Hybrid Packet/Circuit Switched Datacenter Network: Promises and Challenges

Guohui Wang

IBM T.J. Watson Research Center

in collaboration with Rice, Intel Labs, CMU and UCSD

Big data for modern applications

Scientific data, 200GB images per day

Business data, > 6.5 petabytes

Multimedia data, > 27 petabytes per month in 2006

World wide web, 20 petabytes processed per day,

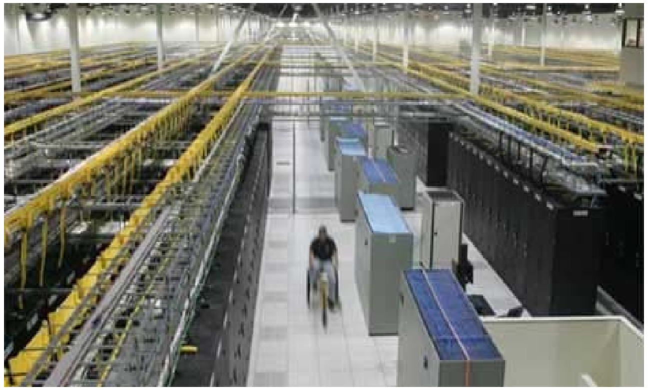
1 exabyte of storage under construction

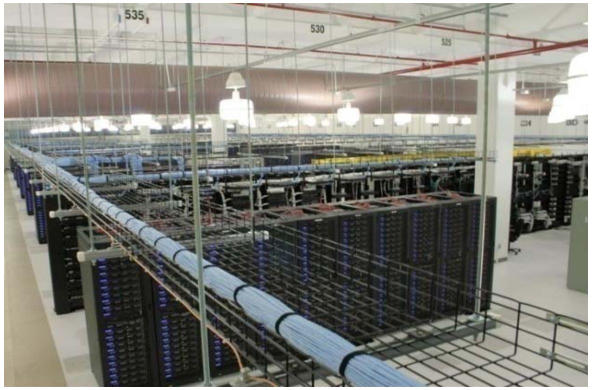
1 exabyte ~= 1000, 000, 000, 000, 000, 000 bytes,
a stack of 1TB disks higher than 15 miles.

Company logos and data are from Internet websites.

Big data in big data centers

Outside: larger than a football field



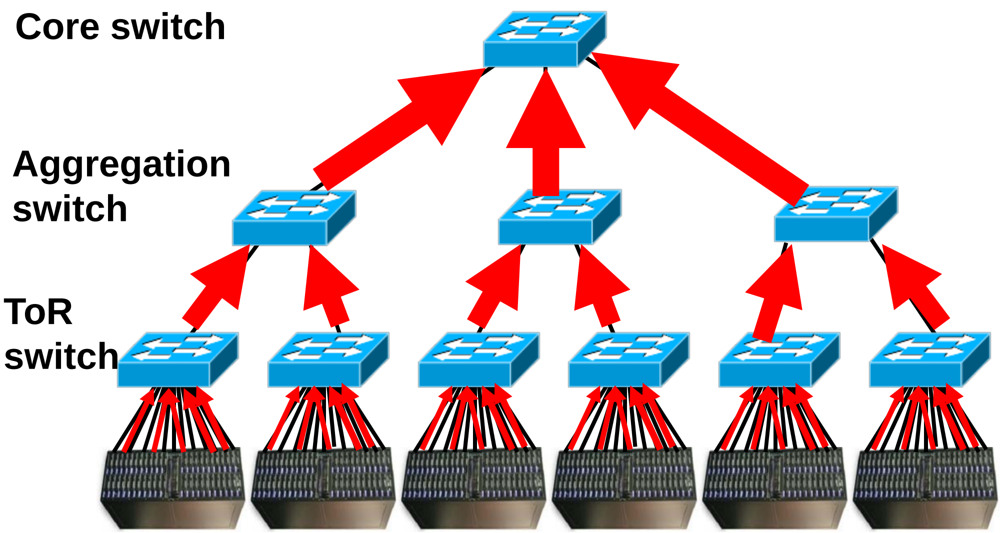


Inside: thousands of server racks

Data center pictures are from Internet websites.

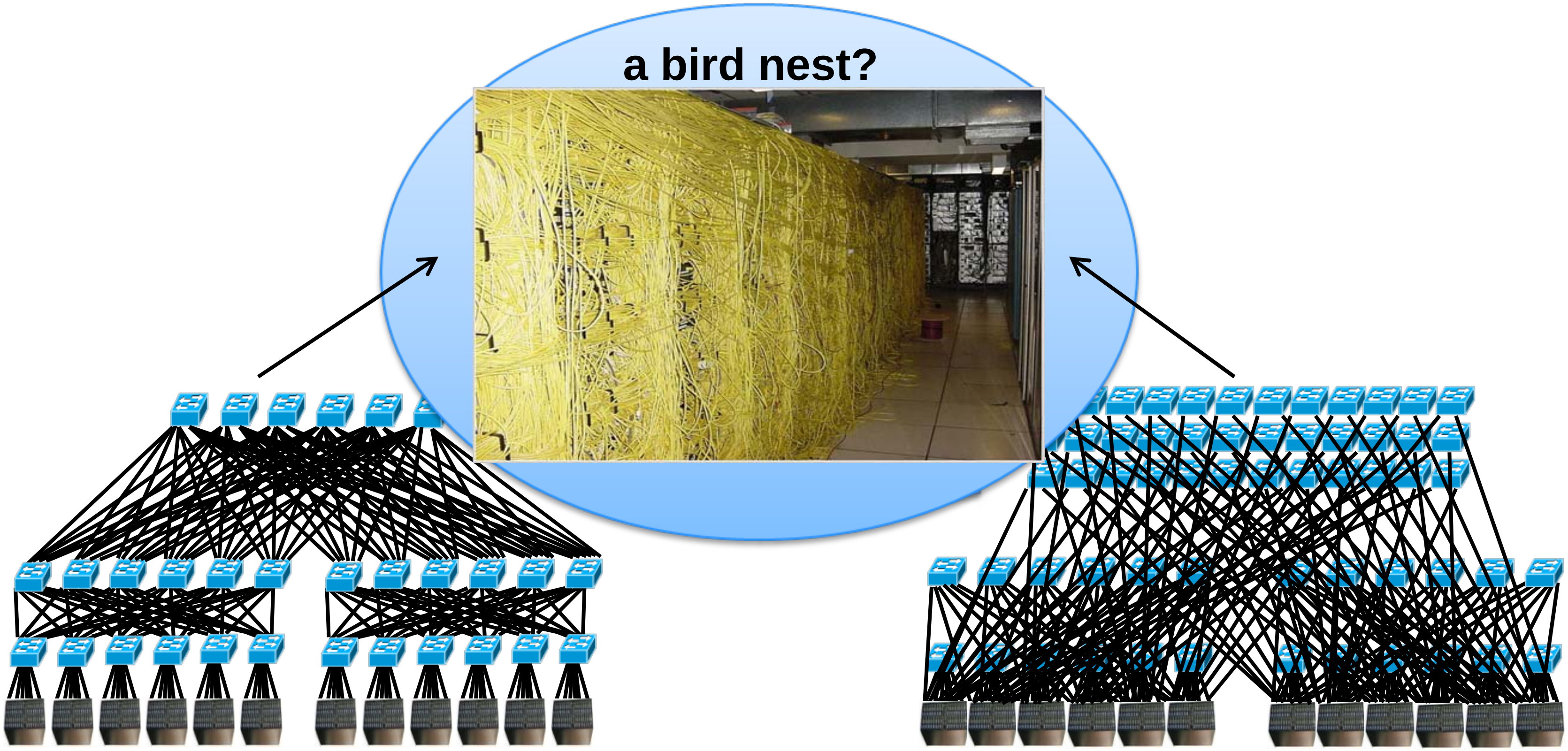
Bandwidth bottleneck in data center networks

- Traditional data center network:
  - tree-structure Ethernet



Severe bandwidth bottleneck in aggregation layers.

