Toward Runtime Power Management of Exascale Networks by On/Off Control of Links

Ehsan Totoni
University of Illinois-Urbana Champaign, PPL
Charm++ Workshop, April 16 2013

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

Power challenge

- Power is a major challenge
- Blue Waters consuming up to 13 MW
  - Enough to electrify a small town
  - Power and cooling infrastructure
- Up to 30% of power in network
  - Projected for future by Peter Kogge
  - Saving 25% power in current Cray XT system by turning down network
    - Work from Sandia

Network link power

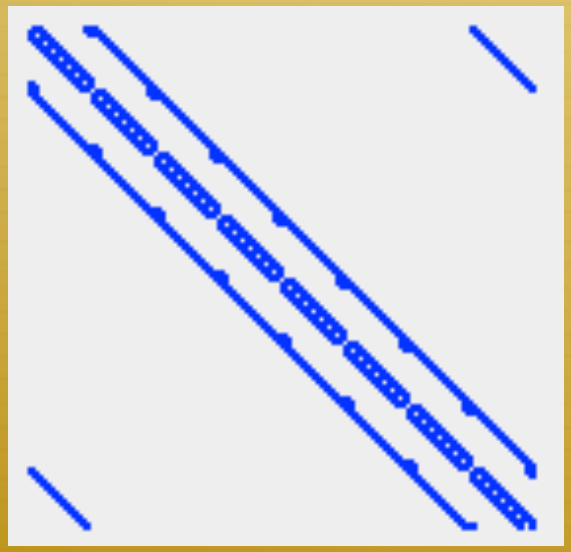
- Network is not “energy proportional”
  - Consumption is not related to utilization
  - Near peak most of the time
  - Unlike processor
- Recent study:
  - Work from Google in ISCA’10
  - 50% of power in network of non-HPC data center
  - When CPU’s underutilized
- Up to 65% of network’s power is in links

Exascale networks

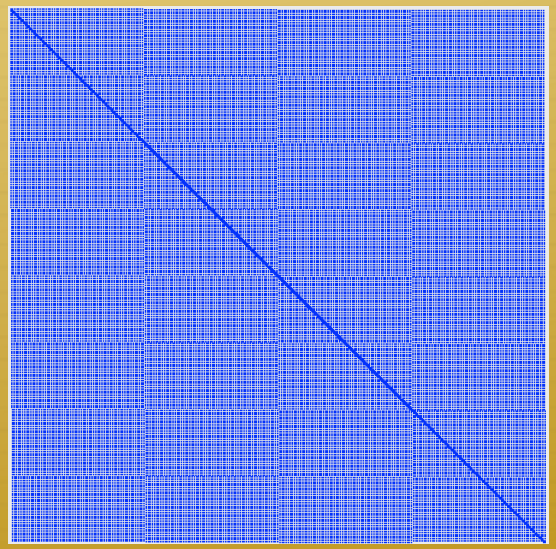
- Dragonfly
  - IBM PERCS in Power 775 machines
  - Cray Aries network in XC30 “Cascade”
  - DOE Exascale Report
- High dimensional Tori
  - 5D Torus in IBM Blue Gen/Q
  - 6D Torus in K Computer
- Higher radix -> a lot of links!

Communication patterns

- Applications' communication patterns are different
- Network topology designed for a wide range of applications

