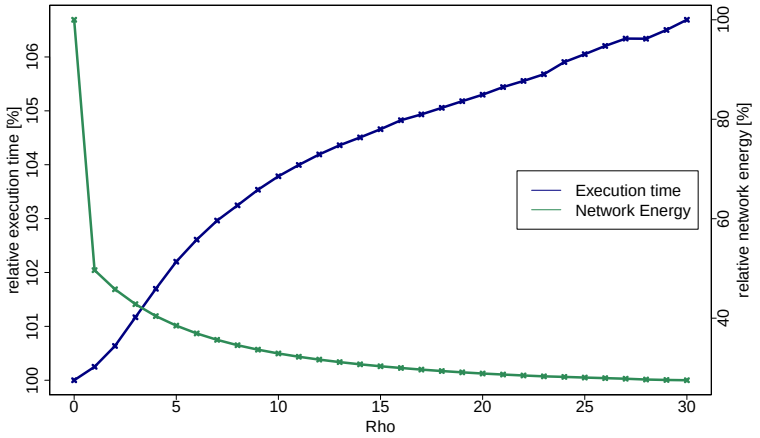
To illustrate the impact of this parameter, Figure 6.7 depicts the energy consumption (green) and the execution time (blue) for multiple combinations of ρ and t_{trans}. With no loss of generality, the analyses are performed with 64 rank NAMD-apoa1 traces. As stmv, apoa1 is a molecule that serves as input data for NAMD. However, apoa1 is significantly smaller, which facilitates the simulations of this variety of different configurations. The same trends can be observed for all other workloads.



(a) $t_{trans} = 1\mu s$



(b) $t_{trans} = 10\mu s$



(c) $t_{trans} = 100\mu s$

Fig. 6.7 Energy consumption (green) and execution time (blue) for different transition times and ρs (NAMD apoa1).

As an overall trend, the energy savings reach a plateau at about $\rho = 10\%$, with a flattening curve for longer transition times. The increase of execution time is also flattening, however, slower than the energy saving. As a result, a performance loss of 10-15% provides good results, within a reasonable performance. Additionally, the flattening of both curves allows to investigate the effects of a more aggressive energy-saving approach for instance $\rho = 90\%$ and a resulting shorter t_{down}. For the comprehensive studies, the policy of Alonso et al. has also thresholds that work in a similar way. Therefore, a conservative utilization threshold for powering a link on is set to 0.15 and an aggressive one of 0.9.

6.3.3 Evaluation

The design of the energy-saving policies entails a large set of parameters that impact their results. The goal of these evaluations is to determine the impact of these parameters. Furthermore, analyses about the practical energy savings that can be achieved and their impacts on execution time provide useful insights that help to implement energy-proportional interconnection networks.

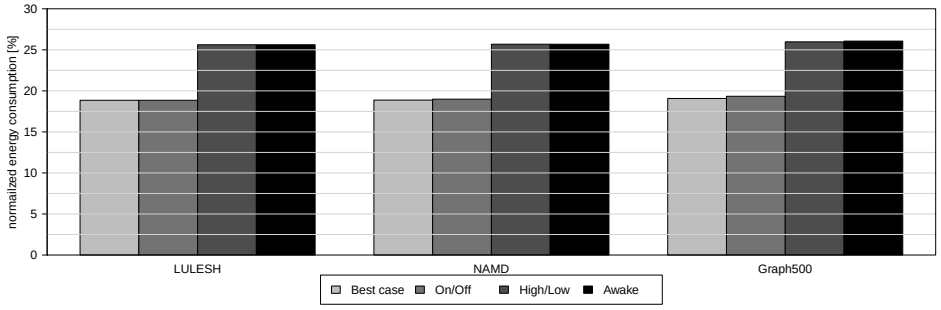
6.3.3.1 Awake

Firstly, the benefits of the awake policy are evaluated, since this policy differs from the other ones with their strict local approach. The awake policy on the other side allows individual linkports to change power states throughout the network by sending awake messages. For their functionality, it is crucial to have one unique path between every sender/destination pair because this approach is based on the assumption that all packets of a single message traverse the same path from their injection to their sink. Since the implemented routing algorithms for k-ary n-tree (destro) and dragonflies (ugal) are partly adaptive, these topologies and their respective routing algorithms are not suitable for the awake policy. Hence, this first analysis is limited to the 3D torus and with no loss of generality to the three applications Graph500, NAMD, and LULESH.

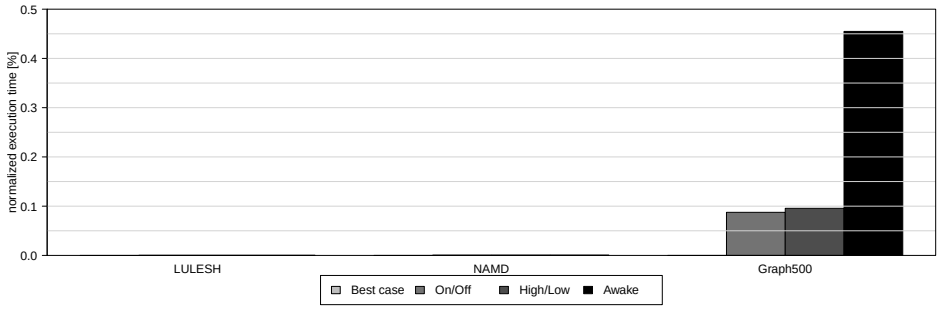
Tori are flat, direct networks, which are widely used in large scale systems due to their good scaling behavior. In order to provide favorable conditions for the energy-saving policies, a network with a transition time $t_t = 1\mu s$ is assumed, which is within the range of technical possible transition times but close to the lower limit. For the maximally allowed performance loss, $\rho = 50\%$ was selected,

which represents a middle course between a conservative energy saving and a very aggressive approach. Furthermore, this first evaluation includes a best case study, which assumes a perfect network that can switch links on and off immediately and, therefore, enables maximum energy saving without any performance loss.

The results for these studies are depicted in Figure 6.8a and Figure 6.8b. The former indicates the network energy that is consumed relative to a regular network without any energy-saving, that runs the same applications. Here, the network energy includes power consumed by all links as well as the power consumed by the switch core logic. The latter shows the changes in execution times relative to the regular network without energy saving.



(a) Energy consumption



(b) Execution time

Fig. 6.8 3D Torus: Energy saving results for all policies including a best case study and the awake policy (normalized to energy and execution time without energy saving).

Overall, the first setup provides very promising results regarding the energy-saving potentials and effects on the execution time. The difference in bandwidth from 1 and 12 GB/s between the high and low power state result in an 8.3% offset that represents the lower bound for energy savings. Therefore, a small transition time of $1\mu s$ facilitates the on/off and high/low policies, so that they

are operating close to the theoretical optimum.

This does not hold true for the awake policy. The energy consumption for this policy is even slightly higher than high/low for all workloads and the execution time is also slightly longer than for the other policies. This holds also true for other transition times, as presented in [12]. Although the results are similar compared to the other policies, the complexity of implementation for the awake policy is significantly increased. The concept of hiding latency and improving execution time does not translate to further advantages over both other policies. As a consequence, the following studies are refrained from adapting this policy to other topologies, since there no additional benefits compared to the high/low policy.

6.3.3.2 Topology-aware energy saving

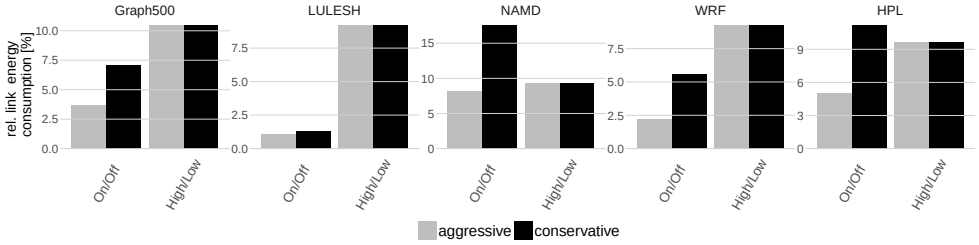
In the following, different topologies are evaluated with the two local-based policies: on/off and high/low. Furthermore, the k-ary n-tree is used to compare these technology-agnostic policies to a highly specialized one. All evaluations are performed with a rather conservative and an aggressive approach. The former means a smaller parameter ρ of 10%, which translates to a t_{down} of $1ms$, the letter indicated $\rho = 90\%$ and $t_{down} = 0.1ms$, assuming a transition time $t_t = 100\mu s$. This transition time is selected as an upper bound of commonly assumed transition times to gain more insights about how it affects energy saving.

Contrary to the previous study, the energy consumption of these evaluations is only based on link power. Link power makes up the largest part of the overall network power consumption and scales linearly with the number of links. Switch core power has an exponential scaling scheme, however, in absolute measures the link power exceeds the switch core power by far. Since the power measurements, which build the basis for this power analysis, are only evaluated for one particular radix, the effects of switch core scaling can hardly be estimated. Accordingly, it is fairer to compare only link power, since all topologies use switches at different radices.

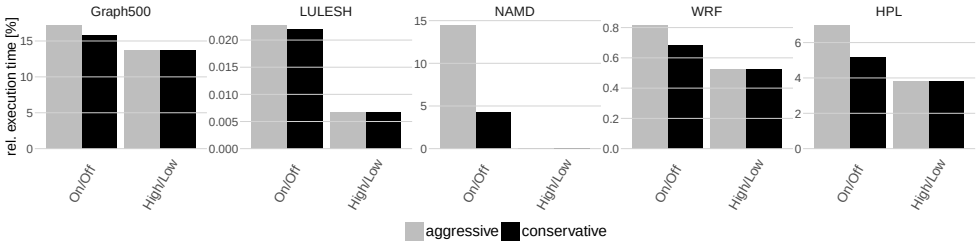
3D Torus

The first evaluations of the remaining on/off and high/low policy are also performed on the 3D torus and the resulting energy consumptions are depicted in

Figure 6.9a. The graph shows the relative link energy consumption for each workload and policy combination. The results are normalized to the network energy consumption that is consumed by running the same application on a system without energy-saving capabilities. Analog Figure 6.9b shows the relative changes in the execution time compared to a run without power savings.



(a) Link energy consumption



(b) Execution time

Fig. 6.9 3D Torus: Link energy saving results (normalized to energy and execution time without energy saving).

Among all applications and both policies, link energy is significantly reduced by more than 90%, where the combination of Lulesh and the on/off policy provides the best result by saving almost 99% energy. As expected, the on/off policy enables more energy savings than high/low, whose purpose is to provide better performance with slightly higher energy consumption. For all workloads, the aggressive approach with a smaller t_{down} provides better results in terms of lower energy than the conservative approach. A larger time t_{down} ensures that links remain in an active state when multiple small messages are on the network. This results indeed in shorter execution time since links have less often to be reconfigured, but also in higher energy consumption. As intended, the high/low policy provides a lower execution time for all applications and even keeps the performance loss within 1% for three out of five. Also regarding energy saving, the high/low policy provides good results, considering the minimal energy consumption of 8.3%, which is caused by the inability to switch links completely off. Surprisingly, there is almost no difference between the aggressive

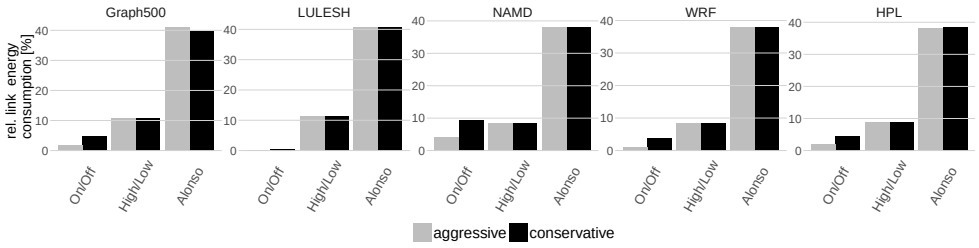
and conservative approach of the high/low policy for energy as well as execution time. This indicates, that power states are seldom changed and most links remain in the low power state.

K-Ary N-Tree

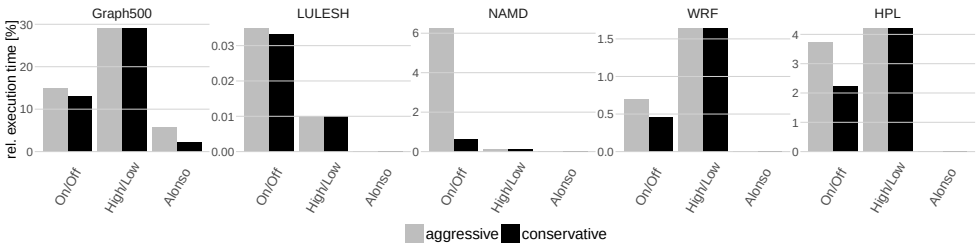
The k-ary n-tree is a special implementation of the fat-tree. Compared to other topologies, the fat-tree and the k-ary n-tree in particular have been in the focus for network energy savings so far. This includes studies of Alonso et al. [11], [79], which are used here to compare with the two topology agnostic ones. Since the number of stages and the switch radix have a considerable impact on the network behavior, both two and three stages configurations are implemented.