Previous solutions for increasing data center network bandwidth

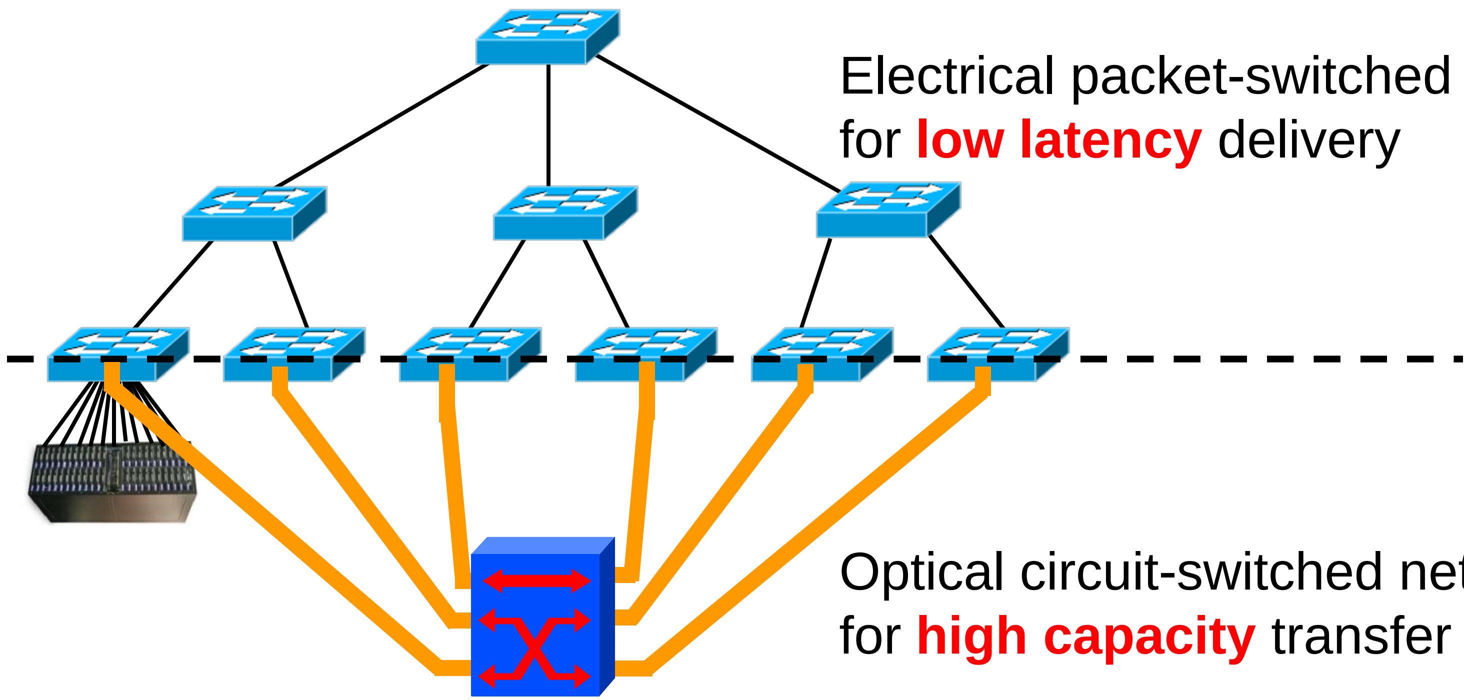


FatTree **HyperCube**

1. Hard to construct

2. Hard to expand

An alternative: hybrid packet/circuit switched network architecture



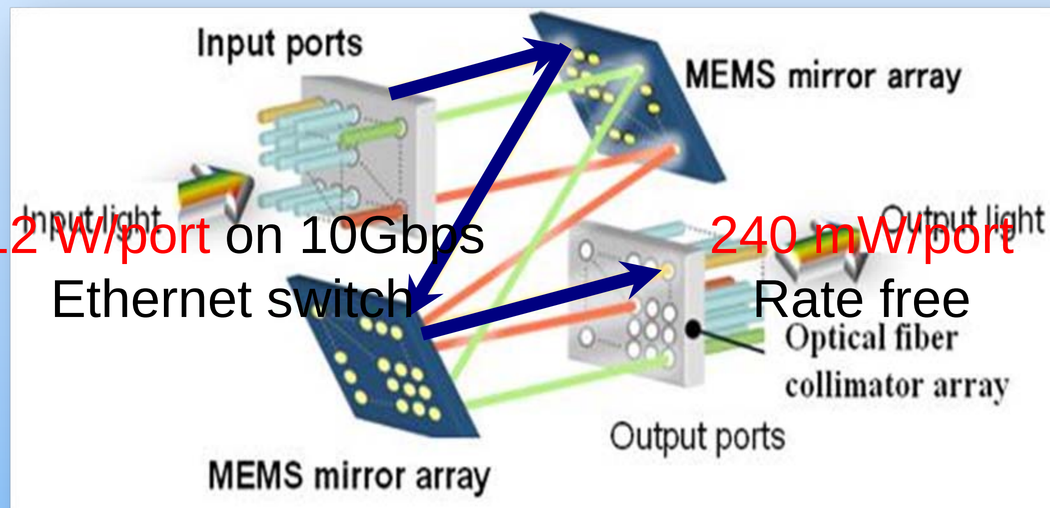
Electrical packet-switched network for **low latency** delivery

Optical circuit-switched network for **high capacity** transfer

- Optical paths are provisioned rack-to-rack
  - A simple and cost-effective choice
  - Aggregate traffic on per-rack basis to better utilize optical circuits

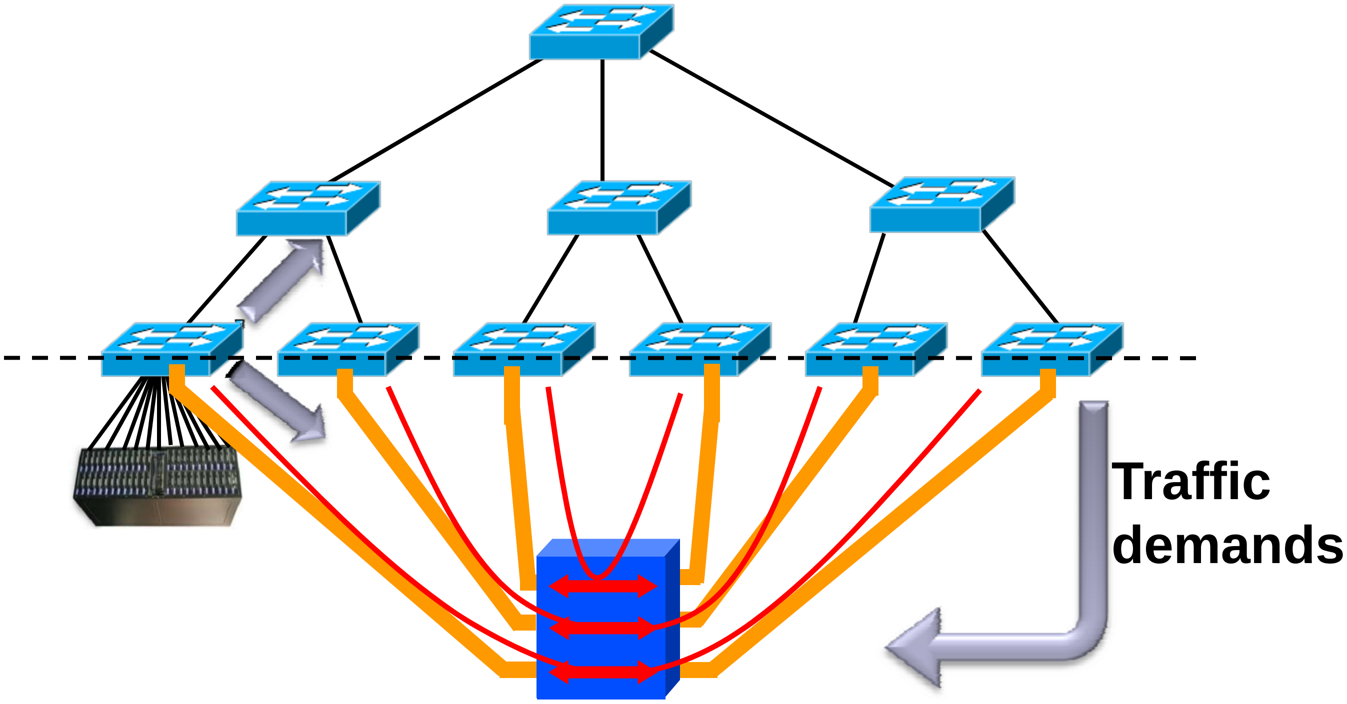
Optical circuit switching v.s. Electrical packet switching

	Electrical packet switching	Optical circuit switching
Switching technology	Store and forward	Circuit switching
Switching capacity		
Energy efficiency	12 W/port on 10Gbps Ethernet switch	240 mW/port Rate free
Switching time		



e.g. MEMS optical switch

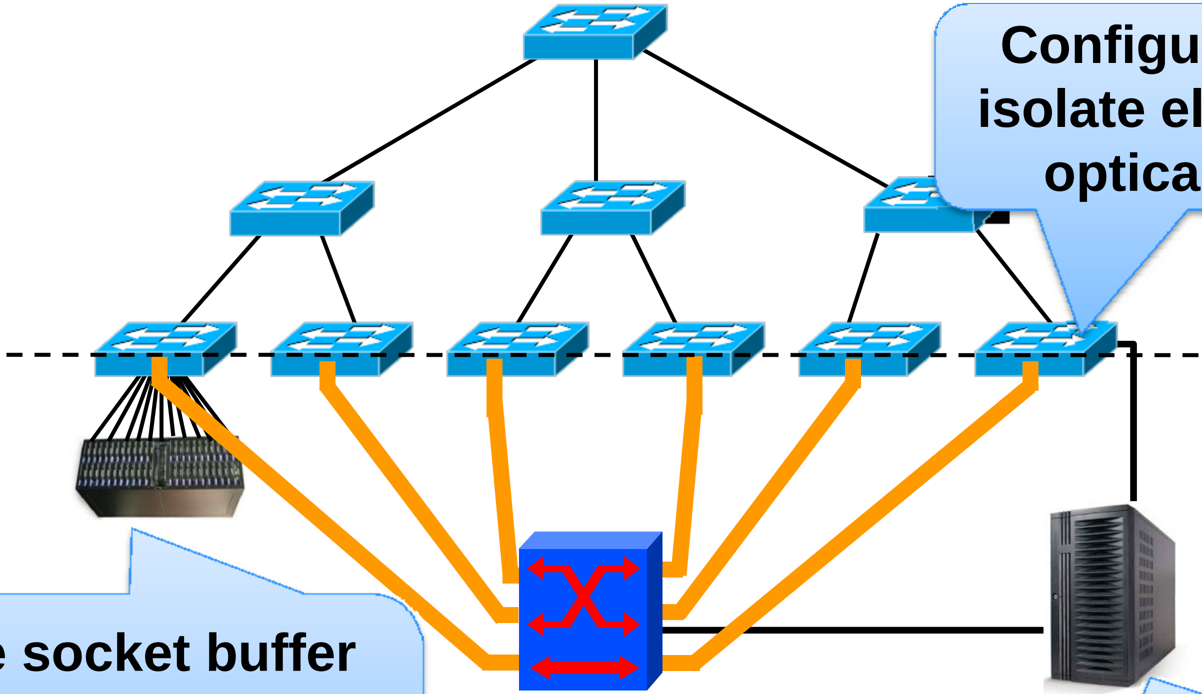
Design requirements



- Control plane:
  - Traffic demand estimation
  - Optical circuit configuration

- Data plane:
  - Dynamic traffic de-multiplexing
  - Optimizing circuit utilization (optional)

c-Through design



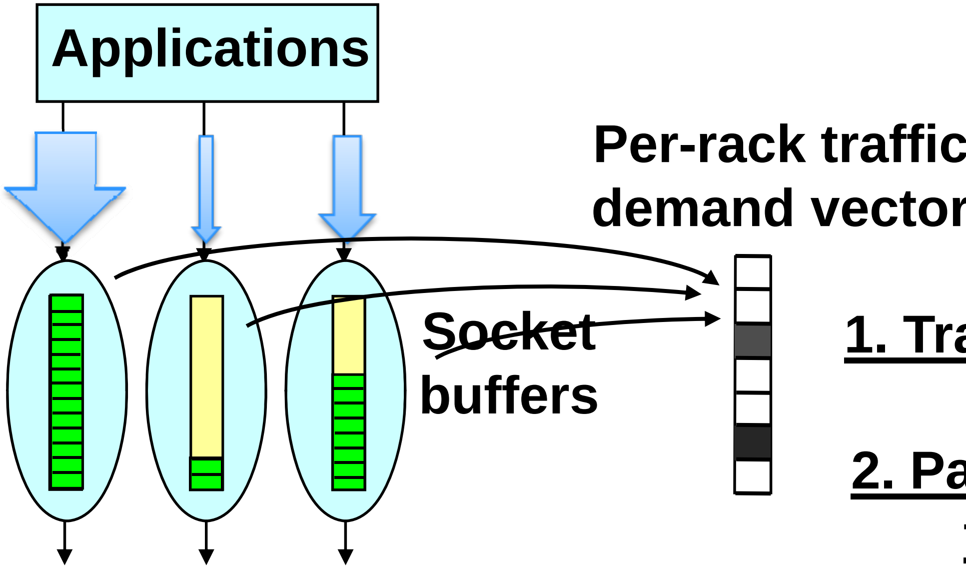
Configure VLAN to isolate electrical and optical network

