PowerHerd: Dynamic Satisfaction of Peak Power Constraints in Interconnection Networks

Li Shang Li-Shiuan Peh Niraj K. Jha

Department of Electrical Engineering
Princeton University, Princeton, NJ 08544
{*lshang,peh,jha*}*@ee.princeton.edu*

ABSTRACT

Power consumption is a critical issue in interconnection network design, driven by power-related design constraints, such as thermal and power delivery design. Usually, off-line worst-case power analysis is used in network design to guarantee safe on-line operation, which not only increases system cost but also constrains network performance. In this work, we present an on-line mechanism, called PowerHerd, which can dynamically regulate network power consumption, and guarantee that network peak power constraints are not exceeded. PowerHerd is a distributed approach – within the interconnection network, each router dynamically maintains a local power budget, controls its local power dissipation, and exchanges spare power resources with its neighboring routers to optimize network performance.

Experiments demonstrate that PowerHerd can effectively regulate network power consumption meeting peak power constraints with negligible network performance penalty. Armed with PowerHerd, network designers can focus on system performance and power optimization for the average case rather than the worst case, thus making it possible to employ a more powerful interconnection network in the system.

Categories and Subject Descriptors

C.1.4 [**Parallel Architectures**]: [Distributed Architectures]; C.2.1 [**Network Architecture and Design**]: [Packet-Switching Networks]

General Terms

Design, Management, Performance

Keywords

low-power, thermal management, interconnection networks

1. INTRODUCTION

Interconnection networks have proliferated to a wide range of high-performance parallel and distributed systems, from clusters of servers to terabit routers, from server blades to chip-multiprocessors. In many of these systems, interconnection networks consume a substantial fraction of total system power – the power consumed by the interconnection network circuitry in a terabit IP router is about a third of the power dissipated by the entire line card; the integrated router and links of the Alpha 21364 microprocessor consume about 20% of total chip power [8].

Power is a key design constraint in many of the above systems. More specifically, designers are faced with tight *peak* power constraints, as cooling mechanisms and power supply/delivery circuits have to handle worst-case system power. Cooling costs can be prohibitive (consider the high power budgets of industry-standard six-foot server racks (5,000W) [11]). Besides, over 50% of electronic failures are temperature-related [18], given that circuit reliability is exponentially dependent on the operating temperature. Power supply and delivery design have become increasingly challenging and costly as well.

Designers of interconnection networks are typically given a tight peak power budget that they have to adhere to at design-time. Since the power consumption of interconnection networks varies greatly with input traffic, designers usually have to assume a worst-case power profile of the network based on maximum switching activity of each hardware component, even though peak activity may rarely occur. This distribution of system-wide peak power budget to each node in the network based on worst-case hardware power dissipation degrades system performance – a higher-performance network architecture will have to be traded off for a simpler lower-performance alternative if the former's worst-case power consumption exceeds the peak power budget, even if average-case power consumption is significantly below the peak power constraint. As estimated in [1], additional power dissipation above 35-40 W increases total chip cost by more than $\$1/W$. As peak power becomes the primary design constraint, we need to investigate ways to efficiently manage and utilize the limited power budget to optimize network performance.

In this paper, we propose PowerHerd, a distributed run-time mechanism that dynamically monitors and ensures that the power dissipation of interconnection networks meets peak power constraints. Based on a user-supplied peak power constraint for the entire network at design time, PowerHerd parcels this global power budget into local power budgets for each node at run-time. These local power budgets are shared dynamically among adjacent routers in response to actual network traffic, effectively shuffling the power budget around to meet traffic hotspots, delivering high network performance. Within each router, an on-line power estimator uses flit arrival counters to estimate local power consumption. This information is fed into a predictor that estimates future power demand and determines the amount of budget sharing. Simple power regu-

ICS'03, June 23-26, 2003, San Francisco, California, USA

lation mechanisms in each router throttle the power consumption so the local peak power budget is met. To further balance network workload, and alleviate transient bursts of traffic, routing algorithms are tailored to be power-aware, adaptively avoiding paths with low power budgets.

Our simulations show PowerHerd consistently maintaining interconnection network power below the user-supplied peak power constraint, with negligible network performance penalty (less than 1% increase in maximum latency with no throughput degradation). With PowerHerd dynamically guaranteeing peak power constraints for the entire network, network designers can concentrate on optimizing architectures for average-case system performance.

The rest of this paper is organized as follows. In Section 2, we define the problem. In Section 3, we give details of PowerHerd, from its power estimation, to prediction, sharing, throttling and routing. In Section 4, we evaluate the performance of PowerHerd. Section 5 presents prior related work and discusses several design issues. Section 6 concludes the paper.

2. PROBLEM DEFINITION

The peak power consumption of interconnection networks is a critical design constraint, and directly affects their thermal and power delivery design. In most systems, designers are faced with a tight peak global power budget for the entire network. For instance, in server blades, processors are connected with an interconnection network fabric within a PCB, and assigned a fixed power budget driven by cooling and power delivery design constraints. A fraction of this budget is allocated to the clock tree, the processors, and a certain fraction for the inter-processor network. We label this global power budget for the entire network P_{GPB}. In choosing network architectures, designers cannot use network topologies and architectures whose peak power consumption is higher than P_{GPB}. Similarly, in systems such as chip-multiprocessors, the total network power consumption contributed by all routers and links of the on-chip network directly affect thermal solutions, such as packaging, heat sinks, and fans. Hence, a tight P_{GPB} is needed.

The problem tackled by PowerHerd is defined as follows:

Global power budget: Given an interconnection network I with R routers, a global power budget P_{GPB} is a hard peak power constraint for the entire network, such that

$$\sum_{i=1}^{R} P_i = \sum_{i=1}^{R} \frac{E_i}{T_{GPB}} = P_I \leq P_{GPB} \quad (1)$$

where P_i is the power consumption of router i over a pre-defined time period T_{GPB}, E_i is the energy consumption of router i, and P_I is the average power consumption of network I over T_{GPB}. The global power budget P_{GPB} and time period T_{GPB} are both determined by the requirements of the system, and are based on thermal considerations. T_{GPB} is assumed to be equal to the thermal time constant, which varies from 100 microseconds (internal hotspot within on-chip networks) to several seconds (chip-to-chip networks) [13].

With PowerHerd, we propose a way of meeting P_{GPB} constraints by dynamically monitoring and regulating E_i to ensure that the power consumption of the entire network, P_I, over T_{GPB}, does not exceed P_{GPB} under any network workload. Since power consumption affects system temperature incrementally (*i.e.*, transient power variation does not instantly change the temperature), by setting T_{GPB} equal to the thermal time constant, and guaranteeing that the average peak power consumption over each T_{GPB} is always below the thermally-related peak power constraint, we are able to safely satisfy thermal constraints.

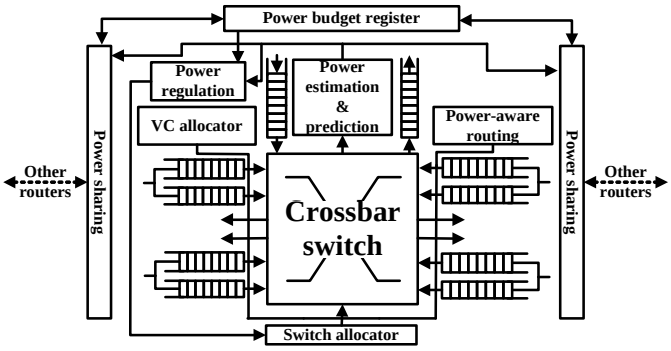


Figure 1: A PowerHerd router.

