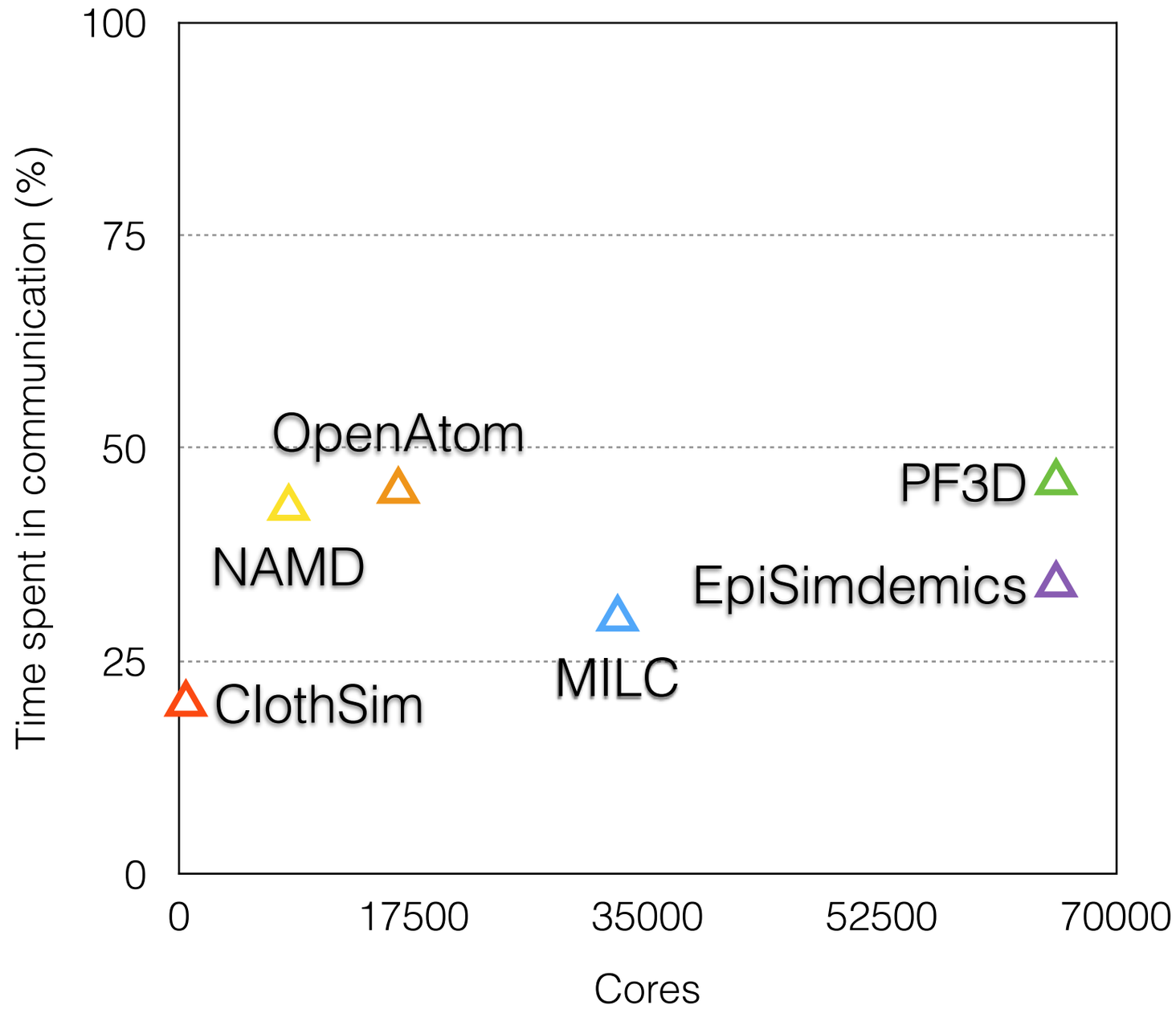
Understanding and Optimizing Communication Performance on HPC Networks

Contributors: Nikhil Jain, Abhinav Bhatele, Todd Gamblin, Xiang Ni, Michael Robson, Bilge Acun, Laxmikant Kale

University of Illinois at Urbana-Champaign
http://charm.cs.illinois.edu/~nikhil/

Communication in HPC

- A necessity, but can be viewed as an overhead
- Can consume half the execution time



Communication in HPC

Complex interplay of several components : hardware, configurable network properties, interaction patterns, algorithms…

As a user, **limited control** over environment and interference

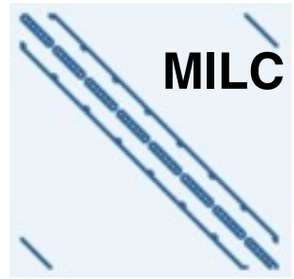
As an admin, how to **best use the system** while keeping users happy

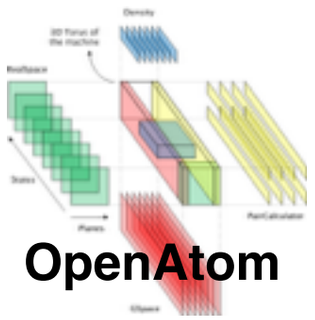
Communication in HPC

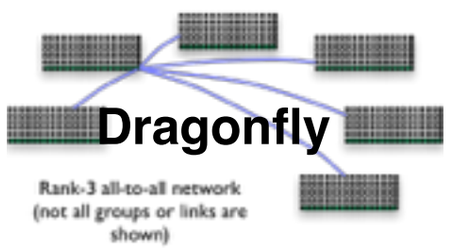
Complex interplay of several components : hardware, configurable network properties, interaction patterns, algorithms…

As a user, **limited control** over environment and interference

As an admin, how to **best use the system** while keeping users happy







Diverse apps

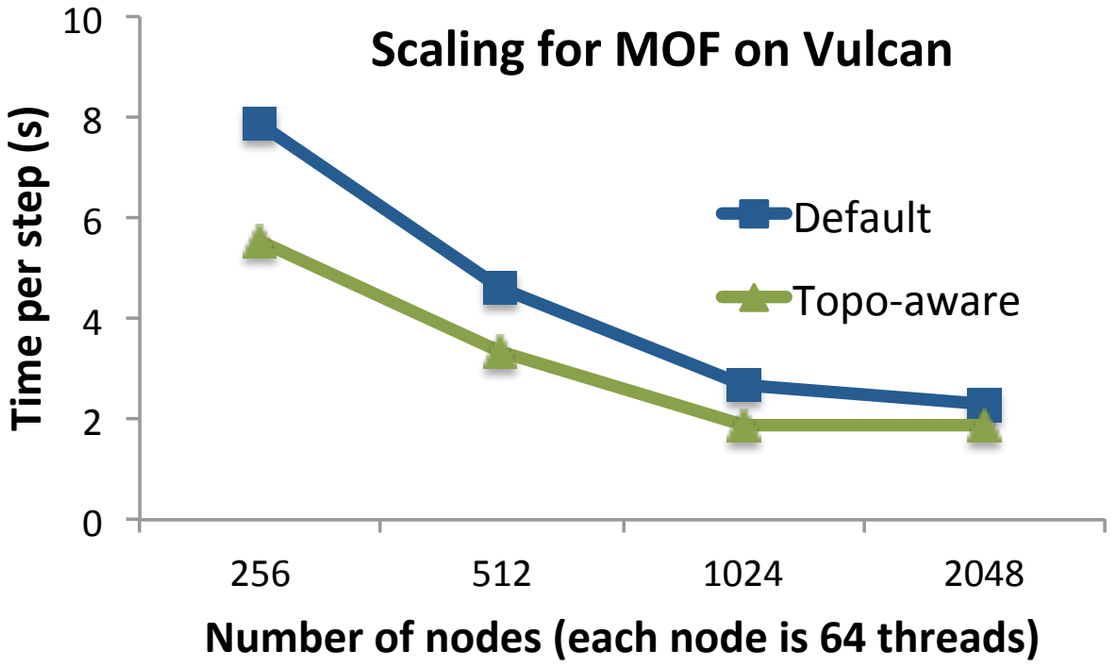
Many systems

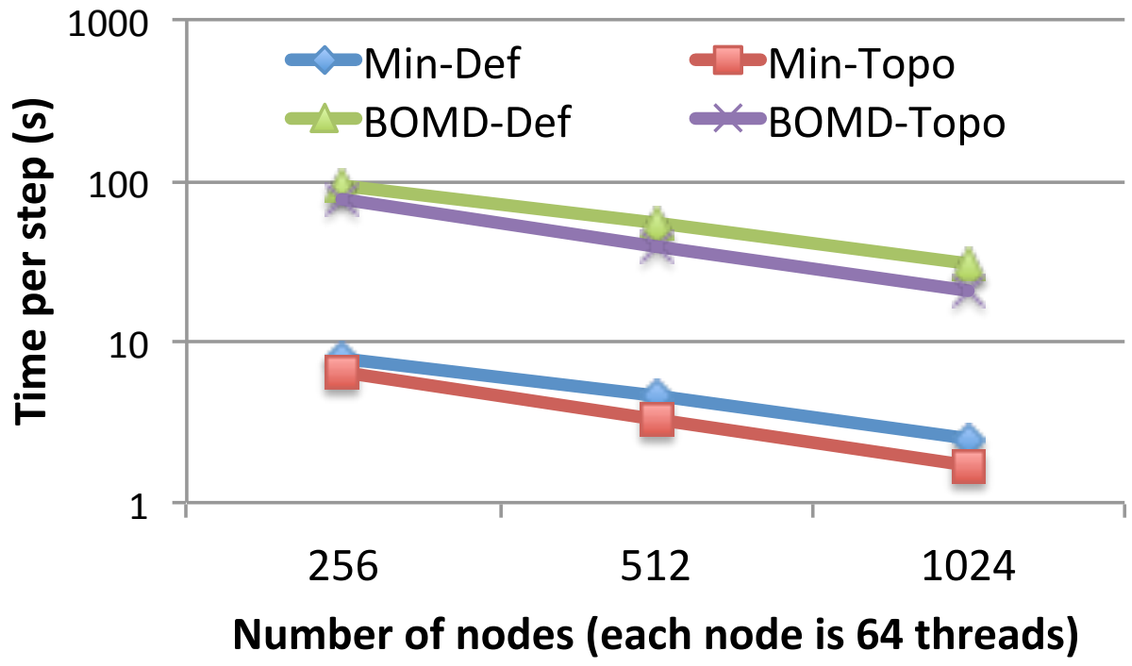
Topology Aware Mapping

- Profile applications for their communication graphs and map them
- Extremely important for Torus-based systems; ongoing work on other topologies

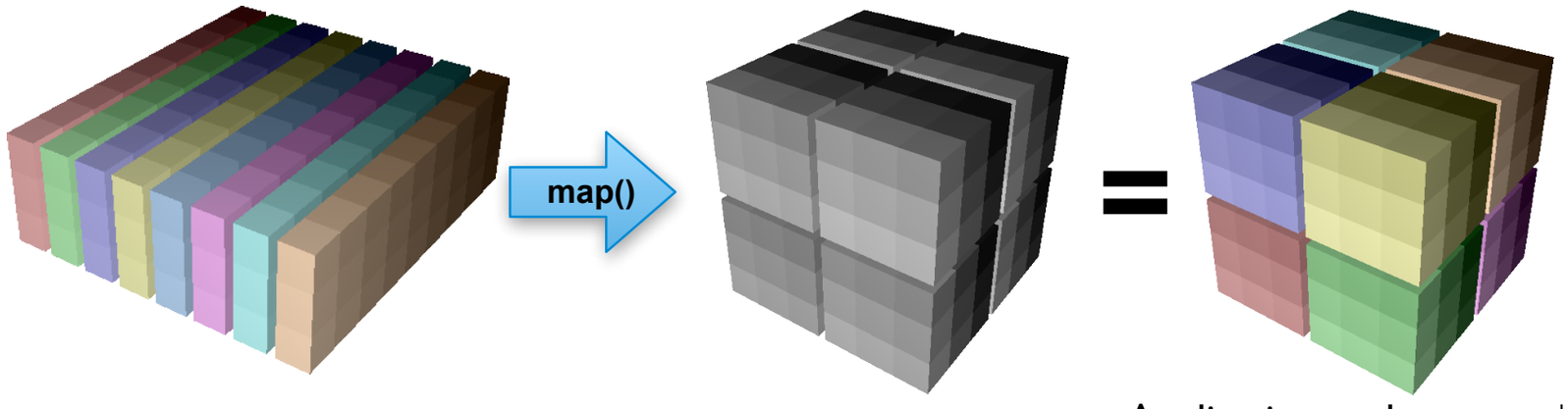
Topology Aware Mapping

- Profile applications for their communication graphs and map them
- Extremely important for Torus-based systems; ongoing work on other topologies
- Use Case: **OpenAtom**