1. Enlarge socket buffer to estimate demand.
2. De-multiplex traffic using VLAN tagging.

Centralized control for circuit configuration

Feasible to build a hybrid network without modifying Ethernet switches and applications!

c-Through - traffic demand estimation and traffic batching

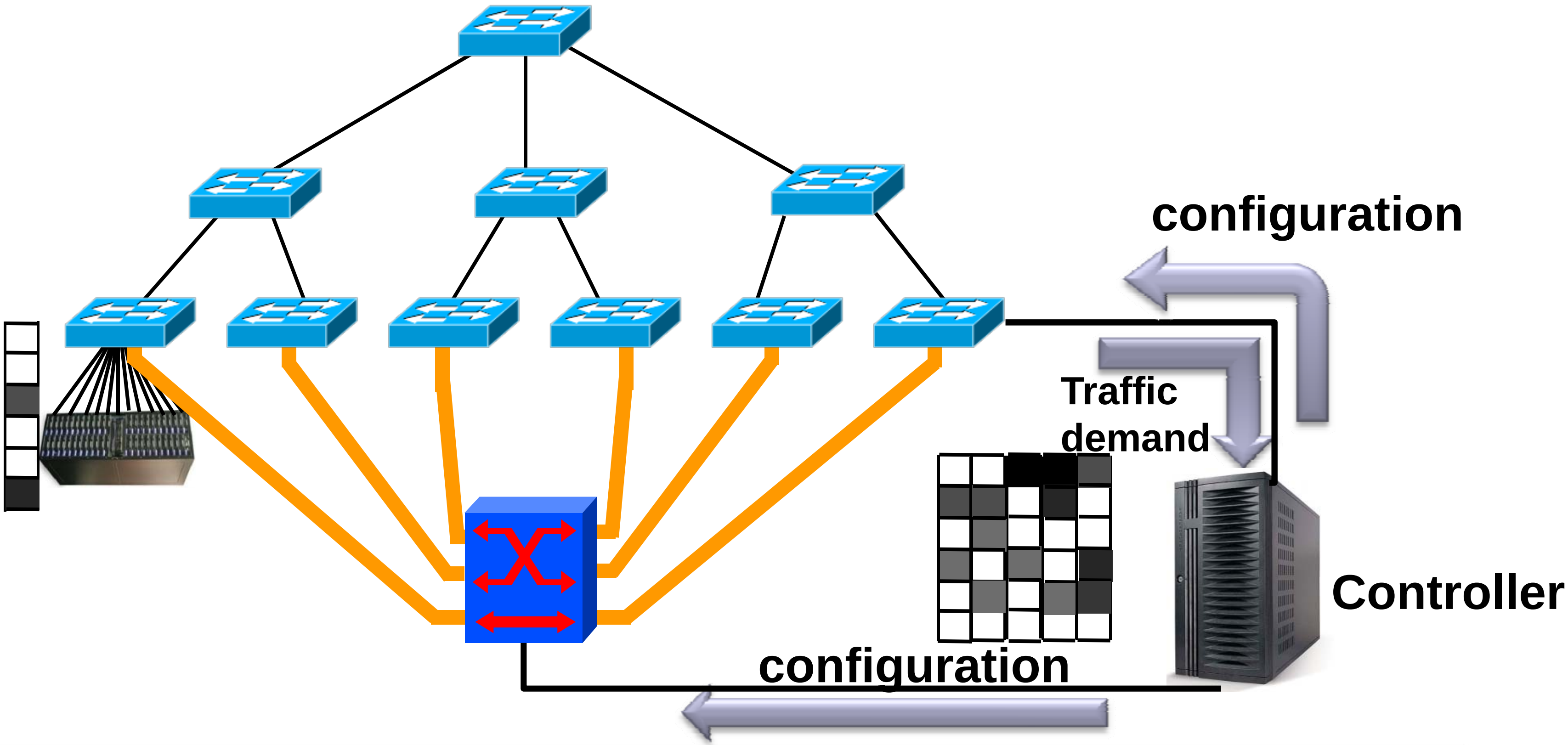


1. Transparent to applications.

2. Packets are buffered per-flow to avoid HOL blocking.

- Accomplish two requirements:
  - Traffic demand estimation
  - Pre-batch data to improve optical circuit utilization

c-Through - optical circuit configuration

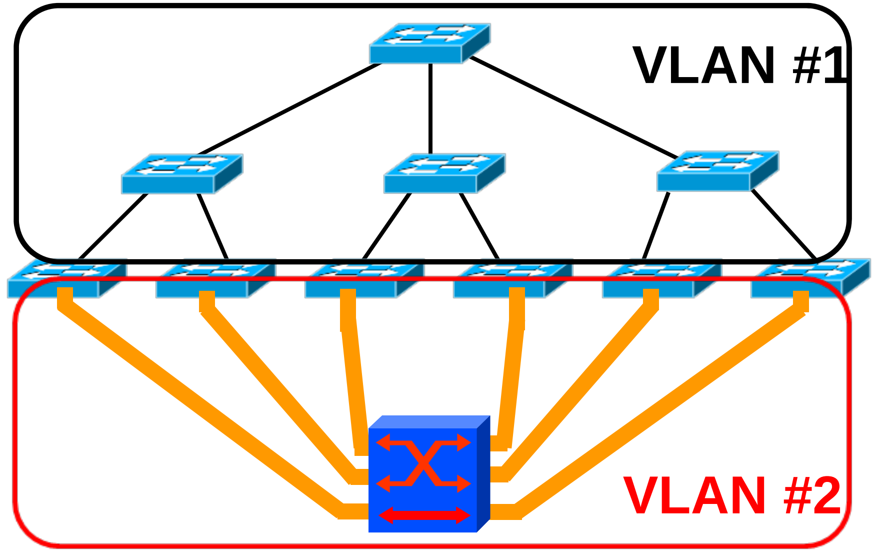


Centralized controller computes optimal configuration

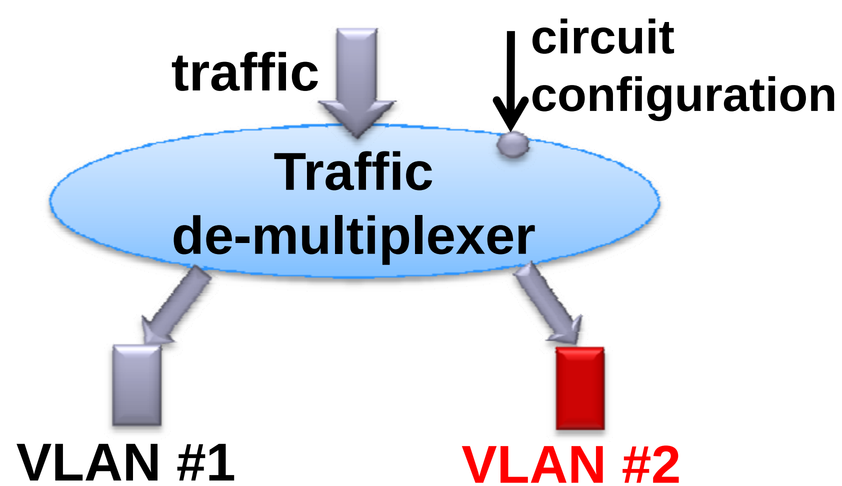
Many ways to reduce the control traffic overhead

c-Through - traffic de-multiplexing

- VLAN-based network isolation:
  - No need to modify switches
  - Avoid the instability caused by circuit reconfiguration

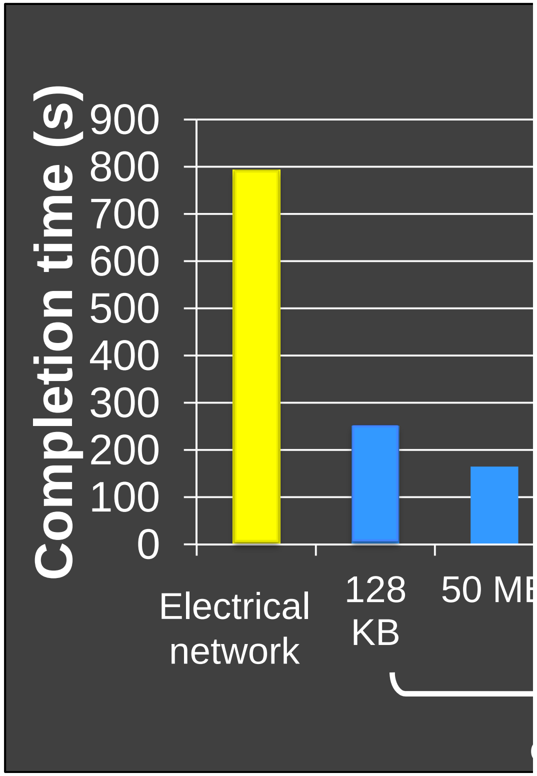


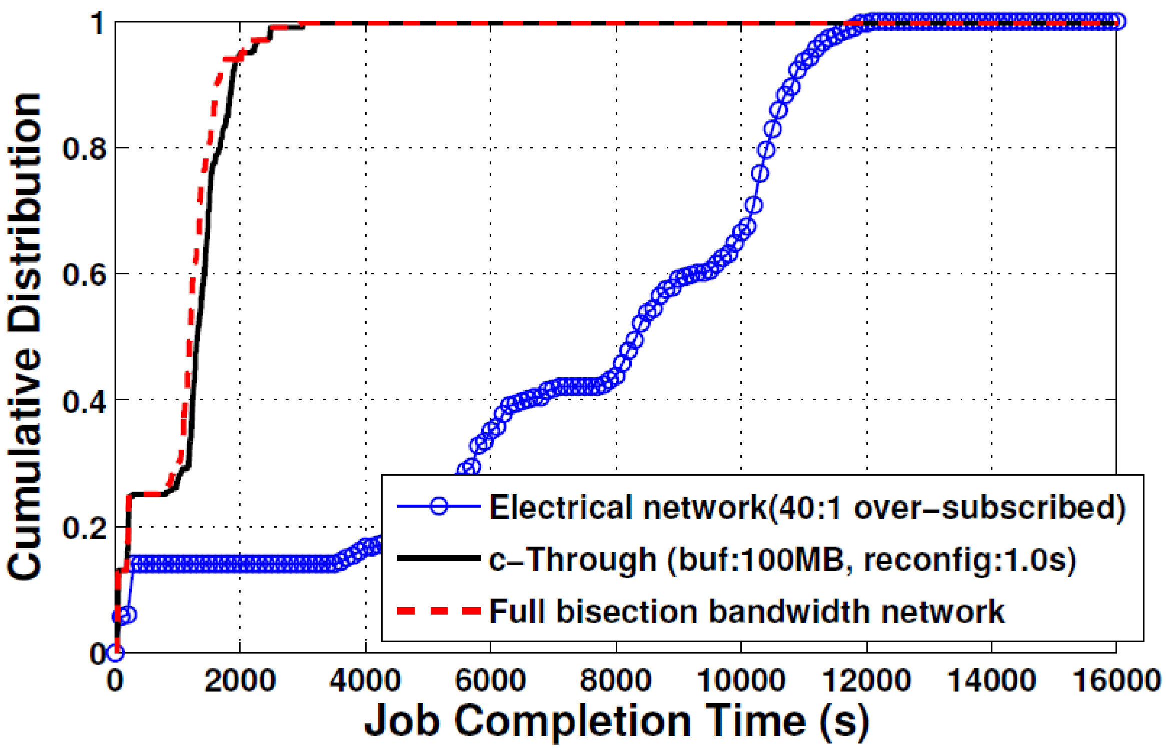
- Traffic control on hosts:
  - Controller informs hosts about the circuit configuration
  - End-hosts tag packets accordingly