As mentioned above, PowerHerd targets a fairly long T_{GPB} based on thermal time constants. As described in [6, 7], transient power, also called the di/dt problem, is a critical issue in power supply and delivery design as well, and involves a much shorter T_{GPB}, on the order of hundreds of cycles. We discuss ways to extend PowerHerd to address a short T_{GPB} in Section 5. In addition, while our focus throughout the paper is on satisfying a single global peak power constraint, P_{GPB}, for the entire network, there are systems in which the local peak power consumption at each node needs to be capped as well, e.g., in on-chip networks with local thermal hotspots. In Section 4.3, we present an extension of PowerHerd to ensure that both local peak power constraints at each node as well as global constraints for the whole network are met.

3. DESCRIPTION OF POWERHERD

PowerHerd consists of power estimation, prediction, sharing, regulation, and routing mechanisms built into each router hardware, as shown in Figure 1. First, the user-defined global power budget, P_{GPB}, is divided evenly and stored in the power budget register at each router. This register forms the local power constraint, P^i_{LPB}, for each router i. Each router *estimates* its own power consumption at run-time, based on actual activity. This estimate is then used to *predict* future activity and future power consumption. Based on the estimate and predicted demand, each router decides if it has spare power to *share* with neighboring routers, updating the power budget register accordingly to reflect the current power budget allocated to the router. At each cycle, power regulation mechanisms dynamically *throttle* the switch allocator, adjusting the router power consumption to keep it in line with its allocated budget. Finally, the *routing* protocol at each node adjusts its routing decisions in response to the current power budgets of neighboring routers, steering network traffic towards routers with an excess power budget.

3.1 Dynamic Power Estimation

In an interconnection network, power is dissipated when flits[1] traverse routers and links. As shown in Figure 2(a), flits flow through an interconnection network invoking a stream of operations – in a typical virtual-channel router, when a flit arrives at a router, it (1) is written into the input buffer, undergoes (2) routing and (3) virtual-channel allocation (head flits only), (4) competes for the crossbar switch, (5) gets read from the input buffer, then traverses (6) the switch, and (7) the outgoing link, and finally arrives at the downstream router where the entire flow repeats. Each operation dissi-

[1]A flit is a flow control unit, a fixed-size segment of a packet.

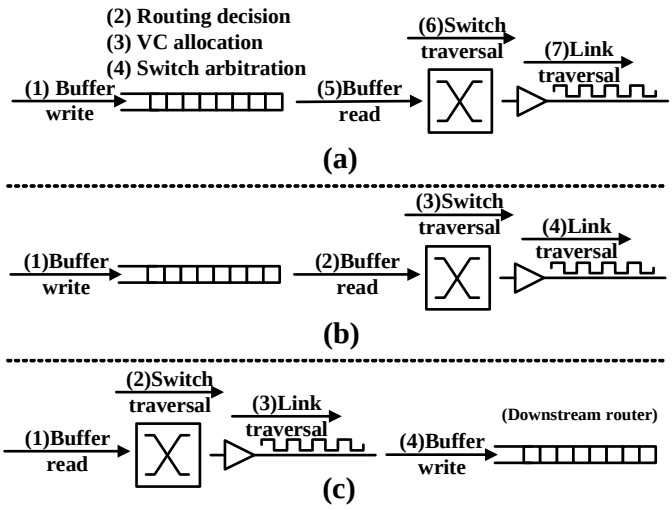


Figure 2: Simplification of router operations for on-line power estimation.

pates dynamic power.[2]

The power consumed by each operation varies significantly, as illustrated in [15] for an on-chip 4-by-4 torus interconnection network with $3mm$ 256-bit links between routers, each router clocked at 2GHz, with three virtual channels, and 64 flit buffers per port, in the $0.10\mu m$ CMOS technology with $V_{dd} = 1.2V$. As shown in Figure 3, input buffers, crossbar switch and link circuitry dominate network power consumption. The power consumption of arbitration and allocation logic is negligible. These components were similarly found to be dominant power consumers in chip-to-chip networks. We thus derive a simplified router operation sequence for power estimation: (1) input buffer write, (2) input buffer read, (3) switch traversal, and (4) link traversal, as shown in Figure 2(b). Then, we right-shift this simplified router operation sequence, obtaining a new sequence: (1) input buffer read, (2) switch traversal, (3) link traversal, and (4) input buffer write (at downstream router), as shown in Figure 2(c). Compared with operation sequence (b), sequence (c) allows more efficient on-line power estimation and regulation, which is explained in detail next.

In routers, power consumption is highly correlated with the switching activity of the arriving flits at the boundaries of each functional block, as demonstrated in [15]. Flit data do not get processed and altered within blocks, they are simply relayed[3]. Furthermore, these functional blocks are highly symmetric. Therefore, the inter-bit difference in capacitive load is small. Hence, capturing aggregated switching activities at the edge of each functional block is sufficient for on-line power estimation. We thus propose a regression-based approach for modeling the average power consumption of each operation in Figure 2(c), as discussed next.

Input buffers: In a router, buffer read and write are the two operations that trigger dynamic power dissipation in input buffers. The average power consumption for read, P_{buff_read}, and write, P_{buff_write}, operations over time T can be estimated based on four variables: number of read operations N_{buff_read}, number of write operations N_{buff_write}, and aggregated switching activities of each

[2]We focus on dynamic power dissipation, though PowerHerd can be readily extended to include leakage power.

[3]There are updates to header information, such as virtual channel ID, but these changes to a few bits are minute as compared to the entire data packet.

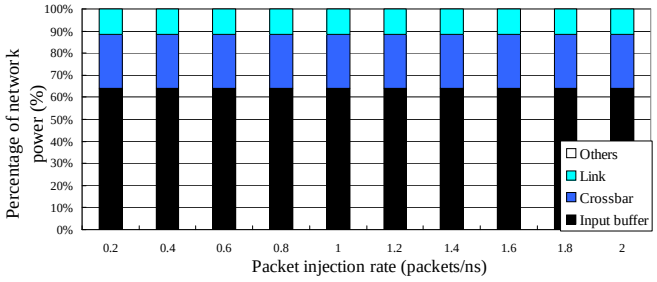


Figure 3: Power consumption distribution within an on-chip interconnection network.

read (write) operation, $S^i_{buff\_read}$ ($S^i_{buff\_write}$), as follows (in the following equations, C's denote regression coefficients that are obtained off-line using linear regression techniques [9]):

$$\begin{aligned} P_{buff\_read} &= (C^1_{buff\_read} \times \sum_{i=1}^{N_{buff\_read}} S^i_{buff\_read} \\ &+ C^2_{buff\_read} \times N_{buff\_read} + C^3_{buff\_read})/T \end{aligned} \quad (2)$$

$$\begin{aligned} P_{buff\_write} &= (C^1_{buff\_write} \times \sum_{i=1}^{N_{buff\_write}} S^i_{buff\_write} \\ &+ C^2_{buff\_write} \times N_{buff\_write} + C^3_{buff\_write})/T \end{aligned}$$