Two Stages

This configuration is composed of 64-port switches, which results in 1024 nodes following the design rules of a k-ary n-tree. However, the unused half of the network is not considered in energy consumption to ensure fair and comprehensive comparisons. The resulting relative link energy consumption and the execution time are depicted inFigure 6.10.



(a) Link energy consumption



(b) Execution time

Fig. 6.10 K-ary n-tree (two stages): Link energy saving results (normalized to energy and execution time without energy saving).

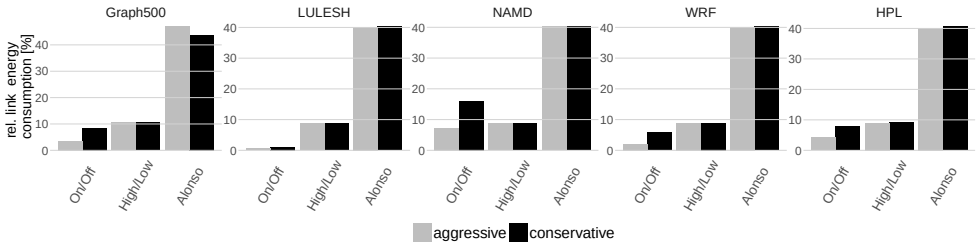
For all applications, the same trend emerges regarding energy savings. On/off provides the best energy-saving capabilities, followed by the high/low policy, which is again close to its optimum of 8.3% and a significantly higher energy consumption of Alonso's policy. Again, the aggressive approach enables higher energy-saving possibilities for the on/off policy than the conservative one, while both approaches do not differ for high/low and Alonso. The signification higher energy consumption of Alonso's policy is caused by the minimal connection tree, that establishes guaranteed fast connectivity between all nodes. This reflects also in the execution time. While on/off and high/low increase the execution time notably, the effects of Alonso's policy on the execution time are almost neglectable.

The Graph500 workload is the only exception to this. While the increase of execution time is less than five percent for all polices and remaining workloads, this increase ranges from almost 30% to 2.5% and 5%, respectively for the Graph500. Surprisingly, in three out five applications (Graph500, WRF, and HPL) on/off does not only provide higher energy-saving capabilities but also a better performance than high/low, which is in complete contrast to the torus results and its design goal. This can be attributed to the semi adaptive routing algorithm, which favors the on/off policy since the upwards direction is selected randomly between all available possibilities.

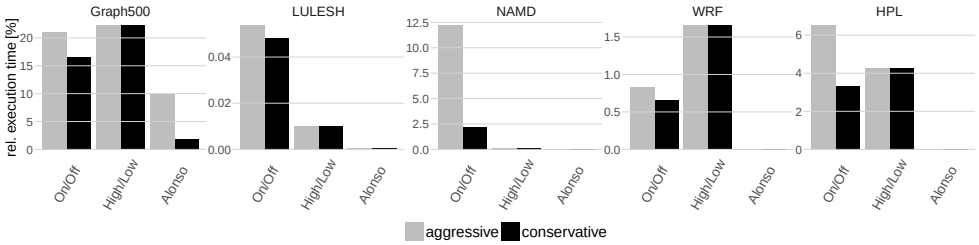
Three stages

A three-staged k-ary n-tree consists of more links than the two previous topologies. With the constant amount of traffic specified by the traces, the overall utilization here is lower than compared to the two previous topologies. Nevertheless, this does not affect the overall trends, as shown in Figure 6.11.

Compared to the two-staged k-ary n-tree there are only slight differences between these two setups and almost all trends remain the same here. While for HPL and on/off the execution time gets slightly worse with the additional stage, the relative execution time increase almost doubles for the NAMD workload. On the other hand, the performance of high/low and at the Graph500 workload increases by almost 10%.



(a) Link energy consumption



(b) Execution time

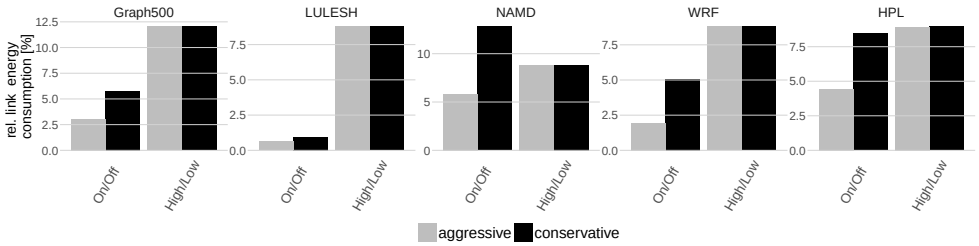
Fig. 6.11 K-ary n-tree (three stages): Link energy saving results (normalized to energy and execution time without energy saving).

Dragonfly

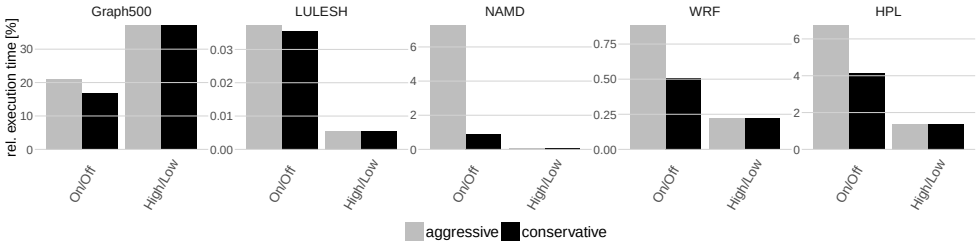
Dragonfly networks and variations, such as the dragonfly+ and slimfy, are becoming increasingly popular due to their low diameter and overall lower hardware costs. Compared to the previous network configurations, the dragonfly is equipped with the lowest number of total links as well as the lowest number of potentially costly optical links. The letter assumes, global links in the dragonfly, wrap-around links in the torus, and last-stage links in the fat-tree to be longer optical links.

Following the design rules for load balancing, the dragonfly is composed of 28 port switches, which form a 756 node network. As in the previous k-ary n-tree, all unused parts of the network are not considered for energy consumption. The relative energy consumption and relative execution time increases for this topology are depicted in Figure 6.12a and Figure 6.12b, respectively.

Throughout all combinations, the previous trends are continuing. On/off enables more energy-saving than high/low, except the conservative approach for NAMD, and the aggressive approach causes less energy consumption than the conservative one. The trends for execution time remains also the same. While high/low provides the best and the aggressive on/off approach the worst performance. The conservative on/off approach is right in between the other two.



(a) Link energy consumption



(b) Execution time

Fig. 6.12 Dragonfly: Link energy saving results (normalized to energy and execution time without energy saving).

The only exception is, similar to the k-ary n-tree, the graph500 which results in larger execution times for the high/low policy, compared to on/off. Besides the overall trends, all policies provide the best absolute energy savings in the dragonfly. The only exception, again, is the graph500, in which high/low results in slightly higher energy consumption than the torus. This is probably caused by the longer execution time, which also impacts energy consumption.

6.4 Combining Energy Saving Policies and Congestion Management

While the on/off and the high/low policy show promising results, the previously shown studies focus on these policies as the only network management entity. However, there are other commonly used network management mechanisms that are operating on interconnection networks. Probably the most common one is congestion management, which aims to avoid or resolve congestions in the network to ensure low latency and high throughput in the entire network.

There are no studies available that investigate the effects of congestion management and energy-saving mechanisms on each other and the overall performance. A mutual use could provide benefits, especially for energy saving. The reduced

bandwidth for energy-saving purposes can resemble a congested network. If both strategies complement each other, it can help to further improve performance while reducing network energy. However, both strategies follow fundamentally different approaches to reach their respective goals. While energy-saving policies aim to bundle traffic flows to utilize few links as much as possible, congestion management aims to split up traffic flows evenly over the network to reduce potentially harmful interaction between them.

6.4.1 Congestion Management

Congestion consists of intense traffic that clogs paths within the network [83]. This slows down traffic and degrades network performance. The origin of congestion is contention, which occurs when several packet flows simultaneously request access to the same output port in a switch. Moreover, congestion also occurs when a destination node is not able to remove packets from the network at the same speed they are received. In these cases and assuming lossless networks, any packet stored in a switch or NIC remains blocked in the buffer until there are enough resources available for its transmission. These blocked packets delay the advance of other packets in the same buffer. If this situation persists in time, the buffers fill up and finally, the flow-control backpressure propagates this congestion to other switches. Eventually, congestion may spread throughout the network reaching the source nodes, increasing packet latency, and degrading network performance.

In a congestion situation, not only the flows contributing to congestion (hot flows) are affected by the traffic jam. The flows not contributing to congestion (cold flows) end up advancing at the same speed as the hot ones because both share the same buffers. This situation is a particular case of the Head-of-Line Blocking (HoL) effect. HoL blocking occurs when a packet, which requests a busy port is blocked. This prevents other packets stored behind it in the same buffer from advancing, even if these packets are requesting free ports [20]. Therefore, in a congestion situation hot flows may pass the HoL blocking to cold flows if a hot packet is blocked at the head of a queue containing cold packets.

Currently, there exist two main approaches to deal with congestion. The first one is injection throttling [84], which is also included in the InfiniBand specification [17]. When switches detect congestion, they inform the source nodes

contributing to congestion to reduce their injection rates. Once congestion is removed, its derived problems, such as HoL blocking, are removed too. However, this technique does not scale with network size, as notifications may be too slow. Therefore, there are situations in which the source nodes are warned to throttle the injection, but the congestion information is obsolete [85] or the congestion has become irreversible.

The second approach is known as queueing schemes. They prevent HoL blocking by allocating packet flows to different queues or virtual channels (VCs) [20]. There are two different families following this idea. On the one hand, some techniques explicitly identify hot flows and isolate them in dynamically-allocated VCs, such as the mechanism described for ATLAS [86], the RECN mechanism [87], or EcoCC [85]. However, they require additional and expensive resources that are not supported by current commercial interconnection networks. On the other hand, other techniques allocate packets from different flows to VCs according to a static mapping policy, independently of the traffic conditions, the topology, or the routing algorithm. Proposals such as VOQnet [88], VOQsw [89] or DBBM [90] follow this idea. Although most of these techniques use resources available in commercial networks, they only prevent HoL blocking partially or they are not feasible in large networks (e.g. VOQnet). By contrast, other solutions are specially designed to be aware of the topology and the routing algorithm, so that HoL blocking is reduced more efficiently and/or by using fewer resources. For instance, queueing schemes such as OBQA [91] and vFTree [92] have been devised for fat-tree topologies [16] using the DESTRO [93] and D-MOD-K routing algorithms [94], respectively. IODET [95] considers the torus topology [16] and its dimension order routing algorithm. BBQ [96] is designed for the KNS topology [97] with the Hybrid-DOR routing algorithm. H2LQ [98] is tailored to Dragonfly topology using its minimal routing [29]. SF2LQ [99] is intended for Slim Fly networks with its minimal routing [100].

6.4.2 Methodology

Similar to the previous evaluations, there is a huge design space from which simulation settings can be elaborated. The goal of these analyses is to investigate how energy-saving policies and queueing schemes interact with each other. Although both techniques can be tuned by multiple parameters, evaluations in this work

are performed with default parameters. The main difference is the usage of a synthetic traffic pattern instead of application traces.

Network Simulator and Traffic Pattern

As the previous analyses, all evaluations are performed on the cycle-accurate SAURON network simulator. The traffic in the simulated network, however, is synthetically generated and not trace-based any more. The reason for this change is that congestion management needs a high network utilization to show impacts on networks performance. On the other hand, energy-saving policies exploit idle periods which are often caused by computing periods, to reduce energy and the synthetic pattern predetermines energy-saving efficiency. But since the following evaluations focus on the relative effects of both mechanisms on each other, the character of the traffic pattern has only minor effects as long as both mechanisms are utilized. Hence, synthetic hotspot traffic is used to stress the network enough to evaluate the impact of different queuing schemes: This means, 25% of all nodes, which are randomly selected, send messages to a single "hot spot" node. The remaining nodes generate evenly random traffic on the network. The fixed load is set to 40%, which equals a 40% utilization of the input bandwidth.

Analog to the previous studies, these evaluations are performed on the same three topologies, but in a larger size of 1024 and 1056 nodes, respectively. The 3D torus is composed of 1056 (12x11x8) nodes, the k-ary n-tree is configured in 5 stages and a switch radix of 8, resulting in 1024 nodes, and the dragonfly consists of 33 groups with switches with a radix of 15, resulting in 1056 nodes. The routing algorithms remain the same as in the previous studies.