Applicability of hybrid network

MapReduce performance **Gridmix performance**





Close-to-optimal performance even for applications with all-to-all traffic patterns.

Related work

c-Through

[Wang et al.]

- Rack level optical paths
- Estimating demand from server socket buffer
- Traffic control in server kernel

Helios

[Farrington et al.]

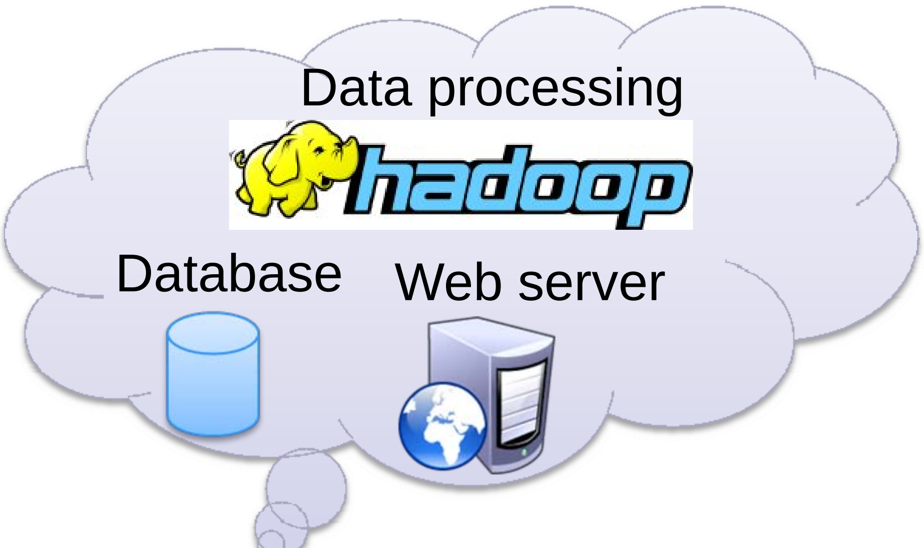
- Pod level optical paths
- Estimating demand from switch flow counters
- Traffic control by modifying switches

Others

- **IBM System-S**: hybrid network for stream processing
- **Proteus** *[Singla et al.]*: all optical data center network using WSS
- **DOS** *[Ye et al.]*: all optical data center network using AWGR

Circuit control in the “wild”

- Sharing is the key of cloud data centers



- Share at fine grain
- Complicated data dependencies
- Heterogeneous applications

Problems of existing systems

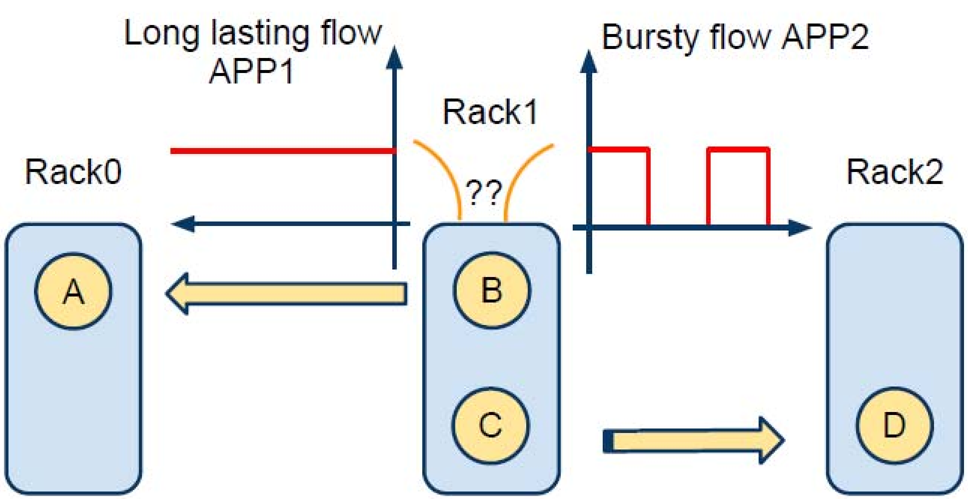
1. Treating all traffic as independent flows
   – Suboptimal performance for correlated applications

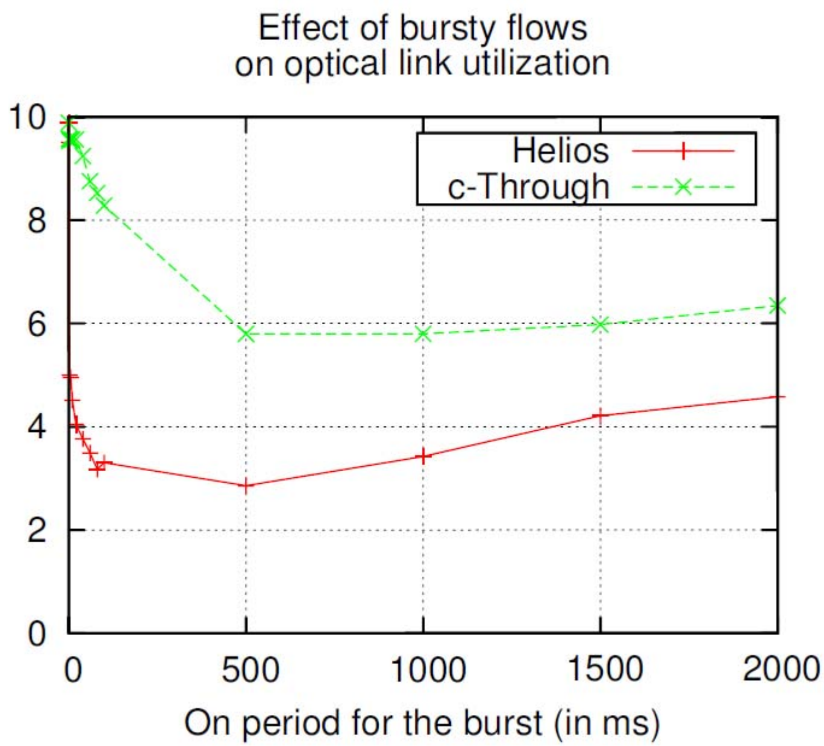
2. Inaccurate information about traffic demand
   – Vulnerable to ill-behaved applications

3. Restricted sharing policies
   – Limited by the control platform of Ethernet switches

Problem 1: Inaccurate demand

- Effect of bursty flow

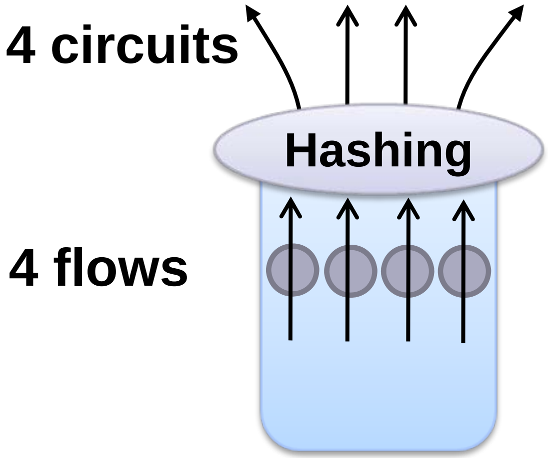




A single bursty flow can reduce circuit throughput by half.

Problem 2: Restricted sharing policy

- An example:
  - Random hashing over multiple circuits.