Crossbar switch: The crossbar switch relays flits from input buffers to output ports. The boundary switching activities of the crossbar switch, i.e., $S^i_{crossbar\_I}$ (for input), $S^i_{crossbar\_O}$ (for output), and the number of switch traversal operations, $N_{crossbar}$, are used to model power consumption in a crossbar switch, as follows:

$$\begin{aligned} P_{crossbar} &= (C^1_{crossbar\_I} \times \sum_{i=1}^{N_{crossbar}} S^i_{crossbar\_I} \\ &+ C^1_{crossbar\_O} \times \sum_{i=1}^{N_{crossbar}} S^i_{crossbar\_O} \\ &+ C^2_{crossbar} \times N_{crossbar} + C^3_{crossbar})/T \end{aligned} \quad (3)$$

Links: The power consumption of link circuitry is similarly modeled based on input switching activity, S^i_{link}, and the number of link traversal operations, N_{link}:

$$P_{link} = (C^1_{link} \times \sum_{i=1}^{N_{link}} S^i_{link} + C^2_{link} \times N_{link} + C^3_{link})/T \quad (4)$$

The total router power consumption is thus estimated as in Figure 2(c):

$$P_{total} = P_{buff\_read} + P_{crossbar} + P_{link} + P_{buff\_write} \quad (5)$$

The aggregate switching activity parameters in the different functions are not independent. Switching activity is the Hamming distance between adjacent flits, which changes when either flit data change or when the flit sequence is shuffled. In wormhole routers, the flit sequence may only change during crossbar traversal. In virtual-channel routers, read and write flit sequences may be different due to the interleaving of virtual channels on a physical channel. In the operation sequence shown in Figure 2(c), $S_{buff\_read} \equiv S_{crossbar\_I}$, $S_{crossbar\_O} \equiv S_{link}$, and $S_{link} \equiv S_{buff\_write}$, where S_{buff_write} is the switching activity introduced by buffer write operations in the downstream router.

The router power model can now be simplified as follows:

$$P_{total} = (C^1 \times \sum_{i=1}^{N_{flit}} S^i_{buff\_read} + C^2 \times \sum_{i=1}^{N_{flit}} S^i_{crossbar\_O} + C^3 \times N_{flit} + C^4)/T \quad (6)$$

where N_{flit} is the number of flits traversing this router during time T, and

$$C^1 = C^1_{buff\_read} + C^1_{crossbar\_I}$$
$$C^2 = C^1_{crossbar\_O} + C^1_{link} + C^1_{buff\_write}$$
$$C^3 = C^2_{buff\_read} + C^2_{crossbar} + C^2_{link} + C^2_{buff\_write}$$
$$C^4 = C^3_{buff\_read} + C^3_{crossbar} + C^3_{link} + C^3_{buff\_write}$$

Only two aggregate switching activity parameters, S_{buff_read} and $S_{crossbar\_O}$, need to be captured on-line (we need extra hardware to capture the switching activity at the network interface). If we had based our estimation on the router operation sequence shown in Figure 2(b), one more switching activity parameter (for input buffer write) would need to be monitored.

We use temporal and spatial sampling techniques to further simplify hardware implementation – assuming the flit width is B bits, for every M flits, switching activities are gathered based on the b bits in the first adjacent pair of flits, i.e., the temporal sampling ratio is $\frac{1}{M}$, and the spatial sampling ratio is $\frac{b}{B}$.

Estimation error: We use the on-chip 4-by-4 torus network described above for evaluating the accuracy of our on-line power estimation technique. This quantitative measurement of error, ϵ, allows designers to factor in the error margin when providing peak power constraints.

We compare our estimates with those given for Orion in [15][4], an architecture-level power model and simulator. In each packet, the payload of the flits is generated by first generating a Gaussian sequence, x_n, and passing it through an autoregressive filter given by the following equation:

$$payload_{flit_i} = \beta \times payload_{flit_{i-1}} + x_n \quad (7)$$

where β is the correlation parameter. A higher β implies that the payload sequence is more temporally correlated. We set $\beta = 0.8$ in our experiments.

As shown in Figure 4, our on-line power estimation technique introduces a maximum error of 7.4% and 3.4% for sampling ratio $\frac{1}{256}$ (temporal sampling ratio $\frac{1}{16}$, spatial sampling ratio $\frac{1}{16}$) and $\frac{1}{128}$ (temporal sampling ratio $\frac{1}{16}$, spatial sampling ratio $\frac{1}{8}$), respectively, at low network traffic, with improved estimation accuracy as network traffic increases. Note that estimation accuracy is more important under high network traffic for PowerHerd to meet peak power constraints. Including the estimation error of Orion [15], the estimation error, ϵ, is within 10%. This error margin needs to be taken into consideration when setting P_{GPB}.

3.2 Dynamic Power Prediction

Due to the dynamic nature of network traffic, network power consumption exhibits both transient fluctuations and long-term transitions, causing on-line power estimation to be noisy. Transient power fluctuations of very short duration do not affect the longer-term power constraints that we are targeting, such as a system's thermal profile. However, these transient fluctuations occlude the actual longer-term power demand profile of the network traffic, and may mislead the power regulation mechanism, degrading the effectiveness of power budget allocation.

[4]Orion has been verified with the MIT RAW on-chip network [14], which shows that Orion introduces 2-3% error with respect to circuit-level power simulations. The authors of Orion are continuing in-depth verification studies with various other networks.

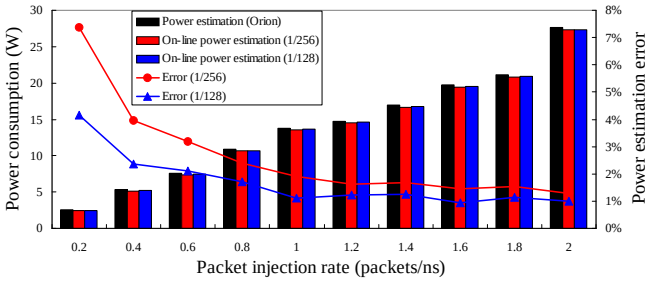


Figure 4: On-line power estimation versus off-line detailed architecture-level power estimation.

To address the above concerns, power estimates are averaged over a pre-defined time window, and smoothed using exponential weighted average, combining both current and past power consumption, before predicting the future power consumption, as follows:

$$Power_{pre} = \frac{W \times Power_S}{W+1} + \sum_{i=1}^{S-1} \frac{W \times Power_i}{(W+1)^{S-i+1}} \quad (8)$$

where $Power_{pre}$ is the predicted router power consumption for the next time window, and $Power_S$ is the average power estimate in the current time window S. $\sum_{i=1}^{S-1} \frac{W \times Power_i}{(W+1)^{S-i+1}}$ is the predicted average power consumption of the current time window from the previous time window. W is a predefined weight, which is set to three in our experiments.

3.3 Dynamic Power Sharing

Within an interconnection network, power demand is not uniformly distributed, and dynamically changes with time as traffic varies. Fixed router power budget allocation can result in serious performance degradation – some routers may require extra power resources to relay their network workload while others may have spare power budget that will be wasted. Hence, it is beneficial to share power budgets among routers and shuffle spare power resources within the network to improve network performance.

Since the interconnection network is a distributed system, a centralized mechanism requiring real-time global information gathering and decision making is hard and expensive to implement. We therefore propose a distributed approach here – power resources are exchanged among neighboring routers through a local power sharing interface.