Energy Saving

The on/off and high/low policy are used here to evaluate the energy-saving capabilities. Contrary to the previous trace-based studies, the synthetic traffic pattern used here is rather disadvantageous for the policies. The increased load on the network leads to high utilizations on the path to the hot spot node, while the even random traffic prohibits long idling periods. In order to get deeper insights into the interaction of energy-saving policies and congestion management, the policies are configured for rather aggressive power saving. The transition time $t_t = 10\mu s$ is shorter than in the previous study to enable energy-saving despite

the rather high utilization. An overview of the used parameters is depicted in Table 6.4.

Parameter	Value
t_t	$10\mu s$
ρ	90%
t_{down}	$11.1\mu s$

Table 6.4 Energy saving parameters for congestion management studies.

Congestion Management

Multiple queuing schemes are also evaluated for congestion management. This includes topology-agnostic ones (VOQsw and DBBM) as well as one queuing scheme that is specially tailored to each topology. The particular queuing schemes are:

VOQsw Virtual Output Queues at switch level [89] can reduce HoL blocking produced by a congestion tree in all topologies. Each input port has one virtual queue for every output port in which packets according to their requested output port are stored.

DBBM Destination Based Buffer Management[90] is also applicable for all topologies. In this scheme, packets are with destination D are mapped to D modulo number of VOQs per port. Here, four virtual output queues are used.

IODET In-Order DETerministic routing[95] is a queueing scheme tailored to direct topologies. It assigns packets to VCs according to the dimension for the next hop.

Flow2SL This technique[101] is a topology- and routing-aware queuing scheme, specially tailored to fat-trees using deterministic routing. Flow2SL defines groups of end nodes and maps flows that have the same source group and the same destination group to the same queue (or VC), while flows that have the same source group but are addressed to different groups are mapped to different queues.

H2LQ Hierarchical Two-Level Queuing [98] is designed for dragonfly networks, in which it reduces HoL blocking and guarantees deadlock freedom. It splits traffic flows up in a standard virtual network (SVN) and an escape virtual network (EVN). The mapping to different VCs in the SVN depends on the destination, while the VCs of the EVN are used to prevent deadlocks. Six VCs for the SVN and two VCs for the EVN are selected for the evaluations.

6.4.3 Evaluation

While hot spot traffic patterns are commonly used to evaluate queueing schemes, it rather not suitable for energy-saving which exploits network idling periods. Hence, congestion management effects might dominate compared to energy-saving policies. Another difference to the previous studies is the usage of different metrics. Throughput and packet latency are the typical metrics to evaluate congestion management and they are also used in this context. Since synthetic traffic has no determined execution time but rather a fixed simulation time, this is no sufficient metric in these evaluations. Although there are no changes in the execution time due to transition times during link reconfiguration, there are differences in the amount of data that is injected into the network at this time. Therefore, the metric energy/data is introduced to evaluate the energy-saving policies.

The baseline for all topologies is one input queue (1q), which means there is basically no congestion management. Additionally, two topology-independent schemes (VOQsw and DBBM) and one scheme tailored to the respective topology are used. All simulations are performed with synthetic hotspot traffic, generating 40% load.

3D Torus

Figure 6.13 depicts the results for throughput, latency, and energy/data in the 3D torus. For the torus, IODET is used as the topology-specific queueing scheme.

Overall, VOQsw provides the worst results regarding performance and energy efficiency, respectively, which are even a decline to the default case without congestion management. These setbacks are caused by the poor fit of VOQsw and the employed xyz-dimension-order routing algorithm, which routes messages firstly in the x-dimension. VOQsw, however, also divides flows in virtual channels

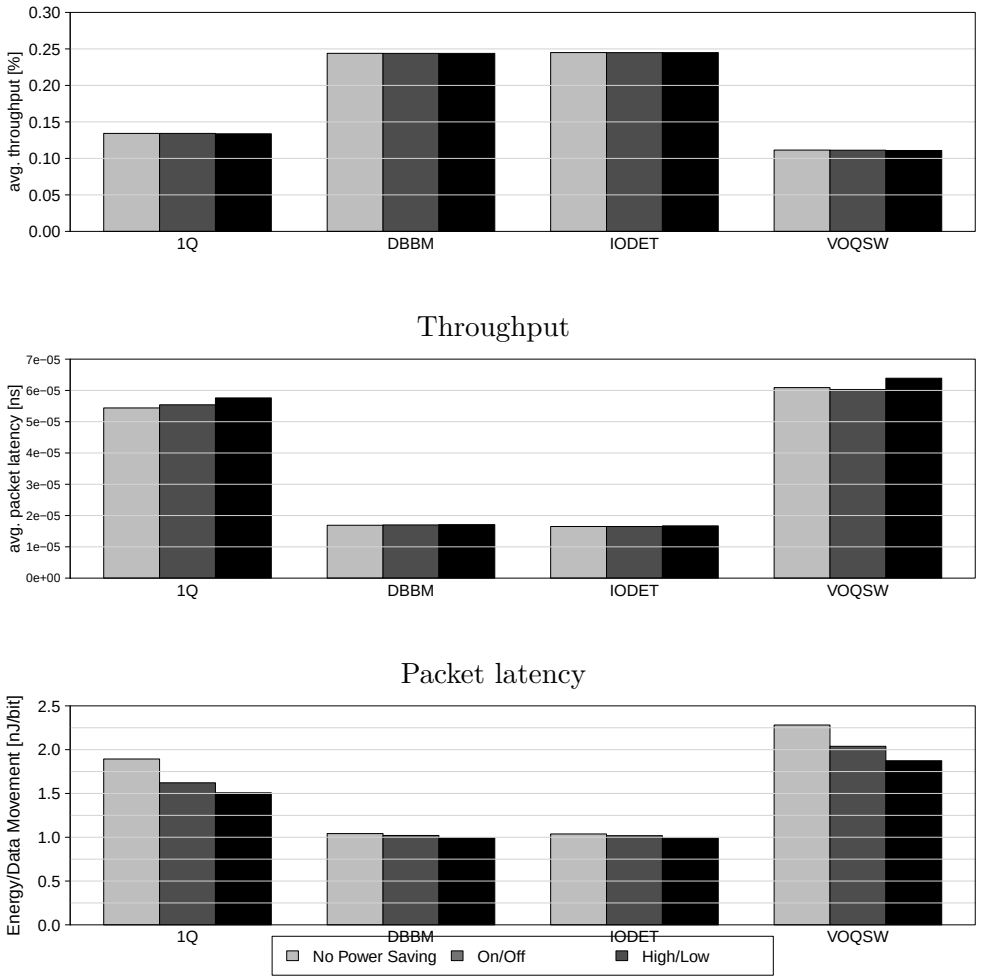


Fig. 6.13 Results for 3D torus.

according to the respective dimension, which deteriorates the situation since buffer size is effectively reduced to only $1/\#ports$. The remaining queuing schemes provide good results without many deferences between them.

Across all configurations, the energy-saving policies follow the same trend. Regarding throughput and latency, there are only little differences between the policies. Only at the 1q and VOQsw configuration does high/low provide slightly worse performance values. However, this results in also better energy efficiency. The reason for these unusual effects is the random nature of synthetic traffic. Because on/off can only switch links on completely, even if only for a small message, there are more links in the network that are in "on" state, providing high bandwidth but also high power consumption. On the other side, the absence of traffic bursts prohibits high/low to switch back to a higher power state, after

a link is set to the lower power state. Overall this results in lower bandwidth and better energy efficiency.

Another finding is that congestion management seems to affect energy efficiency even more than the energy-saving policies. Again, this is likely caused by the unfavorable traffic patterns for the energy-saving policies. Key insights are also the results for DBBM and IODET, which both provide the best performance results and show no significant differences between the different policies. This suggests that a favorable combination of queueing scheme and energy-saving policy can improve performance even in adverse conditions.

K-ary N-tree

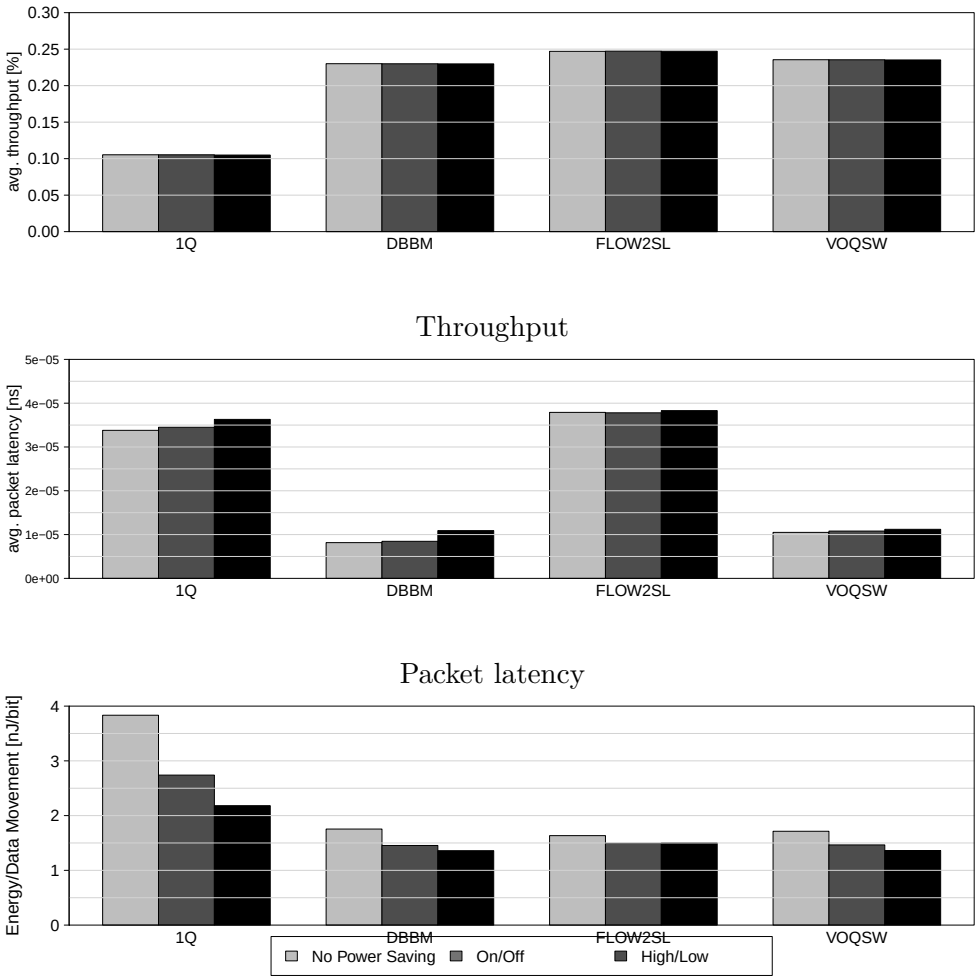
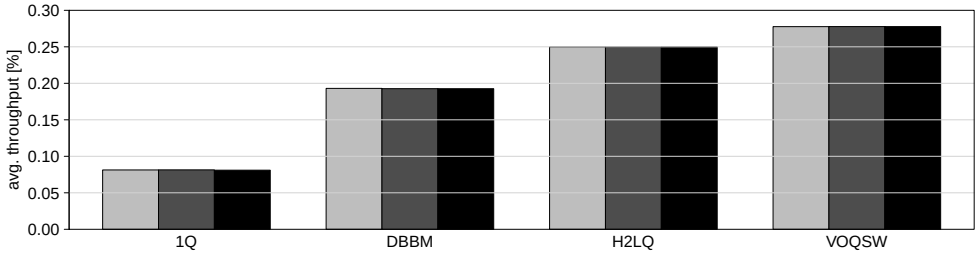


Fig. 6.14 Results for k-ary n-tree.

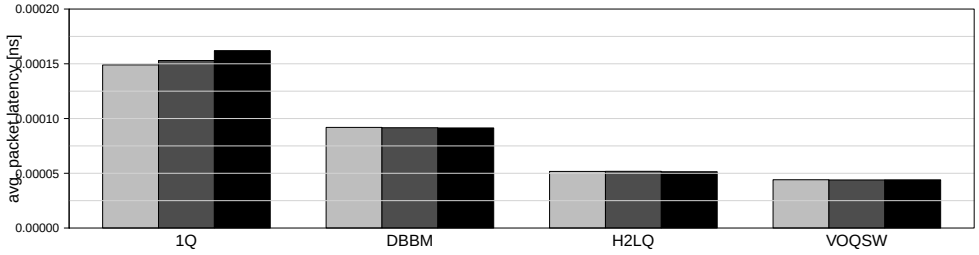
For the k-ary n-tree with the respective topology-specific queueing scheme Flow2SL, the results for performance and energy efficiency are depicted inFigure 6.14. Unlike the k-ary n-tree configurations in the previous studies, this configuration consists of many small switches (radix = 8) in five stages. While all queueing schemes show good results in terms of throughput, the latency of the Flow2SL is worse than without congestion management. A reason for this variation could be the Flow2SL mapping policy, which smartly balances flows among available queues or VCs in the upwards stages of the fat-tree. All flows that are addressed to the same group are mapped together in the same VC, although they are addressed to different destinations. Therefore, the mapping in the downward stages introduces delays as congested flows may share queues with not congested ones if both of them are addressed to the same group. As in the torus, the on/off policy provides better performance, but high/low shows better results in terms of energy efficiency. Overall it seems that the 3D torus provides better energy efficiency, while the k-ary n-tree results in lower packet latency.