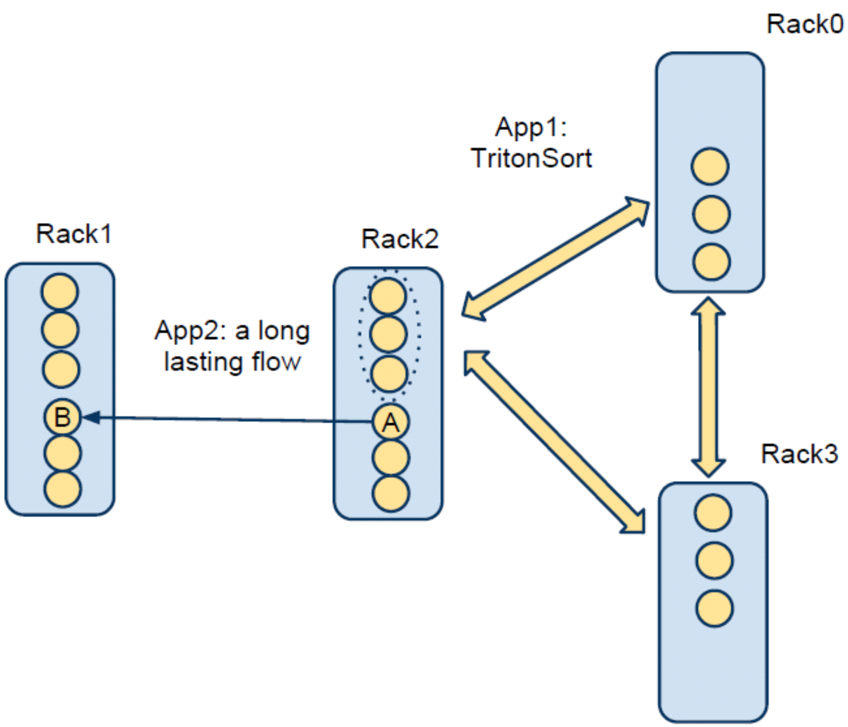
- Hash collision

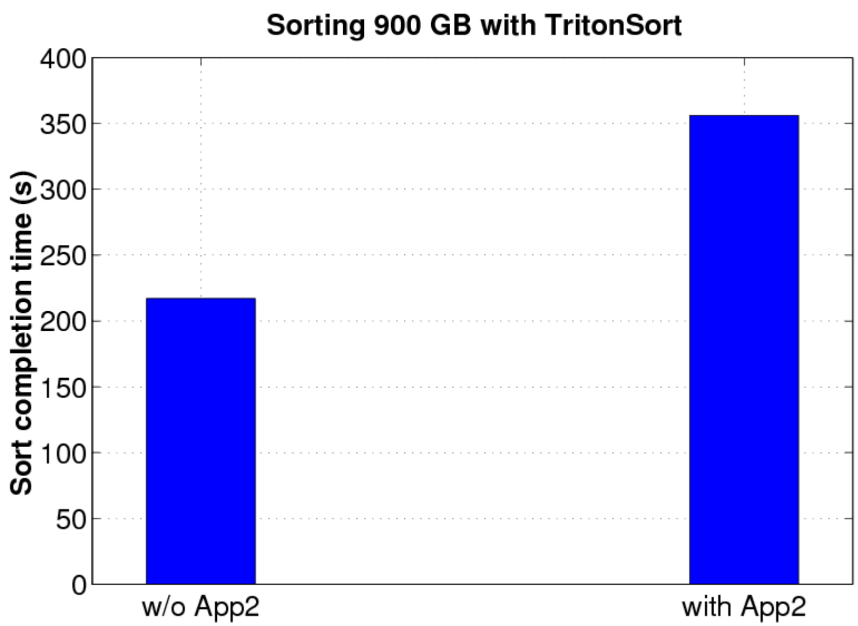
- Limited to random sharing

More flexible sharing policies are needed

Problem 3: traffic dependencies

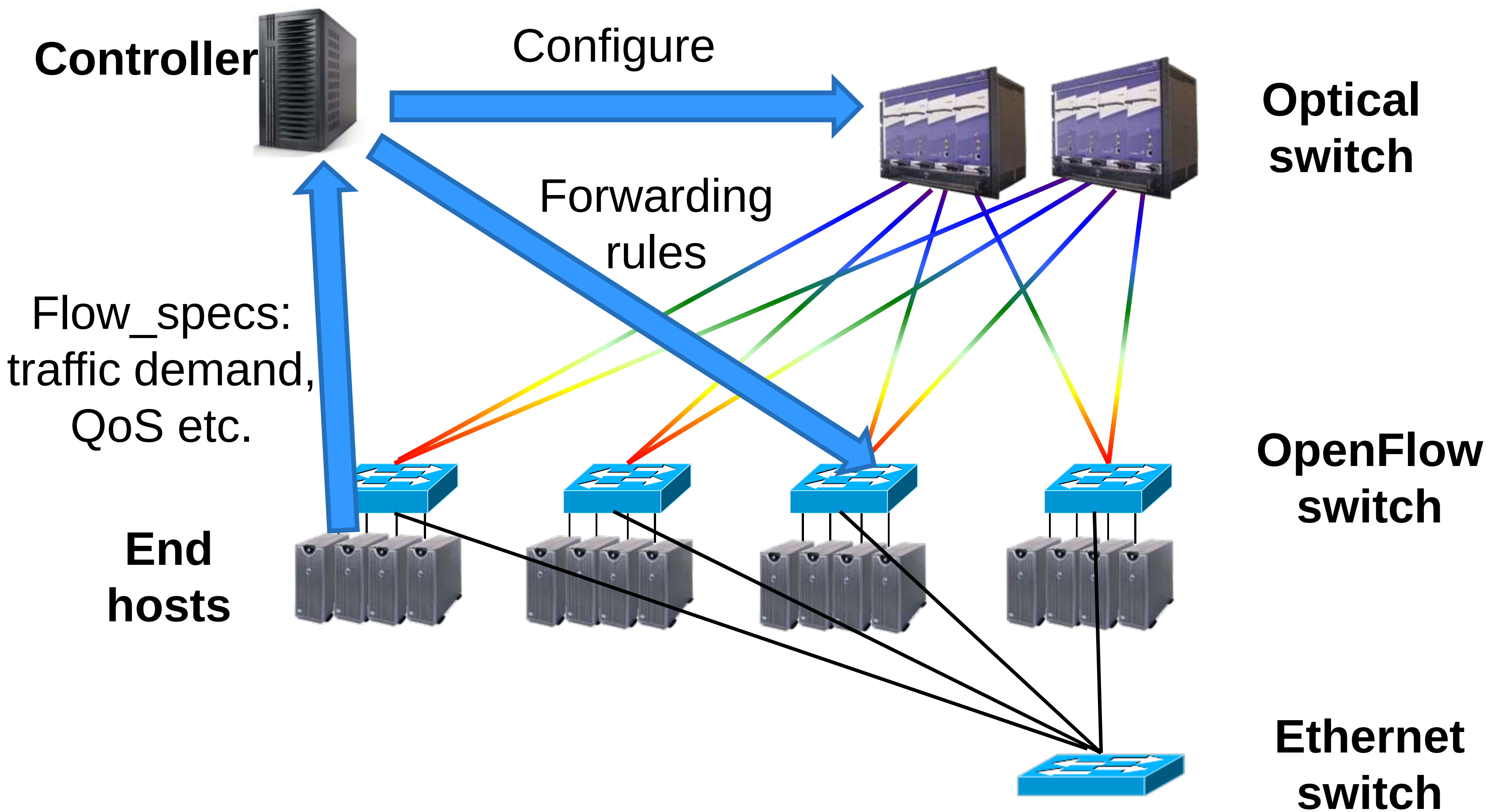
- Effect of correlated flows





Traffic dependencies cause system-wide performance degradations.

OpenFlow based control framework



Demand estimation from historical measurements

Scheduling on flows

Challenges

- Traffic analysis with application semantics
- Circuit scheduling with correlated flows
- Rapid flow table update and aggregation on OpenFlow

Summary

- Hybrid packet/circuit switched data center network
  - c-Through demonstrates its feasibility
  - Good performance even for applications with all to all traffic
  - Challenges remain in circuit control in cloud data centers
- Further explorations:
  - The scaling property of hybrid data center networks
  - Low latency data center network with optical circuits
  - Low cost optical interconnect

Picture from Internet websites.

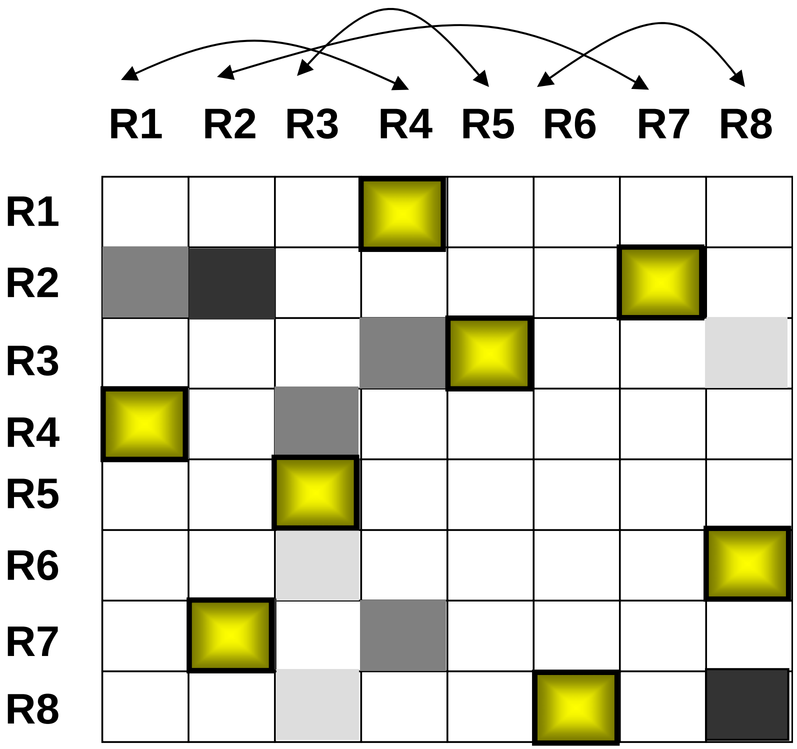
Optical circuit switching is promising despite slow switching time

- [IMC09][HotNets09]: *"Only a few ToRs are hot and most their traffic goes to a few other ToRs. …"*

- [WREN09][IMC10]: *"…we find that traffic at the five edge switches exhibit an ON/OFF pattern… "*

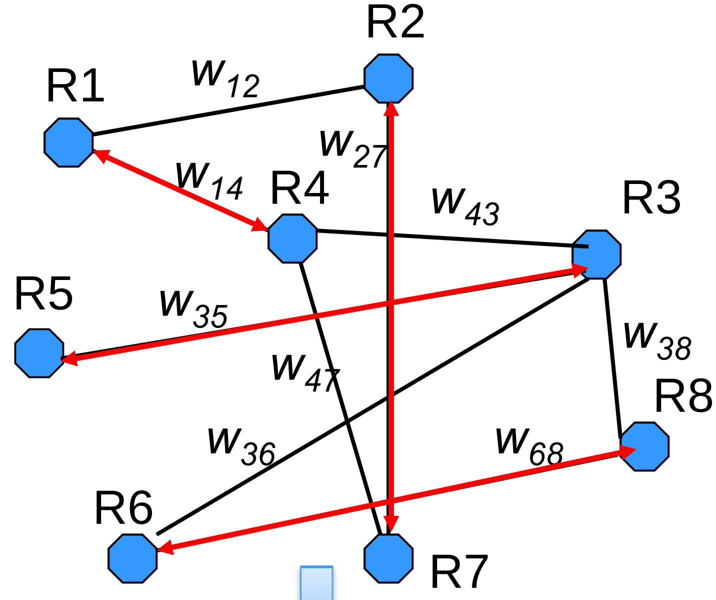
Full bisection bandwidth at packet granularity may not be necessary

Optical circuit configuration algorithm



Graph G: (V, E)