Rubik - Python based tool to create maps

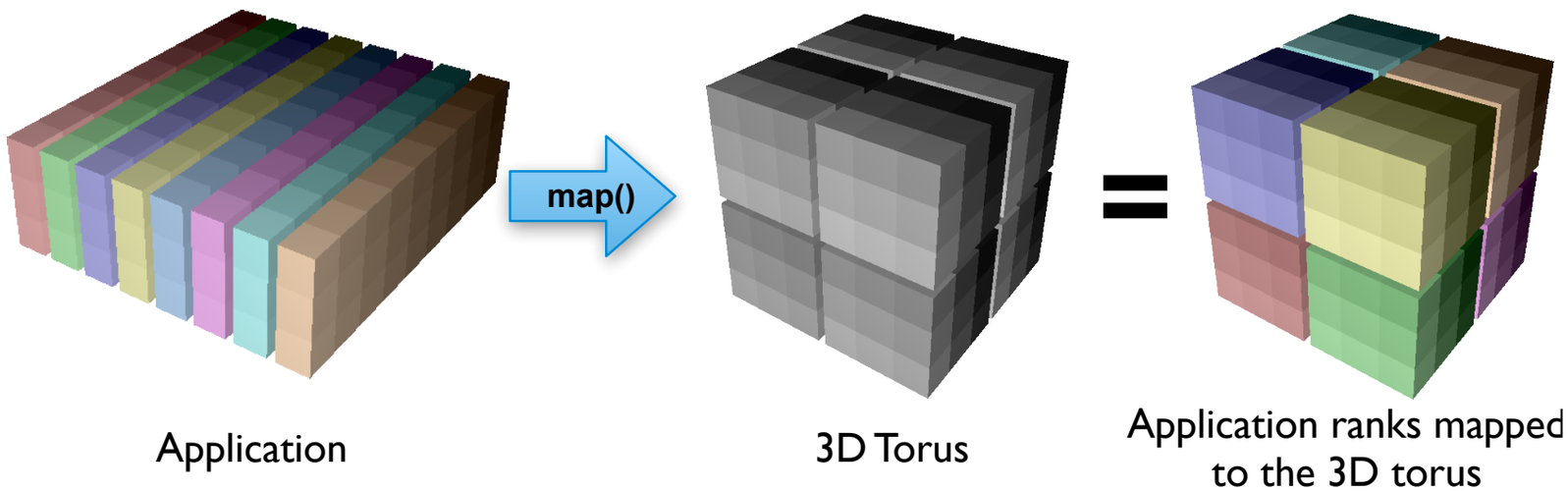


Application

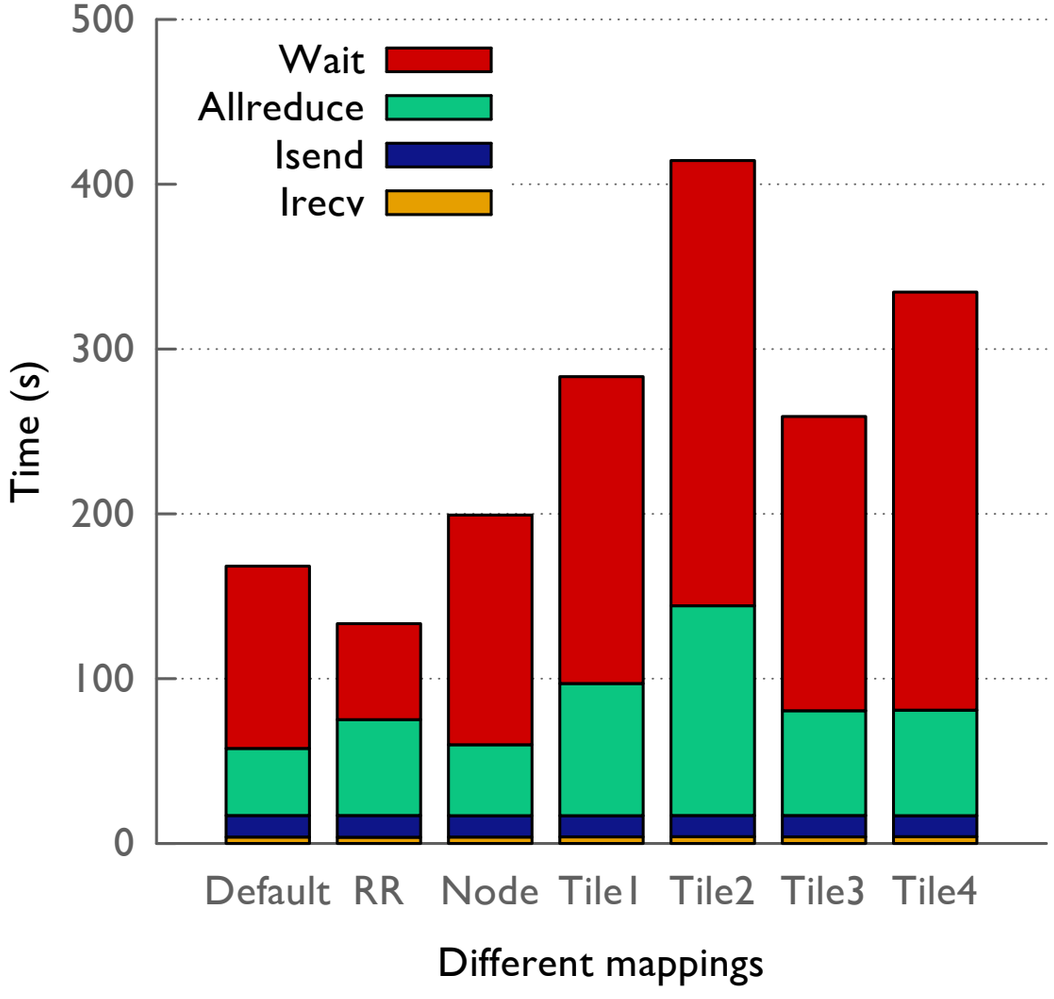
3D Torus

Application ranks mapped to the 3D torus

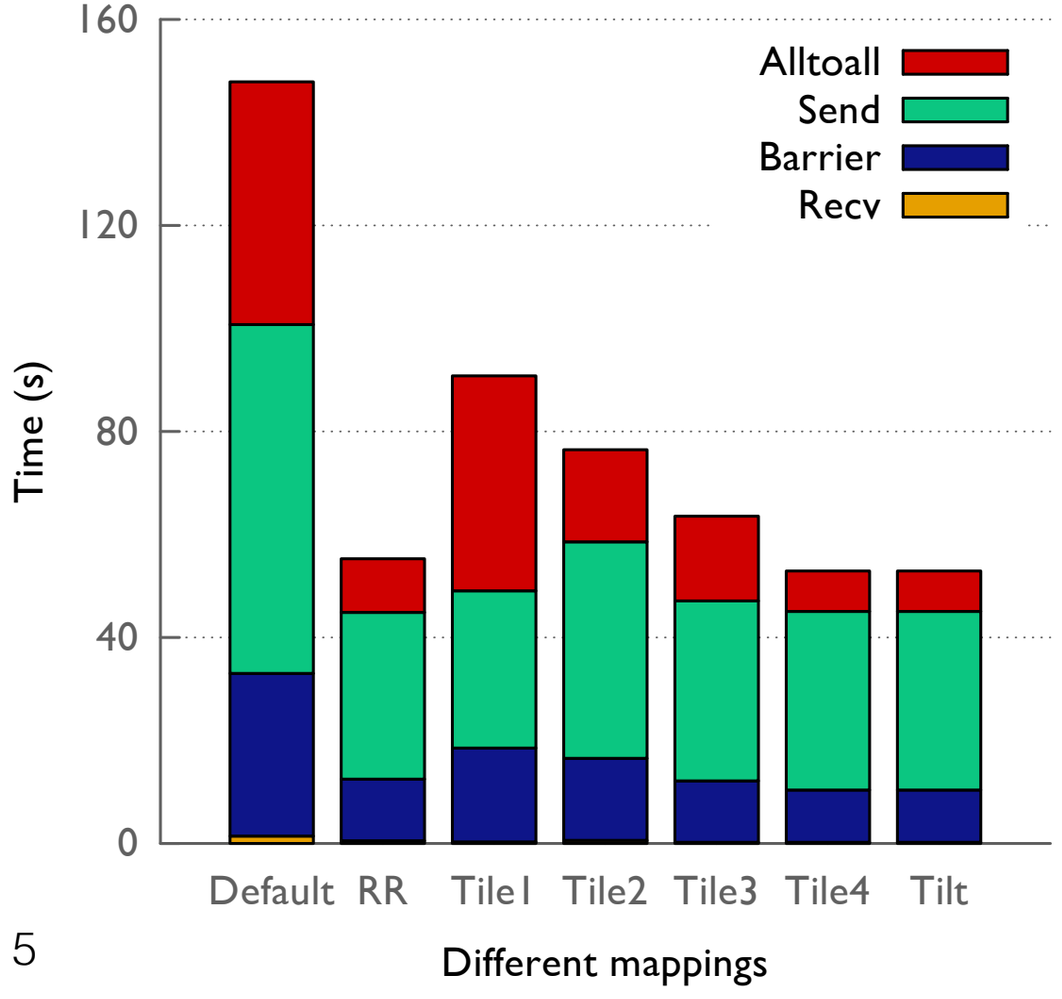
Rubik - Python based tool to create maps



MILC: Time spent in MPI calls on 4,096 nodes



pF3D: Time spent in MPI calls on 4,096 nodes



Understanding Networks

Understanding Networks

- What **determines** communication performance?
  - How can we predict it?
  - Quantification of metrics

Understanding Networks

- What **determines** communication performance?
  - How can we predict it?
  - Quantification of metrics
- What is the **relation** between performance and the entities quantified above?
  - Linear, higher polynomial, or indeterminate
  - Is statistical data related to performance?

Understanding Networks

- What **determines** communication performance?
  - How can we predict it?
  - Quantification of metrics
- What is the **relation** between performance and the entities quantified above?
  - Linear, higher polynomial, or indeterminate
  - Is statistical data related to performance?
- Method 1: **Supervised Learning**
  - More on this in Abhinav's talk

Method 2: **Packet-level Simulation**

Method 2: **Packet-level Simulation**

- Detailed study of what-if scenarios
- Comparison of similar systems

Method 2: **Packet-level Simulation**

- Detailed study of what-if scenarios
- Comparison of similar systems
- BigSim was among the earliest accurate packet-level HPC network simulator (circa 2004)
- **Reviving Emulation and Simulation capabilities of BigSim**
- BigSim + CODES + ROSS = TraceR
  - More on this in the Bilge's talk

Method 3: Modeling via Damselfly

Intermediate methods sufficient to answer certain types of questions

Q1: What is the best combination of routing strategies and job placement policies for single jobs?

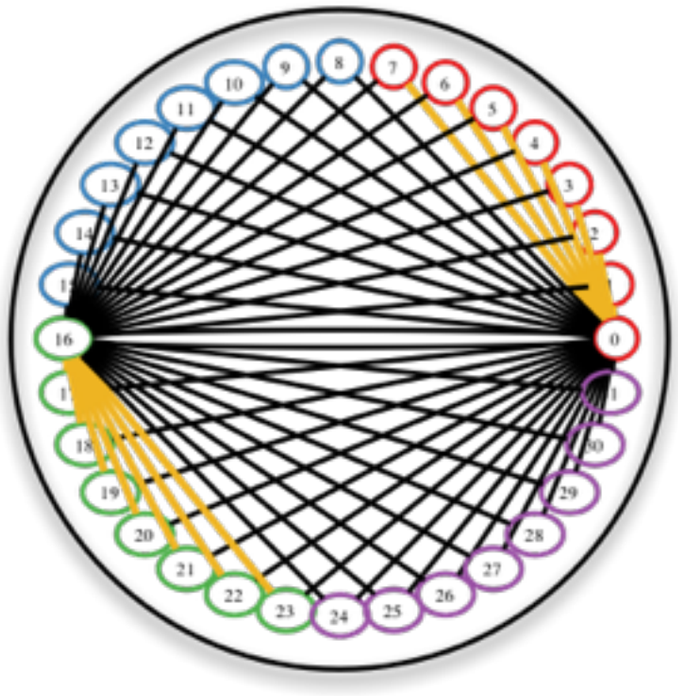
Q2: What is the best combination for parallel job workloads?