Dragonfly

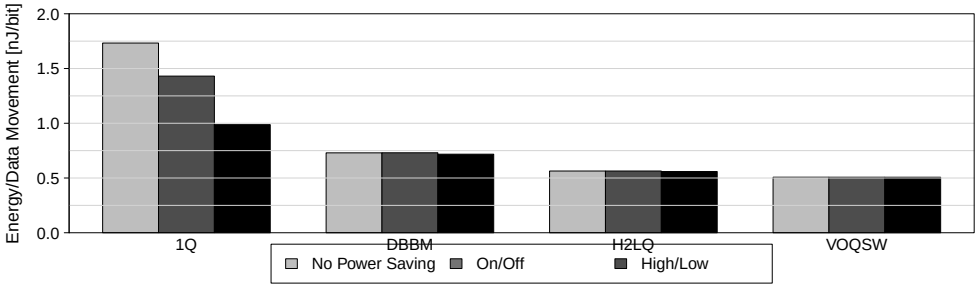
The topology-specific scheme for the dragonfly is H2LQ queueing. All results for this topology are depicted in Figure 6.15. This time all queuing schemes improve performance in terms of throughput and latency compared to the 1q configuration. VOWsw enables the highest throughput and the lowest latency and, therefore, performs slightly better than H2LQ and DBBM. Furthermore, congestion management is increasing link utilization to the effect that there are no significant differences between different energy-saving policies anymore. This effect is also amplified by the design of the dragonfly, which is equipped with few but highly utilized links between different groups. Still, even though energy-saving policies do not improve energy efficiency significantly, they also do not harm performance. Comparing all three topologies, the dragonfly enables the highest energy efficiency among all configurations. This matches the previous studies that focus exclusively on energy saving.



Throughput



Packet latency



Energy/Data

Fig. 6.15 Results for dragonfly.

CHAPTER 7

Discussion

This chapter provides an overview of related works that deal with a similar research question or partial aspects of this work. Furthermore, the previous findings are further evaluated and discussed. Last, this chapter concludes with a short outlook about future work and open research questions.

7.1 Related Work

Although energy efficiency is a hot topic in almost all computing systems, including especially HPC systems and cloud installations, energy-efficient networks are rather a special topic. While there exist several works on energy saving in on-chip networks, there are fundamental design differences and completely different technical constraints, such as physical distances and other signaling issues, or different kinds of topologies compared to interconnection networks. However, they aim to tackle similar problems. Most approaches, including the works of Samih et al. [102], Parikh et al. [103] and Chen et al. [104], [105], rely on power gating to switch routers completely on or off to reduce energy in their network, which is not suitable for interconnection networks.

Of particular inspiration from the NoC-domain is the work of Soteriou and Peh [106]. Although they are also analyzing energy aspects and optimizations for networks at different scale and integration, their methods can be adapted to networks at every scale. Especially the methodology of link idle time distributions and the detailed description of their approach is very helpful.

In regards to interconnection networks, Abts et al. [4] demonstrate the growing need for more energy-proportional networks. The work identifies opportunities for optimization by determining the power consumption analytically. However, a detailed power model or other solutions are not provided

Additionally, Mahadevan et.al. [107] provide a survey and description of network power consumption for a variety of devices. While the authors do not attempt to explain their observations, they completely back up our results, in particular regarding the impact of utilization and number of links to power consumption.

A new approach for gaining deeper insights on the power consumption of interconnection networks is presented by Wang et al. [108], [109]. They developed an activity-based power model for fundamental switch components. This helps to further broaden the understanding of switch power consumption at a lower level and to potentially identify additional sweet spots for energy saving.

Taylor Groves et.al.[110] provide a static performance and power analysis for large dragonfly networks. While they focus only on one topology at another scale, they observe also a very low utilization in interconnection links. Additionally, they scale down underutilized links statically to reduce bandwidth and, thereby, save 6-10% of total system power.

Another approach to reducing energy consumption in interconnection networks is adapting software that runs on a system. Dickov et al. [111] are using MPI data compression to reduce the overall network traffic. The authors argue that a slower and, therefore, more energy-efficient network is sufficient for the reduced load. In another work, Dickov et al. [81] adapt PMPI, the MPI profiling layer, to intercept MPI calls in order to hide transition times when changing power state. While in this approach links are always available when needed, the overhead is shifted to PMPI. Similarly, Hendry [112] proposes Asynchronous Circuit Programming (ACP), which allows the programmer to closely interact with the underlying hardware through a high-level interface to establish communication channels and additional features for HPC applications. Such an approach can guarantee near-optimal state selection as explicit knowledge about application behavior is present, however, the responsibility is shifted from architecture to the programmer with substantial implications on programming complexity.

Also, in [74] Venkatesh et al. introduce an energy-efficient MPI runtime. While the authors focus on optimizations regarding the MPI layer, this work

rather optimizes power consumption at the link level and independent of a particular programming language. However, the here introduced energy-saving policies adapt their algorithm to calculate the threshold for switching network links into a low-power state.

On the hardware side, Kim et. al [82] introduce traffic consolidation for energy-proportional high-radix networks (TCEP). This concept, which relies on path diversity, and defines a minimal set of links that are needed to ensure connectivity, uses management software that centralizes traffic to the selected links. The remaining links can be switched on and off or set to a "shadow state" in which the link is physically on, but not available for traffic. However, this approach relies heavily on high-radix networks with high path diversity, full adaptive routing, and global network management. In contrast, this work's policies function technology agnostic and use local information that is available in each link port.

Principally, different techniques including ACP, continuous clock locking for improved transition time, traffic consolidation, and technology-dependent features can be combined. In particular, link-level power saving is naturally compatible with high-level power-saving methods. Similarly, the combination of energy-saving and congestion management combines also low-level methods with higher-level network management.

Saravanan and Cerpente [5] explore the impacts of Energy Efficient Ethernet (EEE) on HPC applications and introduce an approach similar to our on/off policy. The authors also propose a concept in which power-downed links are regularly woken up to ensure continuous clock locking and word alignment. In [9], Saravanan and Cerpente extend their studies by investigating energy-saving possibilities using an additional EEE sleep state "Fast-Wake". In contrast to this work, these works focus on specific EEE features. Furthermore, topology-pending effects, such as congestion along a certain routing path due to switched off links on this path, are not considered in these studies.

Closely related to our work is the work introduced by Alonso et al. [10]. The authors propose two different energy-saving techniques for regular interconnection networks, which include all networks that are connected by high-degree switches. First, they save energy by switching links between different switches or nodes on and off. Second, they adjust their strategy by a dynamic bandwidth reduction instead of instantly switching links completely off. In order to implement this

gradual reduction, they use several parallel links for every dimension that can be switched on and off individually. This very hardware-specific set-up is also the main difference between their and our approach. Instead of proposing a solution for a specific technology (InfiniBand), this work rather explores fundamental technology aspects and constraints. However, their solution is effective and inspiring. Furthermore, the same authors propose a policy for on/off links specifically tailored to fattrees [11], [79]. Their policy is also used in this work to compare with the topology agnostic ones.

Andujar et al. [113] introduce a new power-aware routing algorithm for fat-tree and torus networks. This routing algorithm is an addition to energy-saving policies. It takes into account which links are in a low power state by utilizing other links instead and thereby reducing impacts on performance. This is a very useful extension that increases the usability of power states in hardware, such as EEE.

7.2 Workload Analysis

Analyses of exascale proxy-applications regarding their communication patterns are important to understand the energy-saving potential of HPC applications. Although these are static analyses that do not provide any temporal information, they reveal useful insights about traffic flows and idle times.

7.2.1 Locality and Selectivity

At the application level, the new metrics rank locality and selectivity are introduced. They are particularly useful to quantify common traffic density plots, which enables their usage as an input parameter for abstract models.

Overall, most applications show a rather low selectivity even compared to their number of peers. Then again, rank locality decreases significantly with the number of ranks. This indicates that although there are just a few distinct communication partners, these communication partners are often not direct neighbors. Instead, the segmentation of ranks follows a multi-dimensional pattern in which ranks with distant IDs are co-located. This finding coincides with the study about the dimensionality of the underlying problem, in which many applications show 2D

or 3D characteristics. Furthermore, this explains why a large share of inter-node traffic remains when modeling a multi-core system with up to 48 cores/node.

However, these results and especially the low selectivity indicate that an optimized mapping could decrease inter-node traffic significantly. To identify such a mapping and profit from faster on-chip communication, a deeper understanding of communication pairs is necessary. Another approach to reduce network traffic and to shorten long network paths is tailoring a network topology to a given application, as the dimensionality analyses suggest. If the topology matches the underlying problem, the nearest neighbor communication could be exploited even if communication pairs are distributed between all ranks. Still, the best approach and the degree of possible optimization is highly depending on the application.

7.2.2 Topology Effects

Following the application-level analyses, the behavior of the proxy-applications in three different topologies is modeled.

Surprisingly, the dragonfly shows the highest average hops for most applications. This is particularly unexpected since the dragonfly is often considered as a low-diameter topology. Especially, the division of nodes in many different groups prevents it from exploiting locality that is present at the application level. However, the design rules for load balancing result in rather small groups, another design with larger groups would probably not harm traffic flows but improve locality effects. Furthermore, as the maximum distance for a dragonfly is bound to five hops, further scaling the number of ranks would benefit this topology.

Despite its large diameter, the 3D torus provides the best locality properties for many applications at a different scale. This might be due to the structure of underlying problems. As shown in the dimensionality study, many applications have a three-dimensional character, which can be perfectly mapped into a 3D torus. Nonetheless, at larger scales of 256 ranks or more the increasing diameter exceeds these dimensional benefits and becomes dominant. The fattree proves to be a solid fit that ranks right in the middle between both other topologies for most applications and scales.

Another finding is that there is no explicit correlation between application-level and system-level locality. Deriving topology effects directly from the MPI

level would provide many benefits and help to improve network performance and energy savings easily. One of the few trends that can be recognized is that a low selectivity and rank distance often indicate a 3D torus to be the best fit, but even this finding does not hold true for all applications. Overall, it is hardly possible to drive absolute findings for all cases, even if there are some indications. One way to improve this situation could be the division of applications in distinguished subclasses. However, one goal of the section of exascale proxy applications was to cover all kinds of different communication and computation behaviors. It is reasonable to assume that they also differ significantly regarding their locality properties. Hence, further studies with different subsets of similar applications could be useful to identify correlations between these metrics.

7.2.3 Network Utilization

The analyses show, that over all combination of scale, application, and topology, the network utilization is very low. This emphasizes the huge energy saving potentials in interconnection networks. Although they are only sparsely used compared to the entire execution time, they are projected to consume 20%-30% of the system power in the near future.

Taking a closer look at the different topologies, it shows that the fattree has the highest utilization of all topologies for most applications. The dragonfly and the torus show a two-folded trend. While at a small scale the dragonfly has higher utilization, the torus exceeds the dragonfly on higher scales. The reason for this trend the links/node ratio of the different topologies: with as many global links as nodes and twice as many switches per group, the ratio of links/node in the dragonfly varies from 3.5 to 3.8 for the used configurations, while a torus has a constant ratio of 3 and a fat tree one of below three. The turn around in the trend at particular scales for the dragonfly and torus is further based on their particular locality properties. While the link/node ratio favors the torus for a higher utilization, its good locality reduces the number of hops, resulting in lower network utilization. With increasing diameter locality worsens and consequently packet hops and utilization, respectively, increase. However, the absolute numbers are too low to enable reasonable comparisons. Also note, that this study considers shortest-path routing, while in practice usually adaptive routing is used in dragonfly networks, which often results in even longer paths.

7.3 Energy Savings

Next, in Chapter 6, energy-saving policies are introduced and evaluated. The key findings and different aspects of energy-saving are discussed in the following section. This includes the three energy-saving policies, as well as important parameters and the setup they are running at, e.g. topology aspects.

7.3.1 Policies

Three different energy-saving policies are introduced in the context of this work. Two of them operate entirely on link-level, where the on/off policy aims for optimizing energy savings and the high/low policy intent to equilibrate energy savings and good performance. The third one (awake) is located between a link-level and system-level approach and targets to further improve performance.

Overall, all policies provide surprisingly good results. Energy savings of more than 90% are achieved in the majority of set-ups. Furthermore, they suggest that especially on/off and high/low operate quite well according to their respective purpose. In nearly all set-ups, on/off enables more energy savings, but increase the execution time more than high/low does and vice versa.