$w_{xy} = vol(Rx, Ry) + vol(Ry, Rx)$

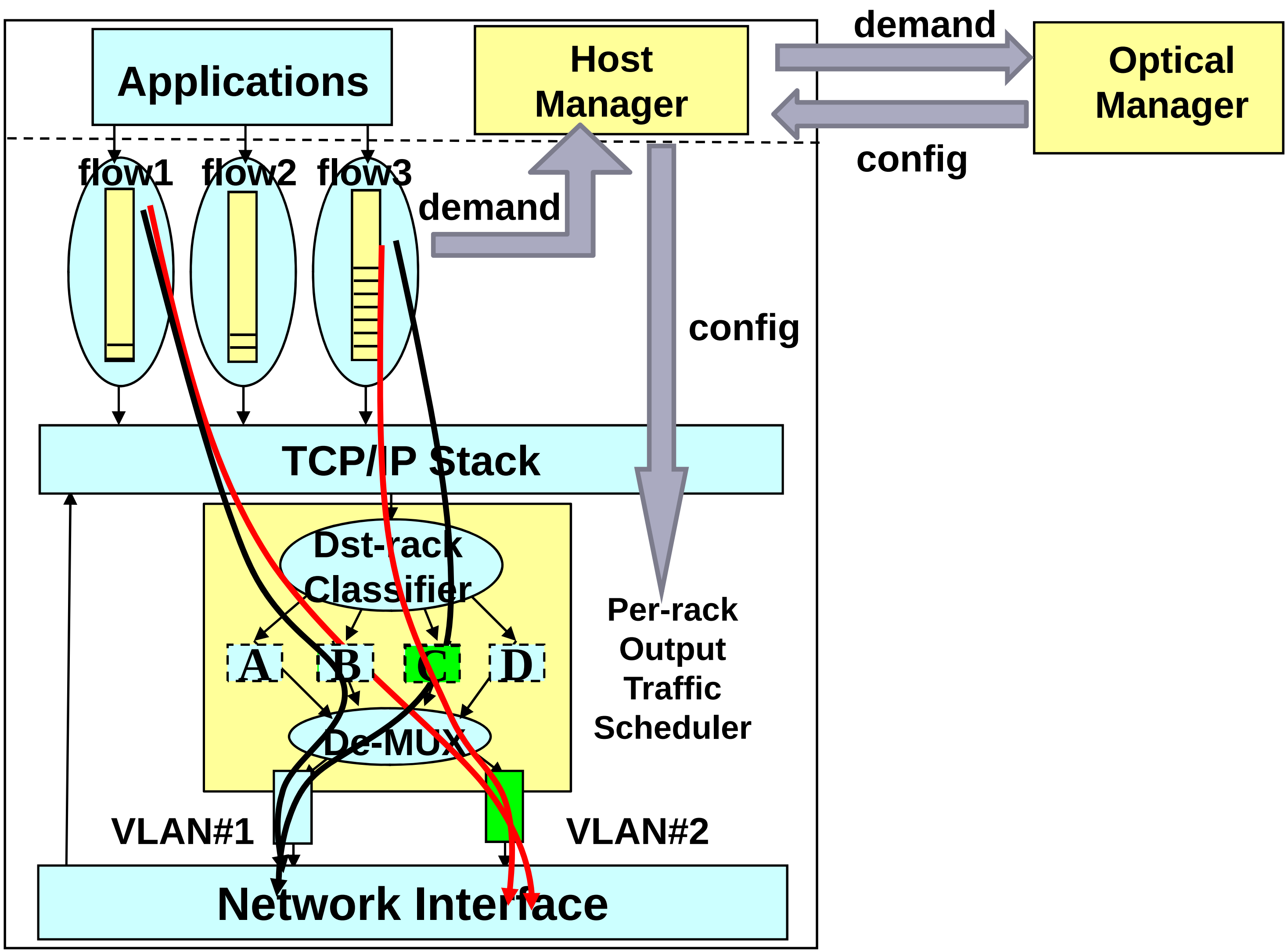


Optical path configuration is a **maximum weight perfect matching** on graph G.

Solved by polynomial time Edmonds' algorithm[1]!

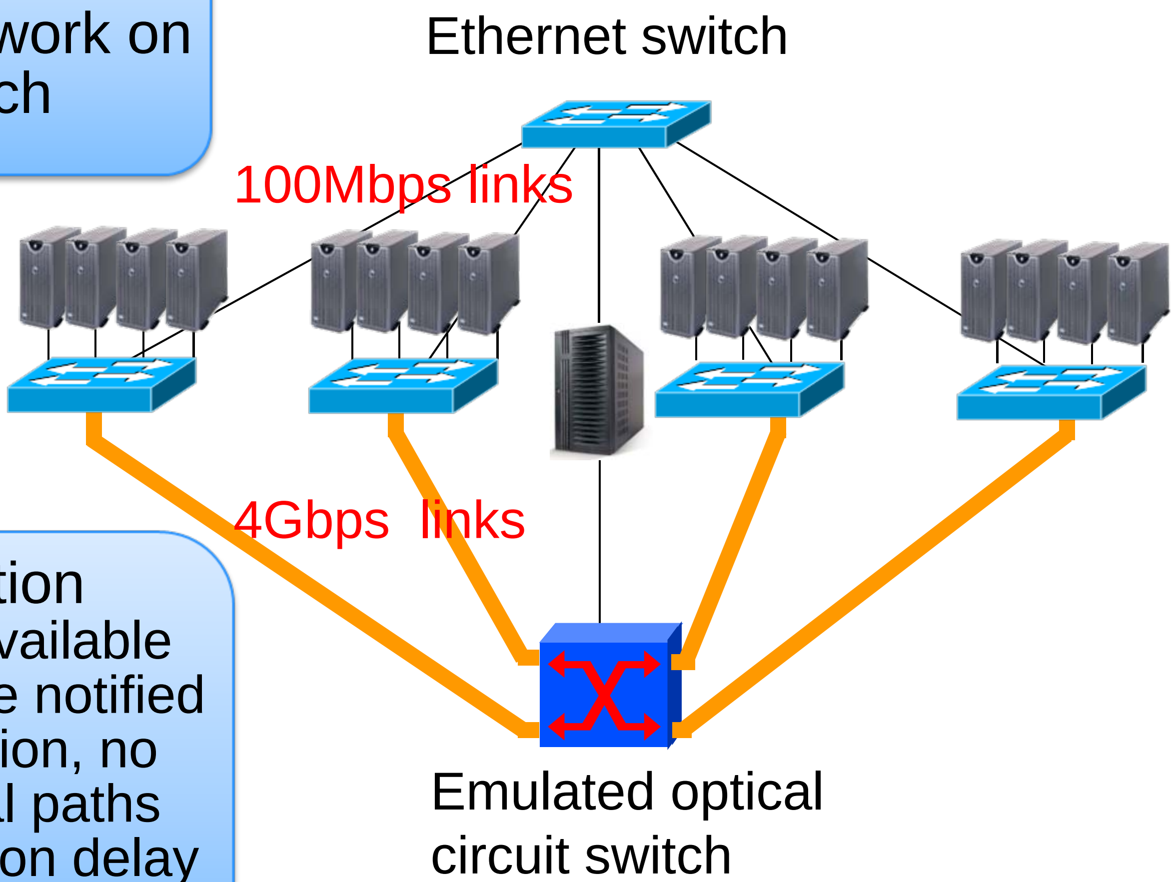
[1] *J. Edmonds, Paths, trees and flowers, Canadian J. of Mathematics, pp 449-467, 1965*

c-Through: implementation



c-Through test-bed

- 16 servers with 1Gbps NICs
- Emulate a hybrid network on 48-port Ethernet switch



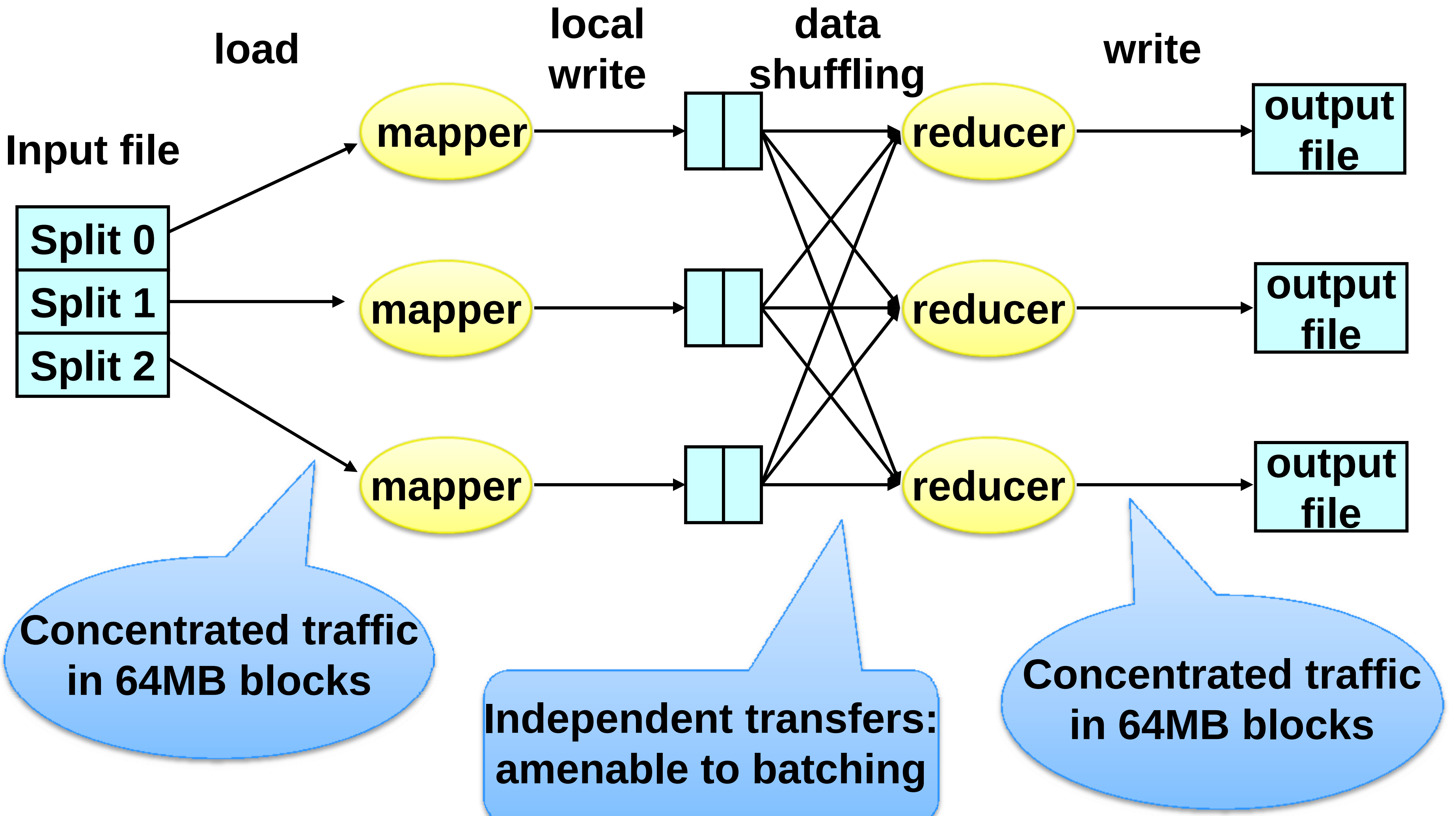
- Optical circuit emulation
  - Optical paths are available only when hosts are notified
  - During reconfiguration, no host can use optical paths
  - 10 ms reconfiguration delay