Q3: Should the routing policy be job-specific or system-wide?

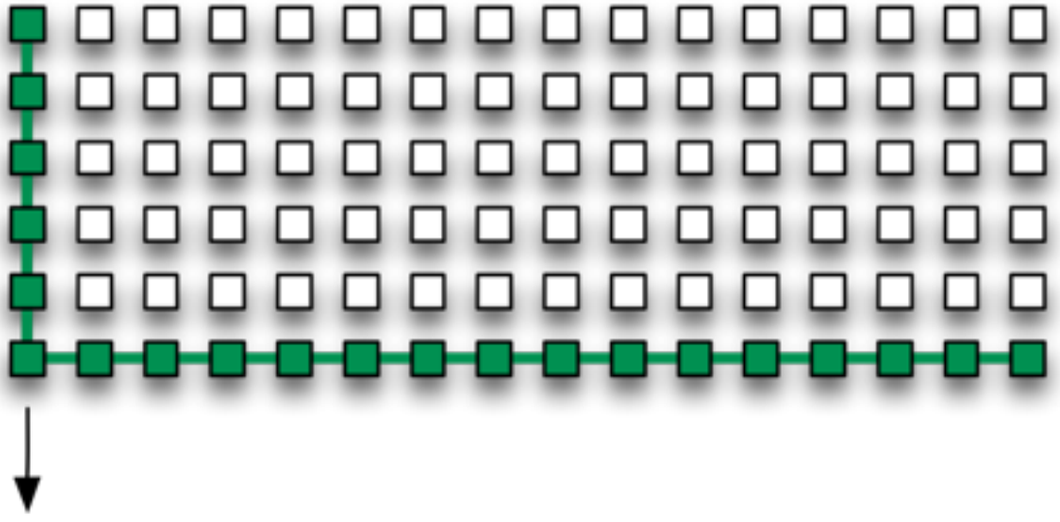
Dragonfly Topology

Level 1: Dense connectivity among routers to form groups

IBM PERCS



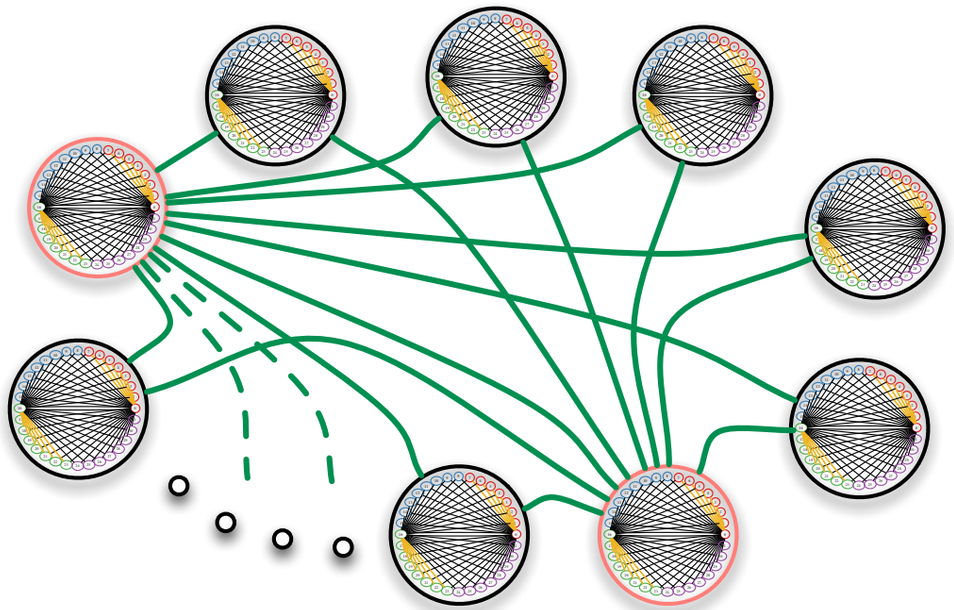
CRAY ARIES/XC30



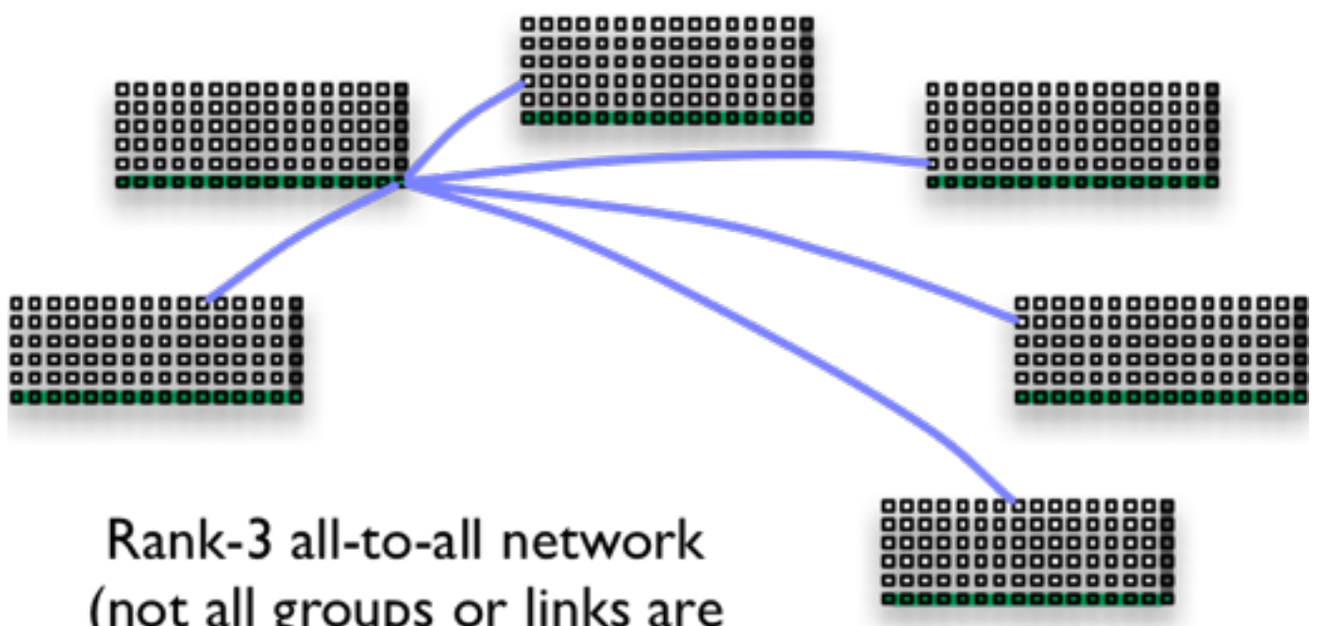
Dragonfly Topology

Level 2: Dense connectivity among groups as virtual routers

IBM PERCS



CRAY ARIES/XC30



Rank-3 all-to-all network
(not all groups or links are shown)

What needs to be evaluated?

Job Placement	Routing	Comm Kernel
Random Nodes (RDN)	Static Direct (SD)	UnStructured
Random Routers (RDR)	Static Indirect (SI)	2D Stencil
Random Chassis (RDC)	Adaptive Direct (AD)	4D Stencil
Random Group (RDG)	Adaptive Indirect (AI)	Many-to-many
Round Robin Nodes (RRN)	Adaptive Hybrid (AH)	Spread
Round Robin Routers (RRR)	Job-specific (JS)	Parallel Workloads (4)

Total cases ~ 360 for
8.8 million cores with 92,160 routers

Model for link utilization

- Input to the model:
    1. Network graph of Dragonfly routers
    2. Application communication graph for a communication step
    3. Job placement
    4. Routing strategy
- Output: The **steady-state traffic distribution on all network links**, which is representative of the network throughput
- Implemented as a **scalable parallel MPI program** executed on Blue Gene/Q
  — **Maximum runtime of 2 hours on 8,192 cores for prediction on 8.8 million cores**

- Initialize two copies of network graph N :
  $\mathbf{N^{Alloc}}$: stores total and per message allocated bandwidth (= 0)
  $\mathbf{N^{Remain}}$: stores bandwidth available for allocation (= capacity)

- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

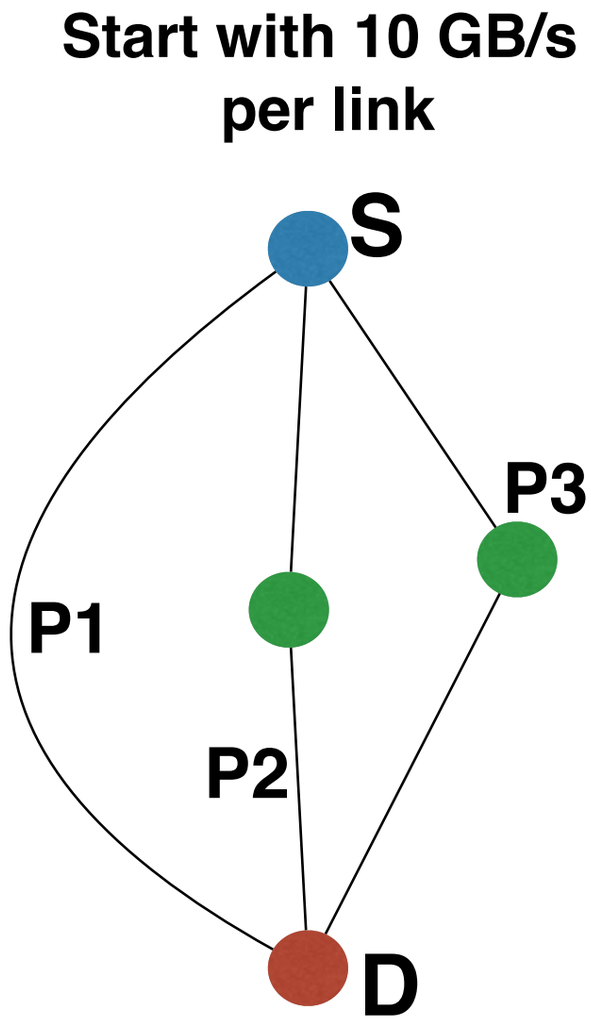
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)

Start with 10 GB/s per link

- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

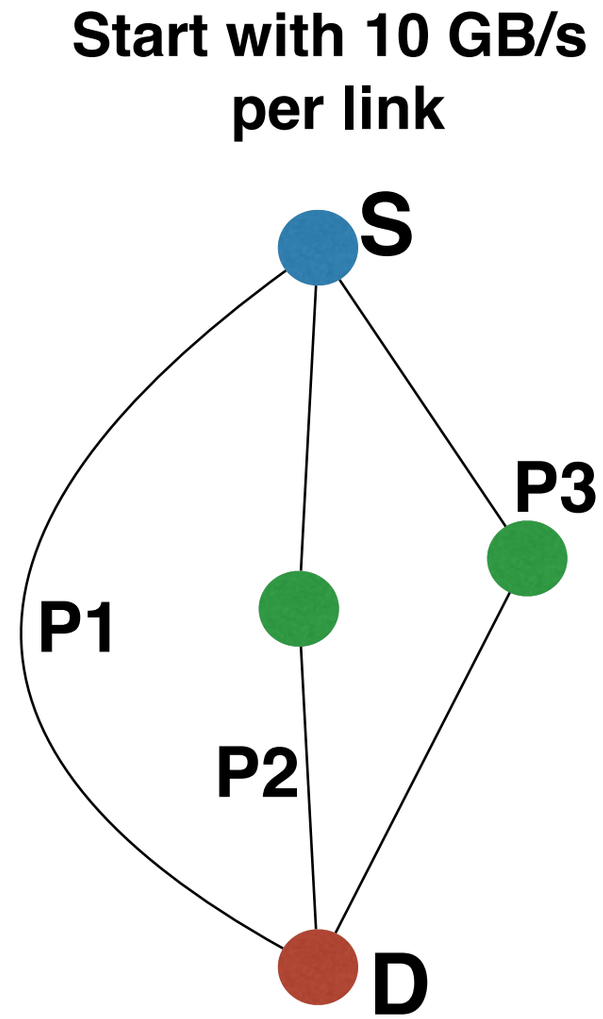
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