For each router i, the average power consumption is evaluated over each time period T_{GPB}. Within each T_{GPB}, the local power budget, P^i_{LPB}, is transformed into a local energy budget, E^i_{LPB}, where $E^i_{LPB} = T_{GPB} \times P^i_{LPB}$. The power sharing problem then becomes an energy sharing problem – for each router i, within each T_{GPB}, if its energy consumption E_i is less than its local energy budget, E^i_{LPB}, it tries to give its spare energy to its neighboring routers. Otherwise, it tries to obtain extra energy resources from its neighboring routers. The power sharing protocol is shown in Figure 5.

Within each T_{GPB}, power sharing is invoked periodically. Each T_{GPB} is partitioned into N time slots, and power sharing is invoked at the beginning of each time slot, so that the power sharing interval $T_{share} = \frac{T_{GPB}}{N}$. Parameter N determines the interval of

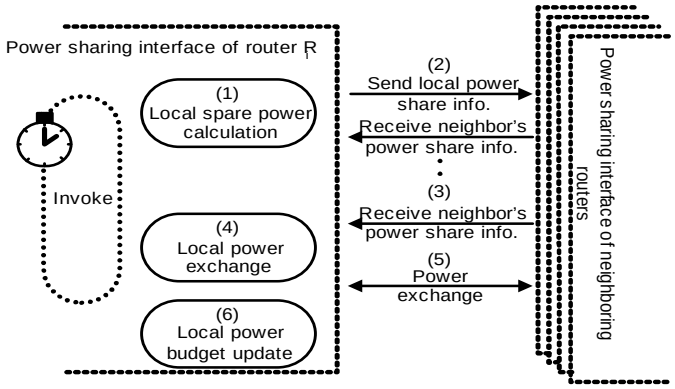


Figure 5: Distributed power sharing protocol.

power sharing being invoked. Potentially, as N increases, the effectiveness of power budget allocation improves, but the overhead introduced by power sharing increases too.

When power sharing is invoked at time slot k, each router calculates the spare energy that can be shared with the neighboring routers based on the following parameters – total energy budget (E^i_{LPB}), total energy consumption from the beginning of this time period T_{GPB} (E_i), predicted energy consumption (E_p) in each time slot, and the remaining time slots within the current period T_{GPB} ($N-k$), as shown below.

$$E^i_{share} = \frac{E^i_{LPB} - E_i - E_p * (N-k)}{N-k} \times \alpha; 0 \leq \alpha \leq N \quad (9)$$

If E^i_{share} is positive, it indicates that router i has spare energy that can be shared with its neighbors. Otherwise, this router requires extra energy. Parameter α varies with time slot k. In the beginning of time period T_{GPB}, k is small, and higher residue energy resources are available. A more aggressive power sharing policy can be employed using a higher value of α. When k is high, which implies we are closer to the end of time period T_{GPB}, less aggressive power sharing should be used to guarantee that the local power budget is enough to support the local traffi c workload.

After calculating the local spare energy that can be shared, routers exchange energy sharing information with their neighbors. Then the energy budgets of adjacent routers need to be updated. This can be implemented through either a pull or push-based approach. In a pull-based approach, needy routers decide how much energy sharing they require from their neighbors, while in a push-based approach, the amount of energy sharing is determined by the routers with extra energy. We use a push-based approach. Each router, if it has energy to spare, gathers the energy demand information from its neighbors, and then decides how to distribute its spare energy budget to these neighbors. We use the following policy: Neighboring routers are sorted based on their energy demand, which determines the priority with which they receive part of the spare energy.

Once energy sharing between neighboring routers is complete, each router updates its local energy and power budgets, as follows:

$$\begin{aligned} E^i_{LPB} &= E^i_{LPB} + e_{share} \\ P^i_{LPB} &= P^i_{LPB} + \frac{e_{share}}{T_{GPB}} \end{aligned} \quad (10)$$

where e_{share} is the change in the energy resources of router i through this round of energy sharing, which can be either positive (if it obtained energy resources from its neighbors) or negative (if it gave energy resources to its neighbors).

3.4 Dynamic Power Throttling

The power throttling mechanism at each router acts to ensure that the router power consumption falls below the budget allocated in the *power budget register*. It is costly to place throttling mechanisms in each router hardware component. Since router operations are triggered one-after-another when a packet flows through the router, we can instead throttle the flow, thus effectively regulating all hardware elements triggered by a flit. We propose a microarchitecture-level technique that integrates power-aware throttling logic into the switch allocator that controls crossbar switch access. Simply speaking, when router power consumption nears the local power budget, no flits are granted crossbar access, effectively gating buffer read, crossbar traversal, link traversal and buffer write at the downstream router, which are the power-dominant operations shown in Figure 2(c). This technique also permits a simple hardware implementation, since only an additional on/off bit is input to the allocator's grant circuits, and can be activated on a cycle-by-cycle basis, dynamically adjusting to varying power budgets and network workloads. While power throttling can also be applied to other components, such as routing and virtual channel allocation logic, this will only impact head flits. In addition, our technique does not introduce any performance degradation, unlike other dynamic power optimization techniques, such as dynamic voltage scaling with links [12] where a performance penalty is introduced during timing synchronization.

3.5 Power-aware Routing

When a packet is injected into an interconnection network, it follows a path from the source to destination, guided by a routing algorithm. Many routing algorithms driven by performance and fault tolerance have been proposed [5]. Here, we propose a power-aware routing protocol that routes packets away from nodes that are tightly power-constrained and thus unable to accommodate increased traffi c. It balances and redirects network traffi c based on a knowledge of the distribution of power resources.

We propose a distributed power-aware routing protocol where routing decisions are made dynamically based on the local power resources available in neighboring routers. Routers exchange power budget information between neighbors – when a router is running close to its allocated power budget, it flags a special $power\_low$ message to all its neighbors, leading them to mark this power-constrained router as a $hotspot$ to be avoided. When a $hotspot$ router has enough breathing room between actual power consumption and its allocated power budget, it sends out another special $power\_up$ message to its neighbors that causes it to be unmarked as a $hotspot$.

In order to avoid sending packets through $hotspot$ routers, the routing logic should have the ability to fi nd alternative routing paths to redirect network traffi c. More alternative routing paths improve performance, but may introduce other problems such as deadlock or livelock. We tailor distributed adaptive routing protocols to be power-aware – in addition to the load on neighboring routers, the power consumption and budget of routers are also taken into consideration. Figure 6 shows a power-aware adaptive routing algorithm for a 2-D torus that is based on a maximally adaptive routing algorithm [5]. Three virtual channels are used: VC0, VC1, and adaptive channel VC2. The fully adaptive channel, VC2, allows fully adaptive minimal routing in order to provide maximum routing flexibility. VC0 and VC1 provide a deadlock-free path allowing momentarily deadlocked packets in the fully adaptive channel to drain through, avoiding a deadlock in the torus. In each router, the route is dynamically selected based on the availability of power resources in neighboring routers. This routing algorithm permits a

```
xoffset = destination[0] −router_address[0]
yoffset = destination[1] –router_address[1]
xdirection = (|xoffset| * 2 <= X size)? true: false
ydirection = (|yoffset| * 2 <= Y size)? true: false
if(xdirection)                    | if(ydirection)
  X = (xoffset < 0) ? X− : X+| Y = (yoffset < 0) ? Y– : Y+
else                              | else
  X = (xoffset < 0) ? X+ : X−| Y = (yoffset < 0) ? Y+ : Y–
if(yoffset != 0)
 if(neighbor_y_is_not_a_hotspot)
    channel.add((Y,VC2))
if(xoffset < 0)
 if(neighbor_x_is_not_a_hotspot)
    channel.add((X,VC0),(X,VC2))
else if(xoffset > 0)
 if(neighbor_x_is_not_a_hotspot)
    channel.add((X,VC1),(X,VC2))
else if(yoffset > 0)
 if(neighbor_y_is_not_a_hotspot)
    channel.add((Y,VC0))
else
 if(yoffset < 0)
  if(neighbor_y_is_not_a_hotspot)
    channel.add((Y,VC1))
```

Figure 6: Power-aware routing protocol for a torus.

simple hardware implementation. As we demonstrate in our simulations later, power-aware sharing enables power guarantees without signifi cant network performance penalty.