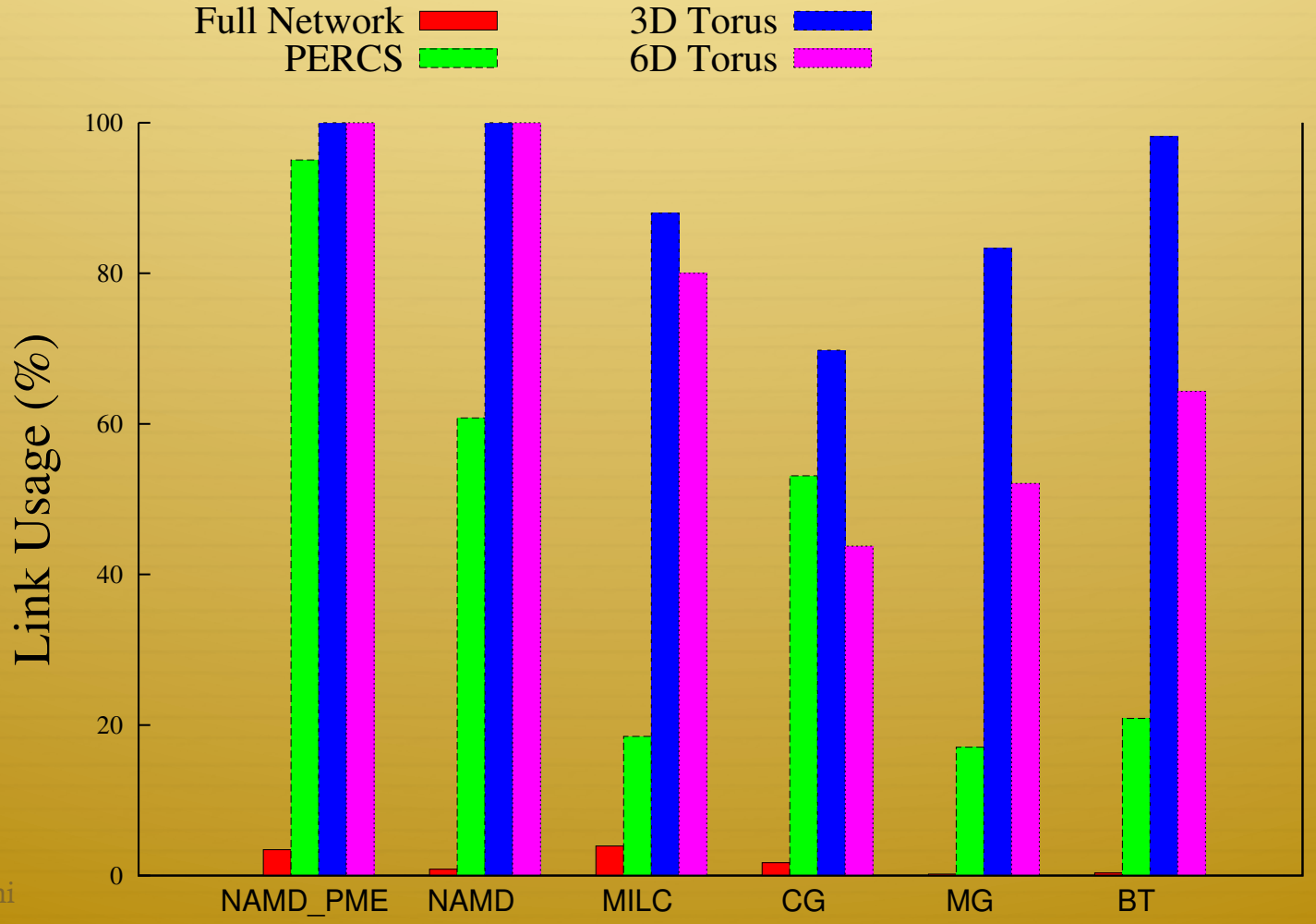


MILC

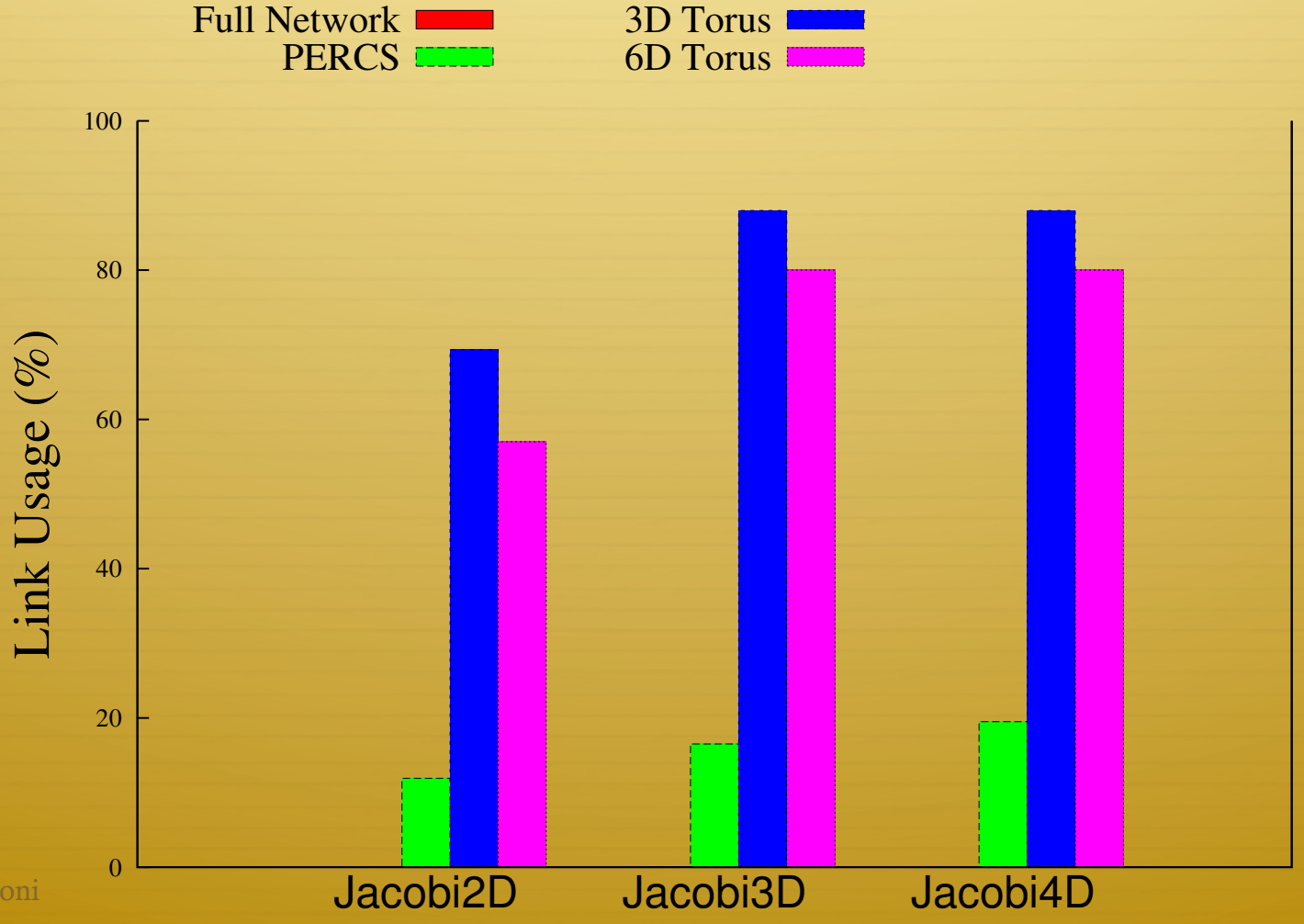


NPB CG

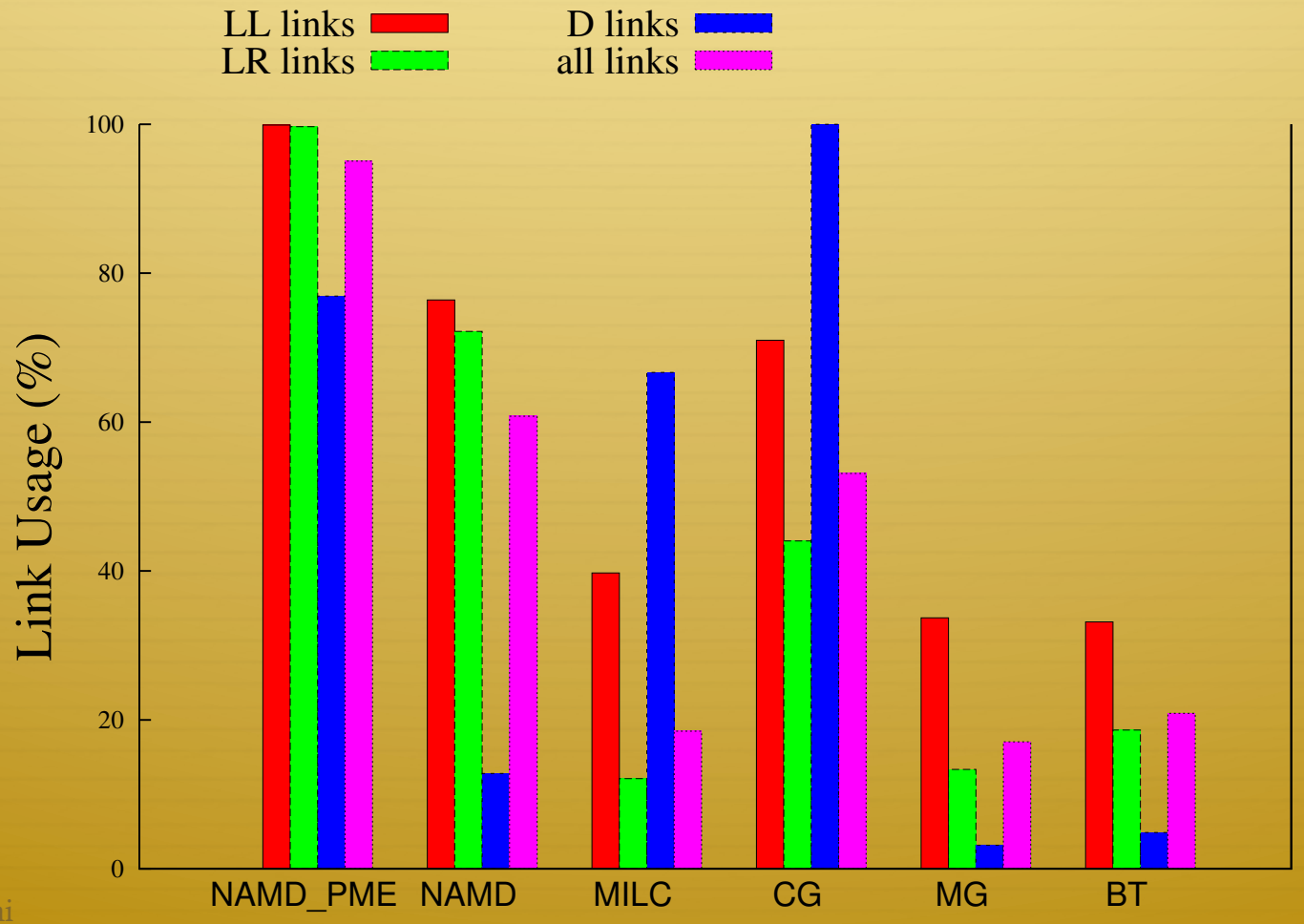
Fraction of links ever used



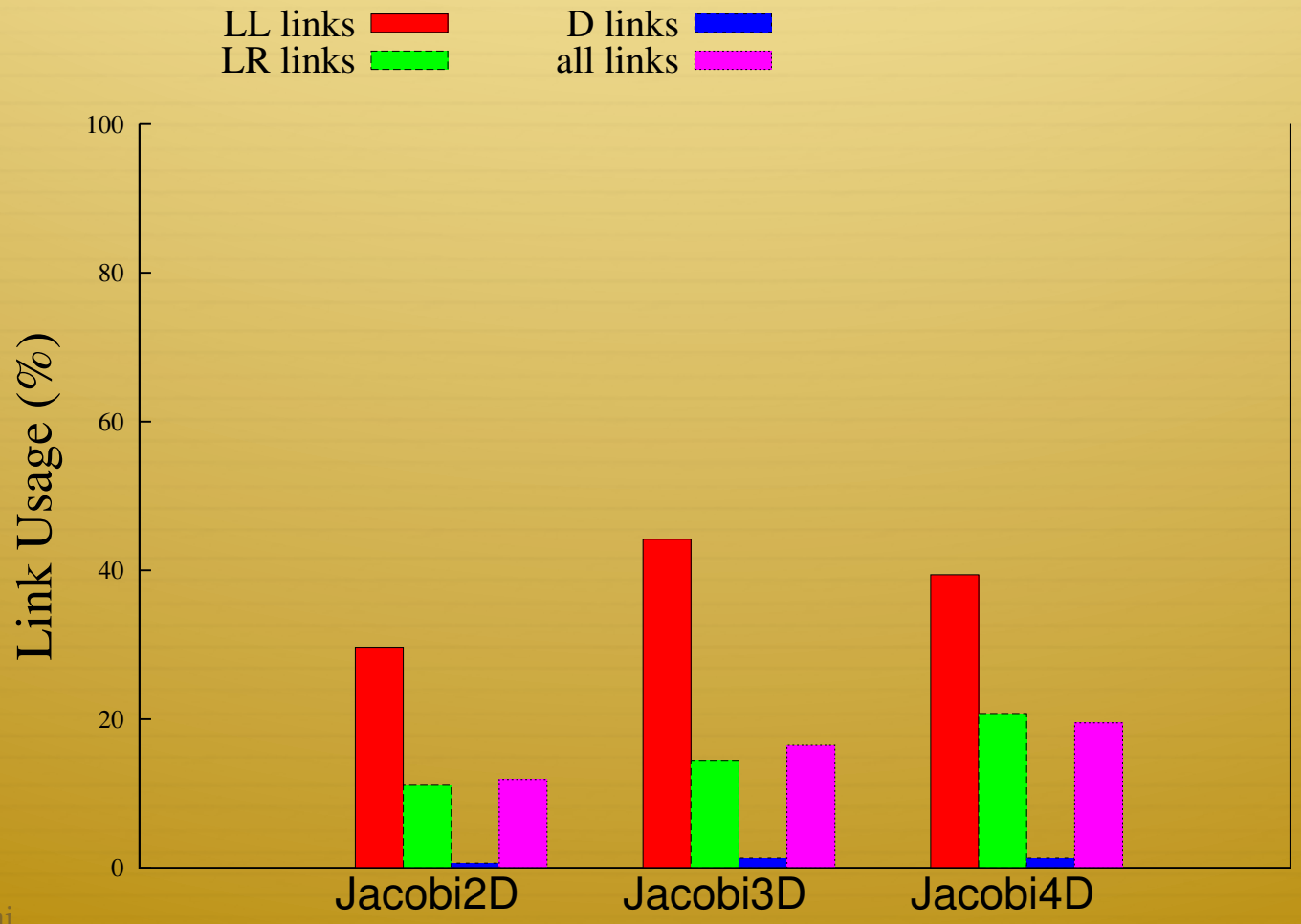
Nearest neighbor usage



More expensive links