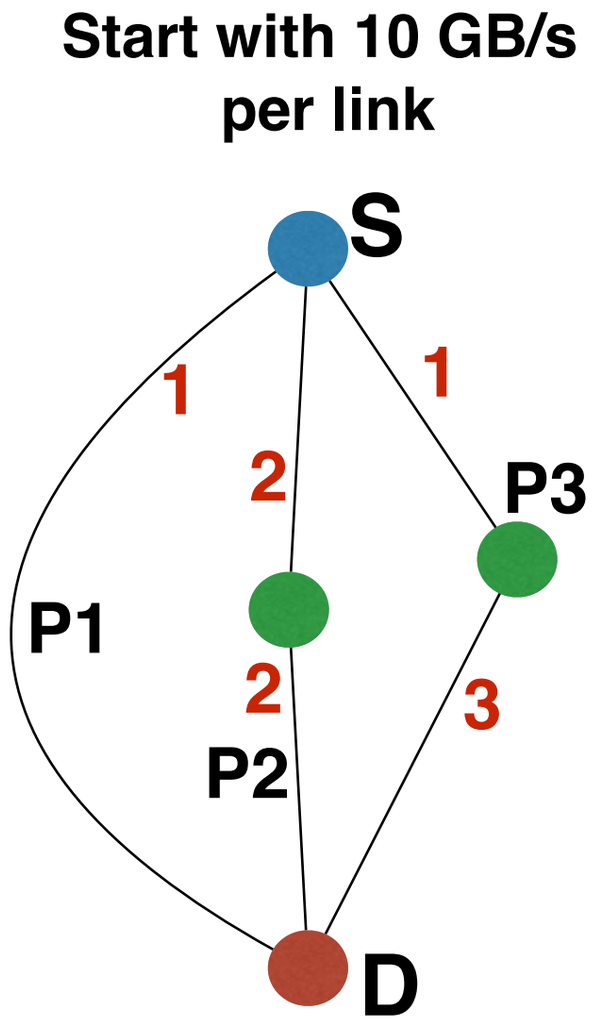
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

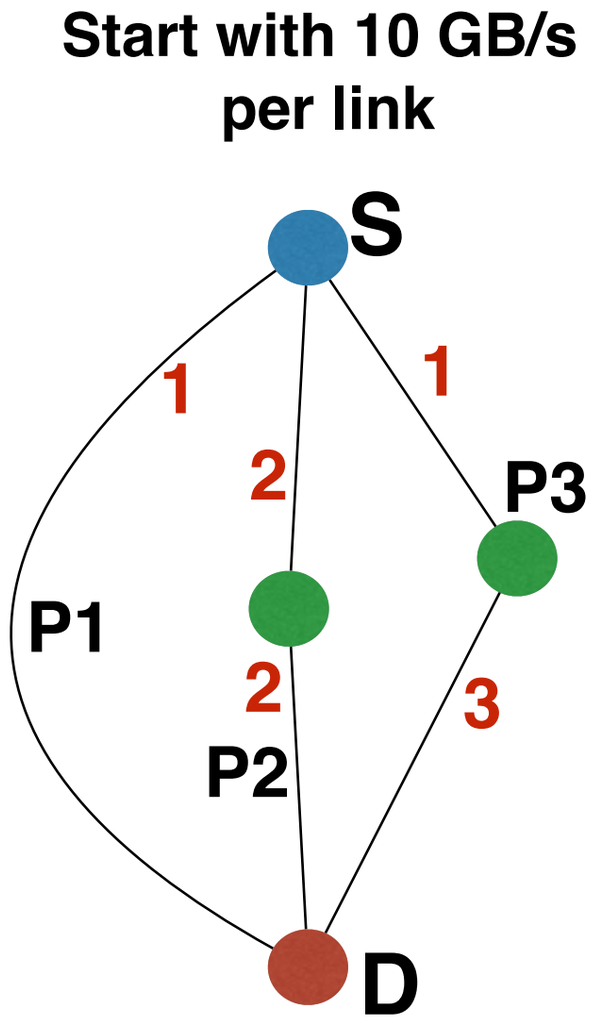
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

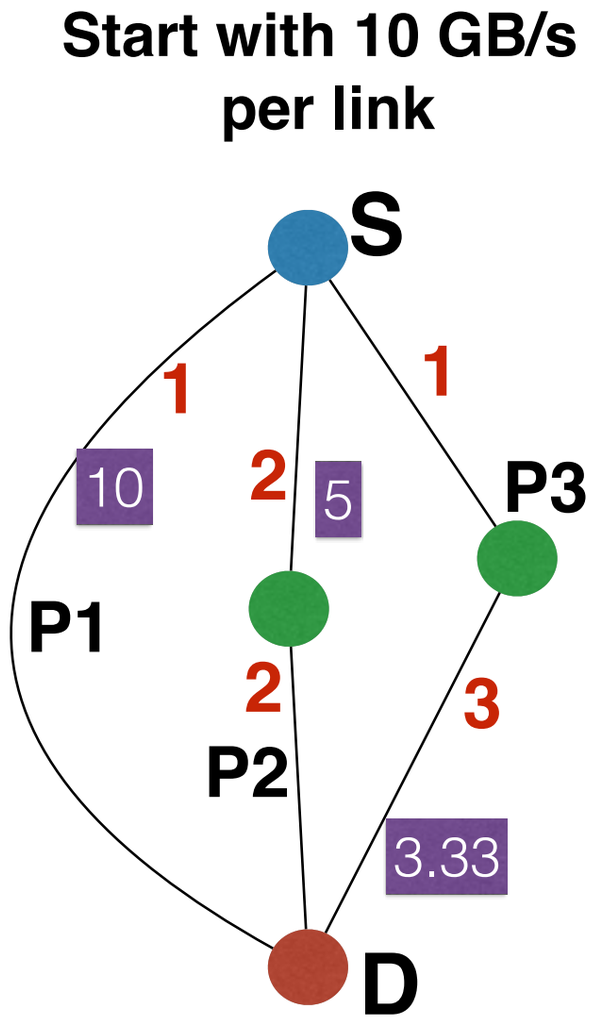
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link
  - for each path p in P(m), compute its availability using **N^{Remain}**



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

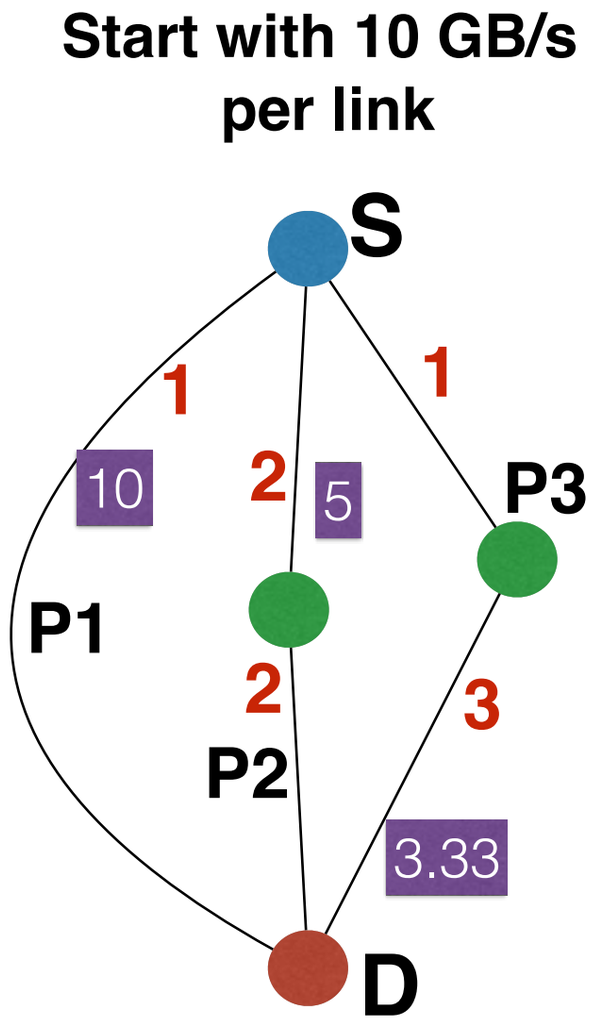
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link
  - for each path p in P(m), compute its availability using **N^{Remain}**



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

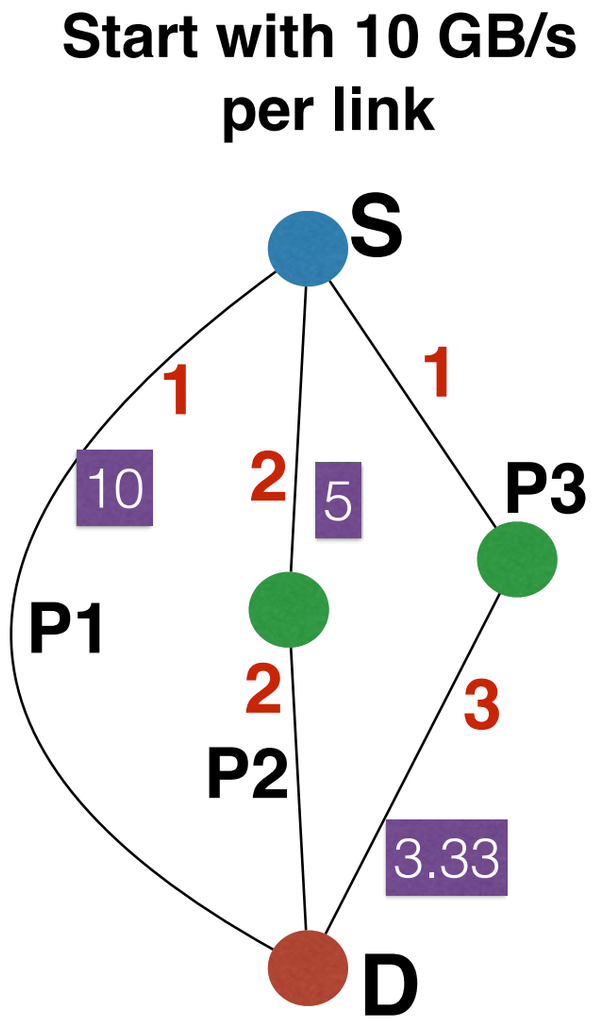
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link
  - for each path p in P(m), compute its availability using **N^{Remain}**
  - using availability, allocate more bandwidth to the messages



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

- Iterative solve for computing representative state **N^{Alloc}**

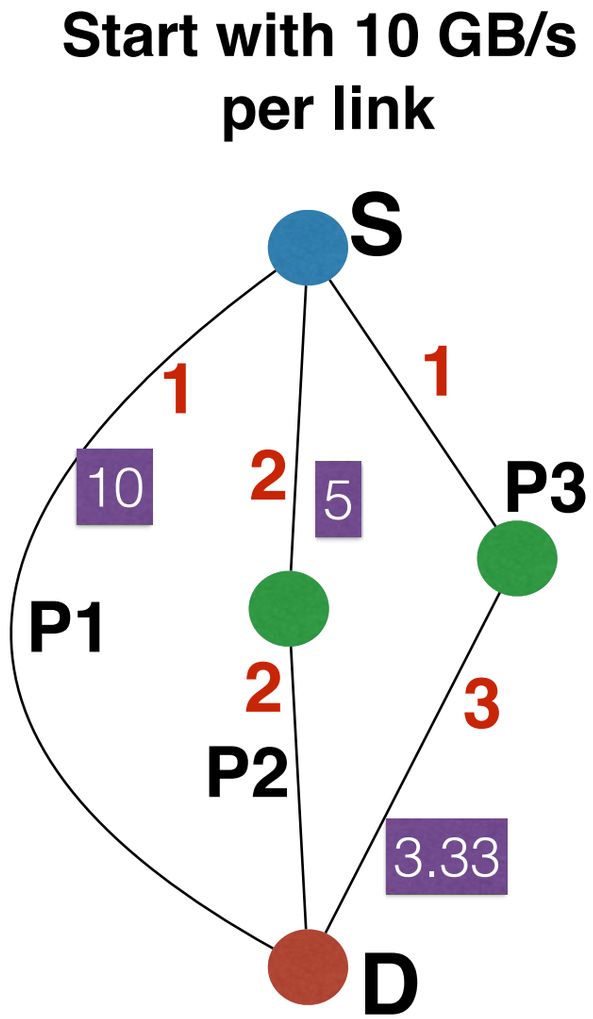
  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link
  - for each path p in P(m), compute its availability using **N^{Remain}**
  - using availability, allocate more bandwidth to the messages
  - update **N^{Alloc}** and **N^{Remain}** to to reflect the new allocations