System performance study

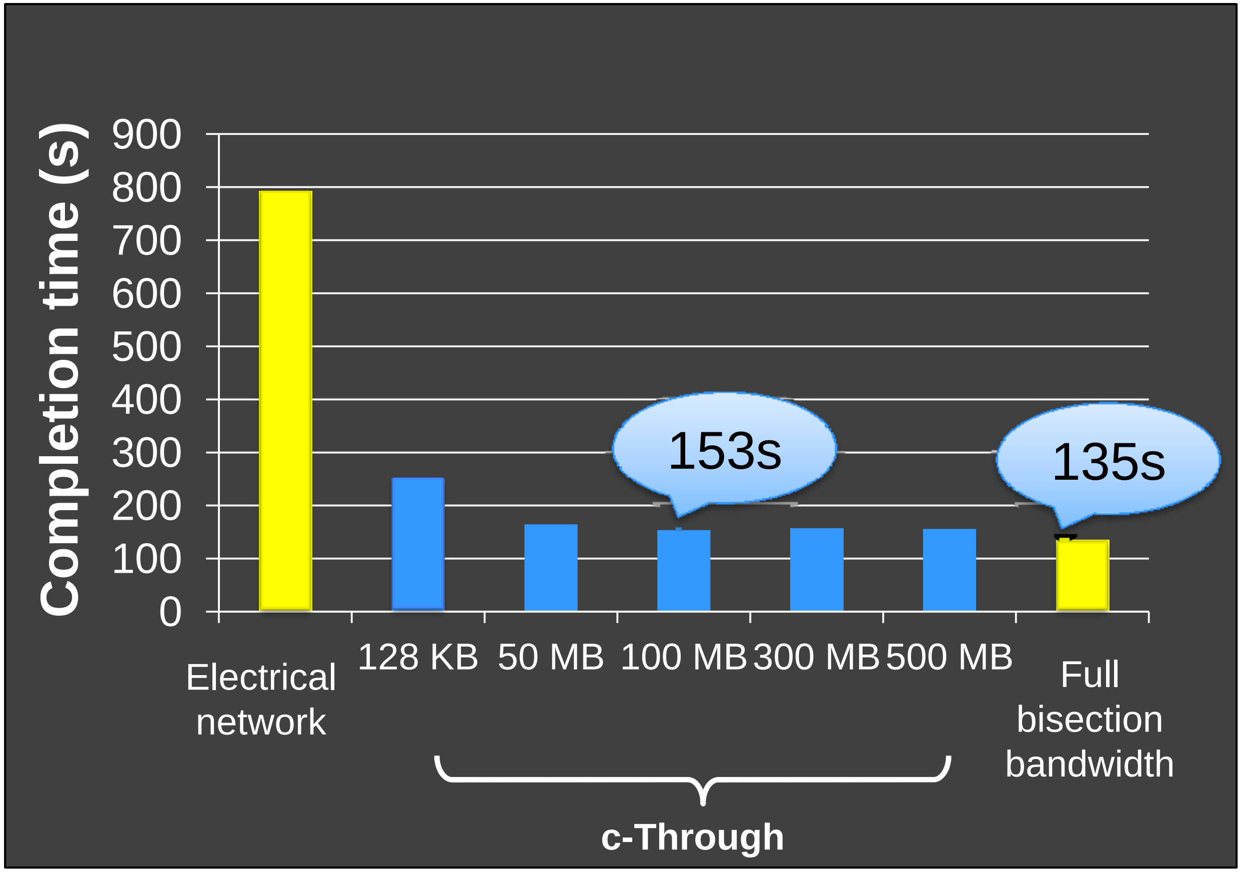
- Basic system performance:
  - Can TCP exploit dynamic bandwidth quickly? **Yes**
  - Does traffic control on servers bring significant overhead? **No**
  - Does buffering unfairly increase delay of small flows? **No**

- Application performance:
  - Bulk transfer (VM migration)? **Yes**
  - Loosely synchronized all-to-all communication (MapReduce)? **Yes**
  - Tightly synchronized all-to-all communication (MPI-FFT) ? **Yes**

MapReduce Overview

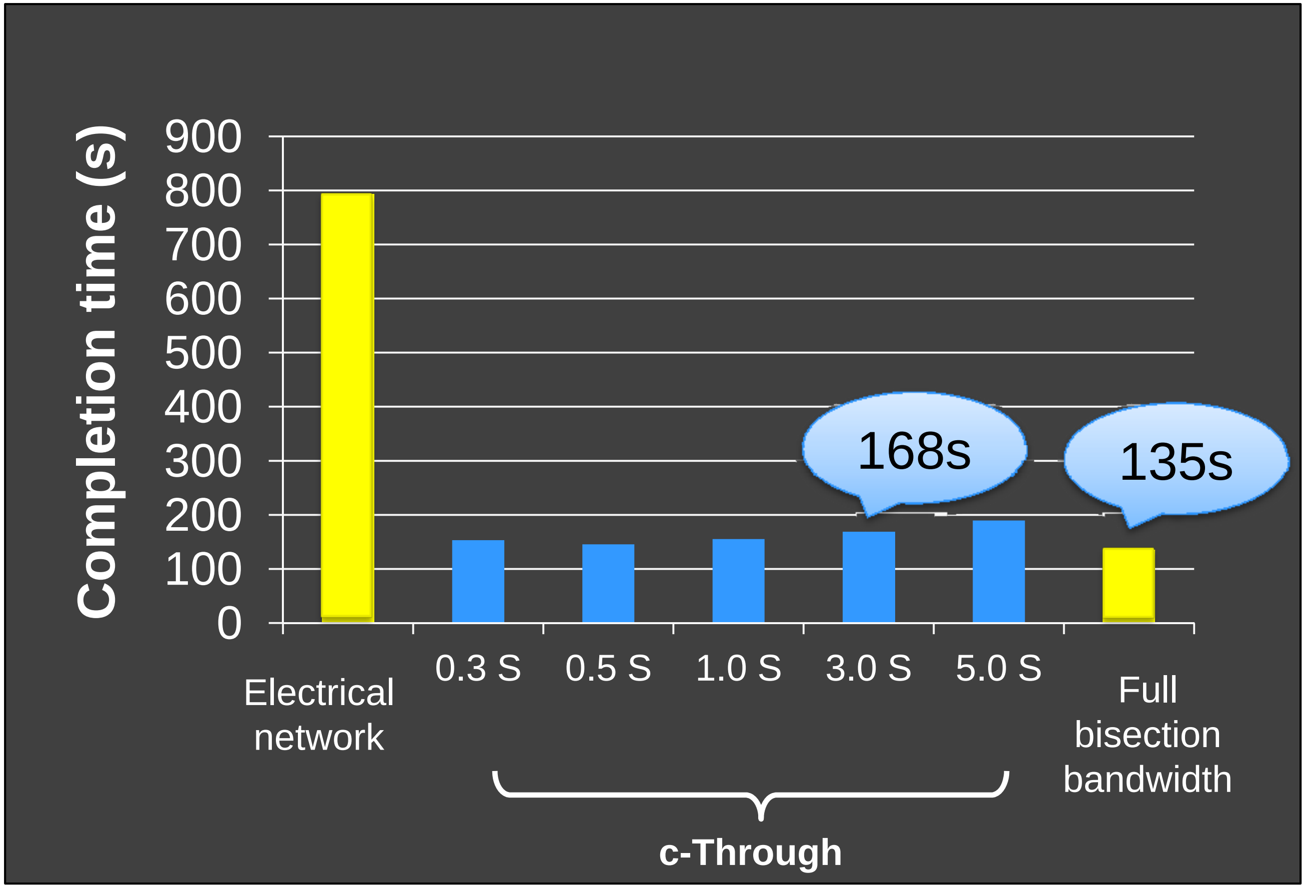


MapReduce sort 10GB random data



c-Through varying socket buffer size limit (reconfiguration interval: 1 sec)