Nearest neighbor



Solution to power waste

- Many of the links are never used
  - For common applications
- Are networks over-built? Maybe
  - FFTs are crucial
  - But processors are also overbuilt
- Let's make them "energy proportional"
  - Consume according to workload
  - Just like processors
- Turn off unused links
  - Commercial network exists (Motorola)

Runtime system solution

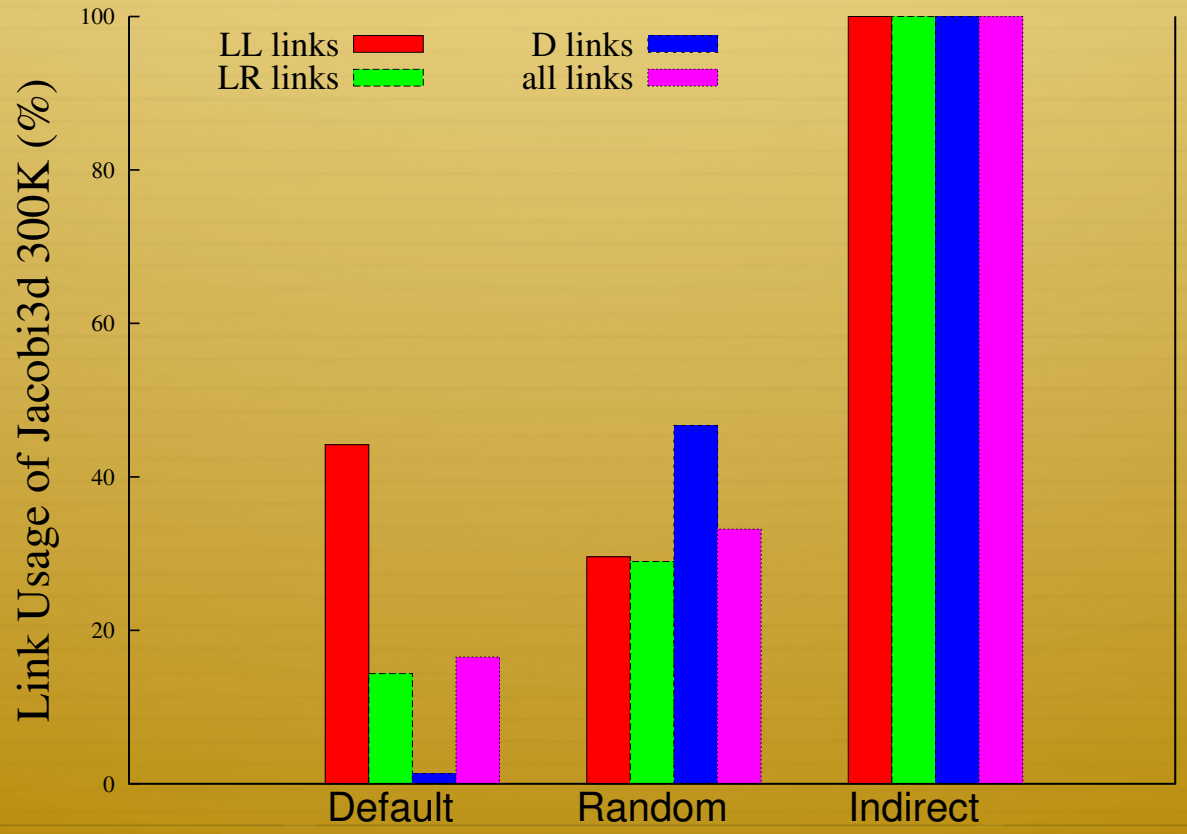
- Hardware can cause delays
  - According to related work
  - Not enough application knowledge
    - Small window size
- Compiler does not have enough info
  - Input dependent program flow
- Application does not know hardware
  - Significant programming burden to expose
- Runtime system is the best
  - mediates all communication
    - knows the application
    - knows the hardware