- Initialize two copies of network graph N :
  N^{Alloc} : stores total and per message allocated bandwidth (= 0)
  N^{Remain} : stores bandwidth available for allocation (= capacity)

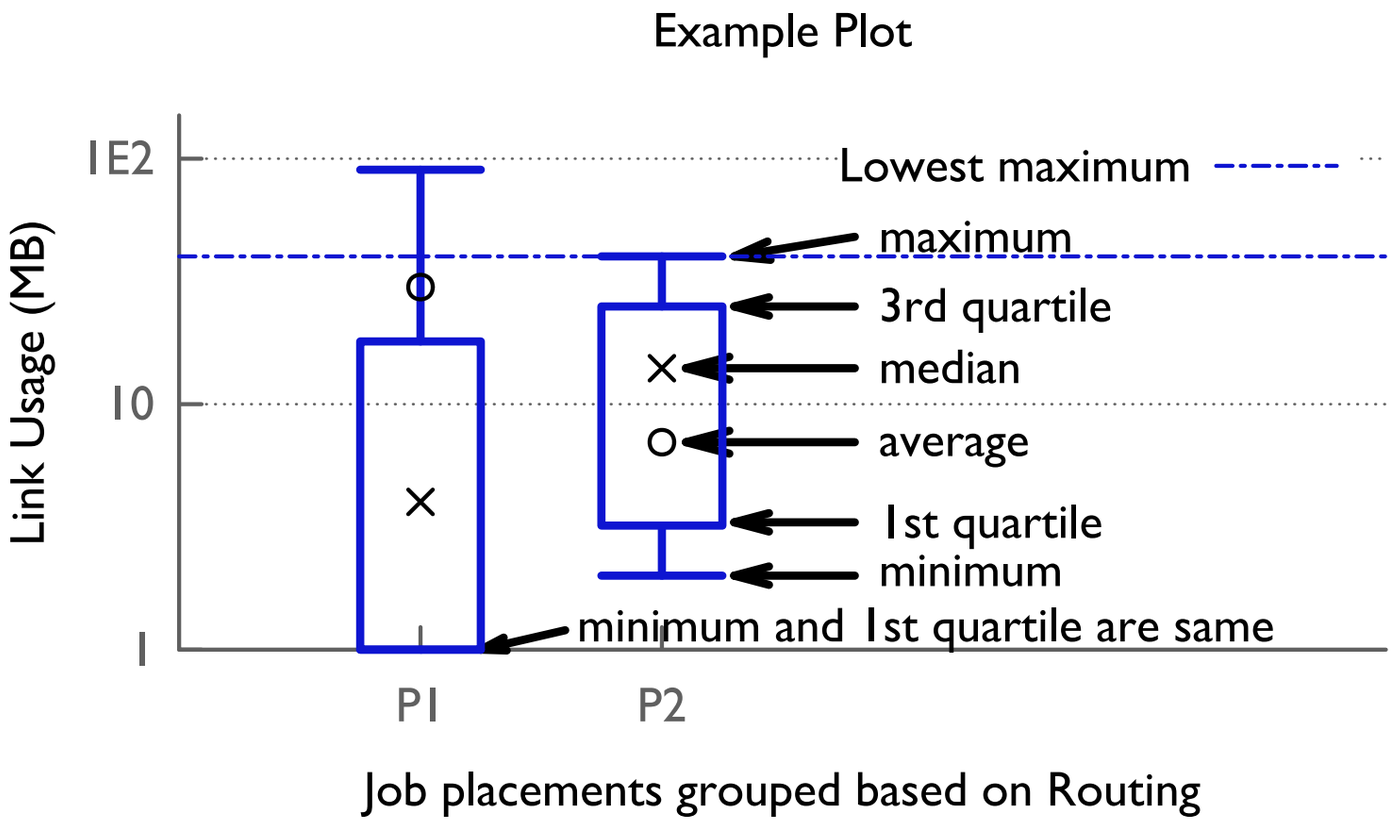
- Iterative solve for computing representative state **N^{Alloc}**

  while a message is allocated additional bandwidth
  - for each message m, obtain the list of paths P(m)
  - using P(m) of all messages, find the request count for each link
  - for each path p in P(m), compute its availability using **N^{Remain}**
  - using availability, allocate more bandwidth to the messages
  - update **N^{Alloc}** and **N^{Remain}** to to reflect the new allocations

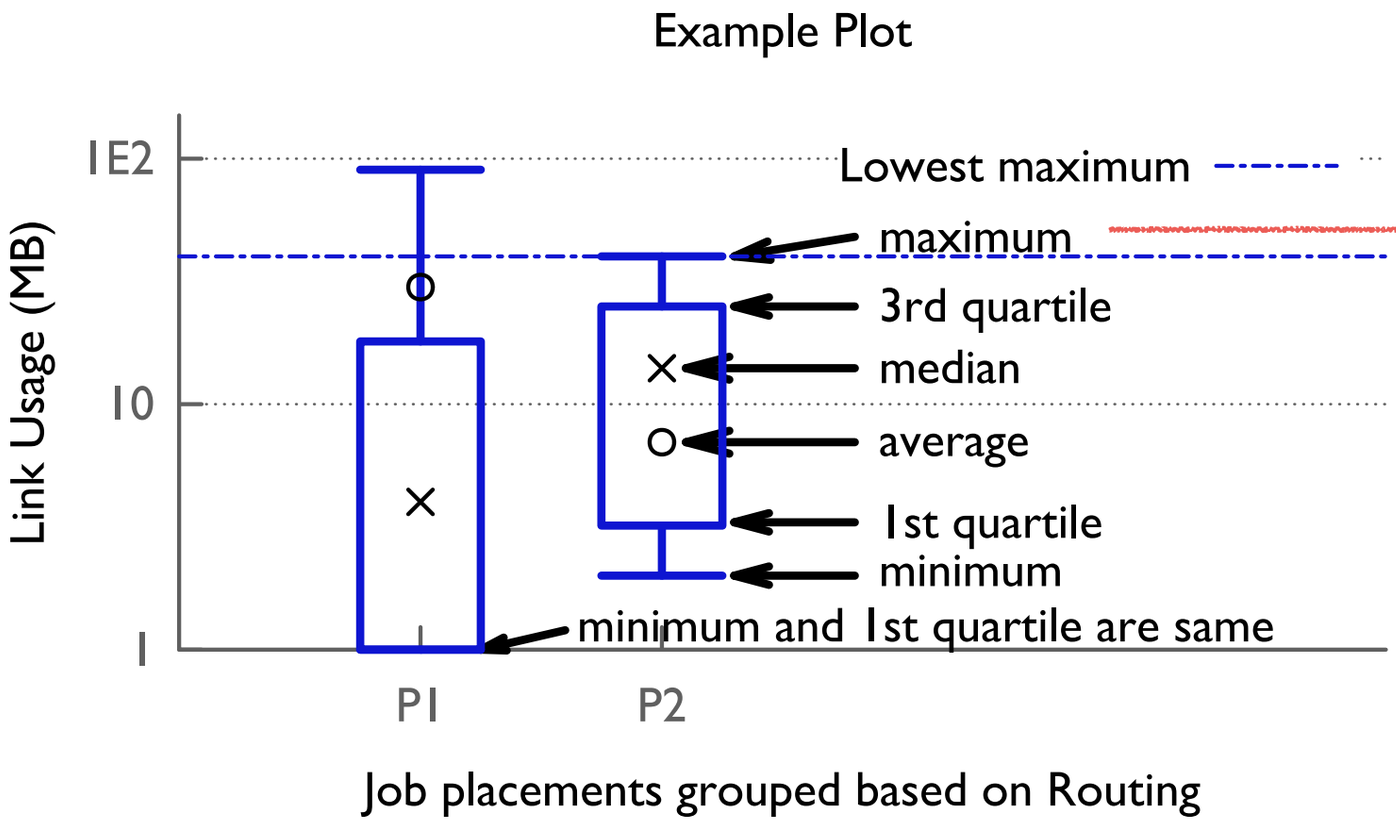


- Use **N^{Alloc}** to compute the distribution of bytes on the given links

How to read the plots?

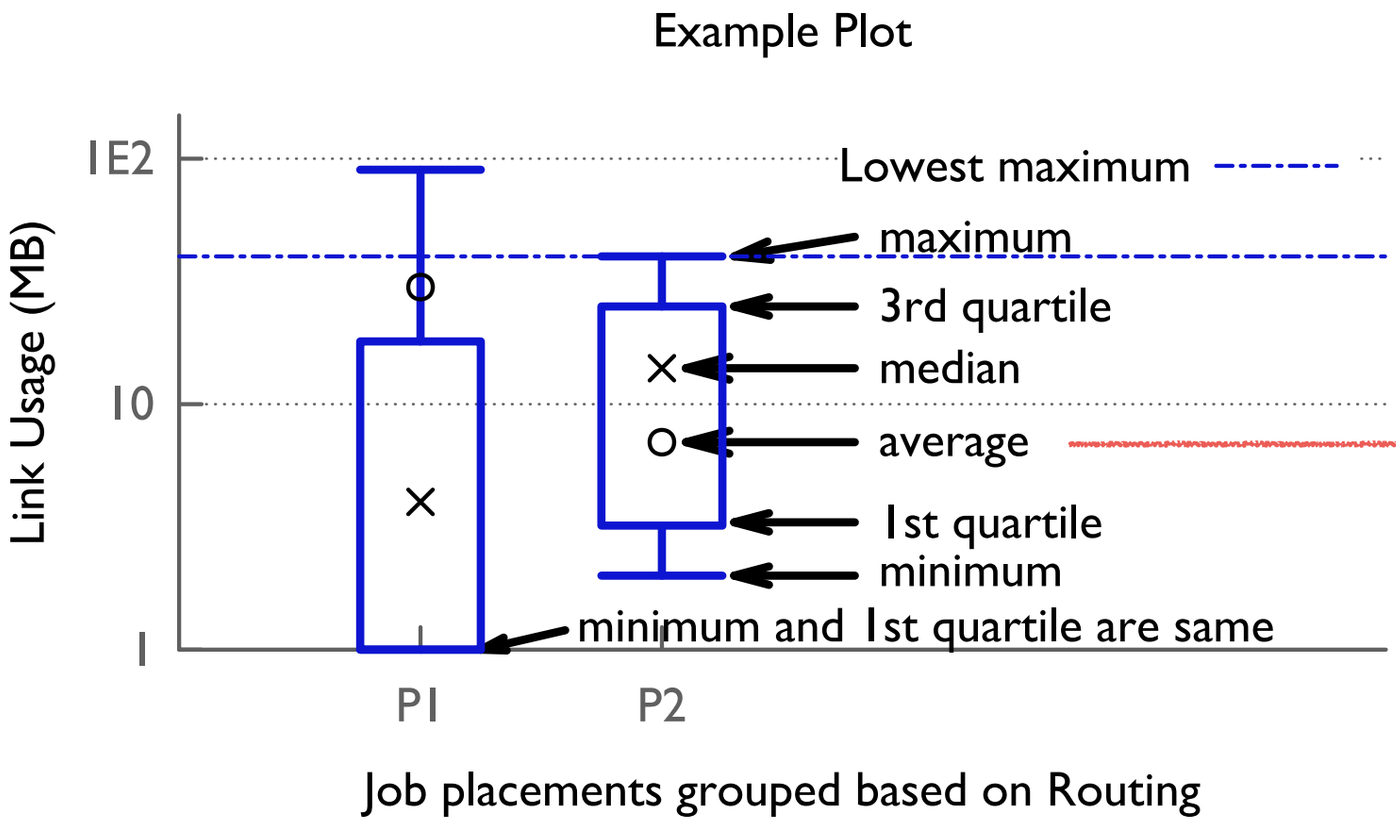


How to read the plots?