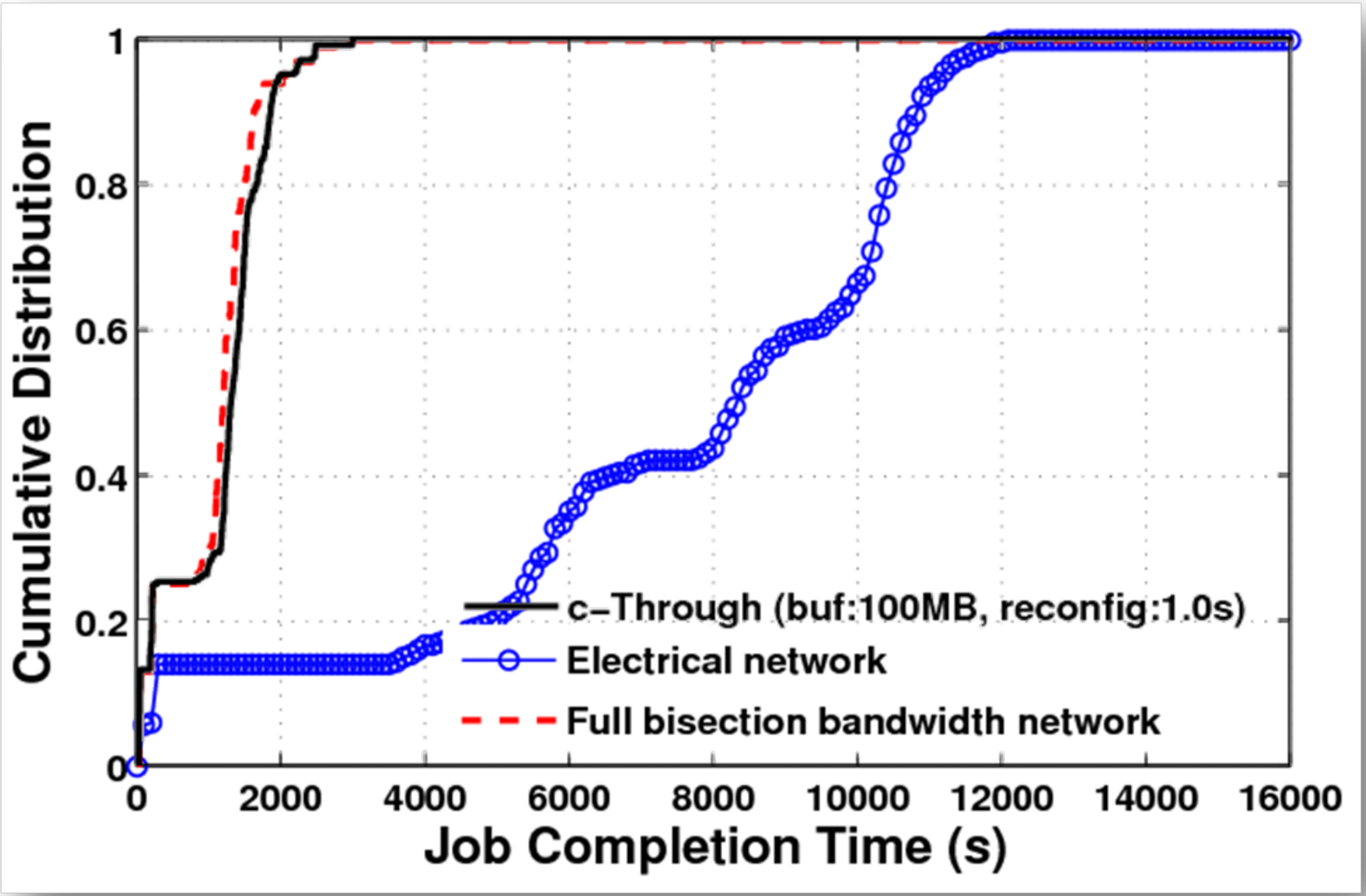
MapReduce sort 10GB random data



c-Through varying reconfiguration interval
(socket buffer size limit: 100MB)

Yahoo Gridmix benchmark

- 3 runs of 100 mixed jobs such as web query, web scan and sorting
- 200GB of uncompressed data, 50 GB of compressed data



Related work

	Prototype system	Optical devices	Modifications required
HPC (IBM) (SC'05)	Simulation	MEMS	Compiler and NIC hardware
IBM System S (GLOBECOM'09)	Yes	MEMS	Specific to stream processing
c-Through (HotNets'09, SIGCOMM'10)	Yes	MEMS	Host software, support all apps
Helios (SIGCOMM '10)	Yes	MEMS	Switch software, support all apps
Proteus (HotNets'10),	Not yet	WSS, MEMS	Unspecified
DOS (ANCS'10)	Simulation	AWGR	Unspecified

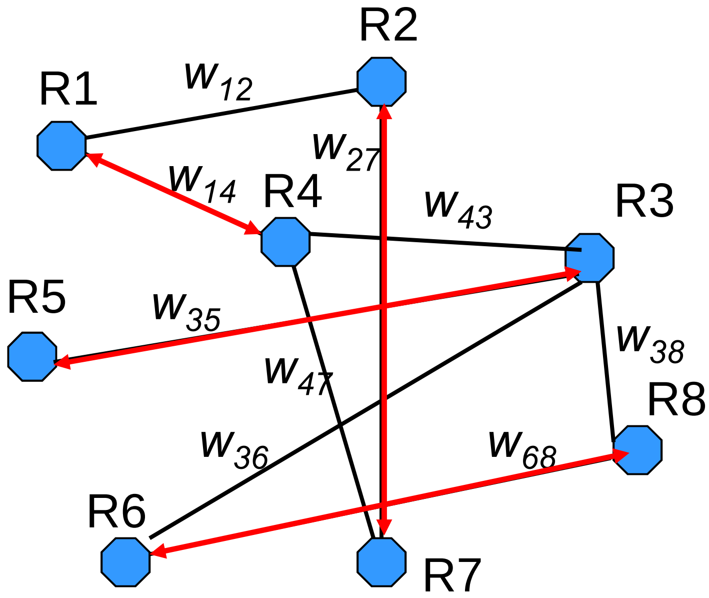
Circuit configuration with correlated traffic

- Problem formulation

Basic configuration: a matching problem

Graph G: (V, E)

w_{xy}= *vol(Rx, Ry) + vol(Ry, Rx)*



Modeling correlated traffic:

Definition of correlated edge groups:

$EG = \{e_1, e_2, \ldots, e_n\}$, *so that*

$w(e_i) \mathrel{+}= \Delta(e_i), i = 1, \ldots, n$

when EG is part of the matching.

Conflicting edge groups:

Two edge groups are conflict if they have edges sharing one end vertex.

Maximum weight matching with correlated edges

Algorithm design (1)

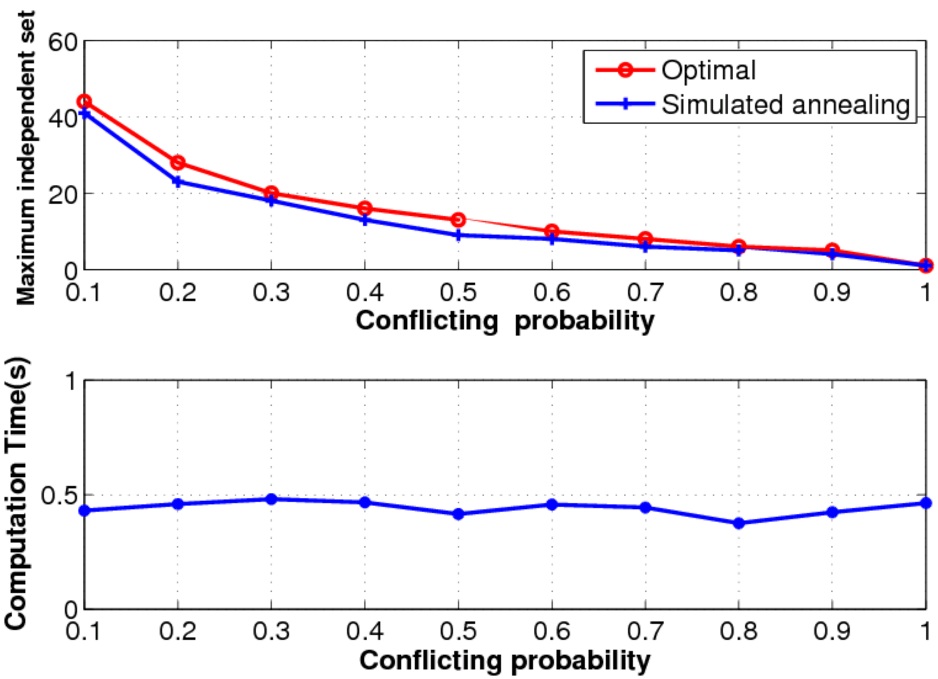
- If there is only one edge group
  - Intuition: test if including the edge group in the match will improve the overall weight.
  - Equation:

$$benefit(EG,G) = \overbrace{Weight\big(EG + Edmonds(G - EG)\big)}^{\textbf{Accept}} - \overbrace{Weight(Edmonds(G))}^{\textbf{Not accept}}$$

- If no conflict among edge groups:
  - A greedy algorithm
    - Iteratively accept all the edge groups with positive benefits;
    - Proven to achieve maximum overall weight;

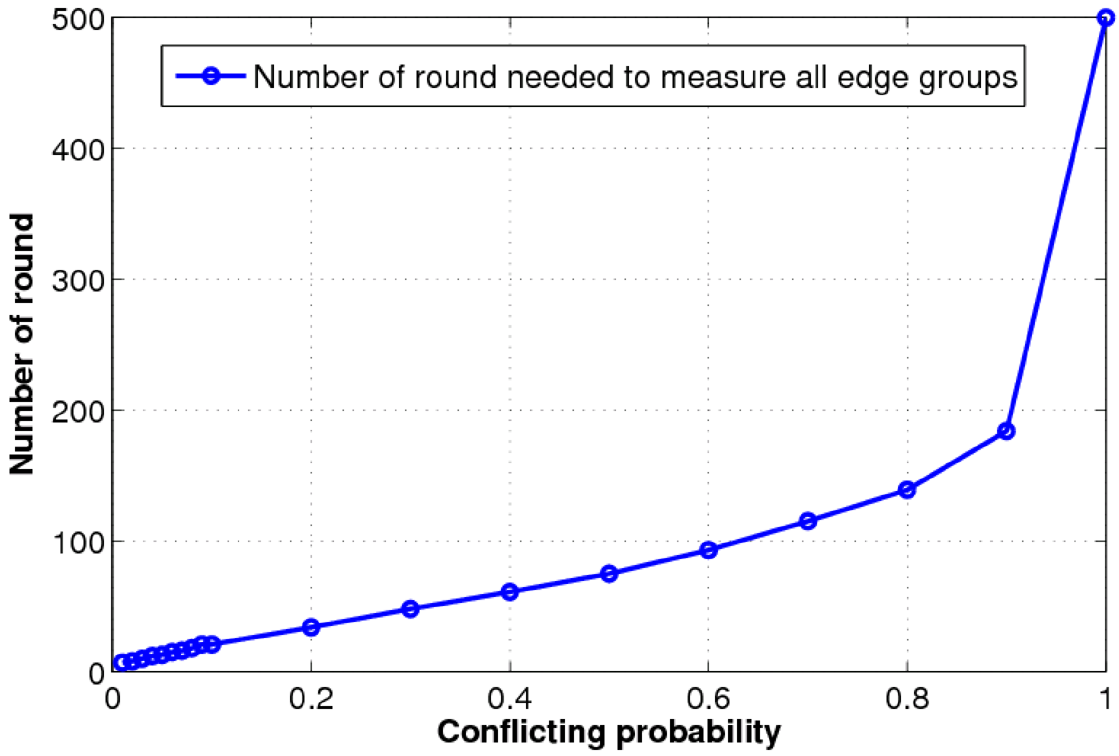
Algorithm design (2)

- If there are conflicts among edge groups
  - *Finding the best non-conflict edge groups is NP-hard.*
    - Equivalent to maximum independent set problem.
  - *An approximation algorithm based on simulated annealing works well.*



Inferring correlated edge groups (1)

- Locations known, demand unknown:
  - Measuring maximal number of non-conflicting edge groups in each round.



Inferring correlated edge groups (2)

- Location unknown, demand unknown:
  - Best effort heuristics:
    - Edge tables

Src	Dst	Appearance	Thr_Avg	Thr_Var	Peak_config

    - Peak_config: Thr > (Thr_Avg+2*Thr_Var)
    - Edges with common surge_config are correlated.
  - Performance limited by the chance that edge groups are included in recent matching results;
    - ~0.04% for a group of 2 edges being selected in a 50-rack network.