3.6 Hardware Overhead

PowerHerd is implemented in each router – while PowerHerd mechanisms are not on the critical path of a router, lower hardware overhead reduces the power and area penalty introduced by PowerHerd. Here, we analyze each functional component of PowerHerd. Dynamic power estimation requires the most hardware complexity. On-line switching activity needs to be determined at each input and output of the crossbar switch, requiring complicated multiplexer and buffer logic. In order to minimize the hardware overhead and still achieve accurate power estimation, we use both temporal and spatial sampling. For example, for the router we assumed in Section 3.1 with 256-bit flits, we chose the sampling rate to be $\frac{1}{256}$, with a $\frac{1}{16}$ contribution from spatial sampling and $\frac{1}{16}$ from temporal sampling. Thus, only 16 out of 256 bit lines need to be monitored. This results in tolerable area overhead and power consumption overhead.

The dynamic power prediction module of PowerHerd is based on exponential weighted average power. We choose the weight such that the divide and multiply operations can be implemented with shift and add operations, reducing hardware complexity. For dynamic power throttling, only disable signals need to be added to the original switch allocation logic. As for dynamic power sharing, an overhead arises from communicating and exchanging the power budget information. Since this occurs only between the neighboring routers, and involves just a few bits of information in each message every 10,000 cycles where the power sharing interval is 5 μs, the overhead is reasonable. Finally, the power-aware routing protocol involves just a minor variation on a simple adaptive routing protocol. The extra overhead occurs with each router having to inform its neighbors when its local power status changes, which is infrequent and can be implemented with very simple hardware.

In short, hardware complexity was foremost in our minds when designing PowerHerd, and we believe it adds negligible complexity to existing routers. We will quantify this overhead through RTL synthesis in the future.

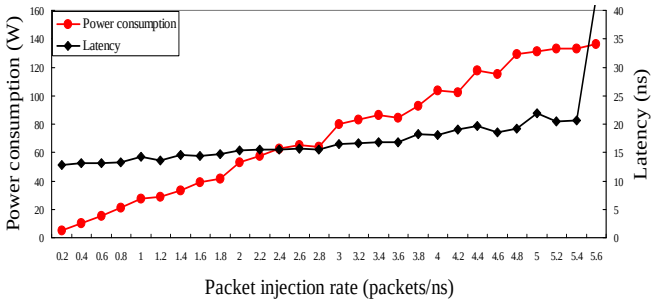


Figure 7: Power consumption vs. network traffic.

4. EXPERIMENTAL RESULTS

For our experiments, we assume a two-dimensional 8x8 on-chip torus interconnection network implemented in 0.10 μm CMOS technology with $V_{dd} = 1.2V$ in a 24mm by 24mm chip, a larger network extended from that assumed in [4]. Each router has three virtual channels and 64 flit buffers, each 256 bits wide, per input port. Following the estimation method in [13], we assume a thermal time constant of 100 μs, i.e., T_{GPB} = 100 μs.

Network traffi c is generated based on the two-level traffi c model proposed in [12] that was demonstrated to exhibit the realistic temporal and spatial variance of bursty network traffi c. The workload consists of concurrent communication sessions with Poisson inter-arrival times, and long-range dependent packet inter-arrival times within each communication session.

Latency, throughput and peak power consumption are the metrics we used to evaluate PowerHerd. Latency spans the creation of the fi rst flit of the packet to ejection of its last flit at the destination router, including source queuing time and assuming immediate ejection. Each simulation is run for 10 million cycles, and the latency of all packets averaged. The saturation throughput of the network is where average packet latency worsens to more than twice the zero-load latency (*i.e.*, the average packet latency when there is no network congestion). Peak power consumption is determined by selecting the maximum average power consumption over each T_{GPB}.

4.1 Effectiveness of PowerHerd in Satisfying Peak Power Constraints

PowerHerd's design inherently ensures that it satisfi es peak power constraints within an error margin of ϵ, where ϵ is dependent upon the accuracy of on-line power estimation, and was quantifi ed in Section 3.1. We next investigate how effectively PowerHerd allocates the peak power budget dynamically, i.e., how does the latency-throughput performance of a network governed by PowerHerd compare to two alternatives to PowerHerd – (1) *IdealMaxPower*: the ideal though unachievable yardstick that PowerHerd should be measured against – a centralized global controller with perfect global information that can allocate the power budget to each local router based on its exact needs, and (2) *StaticAllocPower*: Static design-time distribution of the peak power constraint across the network.

Table 1: Performance of PowerHerd vs. IdealMaxPower. The first three rows show the eight peak power consumption numbers for IdealMaxPower that are used as global power constraints to evaluate the performance of PowerHerd, the corresponding network throughput, and network latencies. The last two rows show the actual peak network power consumption and latency regulated by PowerHerd at each corresponding network traffic under the given P_{GPB}.

(IdealMaxPower) Peak power (W) (P_{GPB})	27.3	33.5	53.3	65.4	86.5	103.4	115.1	136.3
(IdealMaxPower) Throughput (packets/ns)	1.0	1.4	2.0	2.6	3.4	4.0	4.6	5.4
(IdealMaxPower) Latency (ns)	14.2	14.5	15.3	15.8	16.8	18.0	18.5	20.6
(PowerHerd) Peak power consumption (W)	26.9	33.2	52.9	64.7	86.0	102.2	114.6	135.9
(PowerHerd) Latency (ns)	14.3	14.6	15.4	15.9	17.0	18.1	18.7	20.6

We first study network performance vs. power consumption without global power constraints. Figure 7 shows the peak power consumption vs. latency-throughput performance with increasing network traffic. As shown in the figure, with an increase in network traffic, the peak power consumption varies from $5.1W$ to $136.3W$[5]. Since no global power constraints are imposed, the network latency at each injection rate is the optimal performance that can be achieved at the corresponding peak power consumption. This scenario thus mimics *IdealMaxPower*, the ideal centralized global controller that parcels power rapidly and accurately to nodes just when they are needed, with no sharing overhead.

We chose eight global power budgets, P_{GPB}, from the interval $[5.1W, 136.3W]$, and set power sharing interval T_{share} to be 5 μs, i.e., in each T_{GPB}, power sharing will be invoked 20 times. For *StaticAllocPower*, we try to devise a realistic static allocation where the global power budget is statically distributed in proportion to the average power consumption of each router over a 10 million cycle simulation of *IdealMaxPower*. At run-time, each router strictly obeys its local power constraint and does not share its power budget with its neighbors. We believe this scenario realistically models that followed by network designers, where network traffic is first characterized and used to guide design.