Feasibility

- Not probably available for your cluster downstairs
  - Need to convince hardware vendors
  - Runtime hints to hardware, small delay penalty if wrong
- Multiple jobs: interference
  - Isolated allocations are becoming common
    - Blue Genes allocate cubes already
  - Capability machines are for big jobs

Software design choices

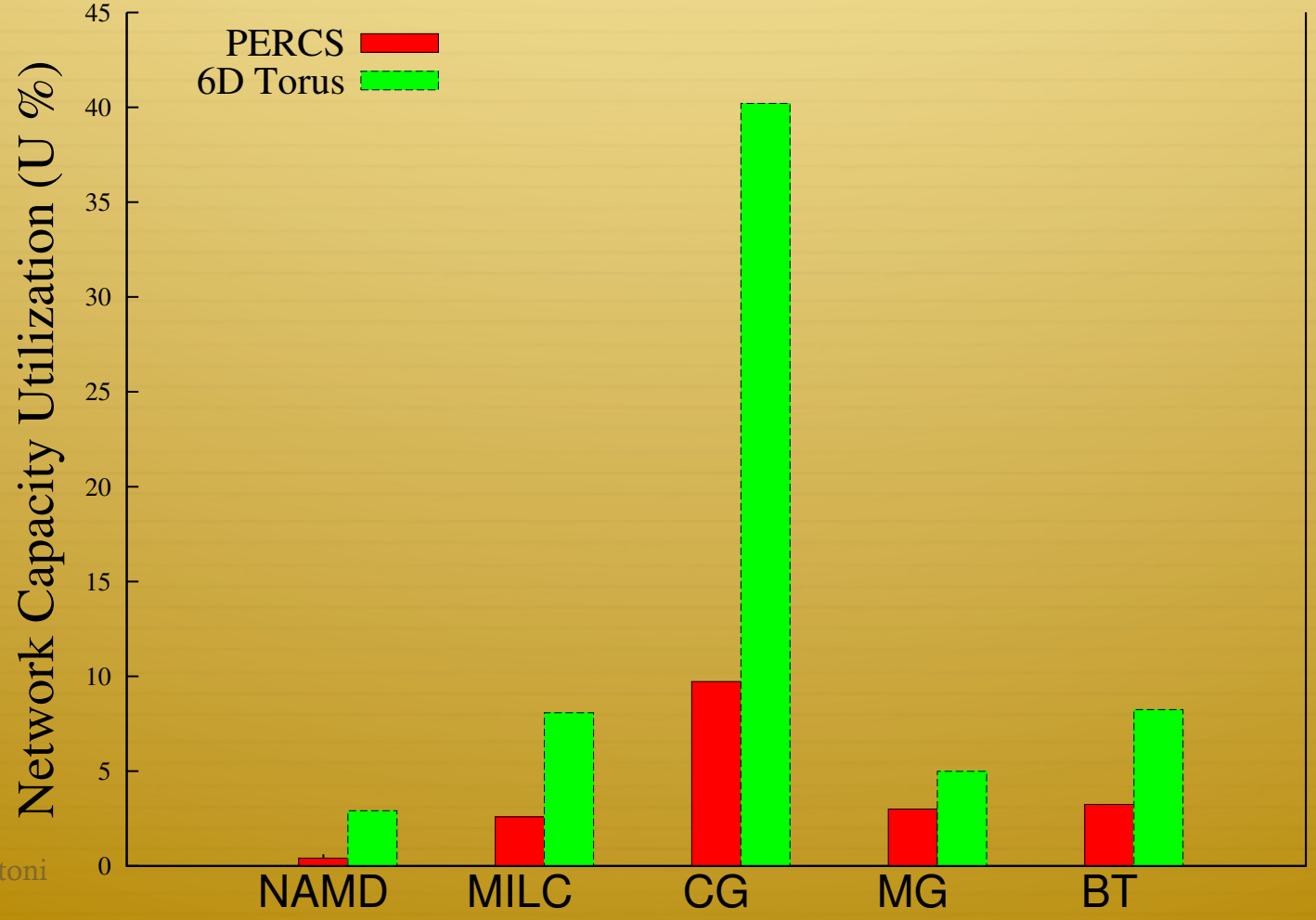
- Random mapping and indirect routing have similar performance but different link usages