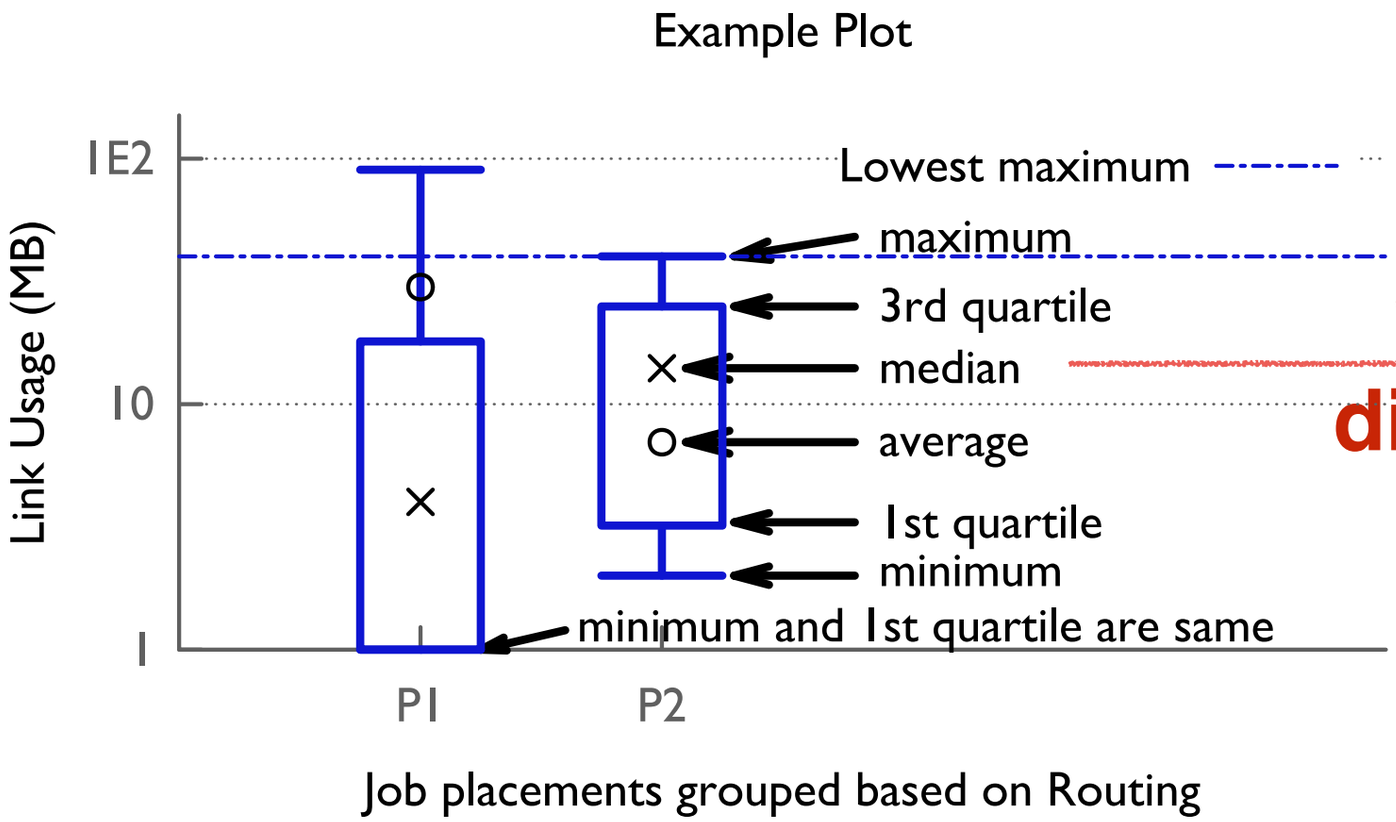
Maximum traffic on any link: indicates network hotspot

How to read the plots?



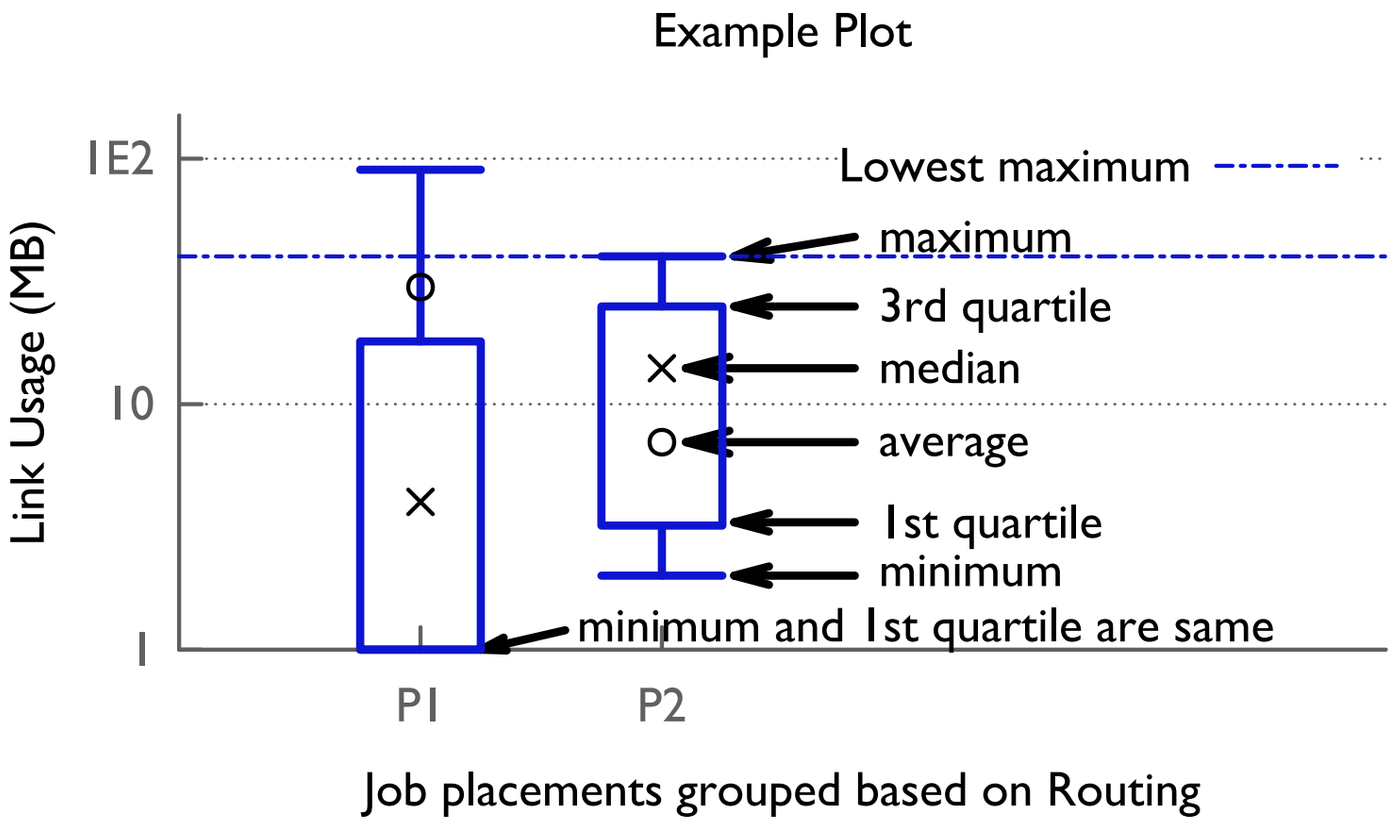
Average traffic on all links: indicates relative merit

How to read the plots?



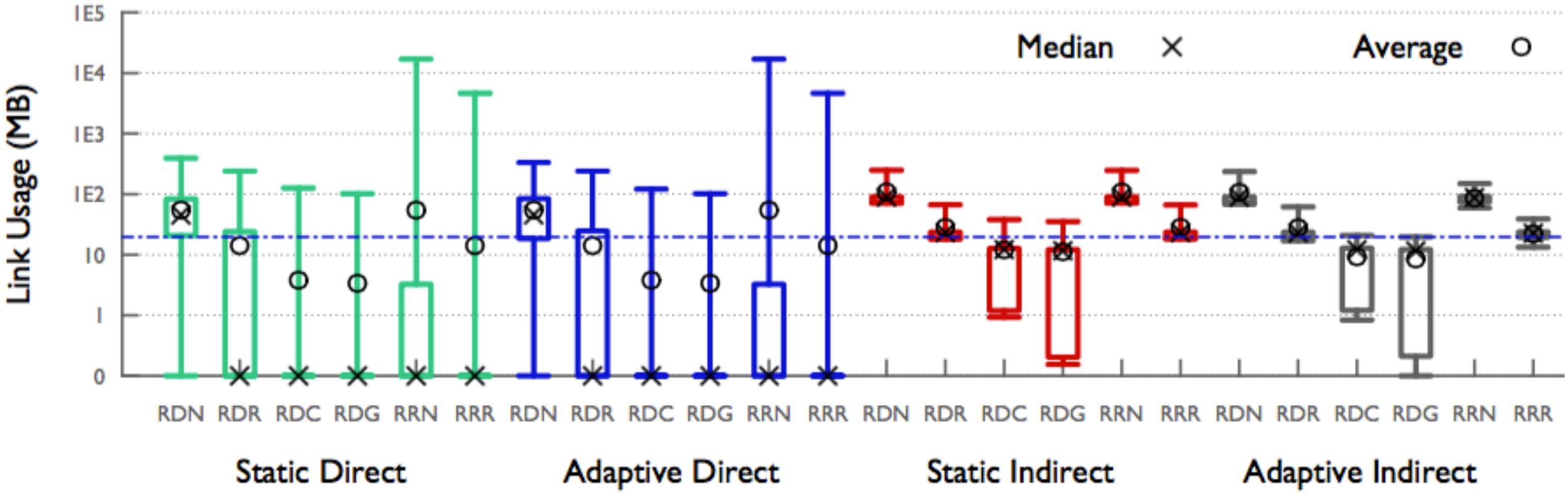
Median traffic: valuable for estimating distribution by comparing with the average

How to read the plots?



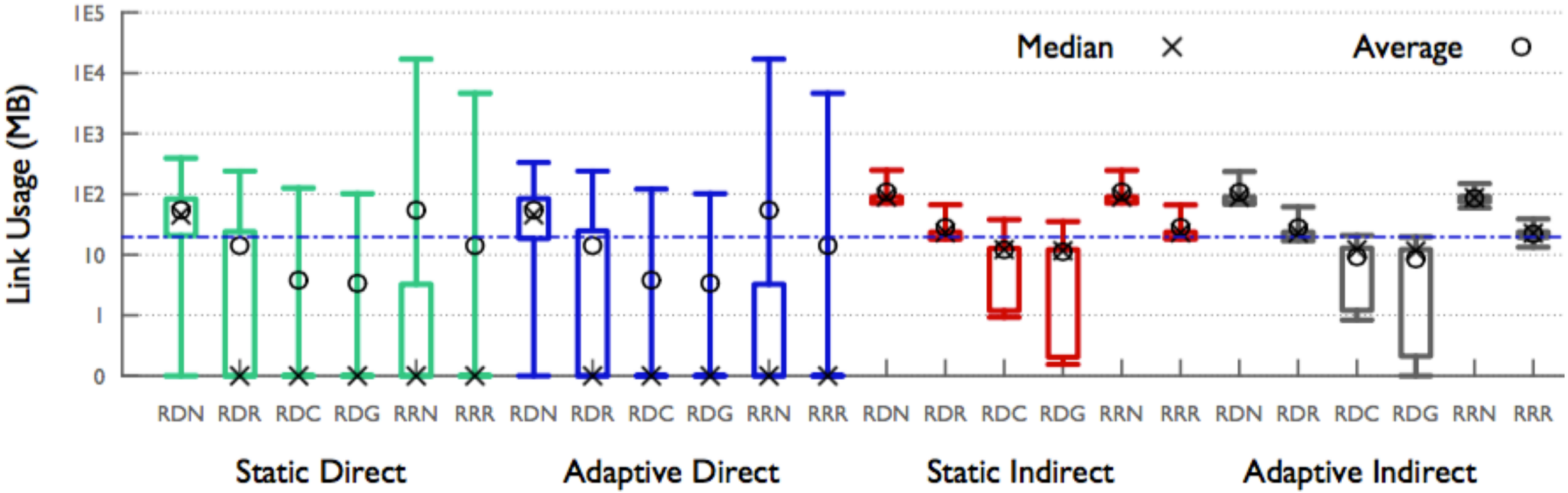
Ideal: distribution with close values for all data points, lower the better