Table 1 compares PowerHerd to *IdealMaxPower*. It demonstrates PowerHerd can effectively regulate network peak power consumption (Row 4) under different global peak power constraints of P_{GPB} (Row 1). Also, it shows that at each global power constraint, PowerHerd maintains good network latency (Row 5) as compared to *IdealMaxPower* (Row 3), introducing less than 1% network latency penalty.

Figure 8 compares PowerHerd to *StaticAllocPower* at different peak power constraints. Clearly, we see PowerHerd arriving at a better latency-throughput performance with the same peak power constraint as *StaticAllocPower*, consistently delivering about twice the saturation throughput. This highlights *StaticAllocPower*'s inability to tackle the variance in traffic and demand for power at each local router, leading to routers that may waste their excess power resources, while others do not have enough power budget to relay traffic and ease congestion.

4.2 Effect of Power Sharing Interval on PowerHerd

In the previous experiments, the power sharing interval, T_{share}, is set to $5\mu s$, i.e., power sharing is invoked 20 times within each T_{GPB} ($= 100\mu s$). In PowerHerd, more frequent power sharing can potentially allocate power more effectively across the network to meet a fast, dynamically changing demand of power resources, leading to better network latency-throughput performance. However, the downside of more frequent power sharing is the communication overhead of exchanging information. We thus investigate the effect of T_{share} on network performance.

In these experiments, we set the global power constraints to be $136.3W$, $103.4W$, and $53.3W$, corresponding to a network throughput of 5.6 packets/ns, 4.0 packets/ns, and 2.0 packets/ns, respectively, for *IdealMaxPower*. Under each global power constraint, we use $5\mu s$, $12.5\mu s$, and $25\mu s$ as the power sharing interval. Figure 9 shows the results. Network performance under *StaticAllocPower* is also shown for comparison purposes. When the power sharing interval increases significantly, PowerHerd effectively degenerates to *StaticAllocPower*, with the exception of power-aware routing. As shown in Figure 9, under each global power constraint, as power sharing interval increases, network performance penalty increases. The performance penalty becomes visible when the power sharing interval increases beyond $12.5\mu s$, as PowerHerd finds it increasingly difficult to track the dynamically changing traffic.

4.3 Effect of Local Power Constraints on Network Performance

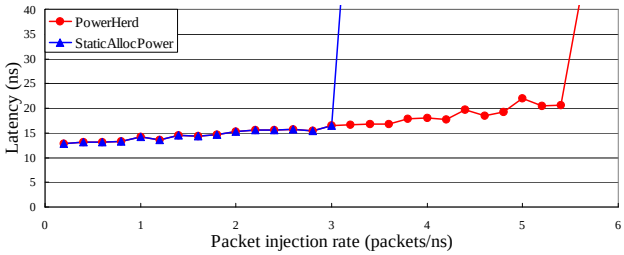
As explained in Section 2, global peak power constraints may not be the only power constraint in interconnection networks. Even when the total network power consumption remains below the global power constraint, bursty network traffic may still introduce local hotspots that can cause problems. In order to avoid hotspots within the network, local power constraints should also be enforced. In general, power constraints can be enforced globally, on a cluster of neighboring routers, or on each individual router. As the scope of peak power constraints tightens, a higher performance impact will be introduced, since PowerHerd can no longer share power freely across the network to maximize performance. We next evaluate the performance penalty introduced due to local power constraints.

We use the same three global power constraints used in the previous section. For each global power constraint, we also impose local power constraints at each router ranging from $2.0P_{avg}$, $1.8P_{avg}$, $1.6P_{avg}$, $1.4P_{avg}$ to $1.2P_{avg}$, where P_{avg} is the static power budget of each router based on the router's average power consumption. We also choose the power sharing interval to be 5 μs to eliminate the impact of insufficient power sharing on PowerHerd's performance. Figure 10 shows the results. Even with sufficient power sharing, as local power constraints tighten, a higher performance penalty is introduced.

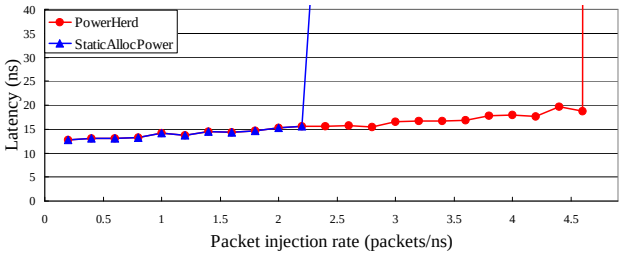
4.4 Impact on Interconnection Network Design

Current interconnection network design is based on design-time worst-case power analysis. This leads to network topologies of lower peak power consumption being used, despite poor network performance. Here, we compare two network design choices for an

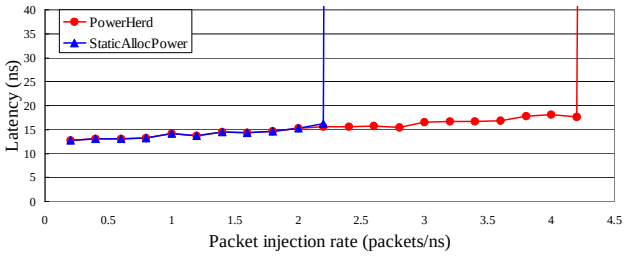
[5]Results are not monotonically increasing because the network traffic workload is bursty. Network traffic at statistically lower packet injection rates may contain temporary larger bursts that cause higher peak power consumption.