Power model

- We saw many links that are **never used**
- **Used** links are not used all the time
  - For only a fraction of iteration time
  - Compute-communicate paradigm
- A power model for "network capacity utilization"
  - "Average" utilization of all the links
  - Assume that links are turned magically on and off
    - At the exact right time
  - No switching overhead
  - Example: network used one tenth of iteration time

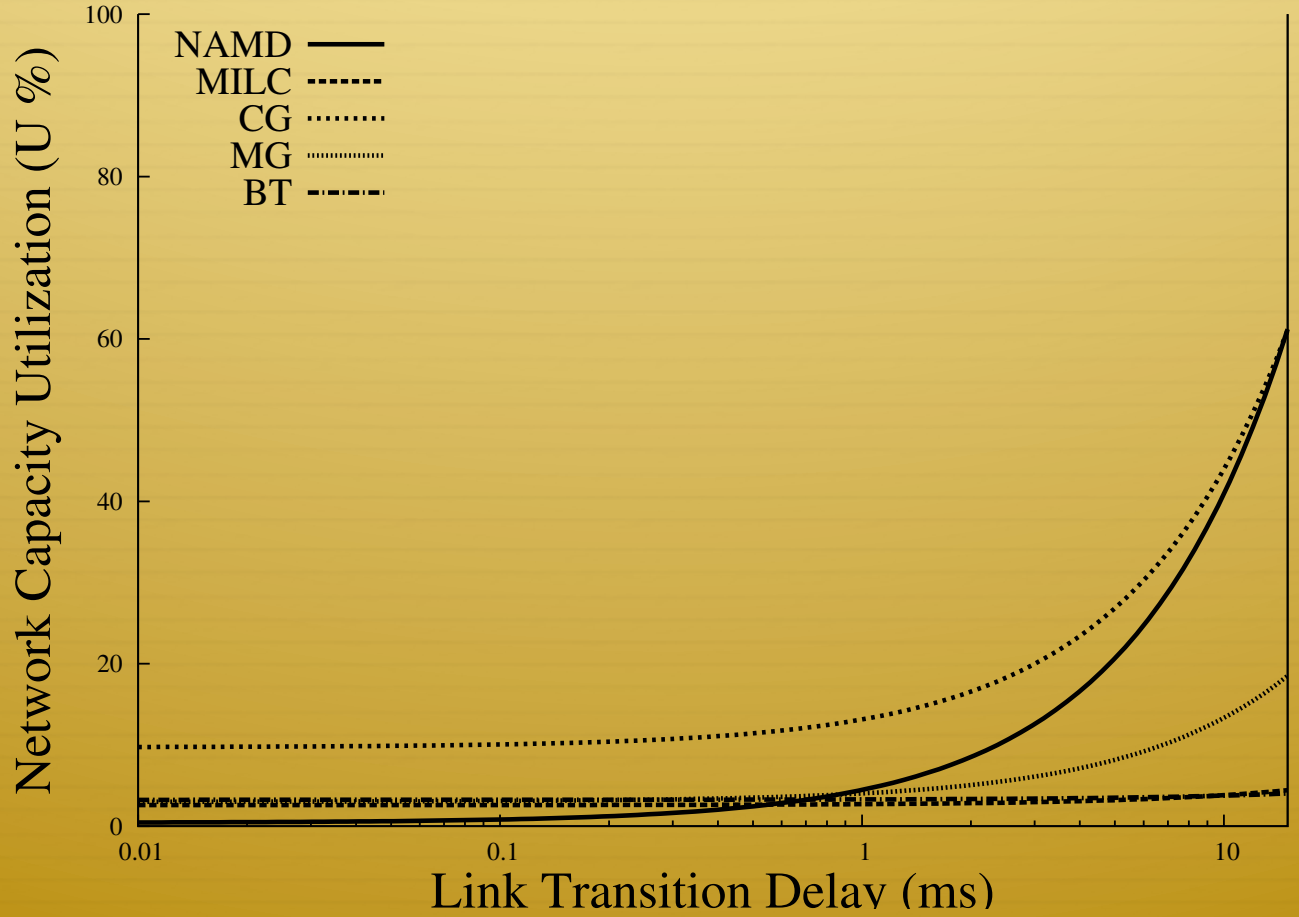
Model results