Single job: Unstructured Mesh
6-20 partners with 512 KB messages



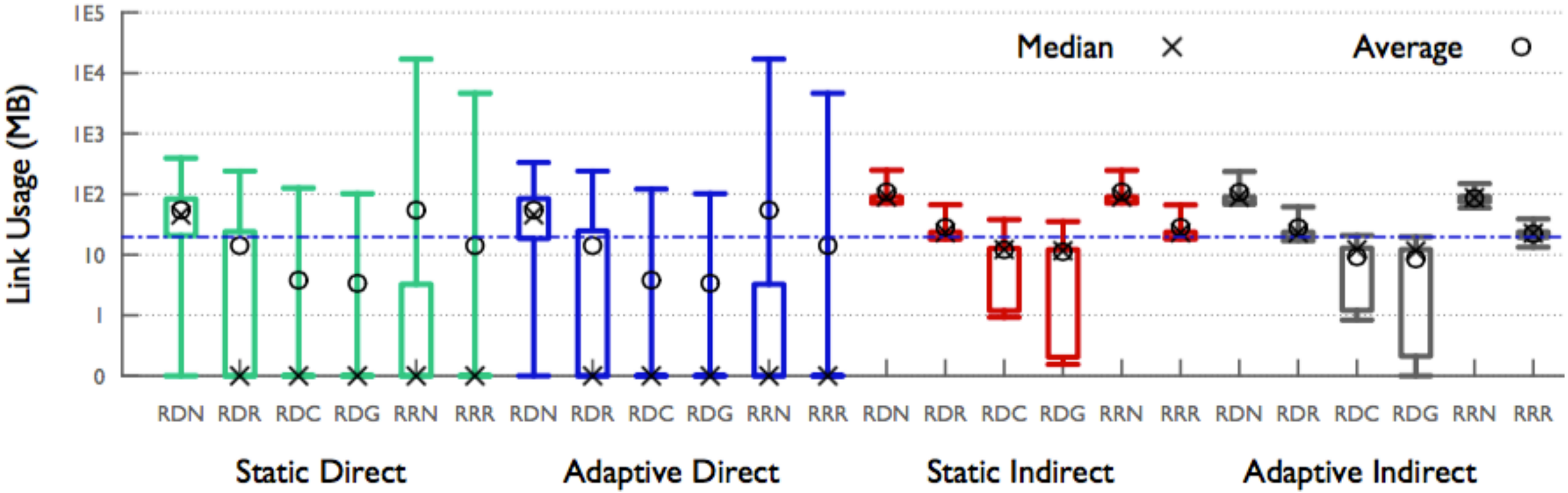
Job placement: blocking reduces
the maximum (up to 90% drop)
and average (up to 92% drop)

Single job: Unstructured Mesh
6-20 partners with 512 KB messages



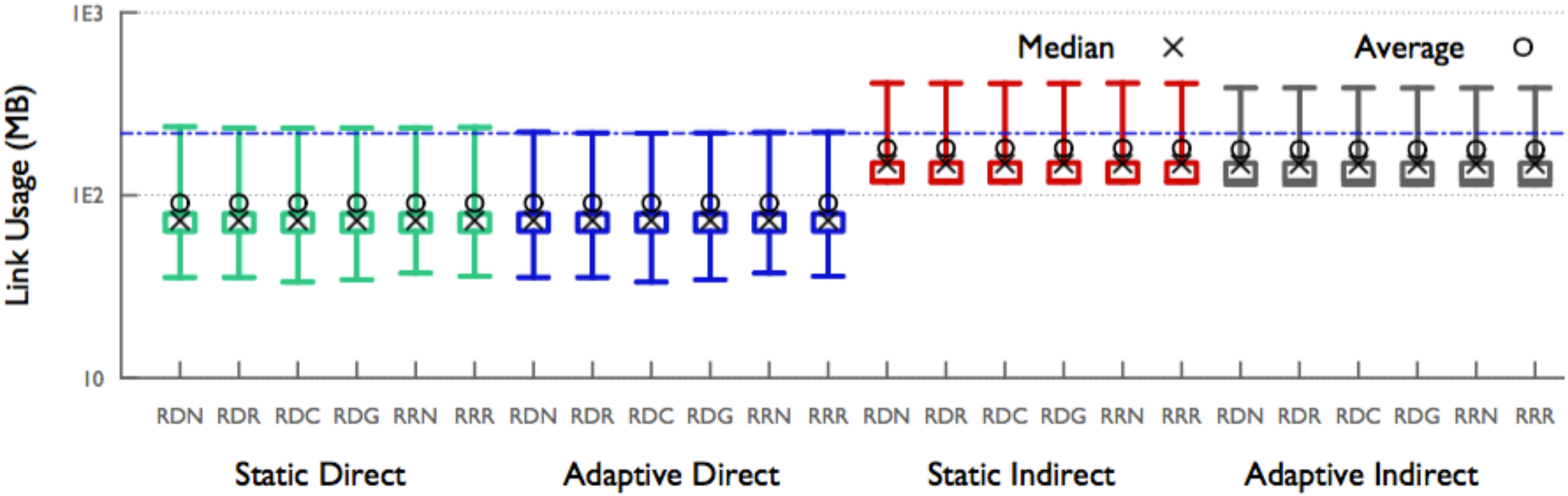
Indirect routing: increases average,
but reduces maximum by 50% in the best case

Single job: Unstructured Mesh
6-20 partners with 512 KB messages



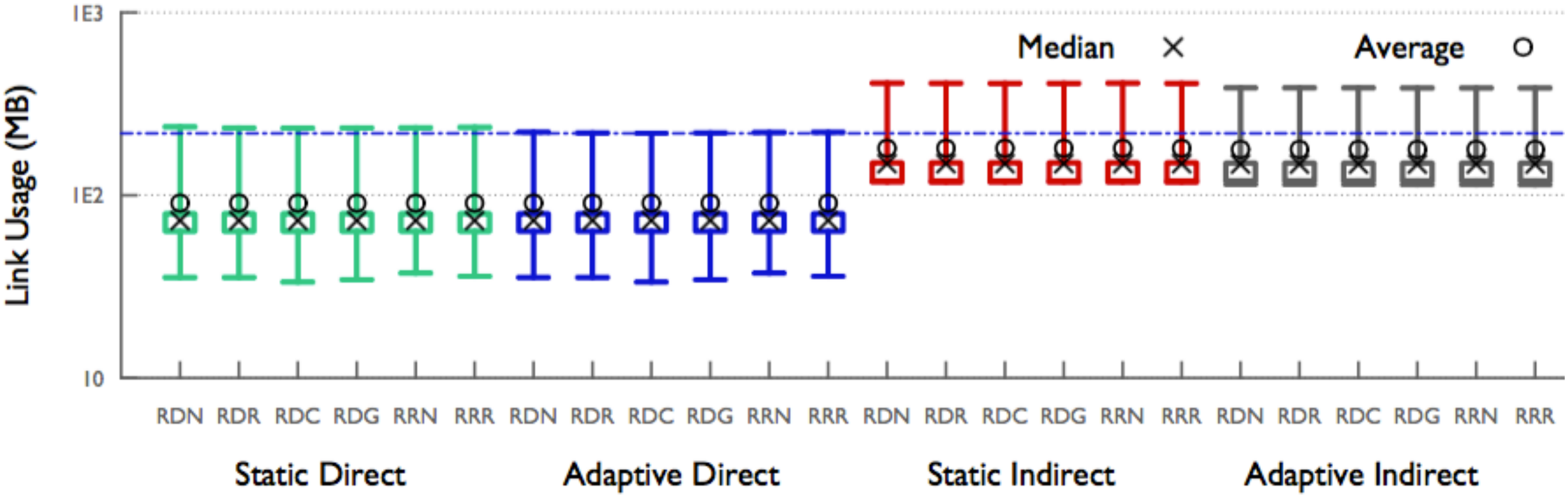
Adaptivity: similar distribution as static,
but with lower maximum
AI leads to 50% reduction in maximum traffic;
hybrid does worse than AI

Single job: Random Neighbors
6-20 partners with 512 KB messages



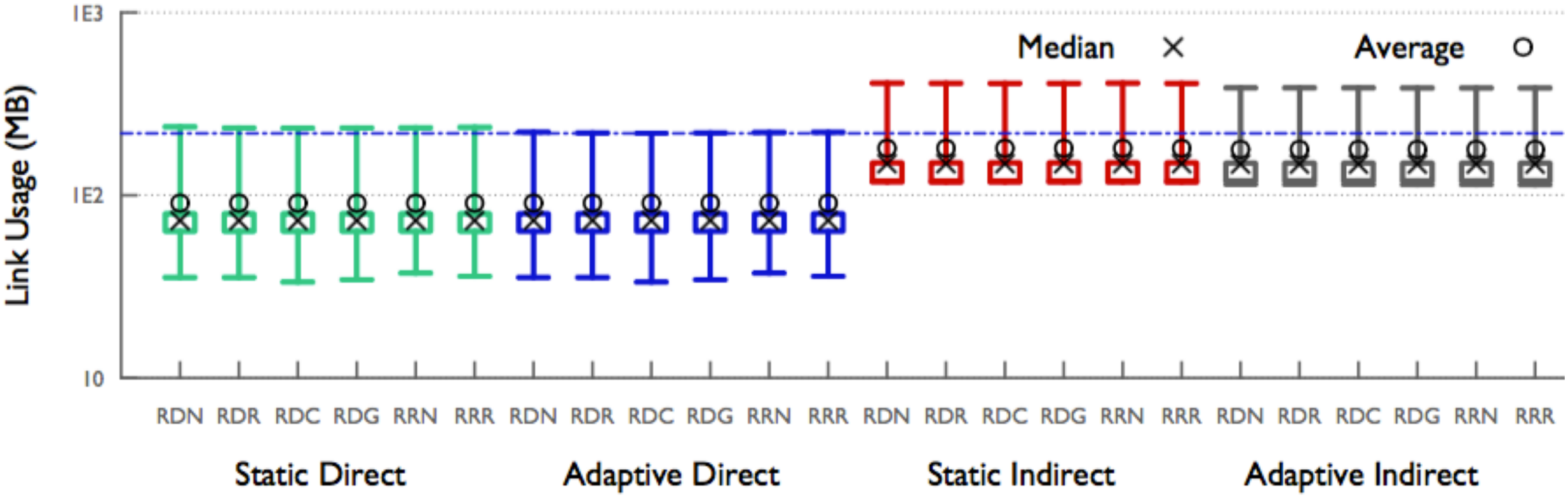
Job placement: negligible impact!