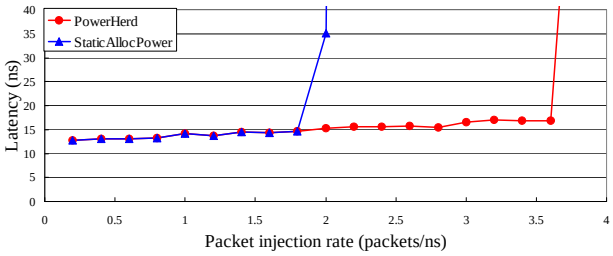
(a) Global power constraint = 136.3W



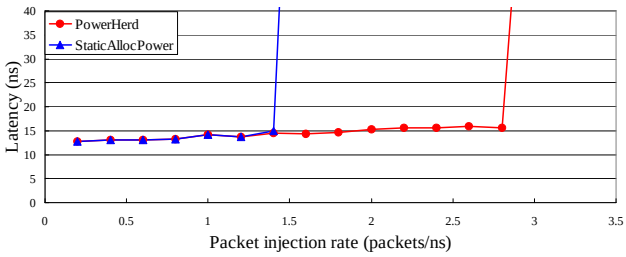
(b) Global power constraint = 115.1W



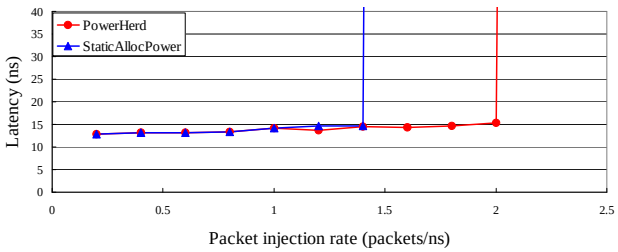
(c) Global power constraint = 103.4W



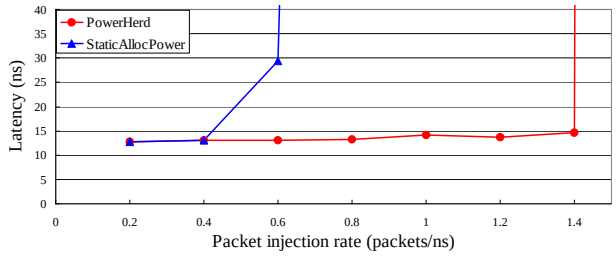
(d) Global power constraint = 86.5W



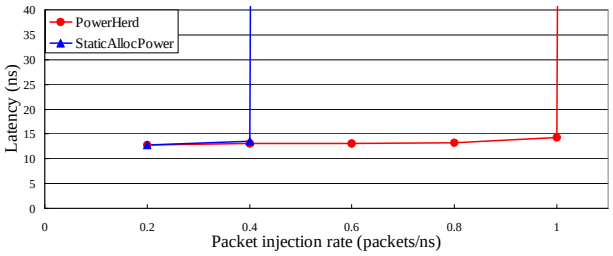
(e) Global power constraint = 65.4W



(f) Global power constraint = 53.3W



(g) Global power constraint = 33.5W

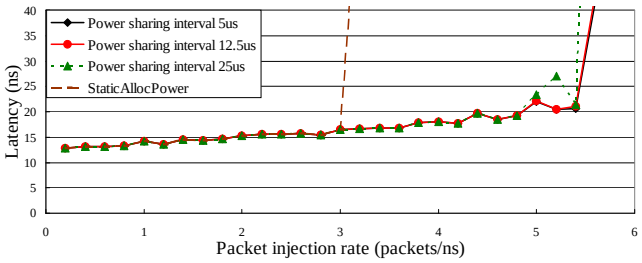


(h) Global power constraint = 27.3W

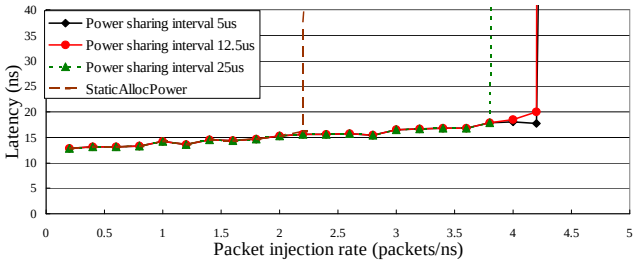
Figure 8: Latency-throughput of PowerHerd vs. StaticAllocPower under varying peak power constraints.

n-node network – an n-node ring and a $\sqrt{n}$-ary 2-cube (torus).

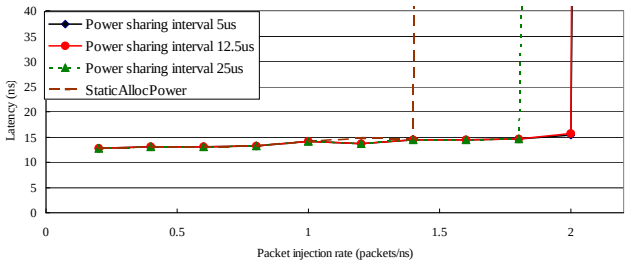
For uniform random traffi c, the average hop count of a torus network ($\frac{\sqrt{n}}{2}$) is much lower than an n-node ring ($\frac{n}{4}$). However, in a torus, the number of links and ports are doubled and more complex crossbar switch logic is used. Hence, while performance in terms of latency-throughput is better, power consumption is also considerably higher than an n-node ring. This may result in a designer giving up torus for a ring, in order to remain within worst-case



(a) Global power constraint = 136.3W

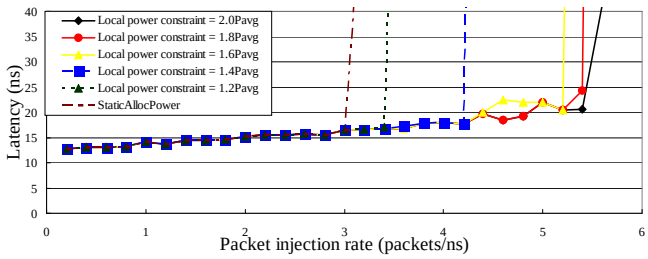


(b) Global power constraint = 103.4W

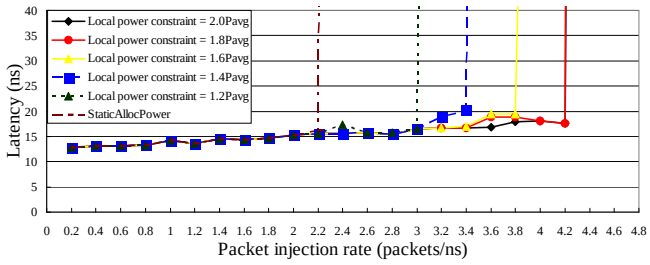


(c) Global power constraint = 53.3W

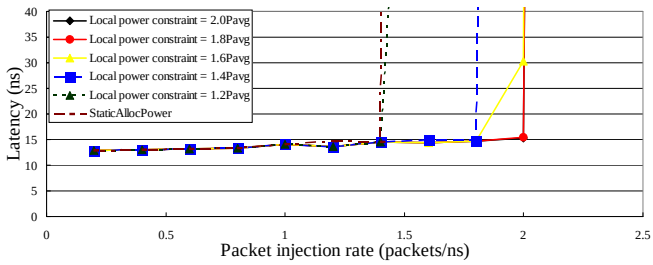
Figure 9: Effect of power sharing interval on PowerHerd's latency-throughput performance.



(a) Global power constraint = 136.3W



(b) Global power constraint = 103.4W



(c) Global power constraint = 53.3W

Figure 10: Effect of local power constraints on PowerHerd.

peak power constraints. This experiment explores the impact of PowerHerd, which makes it possible to still use a high-performance topology such as a torus, by dynamically regulating and guaranteeing that the peak power constraint is met, while enhancing network performance.

In this experiment, we compare a 16-node ring vs. a 4x4 torus. We set the peak power constraint to the power consumption of the 16-node ring under its maximum throughput, $P_{GPB} = 36.3W$. The worst-case power consumption of a 4x4 torus is $54.1W$, making it unsuitable for use at this P_{GPB} without PowerHerd. We then evaluate the latency-throughput of a 4x4 torus armed with PowerHerd. For comparison purposes, we set the input buffer size of each router port to be the same in the ring and torus, and construct different crossbar and link power models based on the structure of the ring and torus. Related parameters are the same as those in Section 3.1.

The network latency-throughput of a 16-node ring and 4x4 torus are shown in Figure 11. At the same peak power constraint of $36.3W$, it shows a 4x4 torus with PowerHerd delivering approximately half zero-load latency and twice the saturation throughput as compared to a 16-node ring. It clearly demonstrates that with PowerHerd, even under tight power budget constraints, high-performance network topologies with potentially infeasible peak power consumption can still be used, enriching design choices and improving system performance.