Regarding high/low, the results show that also this policy is operating close to its optimum. Since links are never switched off completely by this policy, there remains a constant energy consumption of the low power state that cannot be undercut. In the low power state, only one out of twelve parallel lanes inside one link remains active, which means that a baseline of 1/12 or 8.33% of the overall power consumption represents the upper bound for energy-saving by this policy. The multitude of results that are close to this optimum suggests, that many links that are put to the lower power state are never utilized enough, that the link switches back to the high power state. This affirms that networks are usually over-provisioned if a bandwidth reduction of 91.66% only shows little to no impact on the overall execution time. On the other hand, the threshold when a link is switched back to a high power state only relies on the bandwidth difference between both states. Hence, studies with even slower low states would provide interesting results about the optimum distribution of power states.

While both policies at link-level work well, the awake policy falls short of its expectations. The purpose of this policy was to further improve performance

compared to the high/low policy. The resulting energy consumption and execution times are comparable to the other policy but do not exceed their results. However, the implementation of this policy involves considerable additional effort, due to its awake mechanisms and the introduction of the new awake message type. The cost-benefit analysis states that both other policies are preferable over this one. In addition, this policy requires a strict deterministic routing, which is usually not used in fat-tree and dragonfly topologies. The absent performance improvements are presumably due to the lack of coordination between different awake messages and unintentional interactions among them. This suggests that a hybrid approach between a link- and system-level management is difficult to implement without a central management unit, especially if it only relies on local information.

7.3.2 Energy-Saving Parameters

Since the awake policy did not provide the expected results, further studies focus on the two remaining policies. Note, that these studies only took link power into account due to the variations in switch radices. Overall, both policies show good results in their respective domain. Especially the differences between the aggressive and conservative approaches stand out. The on/off policy with an aggressive approach enables more energy savings than the conservative one but at the costs of a further increased execution time. While this behavior was to be expected, the significant differences between both approaches are rather surprising. As depicted in Figure 6.7, the energy consumption reaches a plateau quickly with an increasing ρ, the execution time, however, continues to increase. Hence, ρ should be selected as small as possible after energy-saving have reached the plateau. This indicates that in some configurations the parameter $\rho = 10\%$ was two low and a sightly higher selection could enable even more energy savings at nearly the same performance. The opposite trend is shown for high/low. Here, the aggressive and conservative approach have almost identical results, which suggests that in both policies links are mainly operating at the low power state. As discussed previously, this indicates that the network utilization never or seldom exceeds the threshold to switch links back to the high power state and that overall the network is over-provisioned for this application.

Another important parameter for energy saving is the transition time.

Through the course of the evaluation, different transition times are used. As expected, it shows that smaller transition times facilitate energy-saving policies. Thereby, smaller transition times do not only reduce the overhead while changing power states but also increase the energy-saving opportunities. In both policies, t_{down}, which determines the time an idling link remains active before it is switched off or to the low power state, is proportional to the transition time and, therefore, smaller transition times enable exploiting smaller idle periods in the traces. These findings are also backed up by further studies [73].

7.3.3 Topologies

Regarding the 3D torus, both policies provide expected results. The only exceptions are the NAMD and HPL workload, for which the on/off policy in the conservative setting result in fewer energy savings than the high/low policy. As discussed previously, a larger ρ would probably be beneficial here. However, also the aggressive approach enables only slightly more energy-savings than high/low. This can is caused by higher execution times, which also affect energy consumption. Additionally, the high execution time increases for both policies while running the Graph500 are notable. Since this trend continuous in all other topologies, it suggests, that the Graph500 traffic pattern is adverse for energy-saving policies. If this is particularly Graph500 specific or rather caused by traffic patterns that heavily rely on collective operations in general, remains to be analyzed in further studies.

The k-ary n-tree topology was evaluated in two different designs (two and three stages) and the energy-saving policies are compared to another policy that is tailored specifically to this topology. Regarding the number of stages, both set-ups show overall similar results and there are only slight differences in the absolute numbers. However, the results focus only on link power consumption, and fewer stages are required for higher radix switches. Therefore, the rising core power consumption of these switches could further affect energy consumption.

Surprisingly, for the Graph500, WRF, and HPL, the high/low policy results in larger execution times that on/off. This performance is likely caused by the routing algorithm, which is particularly designed for load balancing. The adaptive character in the upward direction of this algorithm is able to use few fast links that remain active by the on/off policy, where traffic flows are spread

evenly over the links in low power states while operating at high/low. This impacts particularly patterns that continuously send messages instead of bursts in a global communication phase.

The comparisons between Alonso's policy and the two newly introduced ones provide contrary results. Both on/off and high/low provide significantly higher energy savings (about 60% for Alonso and more than 90% for on/off), however, Alonso's policy results in almost no increase in execution time. Both can be attributed to the design of this policy, which includes a minimal set of links, that remain at full bandwidth in order to provide full connectivity. On the one hand, this limits the total amount of power that can be saved, on the other hand, there is always an available path for traffic flows and if this minimal set of links is over-utilized, additional links can be switched on. While both approaches provide advantages and drawbacks, the implementation of Alonso's policy could solve the problems of potential performance issues that are particular to be avoided in HPC systems. However, a combination of both approaches could potentially bring the benefits of both together.

Last, the policies in the dragonfly topology, show the same overall trends as in the 3D torus, including the better energy-saving for high/low compared to the conservative approach of on/off while running the NAMD workload. However, the dragonfly outperforms the torus in total numbers for execution time and energy savings. This back ups the overall trend that dragonfly provides not only good performance results but is also highly suitable for energy saving in the interconnection network.

7.4 Congestion Management

Last, the interactions of the energy-saving policies and congestion management as another important network management unit are studied. Potential obstacles are the contrary goals of both techniques. While energy-saving is based on the concept that traffic flows should be combined to switch off unused links, congestion management contrarily aims to split traffic flows across the entire network.

Generally, the studies provide positive results, which suggests that congestion management and energy-saving policies work well together. In all evaluated setups there is no significant negative effect on performance and energy efficiency.

The only exceptions are minor increases in latency in few configurations. However, there is always a combination of an energy-saving policy and a queueing scheme for every topology which provides almost the baseline packet latency.

Another finding that might surprise on the first sight is that the queueing schemes provide more benefits for energy efficiency that the energy-saving techniques. This can be explained by the synthetic traffic pattern that is disadvantageous for the energy-saving policies. The rather high network utilization, which is necessary to see the effects of congestion management, prohibits the energy-saving policies from reaching their potential that they have shown previously. Despite the rather obstructive traffic patter, the energy-saving policies show little improvements regarding energy efficiency in most cases. Additionally, even in other configurations in which they do not impact energy efficiency, they have also no impact on performance.

The uniform traffic has also an unexpected effect on the high/low policy that usually aims to provide better performance for the cost of fewer energy savings. However, this behavior is turned around in these studies. While high/low shows in most cases a slightly worse latency and throughput, it enables the highest energy efficiency in all configurations. The reason for this is the evenly distributed network traffic. While there are idling periods that are long enough to switch in the lower power state, the absence of traffic bursts prevents the buffers from filling up and, therefore, the links remain in the low power state for the residual execution time.

Overall, these results suggest that advanced queueing schemes can maximize the utilization of congested links. Especially in a dragonfly network, queueing schemes are able to increase the average link utilization from 11.2% up to 28.1%. This is backed up by the small differences between the energy-saving policies: the increased link utilization causes a decreasing potential for energy savings since links are significantly less idling, resulting in almost no power state changes. Therefore, more realistic workloads based on application traces should be evaluated to analyze if these initial insights that both policies are compatible holds in more realistic scenarios and to gain a comprehensive and deeper understanding.

Last, it should be noted, that both techniques are tested in their default configuration. It stands to reason that coordination between both would enable further benefits such as energy efficiency. Compared to hardware properties (42.7

$\frac{nJ}{bit}$ at 100% network utilization), measured efficiency is worse by one order of magnitude in the best case. Therefore, further studies on actual data movement costs in terms of energy under realistic conditions would be useful to evaluate sustained energy efficiency.

7.5 Outlook

This work has shown that there is a wide range of opportunities for energy savings in interconnection networks. Hence, it is important to further exploit these opportunities to shift future interconnection networks to more energy-proportionality. Considering the increasing impact of the network on the overall energy consumption predicted by the IRTS and Abts et al. [4], there needs to be a change in the focus of new hardware designs. Several works have studied energy aspects in networks and proposed multiple solutions, ranging from large savings if tolerating longer execution times to moderate savings without notable performance decreases. Although, many new interconnection hardware is lacking respective features, some vendors or standards, such as EEE enable different power states to improve energy efficiency.

One remaining research question is the best granularity for power states. As long as they do significantly vary in their efficiency, coarse-grained power states seem to be sufficient to operate in few distinguished configurations and exploit long idle periods and low utilization. On the other hand, an almost full energy-proportional network would enable fine-grain adjustments correlated to the actual utilization.

Another hardware-specific topic is transition time optimization. The studies in this and a previous work [12] have shown, that shorter transition times improve energy saving significantly. Besides the obvious performance improvements, this would also enable the possibility of more aggressive energy saving that can exploit even short idle periods.

Regarding energy-saving mechanisms, this work has shown, that even simple link-level-based approaches provide good results. Furthermore, especially the studies about the collaboration of energy-saving policies and congestion management suggest that the combination of high- and low-level energy-saving methods could be beneficial for the overall efficiency. However, global distribution of network statuses in real-time remains a key challenge for these high-level concepts.

While code instrumentation is a good approach, in theory, it shifts the duty from the interconnection hardware to software engineers. Especially against the background that many HPC applications are based on scientific models that run on rather ancient code, it is hard to imagine that many of these applications would be instrumented to improve energy efficiency. Other approaches, however, such as global guiding of traffic flows to improve utilization of particular links [82] or the introduction of new routing algorithms that are tailored to improve energy-saving properties [113] would be promising targets for such combinations. Adaptive routing could generally be beneficial for energy saving mechanisms, since links that are currently not available due to being switched off or being in reconfiguration, could be bypassed spontaneously. However, the clash of three network management techniques (including congestion management) has the potential for conflicts, if they are not coordinated. Further studies about different combinations would be interesting and potentially yield improvements for both, performance and energy efficiency.

Last, a wider range of application analyses could also contribute to reducing network energy. On the one hand, a more detailed understanding of communication patterns and especially temporal manner of traffic flows would form the basis for more advanced and aligned energy-saving techniques. On the other hand, applications themselves could provide the potential for energy reduction in the network design. For instance, many applications try to overlap communication latencies with computation. This could be exploited by lowering the overall bandwidth so that data arrive just in time when they are needed. This MPI slackness could be determined by statically analyzing the leeway this overlapping produces.

CHAPTER 8

Conclusion

Energy is a key factor in the design and operation of HPC systems. In addition to economical motivations, energy consumption is also limited by technical constraints, including cooling capabilities and power distribution, as well as ecological reasons, such as the minimization of carbon footprints. With processors and memory becoming increasingly energy-proportional, interconnection networks increase their share of the total power consumption and rise to the focus for further energy-saving capabilities. However, networks fundamentally differ in their underlying technology and access patterns from other components. Hence, a comprehensive and complete study of energy-saving capabilities is key to further increasing energy-efficiency.

One major difference between interconnection networks and other components that are CMOS-based is the underlying CML technology. This logic is widely used in the design of serialization technology and dominates the power consumption of networks. However, established power-saving approaches from other components, such as DVFS, cannot be applied here due to the constant current of CML. As a result, when only hardware is considered, energy-saving in interconnection networks can be best achieved by adjusting link width to the current utilization. Another important performance issue is the transition time, in which the link training is performed after every power state change. This delay prevents messages from being sent, which likely increases the execution time of an application, and, therefore, limits the potential number of link width adjustments.

To address this issue, this work performs further analyses of application

executions, which provide insights about the frequency of required adjustments from the perspective of the software. Commonly, interconnection networks have a reputation as a performance bottleneck, since inter-node data movements are much more expensive in terms of time compared to local data access. This work provides broad studies about the actual communication patterns and the locality effects in a wide range of exascale proxy-applications. In addition, two new metrics (locality and selectivity) are introduced to obtain objective and comparable results. Although there are partly intensive communication phases, the overall network utilization is lower than 1% for all but one application and long idle periods occur between communication phases. Furthermore, the new selectivity metric identifies that ranks send 90% of their overall point-to-point communication to only a small subset of other ranks. This suggests that an advanced, coordinated mapping could further reduce network traffic. Overall, all studied applications have shown a sizeable potential for energy saving; extensive non-communication periods require only a moderate number of link width adjustments.

Based on these insights, three different energy-saving policies are introduced and studied for their effects on energy consumption and performance. All three policies are implemented for evaluations in an existing event-based, cycle-accurate network simulator that is further extended with energy features. With this simulator, the impacts of various design parameters are studied. These policies are each based on two different power states, a fast state and a slow state, where the fast power state is implemented as a link operating at maximum bandwidth. The first policy (on/off) switches links on and off, depending on their current utilization. The second policy (high/low) uses a slow power state that provides only a low bandwidth for the benefits of reduced power consumption instead of switching links entirely off. This low power state enables the possibility to transfer small messages without link reconfiguration and ensures full connectivity throughout the entire network. The awake policy is based on high/low but shifts from a strict local approach to a hybrid local/system one, in which all nodes can configure far links in a network. However, the last policy does not provide further improvements over the first two and demands a significant additional expenditure in the hardware design. On/off and high/low provide overall promising results and enable significant energy savings of more than 90% in almost all configurations with only a moderate increase in execution time. The results of both policies

correspond to their respective design goals; on/off provides more energy savings at the costs of higher energy consumption, and vice versa when using high/low.