Single job: Random Neighbors
6-20 partners with 512 KB messages



Indirect routing: shifts the graph upwards and increases all quartiles;
100% increase in maximum and average

Single job: Random Neighbors
6-20 partners with 512 KB messages



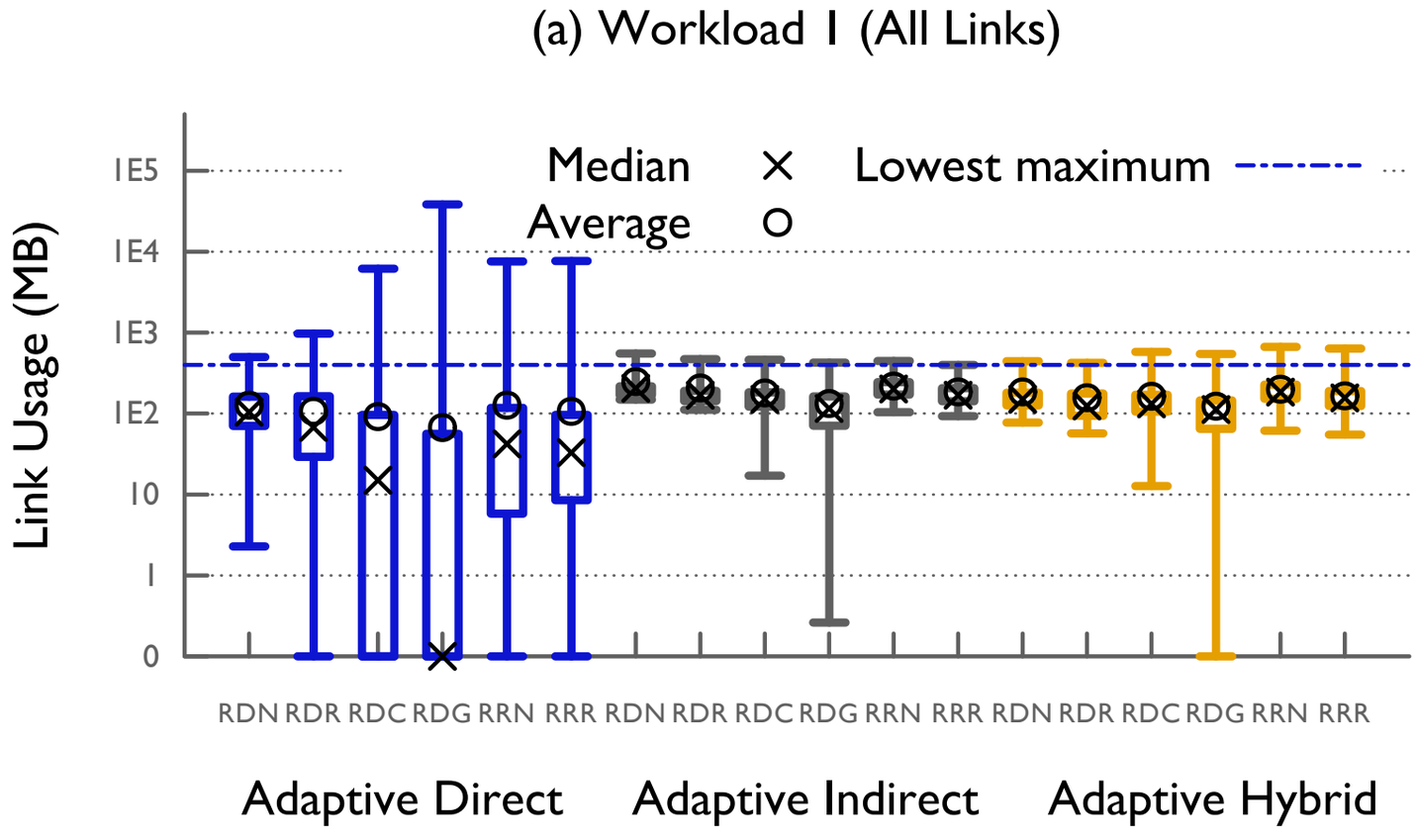
Adaptivity: Minor gains, 10% reduction in maximum
hybrid does better than AI

Parallel Workloads: % Core Distribution

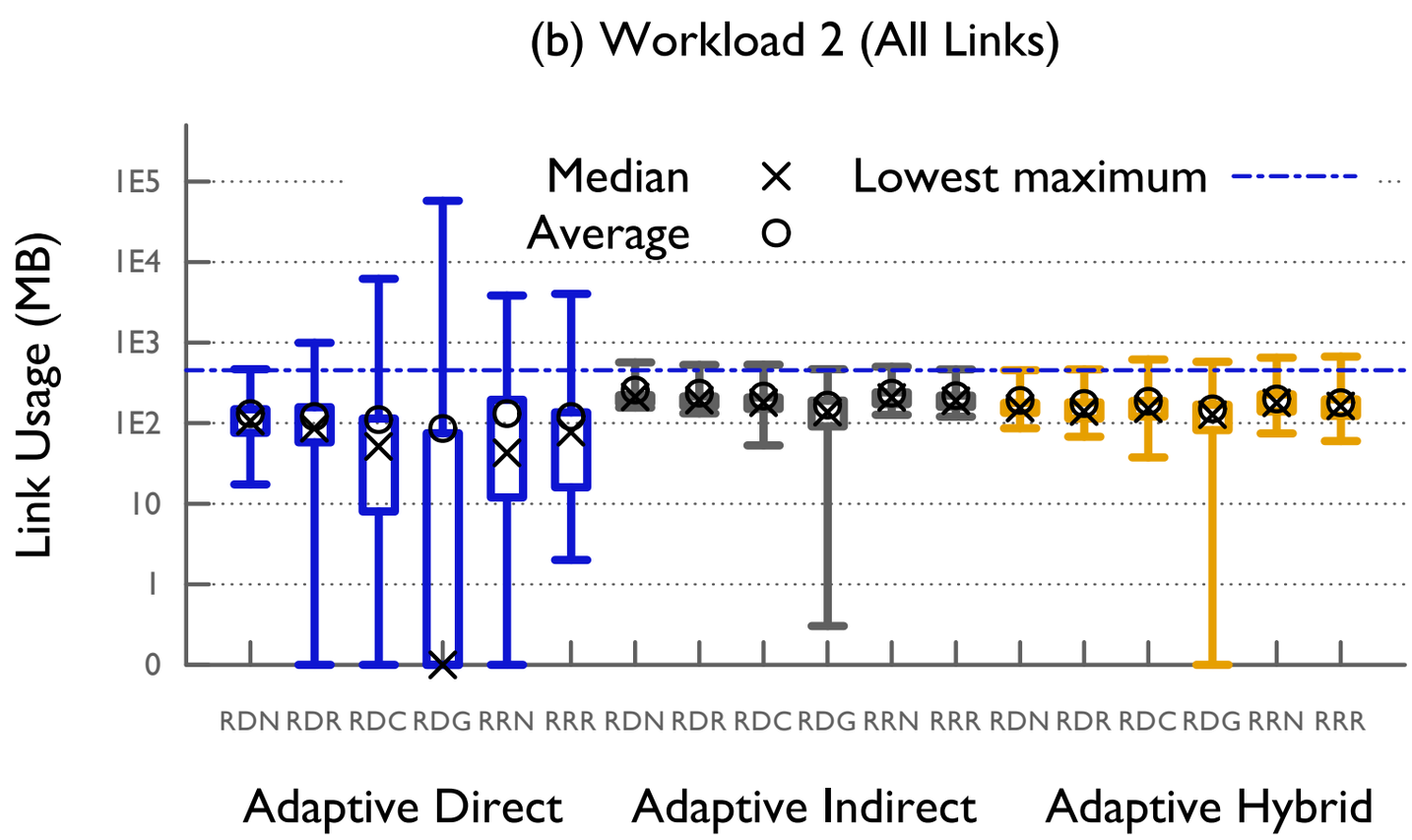
Comm Pattern	Workload 1	Workload 2	Workload 3	Workload 4
Unstructured Mesh	20	10	20	40
2D Stencil	10	10	40	10
4D Stencil	40	20	10	20
Many to many	20	40	10	20
Random neighbors	10	20	20	10

Workloads

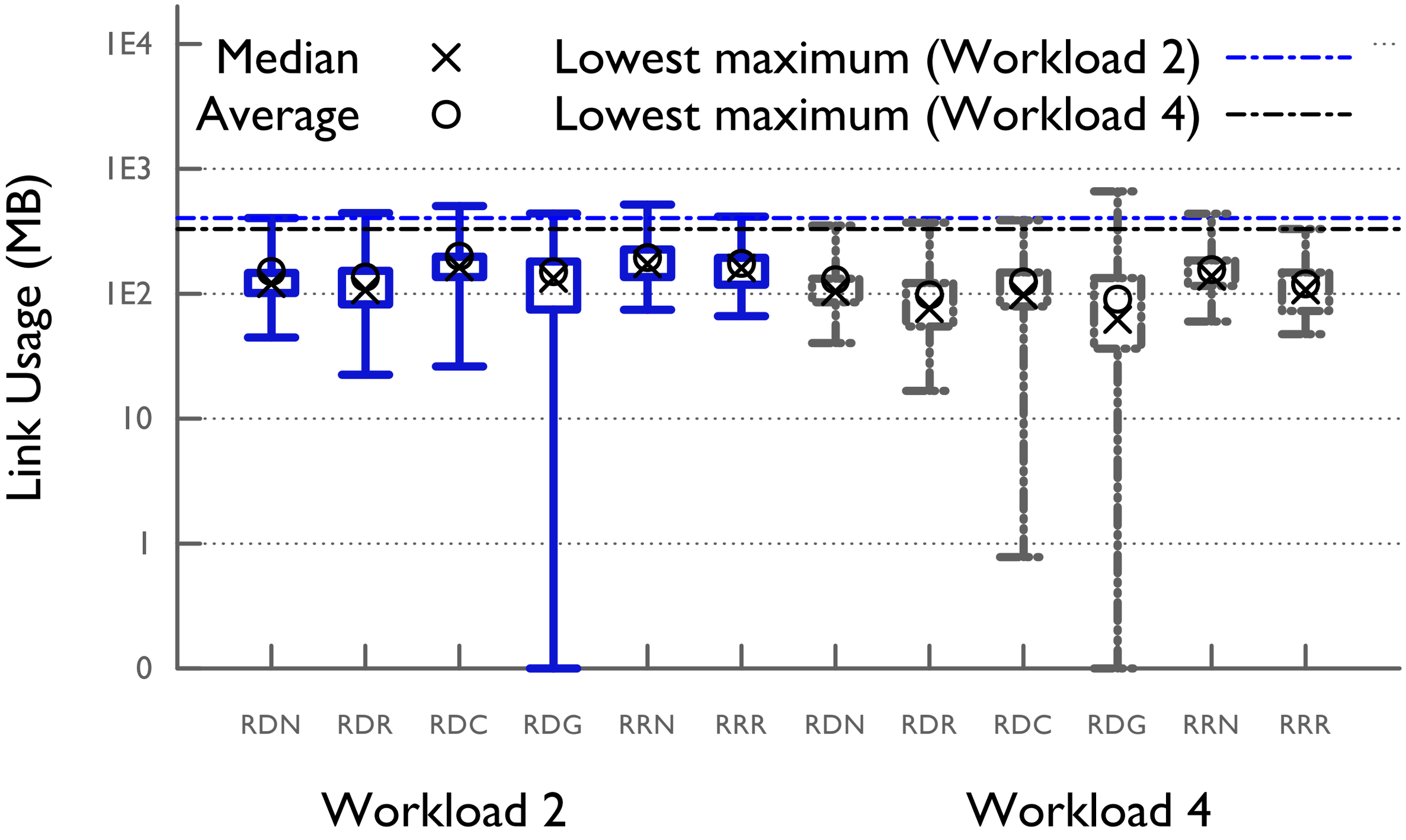
- Adaptivity reduces the maximum traffic by 35%



- Hybrid with RDN/ RDR shows lowest data points



Job-specific Routing



Summary

Summary

- Fast analytical model enables studies with a large number of scenarios

Summary

- Fast analytical model enables studies with a large number of scenarios
- Adaptivity results in significantly lower values for maximum and average traffic (up to 50% reduction)

Summary

- Fast analytical model enables studies with a large number of scenarios
- Adaptivity results in significantly lower values for maximum and average traffic (up to 50% reduction)
- **Q1.** What is the best combination for single job runs?
  - Depends on the job being run!
  - Patterns with communication among near-by MPI ranks benefit by blocking
  - Indirect routing is better when the communication pattern is not sufficiently spread by the application or job placement
  - Hybrid routing provides similar distribution as Adaptive Indirect, but its data points are shifted depending on the communication pattern

Summary

Summary

- **Q2.** What is the best combination for parallel workloads?
  - Similar distributions are observed irrespective of the jobs proportions in the workloads!
  - Adaptive Hybrid combines the best of both worlds
  - Randomized placement with node/router based blocking is good

Summary

- **Q2.** What is the best combination for parallel workloads?
  - Similar distributions are observed irrespective of the jobs proportions in the workloads!
  - Adaptive Hybrid combines the best of both worlds
    - Randomized placement with node/router based blocking is good
- **Q3.** Is it beneficial to use job-specific routing?
  - Yes, provides similar distribution as the best routing while reducing the values of the data points such as the maximum

Relevant publications

- Predicting application performance using supervised learning on communication features. SC 2013.

- Mapping to Irregular Torus Topologies and Other Techniques for Petascale Biomolecular Simulation. SC 2014.

- Maximizing Network Throughput on the Dragonfly Interconnect. SC 2014.

- Improving Application Performance via Task Mapping on IBM Blue Gene/Q. HiPC 2014.

- Identifying the Culprits behind Network Congestion. IPDPS 2015.