5. DISCUSSION AND RELATED WORK

Most prior research on interconnection networks has focused on performance optimization or fault tolerance until recently when power surfaced as a critical design constraint. Recent studies have explored off-line power modeling and estimation for interconnection networks and other network fabrics [3, 10, 15, 17]. In this

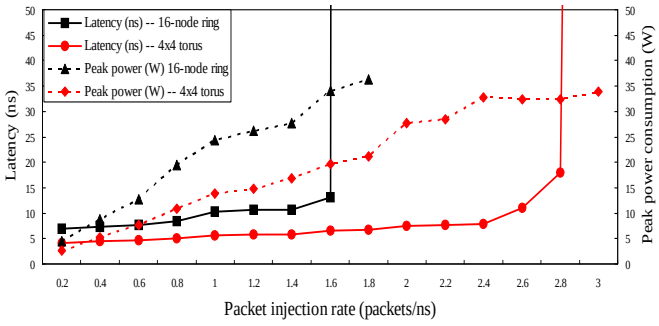


Figure 11: Power-performance of a 16-node ring and a 4-by-4 torus with PowerHerd.

work, we used Orion [15], an architecture-level power-performance simulator to verify the accuracy of our run-time power estimation technique. Researchers have also targeted dynamic power optimization of networks [12] and links [16]. Unlike these prior power optimization studies that aim at reducing *average* power consumption, PowerHerd targets peak power constraints in interconnection networks. Even when huge power savings are achieved in [12], worst-case power consumption cannot be guaranteed to be below a given constraint, and designers may still be faced with having to factor in worst-case power in their choice of network architectures.

Substantial research has explored the power efficiency of microprocessors, with research into thermal management most relevant to this work. Brooks [2] first proposed a dynamic thermal management technique. Skadron [13] further explored this idea and used control-theoretic techniques and thermal models to rigorously tackle thermal issues in microprocessors. Recently, researchers began exploring microarchitectural techniques to address the transient di/dt problem in power supply and delivery [6, 7]. Unlike microprocessors, interconnection networks are inherently distributed in nature. Centralized approaches, such as those previously proposed for microprocessors, are not applicable to networks as there does not exist a single knob in interconnection networks that can be used to regulate the power consumption of the entire network. We thus need an approach like PowerHerd that uses distributed power estimation, sharing and throttling mechanisms.

In this work, PowerHerd addresses the "slow" thermal and average maximum power consumption design constraints. While PowerHerd can also be extended to tackle rapid transient power changes, the key is the timing granularity. As shown in [7], transient di/dt effects can be in the range 50-200MHz, equivalent to tens to hundreds of clock cycles. It is too expensive, if not impossible, to use the same on-line power estimation technique as proposed in this paper to detect such short power transients. Instead, we will need to modify PowerHerd to use coarser-grained estimation techniques – using counters to detect the number of flits sent through each router, and detecting transient power through convolution techniques [6]. Since a packet will invoke the same operation sequence in each router, such convolution operations can be implemented very efficiently. While this approach cannot reach the same power estimation accuracy as our current technique, and will lead to a higher ϵ, the same PowerHerd infrastructure (estimation, prediction, sharing, throttling and routing) can be used. This involves a power-performance tradeoff that has to be explored in order to tackle power-related constraints with widely varying timing granularities.

As a distributed resource allocation mechanism, PowerHerd is more applicable to network-on-chip or network-on-package scenarios, where network components share global power and cooling resources, so that global and local power budgets can effectively regulate global power distribution and local hotspots. For network-on-board applications, such as next-generation server blades, network components are geographically close. Even with separate cooling packages, network thermal behavior is still tightly coupled. Hence, a global power regulation is still needed. For interconnection networks that link chassis-to-chassis or box-to-box, separate power supplies drive each router and inter-router thermal effects are negligible. Hence, the local power budget of each router becomes the only constraint. In these scenarios, dynamic power estimation, prediction, throttling and power-aware routing can still be applied.

6. CONCLUSIONS

Peak power constraints are the most critical constraints faced by systems designers, due to tight cooling costs, thermal issues, and power delivery design challenges. While prior works tackled this issue in microprocessors, they employed centralized approaches that cannot be extended to the distributed nature of interconnection networks.

In this paper, we proposed PowerHerd, where distributed mechanisms in each router dynamically regulate power to ensure that the peak power constraint is not exceeded across the entire network. PowerHerd involves run-time power estimation mechanisms that sample flit activity for estimating the local power consumption at each router. This estimate is then used to predict future demand for power consumption, and a distributed power sharing mechanism shuffles power budgets around neighboring routers in order to track traffic hotspots rapidly and improve network performance. In each router, simple regulation mechanisms are in place to throttle packet flow so as to keep local power consumption below the allocated power budget. Power-aware routing further load-balances traffic in response to power profiles, improving network performance.

Simulations demonstrate that PowerHerd is effective at regulating peak power constraints with a negligible network performance penalty (less than 1% increase in maximum network latency, with no throughput degradation). With PowerHerd, designers can choose network architectures that were originally infeasible as their peak power consumption exceeds the peak power constraint. Our simulations show PowerHerd enabling the selection of a 4-by-4 torus that delivered about half the network latency and twice the network throughput as compared to a ring topology that would be the only feasible choice without PowerHerd.

Acknowledgments

The authors would like to thank Hang-Sheng Wang of Princeton for his help with understanding and using his Orion power models for validating our run-time power estimation. The authors would also like to thank Prof. Kai Li, for the use of his research group's clusters of workstations at Princeton for our network simulations, and anonymous reviewers for their valuable comments. This work was supported in part by DARPA under contract no. DAAB07-00-C-L516 and in part by NSF CAREER grant CCR-0237540.

7. REFERENCES

[1] S. Borkar. Design challenges of technology scaling. *IEEE-MICRO*, 19(4):23–20, July/Aug. 1999.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. International Symposium on High Performance Computer Architecture*, pages 171–182, Jan. 2001.

[3] X.-N. Chen and L.-S. Peh. Leakage power modeling and optimization of interconnection networks. In *Proc. International Symposium on Low Power Electronics and Design*, Aug. 2003.

[4] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. Design Automation Conference*, pages 684–689, June 1999.

[5] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*. Morgan Kaufmann Publishers, San Francisco, CA, 2003.

[6] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural simulation and control of di/dt-induced power supply voltage variation. In *Proc. International Symposium on High Performance Computer Architecture*, pages 7–16, Feb. 2002.

[7] R. Joseph, D. Brooks, and M. Martonosi. Control techniques to eliminate voltage emergencies in high performance processors. In *Proc. International Symposium on High Performance Computer Architecture*, pages 91–102, Feb. 2003.

[8] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 network architecture. *IEEE Micro*, 22(1):26–35, Jan./Feb. 2002.

[9] R. H. Myers. *Classical and Modern Regression with Application*. Duxbury Press, Boston, MA, 1989.

[10] C. Patel, S. Chai, S. Yalamanchili, and D. Schimmel. Power-constrained design of multiprocessor interconnection networks. In *Proc. International Conference on Computer Design*, pages 408–416, Oct. 1997.

[11] RLX. *Features of Server Blades Design*. `http://www.rlx.com`.

[12] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proc. International Symposium on High Performance Computer Architecture*, pages 79–90, Feb. 2003.

[13] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proc. International Symposium on High Performance Computer Architecture*, pages 17–28, Feb. 2002.

[14] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE-MICRO*, 22(2):25–35, Mar./Apr. 2002.

[15] H.-S. Wang, X.-P. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *Proc. International Symposium on Microarchitecture*, pages 294–305, Nov. 2002.

[16] F. Worm, P. Ienne, P. Thiran, and G. De Micheli. An adaptive low power transmission scheme for on-chip networks. In *Proc. International System Synthesis Symposium*, Oct. 2002.

[17] T. T. Ye, L. Benini, and G. De Micheli. Analysis of power consumption of switch fabrics in network routers. In *Proc. Design Automation Conference*, pages 524–529, June 2002.

[18] L.-T. Yeh and R. C. Chu. *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. ASME Press, New York, NY, 2002.