As expected, a shorter transition time is preferred for both energy saving and performance since power states can be changed more quickly and shorter idle periods can be exploited. As link reconfigurations in networks are not usually intended during runtime, link training protocols can take up to multiple milliseconds, but leave plenty of room for optimizations. However, there are also technical constraints that provide a lower bound for feasible transition times. Regarding topologies, the dragonfly provides the best results for energy saving. This is based on the smallest link/node ratio and the concept of few highly used global links and the majority of rarely used local links.

In addition to potential energy-saving policies, most interconnection networks are also equipped with additional management strategies. In such networks, congestion management is the most prominent example represented in nearly all networks. This work analyzes the interactions of these two network management strategies and shows that they complement each other, despite following contrary approaches. For all three evaluated topologies, the energy efficiency increases for the combination of congestion management and energy-saving policies. Congestion management improves performance metrics for the energy-saving policies compared to the baseline without congestion management, except of is a slightly increased latency in the k–ary n-tree.

This work contributes comprehensive studies of energy-saving capabilities in HPC interconnection networks. These studies include a detailed power analysis of network hardware, a study of energy-saving capabilities in HPC software, and introduce two new metrics to quantitatively compare locality in communication patterns. This work further introduces multiple policies that exploit these findings to reduce energy consumption. The policies are evaluated in an energy-aware simulator and studied for their interaction with congestion management. The results of this work provide valuable insights for hardware designers as well as system architects that need guidelines to design their systems in a more efficient way. However, not all issues have been conclusively addressed. Further workload analyses provide a deeper understanding of the concurrency of messages in the network and could help to predict messages. This would allow the network to adjust links before a message is injected in the network and avoid overhead due to transition time. Additional studies on central-managed policies could provide

further benefits if such a policy could rely on additional information as opposed to a local link.

Acknowledgements

First and foremost I wish to express my deepest gratitude to my parents, Barbara and Matthias Zahn, for always believing in me and encouraging me. They have constantly provided me their unconditional and endless support and I will always be grateful for being blessed with such amazing parents.

I am deeply indebted to my supervisor, Prof. Dr. Holger Fröning, Faculty of Mathematics and Computer Science at Heidelberg University, for his encouragement, guidance, invaluable feedback on my work, and countless discussions. His way of approaching research, and in particular of identifying relevant questions has had a profound impact on me.

I wish to express my sincere appreciation to Prof. Dr. Ulrich Brüning, the Computer Architecture Group, and the employees of the EXTOLL GmbH, who provided guidance and assistance in the course of this work. In particular, I want to thank Dr. Maximilian Thürmer and Dr. Sven Kapferer for their continued support. Additionally, I want to thank Steffen Lammel and Armin Schäffer, who also contributed with their master theses to this work. I also appreciate the fruitful discussions with my colleagues Lorenz Braun, Bernhard Klein, and Günther Schindler.

Parts of this work originates from a stay at the Universidad Castilla La-Mancha in Albacete. I am grateful to this institution and all members of the RAAP group for their hospitality. Especially, I want to express my thanks to Pedro Yébenes for providing his network simulator and endless support at the beginning of this work, and Francisco J. Andújar for his support in all trace-related issues. Furthermore, I would like to pay my special regards to Jesús Escudero-Sahuquillo and Pedro Javier García for their consistent support and guidance during this work.

I gratefully acknowledge financial support from the Carl-Zeiss Foundation in the form of a full Ph.D. scholarship as well as the travel grant from the HIPEAC

organization.

A very special thank you to Benjamin Klenk for his generous support and for always being supportive of me and my work. You are a wonderful friend and I wish you all the best. Furthermore, I wish to express my deepest gratitude to Alexander Matz for being there whenever I needed a friend and for his invaluable advice and feedback through the course of this work. I am also very grateful to Christine Harvey and Etienne Dilocker who both helped me in numerous ways during various stages of my Ph.D. All of them have accompanied me through this work and were always beside me during the happy and hard moments to push me and motivate me.

Some special words of gratitude go to my friends and my roommate who have always been a major source of support when things would get a bit discouraging, in particular Juli Xhixho, Sophie Bossert, and Steffen Sikora.

List of figures

List of tables

ASIC Application-Specific Integrated Circuit 37

CDF Cumulative Distribution Function 49

CDR Clock and Data Recovery 13

CML Current Mode Logic 4

CMOS Complementary Metal-Oxide-Semiconductor 33

CPU Central Processing Unit 1

CRC Cyclic Redundancy Check 11

DDT Derived Data Types 57

DoE US Department of Energy 3

DESTRO Deterministic Destination and Stage-based Routing 28

DFS Dynamic Frequency Scaling 90

DVFS Dynamic Voltage Frequency Scaling 90

DVS Dynamic Voltage Scaling 90

DLL Delay-Locked Loop 41

EEE Energy-Efficient Ethernet 5

FLOP Floating Point Operations 49

FIFO First In First Out 24

FLIT Flow Control Unit 11

FPGA Field-Programmable Gate Array 37

GPU Graphics Processing Unit 2

GTL Gate-Level Netlist 37

GUI Graphical User Interface 75

HCA Host Channel Adapter 75

HoL Head-of-Line Blocking 111

HPC High-Performance Computing 3

HTAX High Throughput Advanced X-Bar 37

IC Integrated Circuit 33

ILP Instruction-Level Parallelism 1

INI Initialization 74

ITRS International Technology Roadmap for Semiconductors 4

LAN Local Area Network 9

LUT Look Up Table 24

MIN Multistage Interconnection Networks 19

MPI Message Passing Interface 7

NED Network Description 74

NIC Network Interface Controller 10

NMOS N-Type Metal-Oxide-Semiconductor 33

NoC Network on Chip 9

OSI Open Systems Interconnection 11

OTF Open Trace Format 51

PCIe Peripheral Component Interconnect Express 2

PHIT Physcial Units 12

PLL Phase-Locked Loop 13

PMOS P-Type Metal-Oxide-Semiconductor 33

P2P Point-to-Point 30

RMA Remote Memory Access 32

RTL Register-Transfer Level 37

SAN System Area Networks 9

SoC System on Chip 9

SONAR Simple Offline Network AnalyzeR 48

TAU Tuning and Analysis Utilities 50

TDP Thermal Design Power 4

TLP Thread-Level Parallelism 1

TSMC Taiwan Semiconductor Manufacturing Company 32

UGAL Universal Globally Adaptive Load-balancing 29

UMA Uniform Memory Architecture 10

VC Virtual Channel 27

VOQ Virtual Output Queue 76

WAN Wide Area Network 10

References

[1] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct. 1974, ISSN: 1558-173X. DOI: 10.1109/JSSC.1974.1050511 (cit. on pp. 1, 34).

[2] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2011, pp. 365–376 (cit. on p. 1).

[3] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007, ISSN: 0018-9162. DOI: 10.1109/MC.2007.443. [Online]. Available: https://doi.org/10.1109/MC.2007.443 (cit. on pp. 4, 31).

[4] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 338–347, Jun. 2010, ISSN: 0163-5964. DOI: 10.1145/1816038.1816004. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816004 (cit. on pp. 4, 31, 104, 126, 136).

[5] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "Power/performance evaluation of energy efficient ethernet (eee) for high performance computing," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2013, pp. 205–214. DOI: 10.1109/ISPASS.2013.6557171 (cit. on pp. 4, 127).

[6] *The international technology roadmap for semiconductors 2.0 - executive report*, https://www.semiconductors.org/wp-content/uploads/2018/06/0_2015-ITRS-2.0-Executive-Report-1.pdf, Accessed: 2020-05-19, 2015 (cit. on p. 4).

[7] F. Zahn, P. Yebenes, S. Lammel, P. J. Garcia, and H. Fröning, "Analyzing the energy (dis-) proportionality of scalable interconnection networks," in *2016 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Mar. 2016, pp. 25–32. DOI: 10.1109/HIPINEB.2016.13 (cit. on p. 5).

[8] F. Zahn and H. Fröning, "On network locality in mpi-based hpc applications," in *49th International Conference on Parallel Processing - ICPP (ICPP '20)*, Edmonton, AB, Canada: ACM, New York, NY, USA, Aug. 2020, p. 11. DOI: 10.1145/3404397.3404436 (cit. on pp. 5, 51).

[9] K. P. Saravanan, P. M. Carpente, and A. Ramirez, "Exploring multiple sleep modes in on/off based energy efficient hpc networks," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, Oct. 2015, pp. 54–61. DOI: 10.1109/ICCD.2015.7357084 (cit. on pp. 5, 127).

[10] M. Alonso, S. Coll, J. M. Martinez, V. Santonja, P. Lopez, and J. Duato, "Power saving in regular interconnection networks," *Parallel Computing*, vol. 36, no. 12, pp. 696–712, 2010, ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2010.08.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819110001109 (cit. on pp. 5, 127).

[11] ——, "Dynamic power saving in fat-tree interconnection networks using on/off links," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, Apr. 2006. DOI: 10.1109/IPDPS.2006.1639599 (cit. on pp. 5, 103, 108, 128).

[12] F. Zahn, S. Lammel, and H. Fröning, "On link width scaling for energy-proportional direct interconnection networks," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 2, e4439, 2019, e4439 cpe.4439. DOI: 10.1002/cpe.4439. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4439. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4439 (cit. on pp. 5, 106, 136).

[13] F. Zahn, A. Schäffer, and H. Fröning, "Evaluating energy-saving strategies on torus, k-ary n-tree, and dragonfly," in *2018 IEEE 4th International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Feb. 2018, pp. 16–23. DOI: 10.1109/HiPINEB.2018.00011 (cit. on p. 5).

[14] F. Zahn, P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, and H. Fröning, "Effects of congestion management on energy saving techniques in interconnection networks," in *2019 International Workshop of High-Perfomance Interconnection Networks in the Exascale and Big-Data Era (HiPNEB)*, Feb. 2019, pp. 9–16. DOI: 10.1109/HiPINEB.2019.00009 (cit. on p. 6).

[15] T. M. Pinkston and J. Duato, "Appendix f: Interconnection networks," *Computer Architecture, Fifth Edition: A Quantitative Approach*, 2011 (cit. on pp. 9, 10, 13, 18, 19, 21, 25, 26).

[16] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks - an engineering approach*. IEEE, 1997, ISBN: 978-0-8186-7800-4 (cit. on pp. 9, 15–17, 19, 25, 27, 30, 112).

[17] *Infiniband architecture specification*, https://www.infinibandta.org/, Accessed: 2020-03-24, 2015 (cit. on pp. 9, 111).

[18] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, and R. C. Zak, "Intel® omni-path architecture: Enabling scalable, high performance fabrics," in *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, Aug. 2015, pp. 1–9. DOI: 10.1109/HOTI.2015.22 (cit. on p. 9).

[19] A. Silberschatz, G. Gagne, and P. B. Galvin, *Operating System Concepts*, 8th. Wiley Publishing, 2011, ISBN: 1118112733 (cit. on pp. 10, 11, 25).

[20] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, ISBN: 978-0-12-200751-4 (cit. on pp. 11, 14, 15, 17–19, 111, 112).

[21] N. E. Jerger, T. Krishna, and L.-S. Peh, *On-Chip Networks: Second Edition*, 2nd. Morgan & Claypool Publishers, 2017, ISBN: 1627059148 (cit. on p. 12).

[22] K. Hwang and Z. Xu, *Scalable Parallel Computing: Technology,Architecture,Programming*. USA: McGraw-Hill, Inc., 1998, ISBN: 0070317984 (cit. on pp. 12, 29).

[23] D. E. Culler, A. Gupta, and J. P. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*, 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ISBN: 978-1-55860-343-1 (cit. on pp. 12, 14, 17, 24, 27).

[24] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications," *IEEE Std 802.3-2008 (Revision of IEEE Std 802.3-2005)*, pp. 1–2977, 2008 (cit. on p. 12).

[25] K. S. Stevens, P. Golani, and P. A. Beerel, "Energy and performance models for synchronous and asynchronous communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 369–382, Mar. 2011, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2009.2037327 (cit. on p. 13).

[26] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 775–785, Jun. 1990, ISSN: 0018-9340. DOI: 10.1109/12.53599. [Online]. Available: https://doi.org/10.1109/12.53599 (cit. on p. 14).

[27] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high radix router," in *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005, pp. 420–431 (cit. on p. 14).

[28] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput.*, vol. 34, no. 10, pp. 892–901, Oct. 1985, ISSN: 0018-9340 (cit. on p. 21).