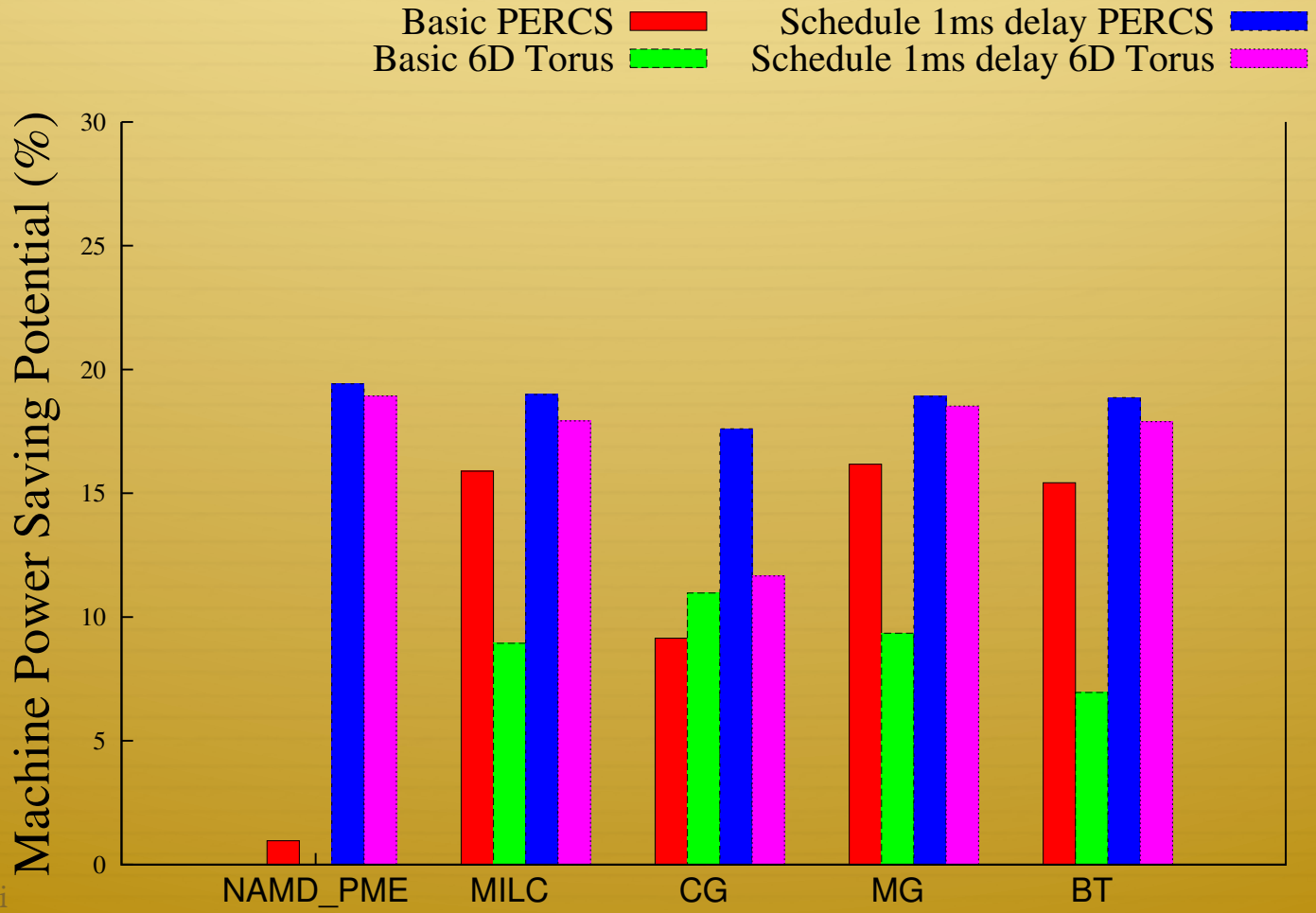
Scheduling on/offs

- Runtime roughly knows when a message will arrive
  - For common iterative HPC applications
  - Low noise systems (e.g. IBM Blue Genes)
- There is a delay for switching the link
  - 10 μ s for current implementation
  - Much smaller than iteration time
  - Runtime can be conservative
    - Schedule “on”s earlier
    - Similar to having more switching delay

Delay overhead



Results summary



Questions?

Are you convinced?