[29] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, Jun. 2008, pp. 77–88. DOI: 10.1109/ISCA.2008.19 (cit. on pp. 22, 24, 28, 29, 57, 103, 112).

[30] E. Hastings, D. Rincon-Cruz, M. Spehlmann, S. Meyers, A. Xu, D. P. Bunde, and V. J. Leung, "Comparing global link arrangements for dragonfly networks," in *2015 IEEE International Conference on Cluster Computing*, 2015, pp. 361–370 (cit. on p. 24).

[31] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, "Dragonfly+: Low cost topology for scaling datacenters," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Feb. 2017, pp. 1–8. DOI: 10.1109/HiPINEB.2017.11 (cit. on p. 24).

[32] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987, ISSN: 0018-9340. DOI: 10.1109/TC.1987.1676939. [Online]. Available: https://doi.org/10.1109/TC.1987.1676939 (cit. on pp. 27, 29).

[33] C. Gomez, F. Gilabert, M. E. Gomez, P. Lopez, and J. Duato, "Deterministic versus adaptive routing in fat-trees," in *2007 IEEE International Parallel and Distributed Processing Symposium*, Mar. 2007, pp. 1–8. DOI: 10.1109/IPDPS.2007.370482 (cit. on p. 28).

[34] L. G. Valiant, "A scheme for fast parallel communication," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 350–361, 1982. DOI: 10.1137/0211027. eprint: https://doi.org/10.1137/0211027. [Online]. Available: https://doi.org/10.1137/0211027 (cit. on p. 28).

[35] A. Singh, "Load-balanced routing in interconnection networks," PhD thesis, Stanford University, 2005 (cit. on p. 29).

[36] S. Kapferer, "Hodology and ecosystem for the design of a complex network asic," https://ub-madoc.bib.uni-mannheim.de/32961, PhD thesis, University of Mannheim, 2012 (cit. on p. 32).

[37] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, fifth. Oxford University Press, 2004 (cit. on p. 33).

[38] R. C. Jaeger and T. N. Blalock, *Microelectronic circuit design /*, 2nd ed. Dubuque Iowa : McGraw-Hill, 2003 (cit. on pp. 33, 35).

[39] J. Rogers, C. Plett, and F. Dai, *Integrated Circuit Design for High-Speed Frequency Synthesis (Artech House Microwave Library)*. USA: Artech House, Inc., 2006, ISBN: 1580539823 (cit. on pp. 33, 35, 36).

[40] K. J. Kuhn, "Cmos scaling beyond 32nm: Challenges and opportunities," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09, San Francisco, California: Association for Computing Machinery, 2009, pp. 310–313, ISBN: 9781605584973. DOI: 10.1145/1629911.1629996. [Online]. Available: https://doi.org/10.1145/1629911.1629996 (cit. on p. 34).

[41] H. Litz, H. Fröning, and U. Brüning, "Htax: A novel framework for flexible and high performance networks-on-chip," Jan. 2010 (cit. on p. 37).

[42] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st. Springer Publishing Company, Incorporated, 2011, ISBN: 9789048195909 (cit. on p. 39).

[43] S. Rumley, D. Nikolova, R. Hendry, Q. Li, D. Calhoun, and K. Bergman, "Silicon photonics for exascale systems," *Journal of Lightwave Technology*, vol. 33, no. 3, pp. 547–562, Feb. 2015, ISSN: 0733-8724. DOI: 10.1109/JLT.2014.2363947 (cit. on p. 44).

[44] Q. Xu, B. Schmidt, S. Pradhan, and M. Lipson, "Micrometre-scale silicon electro-optic modulator," *Nature*, vol. 435, no. 7040, pp. 325–327, 2005. DOI: 10.1038/nature03569. [Online]. Available: https://doi.org/10.1038/nature03569 (cit. on p. 44).

[45] J. Ding, H. Chen, L. Yang, L. Zhang, R. Ji, Y. Tian, W. Zhu, Y. Lu, P. Zhou, R. Min, and M. Yu, "Ultra-low-power carrier-depletion mach-zehnder silicon optical modulator," *Opt. Express*, vol. 20, no. 7, pp. 7081–7087, Mar. 2012. DOI: 10.1364/OE.20.007081. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-20-7-7081 (cit. on p. 44).

[46] W. Bogaerts, P. De Heyn, T. Van Vaerenbergh, K. De Vos, S. Kumar Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. Van Thourhout, and R. Baets, "Silicon microring resonators," *Laser & Photonics Reviews*, vol. 6, no. 1, pp. 47–73, 2012. DOI: 10.1002/lpor.201100017. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/lpor.201100017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.201100017 (cit. on p. 44).

[47] G. Li, A. V. Krishnamoorthy, I. Shubin, J. Yao, Y. Luo, H. Thacker, X. Zheng, K. Raj, and J. E. Cunningham, "Ring resonator modulators in silicon for interchip photonic links," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 6, pp. 95–113, Nov. 2013, ISSN: 1077-260X. DOI: 10.1109/JSTQE.2013.2278885 (cit. on p. 44).

[48] S. T. S. Cheung, B. Guan, S. S. Djordjevic, K. Okamoto, and S. J. B. Yoo, "Low-loss and high contrast silicon-on-insulator (soi) arrayed waveguide grating," in *Conference on Lasers and Electro-Optics 2012*, Optical Society of America, 2012, CM4A.5. DOI: 10.1364/CLEO_SI.2012.CM4A.5. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=CLEO_SI-2012-CM4A.5 (cit. on p. 44).

[49] A. V. Krishnamoorthy, X. Zheng, D. Feng, J. Lexau, J. F. Buckwalter, H. D. Thacker, F. Liu, Y. Luo, E. Chang, P. Amberg, I. Shubin, S. S. Djordjevic, J. H. Lee, S. Lin, H. Liang, A. Abed, R. Shafiiha, K. Raj, R. Ho, M. Asghari, and J. E. Cunningham, "A low-power, high-speed, 9-channel germanium-silicon electro-absorption modulator array integrated with digital cmos driver and wavelength multiplexer," *Opt. Express*, vol. 22, no. 10, pp. 12 289–12 295, May 2014. DOI: 10.1364/OE.22.012289. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-22-10-12289 (cit. on p. 44).

[50] M. Bahadori, S. Rumley, D. Nikolova, and K. Bergman, "Comprehensive design space exploration of silicon photonic interconnects," *Journal of Lightwave Technology*, vol. 34, no. 12, pp. 2975–2987, Jun. 2016, ISSN: 0733-8724. DOI: 10.1109/JLT.2015.2503120 (cit. on p. 44).

[51] K. Xi, Y.-H. Kao, and H. J. Chao, "A petabit bufferless optical switch for data center networks," in *Optical Interconnects for Future Data Center Networks*, C. Kachris, K. Bergman, and I. Tomkos, Eds. New York, NY: Springer New York, 2013, pp. 135–154, ISBN: 978-1-4614-4630-9. DOI: 10.1007/978-1-4614-4630-9_8. [Online]. Available: https://doi.org/10.1007/978-1-4614-4630-9_8 (cit. on p. 45).

[52] S. Di Lucente, N. Calabretta, J. A. C. Resing, and H. J. S. Dorren, "Scaling low-latency optical packet switches to a thousand ports," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 4, no. 9, A17–A28, 2012 (cit. on p. 45).

[53] J. Perelló, S. Spadaro, S. Ricciardi, D. Careglio, S. Peng, R. Nejabati, G. Zervas, D. Simeonidou, A. Predieri, M. Biancani, H. J. S. Dorren, S. D. Lucente, J. Luo, N. Calabretta, G. Bernini, N. Ciulli, J. C. Sancho, S. Iordache, M. Farreras, Y. Becerra, C. Liou, I. Hussain, Y. Yin, L. Liu, and R. Proietti, "All-optical packet/circuit switching-based data center network for enhanced scalability, latency, and throughput," *IEEE Network*, vol. 27, no. 6, pp. 14–22, 2013 (cit. on p. 45).

[54] S. Lammel, F. Zahn, and H. Fröning, "Sonar: Automated communication characterization for hpc applications," in *High Performance Computing*, M. Taufer, B. Mohr, and J. M. Kunkel, Eds., Cham: Springer International Publishing, 2016, pp. 98–114, ISBN: 978-3-319-46079-6 (cit. on pp. 48, 51).

[55] S. Sreepathi, E. D'Azevedo, B. Philip, and P. Worley, "Communication characterization and optimization of applications using topology-aware task mapping on large supercomputers," in *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '16, Delft, The Netherlands: Association for Computing Machinery, 2016, pp. 225–236, ISBN: 9781450340809. DOI: 10.1145/2851553.2851575. [Online]. Available: https://doi.org/10.1145/2851553.2851575 (cit. on p. 48).

[56] I. Lee, "Characterizing communication patterns of nas-mpi benchmark programs," in *IEEE Southeastcon 2009*, Mar. 2009, pp. 158–163. DOI: 10.1109/SECON.2009.5174068 (cit. on p. 48).

[57] R. Riesen, "Communication patterns [message-passing patterns]," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, Apr. 2006. DOI: 10.1109/IPDPS.2006.1639567 (cit. on p. 48).

[58] S. Chodnekar, V. Srinivasan, A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, "Towards a communication characterization methodology for parallel applications," in *Proceedings Third International Symposium on High-Performance Computer Architecture*, Feb. 1997, pp. 310–319. DOI: 10.1109/HPCA.1997.569693 (cit. on p. 48).

[59] J. Kim and D. J. Lilja, "Characterization of communication patterns in message-passing parallel scientific application programs," in *Network-Based Parallel Computing Communication, Architecture, and Applications*, D. K. Panda and C. B. Stunkel, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 202–216, ISBN: 978-3-540-69693-3 (cit. on p. 48).

[60] S. Rumley, R. P. Polster, S. D. Hammond, A. F. Rodrigues, and K. Bergman, "End-to-end modeling and optimization of power consumption in hpc interconnects," in *2016 45th International Conference on Parallel Processing Workshops (ICPPW)*, Aug. 2016, pp. 133–140. DOI: 10.1109/ICPPW.2016.33 (cit. on p. 49).

[61] R. Murphy, "On the effects of memory latency and bandwidth on supercomputer application performance," in *2007 IEEE 10th International Symposium on Workload Characterization*, Sep. 2007, pp. 35–43. DOI: 10.1109/IISWC.2007.4362179 (cit. on p. 52).

[62] R. Murphy, A. Rodrigues, P. Kogge, and K. Underwood, "The implications of working set analysis on supercomputing memory hierarchy design," in *Proceedings of the 19th Annual International Conference on Supercomputing*, ser. ICS '05, Cambridge, Massachusetts: Association for Computing Machinery, 2005, pp. 332–340, ISBN: 1595931678. DOI: 10.1145/1088149.1088193. [Online]. Available: https://doi.org/10.1145/1088149.1088193 (cit. on p. 52).

[63] R. C. Murphy and P. M. Kogge, "On the memory access patterns of supercomputer applications: Benchmark selection and its implications," *IEEE Trans. Comput.*, vol. 56, no. 7, pp. 937–945, Jul. 2007, ISSN: 0018-9340. DOI: 10.1109/TC.2007.1039. [Online]. Available: https://doi.org/10.1109/TC.2007.1039 (cit. on p. 52).

[64] K. Z. Ibrahim, S. Hofmeyr, and C. Iancu, "Characterizing the performance of parallel applications on multi-socket virtual machines," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2011, pp. 1–12. DOI: 10.1109/CCGrid.2011.50 (cit. on p. 52).

[65] B. Klenk and H. Fröning, "An overview of mpi characteristics of exascale proxy applications," in *High Performance Computing*, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds., Cham: Springer International Publishing, 2017, pp. 217–236, ISBN: 978-3-319-58667-0 (cit. on p. 60).

[66] P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, and F. J. Quiles, "Towards modeling interconnection networks of exascale systems with omnet++," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb. 2013, pp. 203–207. DOI: 10.1109/PDP.2013.36 (cit. on pp. 73, 74).

[67] F. J. Andújar, J. A. Villar, J. L. Sánchez, F. J. Alfaro, and J. Escudero-Sahuquillo, "VEF Traces: A Framework for Modelling MPI Traffic in Interconnection Network Simulators," in *Proceedings of 2015 IEEE International Conference on Cluster Computing*, Sep. 2015, pp. 841–848 (cit. on pp. 78, 81, 83).

[68] P. Yébenes, "New queuing schemes to improve the efficiency of hybrid and hierarchical high-performance interconnection network topologies," https://ruidera.uclm.es/xmlui/handle/10578/19674, PhD thesis, Universidad de Castilla-La Mancha, 2018 (cit. on p. 78).

[69] D. Dechev and G. Hendry, "A macroscale simulator for exascale software/hardware co-design," 2013 (cit. on p. 82).

[70] F. J. Andújar, J. A. Villar, J. L. Sánchez, F. J. Alfaro, and J. Escudero-Sahuquillo, "An open-source family of tools to reproduce MPI-based workloads in interconnection network simulators," *The Journal of Supercomputing*, vol. 72, no. 12, pp. 4601–4628, Dec. 2016, ISSN: 1573-0484 (cit. on pp. 83, 85).

[71] Z. Lai, K. T. Lam, C.-L. Wang, and J. Su, "Latency-aware dvfs for efficient power state transitions on many-core architectures," *The Journal of Supercomputing*, vol. 71, no. 7, pp. 2720–2747, Jul. 2015, ISSN: 1573-0484. DOI: 10.1007/s11227-015-1415-y. [Online]. Available: https://doi.org/10.1007/s11227-015-1415-y (cit. on p. 90).

[72] A. McLaughlin, I. Paul, J. L. Greathouse, S. Manne, and S. Yalamanchili, "A power characterization and management of gpu graph traversal," 2014 (cit. on p. 90).

[73] F. Zahn, S. Lammel, and H. Fröning, "Early experiences with saving energy in direct interconnection networks," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Feb. 2017, pp. 33–40. DOI: 10.1109/HiPINEB.2017.10 (cit. on pp. 92, 133).

[74] A. Venkatesh, A. Vishnu, K. Hamidouche, N. Tallent, D. Panda, D. Kerbyson, and A. Hoisie, "A case for application-oblivious energy-efficient mpi runtime," in *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2015, pp. 1–12. DOI: 10.1145/2807591.2807658 (cit. on pp. 93, 95, 126).

[75] I. Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen, Z. DeVito, R. Haque, D. Laney, E. Luke, F. Wang, D. Richards, M. Schulz, and C. Still, "Exploring traditional and emerging parallel programming models using a proxy application," in *27th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2013)*, Boston, USA, May 2013 (cit. on p. 100).

[76] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten, "Scalable molecular dynamics with namd," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005. DOI: 10.1002/jcc.20289. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20289. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20289 (cit. on p. 100).

[77] J. J. Dongarra, "Performance of various computers using standard linear equations software," *SIGARCH Comput. Archit. News*, vol. 20, no. 3, pp. 22–44, Jun. 1992, ISSN: 0163-5964. DOI: 10.1145/141868.141871. [Online]. Available: https://doi.org/10.1145/141868.141871 (cit. on p. 101).

[78] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "Performance impact of process mapping on small-scale smp clusters - a case study using high performance linpack," in *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, ser. IPDPS '02, USA: IEEE Computer Society, 2002, p. 263, ISBN: 0769515738 (cit. on p. 101).

[79] M. Alonso, S. Coll, V. Santonja, J.-M. Martínez, P. López, and J. Duato, "Power-aware fat-tree networks using on/off links," in *High Performance Computing and Communications*, R. Perrott, B. M. Chapman, J. Subhlok, R. F. de Mello, and L. T. Yang, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 472–483, ISBN: 978-3-540-75444-2 (cit. on pp. 103, 108, 128).

[80] P. Reviriego, J. A. Hernandez, D. Larrabeiti, and J. A. Maestro, "Performance evaluation of energy efficient ethernet," *IEEE Communications Letters*, vol. 13, no. 9, pp. 697–699, Sep. 2009, ISSN: 1558-2558. DOI: 10.1109/LCOMM.2009.090880 (cit. on p. 104).

[81] B. Dickov, M. Pericas, P. Carpenter, N. Navarro, and E. Ayguade, "Software-managed power reduction in infiniband links," in *2014 43rd International Conference on Parallel Processing*, Sep. 2014, pp. 311–320. DOI: 10.1109/ICPP.2014.40 (cit. on pp. 104, 126).

[82] G. Kim, H. Choi, and J. Kim, "Tcep: Traffic consolidation for energy-proportional high-radix networks," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2018, pp. 712–725. DOI: 10.1109/ISCA.2018.00065 (cit. on pp. 104, 127, 137).

[83] P. J. Garcia, "Congestion management," in *Encyclopedia of Parallel Computing*, 2011, pp. 378–386. DOI: 10.1007/978-0-387-09766-4_313 (cit. on p. 111).

[84] E. G. Gran, M. Eimot, S. Reinemo, T. Skeie, O. Lysne, L. P. Huse, and G. Shainer, "First experiences with congestion control in infiniband hardware," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, Apr. 2010, pp. 1–12. DOI: 10.1109/IPDPS.2010.5470419 (cit. on p. 111).

[85] J. Escudero-Sahuquillo, E. G. Gran, P. J. Garcia, J. Flich, T. Skeie, O. Lysne, F. J. Quiles, and J. Duato, "Efficient and cost-effective hybrid congestion control for hpc interconnection networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 107–119, Jan. 2015, ISSN: 1558-2183. DOI: 10.1109/TPDS.2014.2307851 (cit. on p. 112).

[86] M. Katevenis, D. N. Serpanos, and G. Dimitriadis, "ATLAS I: a single-chip, gigabit ATM switch with HIC/HS links arid multi-lane back-pressure," *Microprocess. Microsystems*, vol. 21, no. 7-8, pp. 481–490, 1998. DOI: 10.1016/S0141-9331(98)00041-6. [Online]. Available: https://doi.org/10.1016/S0141-9331(98)00041-6 (cit. on p. 112).

[87] P. J. Garcia, F. J. Quiles, J. Flich, J. Duato, I. Johnson, and F. Naven, "Efficient, scalable congestion management for interconnection networks," *IEEE Micro*, vol. 26, no. 5, pp. 52–66, Sep. 2006, ISSN: 1937-4143. DOI: 10.1109/MM.2006.88 (cit. on p. 112).

[88] W. Dally, P. Carvey, and L. Dennison, "The avici terabit switch/router architecture.," in *Proceedings of Hot Interconnects Symposium VI*, 1998, pp. 41–50 (cit. on p. 112).

[89] M. E. Gómez, J. Flich, A. Robles, P. López, and J. Duato, "VOQSW: A methodology to reduce HOL blocking in infiniband networks," in *17th International Parallel and Distributed Processing Symposium (IPDPS, 2003), 22-26 April 2003, Nice, France*, IEEE Computer Society, 2003, p. 46. DOI: 10.1109/IPDPS.2003.1213134. [Online]. Available: https://doi.org/10.1109/IPDPS.2003.1213134 (cit. on pp. 112, 114).

[90] T. Nachiondo, J. Flich, and J. Duato, "Buffer management strategies to reduce hol blocking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 6, pp. 739–753, Jun. 2010, ISSN: 1558-2183. DOI: 10.1109/TPDS.2009.63 (cit. on pp. 112, 114).

[91] J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, J. Flich, and J. Duato, "OBQA: smart and cost-efficient queue scheme for head-of-line blocking elimination in fat-trees," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1460–1472, 2011. DOI: 10.1016/j.jpdc.2011.07.007. [Online]. Available: https://doi.org/10.1016/j.jpdc.2011.07.007 (cit. on p. 112).

[92] W. L. Guay, B. Bogdanski, S. Reinemo, O. Lysne, and T. Skeie, "Vftree - a fat-tree routing algorithm using virtual lanes to alleviate congestion," in *2011 IEEE International Parallel Distributed Processing Symposium*, May 2011, pp. 197–208. DOI: 10.1109/IPDPS.2011.28 (cit. on p. 112).

[93] C. G. Requene, "Low-memory techniques for routing and fault-tolerance on the fat-tree topology," https://riunet.upv.es/bitstream/handle/10251/8856/tesisUPV3368.pdf, PhD thesis, Universidad Politécnica de Valencia, 2010 (cit. on p. 112).

[94] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized infinibandtm fat-tree routing for shift all-to-all communication patterns," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 2, pp. 217–231, Feb. 2010, ISSN: 1532-0626 (cit. on p. 112).

[95] R. Penaranda, C. Gomez, M. E. Gomez, P. Lopez, and J. Duato, "Iodet: A hol-blocking-aware deterministic routing algorithm for direct topologies," in *Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems*, ser. ICPADS '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 702–703, ISBN: 978-0-7695-4903-3. DOI: 10.1109/ICPADS.2012.103. [Online]. Available: http://dx.doi.org/10.1109/ICPADS.2012.103 (cit. on pp. 112, 114).

[96] P. Yebenes Segura, J. Escudero-Sahuquillo, C. Gomez Requena, P. J. Garcia, F. J. Quiles, and J. Duato, "Bbq: A straightforward queuing scheme to reduce hol-blocking in high-performance hybrid networks," in *Euro-Par 2013 Parallel Processing*, F. Wolf, B. Mohr, and D. an Mey, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 699–712, ISBN: 978-3-642-40047-6 (cit. on p. 112).

[97] R. Penaranda, C. Gomez, M. E. Gomez, P. Lopez, and J. Duato, "The k-ary n-direct s-indirect family of topologies for large-scale interconnection networks," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 1035–1062, Mar. 2016, ISSN: 1573-0484. DOI: 10.1007/s11227-016-1640-z. [Online]. Available: https://doi.org/10.1007/s11227-016-1640-z (cit. on p. 112).

[98] P. Yébenes, J. Escudero-Sahuquillo, P. J. Garcśa, and F. J. Quiles, "Efficient queuing schemes for hol-blocking reduction in dragonfly topologies with minimal-path routing," in *2015 IEEE International Conference on Cluster Computing*, Sep. 2015, pp. 817–824. DOI: 10.1109/CLUSTER.2015.138 (cit. on pp. 112, 115).

[99] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, and T. Hoefler, "An effective queuing scheme to provide slim fly topologies with hol blocking reduction and deadlock freedom for minimal-path routing," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, Feb. 2017, pp. 25–32. DOI: 10.1109/HiPINEB.2017.9 (cit. on p. 112).

[100] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, New Orleans, Louisana: IEEE Press, 2014, pp. 348–359, ISBN: 9781479955008. DOI: 10.1109/SC.2014.34. [Online]. Available: https://doi.org/10.1109/SC.2014.34 (cit. on p. 112).

[101] J. Escudero-Sahuquillo, P. J. Garcia, F. J. Quiles, S.-A. Reinemo, T. Skeie, O. Lysne, and J. Duato, "A new proposal to deal with congestion in infiniband-based fat-trees," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1802–1819, Jan. 2014, ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2013.09.002. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2013.09.002 (cit. on p. 114).

[102] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2013, pp. 508–519. DOI: 10.1109/HPCA.2013.6522345 (cit. on p. 125).

[103] R. Parikh, R. Das, and V. Bertacco, "Power-aware nocs through routing and topology reconfiguration," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2014, pp. 1–6 (cit. on p. 125).

[104] L. Chen and T. M. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec. 2012, pp. 270–281. DOI: 10.1109/MICRO.2012.33 (cit. on p. 125).

[105] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2015, pp. 378–389. DOI: 10.1109/HPCA.2015.7056048 (cit. on p. 125).

[106] V. Soteriou and L. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 393–408, Mar. 2007, ISSN: 1558-2183. DOI: 10.1109/TPDS.2007.43 (cit. on p. 125).

[107] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, ser. NETWORKING '09, Aachen, Germany: Springer-Verlag, 2009, pp. 795–808, ISBN: 9783642013980. DOI: 10.1007/978-3-642-01399-7_62. [Online]. Available: https://doi.org/10.1007/978-3-642-01399-7_62 (cit. on p. 126).

[108] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35). Proceedings.*, Nov. 2002, pp. 294–305. DOI: 10.1109/MICRO.2002.1176258 (cit. on p. 126).

[109] H.-S. Wang, L.-S. Peh, and S. Malik, "A power model for routers: Modeling alpha 21364 and infiniband routers," in *Proceedings 10th Symposium on High Performance Interconnects*, Aug. 2002, pp. 21–27. DOI: 10.1109/CONECT.2002.1039253 (cit. on p. 126).

[110] T. Groves, R. E. Grant, S. Hemmer, S. Hammond, M. Levenhagen, and D. C. Arnold, "(sai) stalled, active and idle: Characterizing power and performance of large-scale dragonfly networks," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2016, pp. 50–59. DOI: 10.1109/CLUSTER.2016.52 (cit. on p. 126).

[111] B. Dickov, M. Pericas, P. M. Carpenter, N. Navarro, and E. Ayguade, "Analyzing performance improvements and energy savings in infiniband architecture using network compression," in *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, Oct. 2014, pp. 73–80. DOI: 10.1109/SBAC-PAD.2014.27 (cit. on p. 126).

[112] G. Hendry, "Decreasing network power with on-off links informed by scientific applications," in *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, 2013, pp. 868–875 (cit. on p. 126).

[113] F. J. Andújar, S. Coll, M. Alonso, P. López, and J.-M. Martínez, "Powar: Power-aware routing in hpc networks with on/off links," *ACM Transactions on Architecture and Code Optimization*, vol. 15, no. 4, Jan. 2019, ISSN: 1544-3566. DOI: 10.1145/3293445. [Online]. Available: https://doi.org/10.1145/3293445 (cit. on pp. 128, 137).