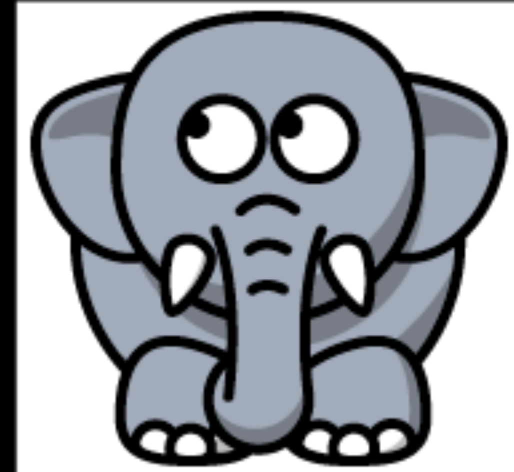
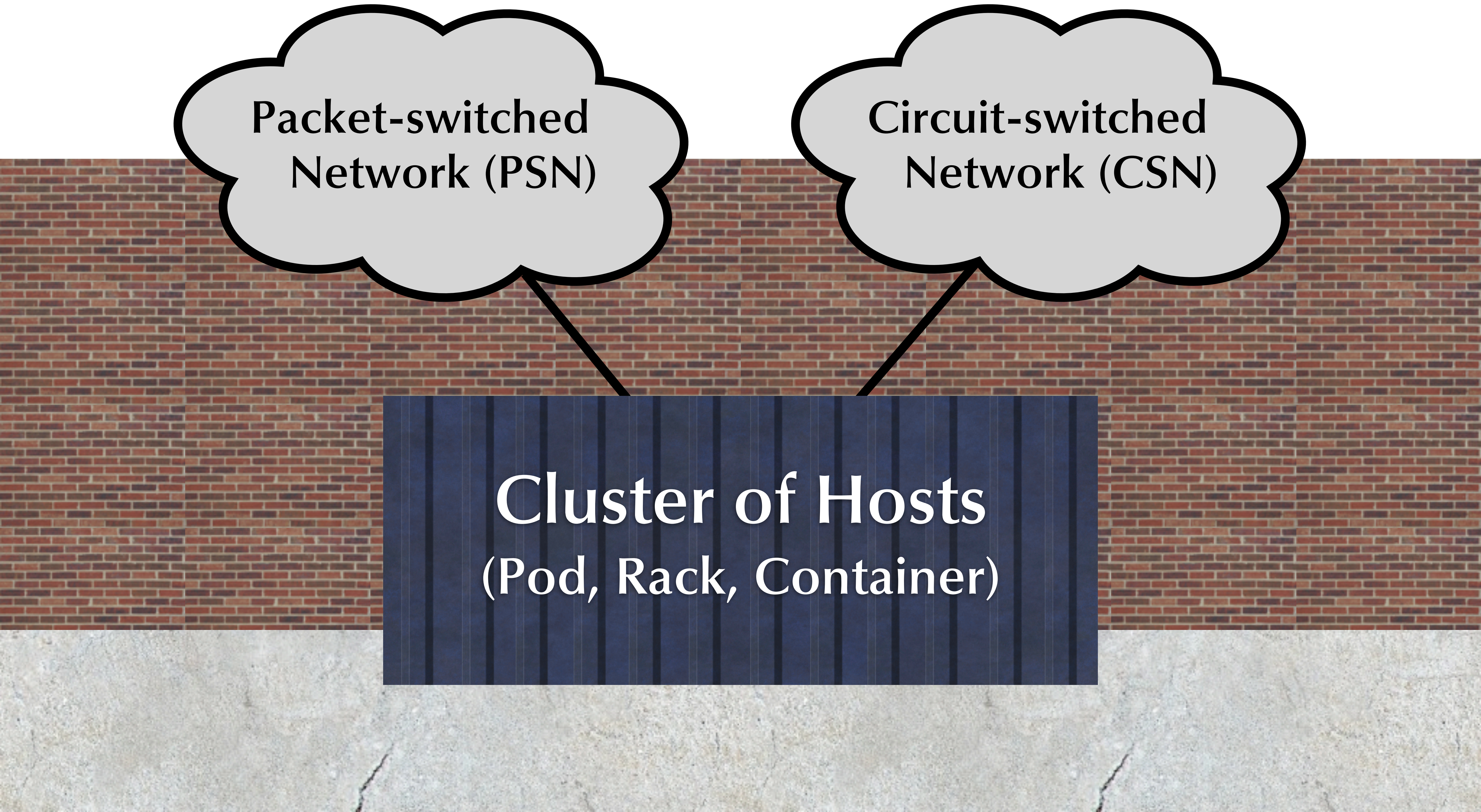
Hunting Mice with μs Circuit Switches

Nathan Farrington
George Porter
Yeshaiahu Fainman
George Papen
Amin Vahdat[†]

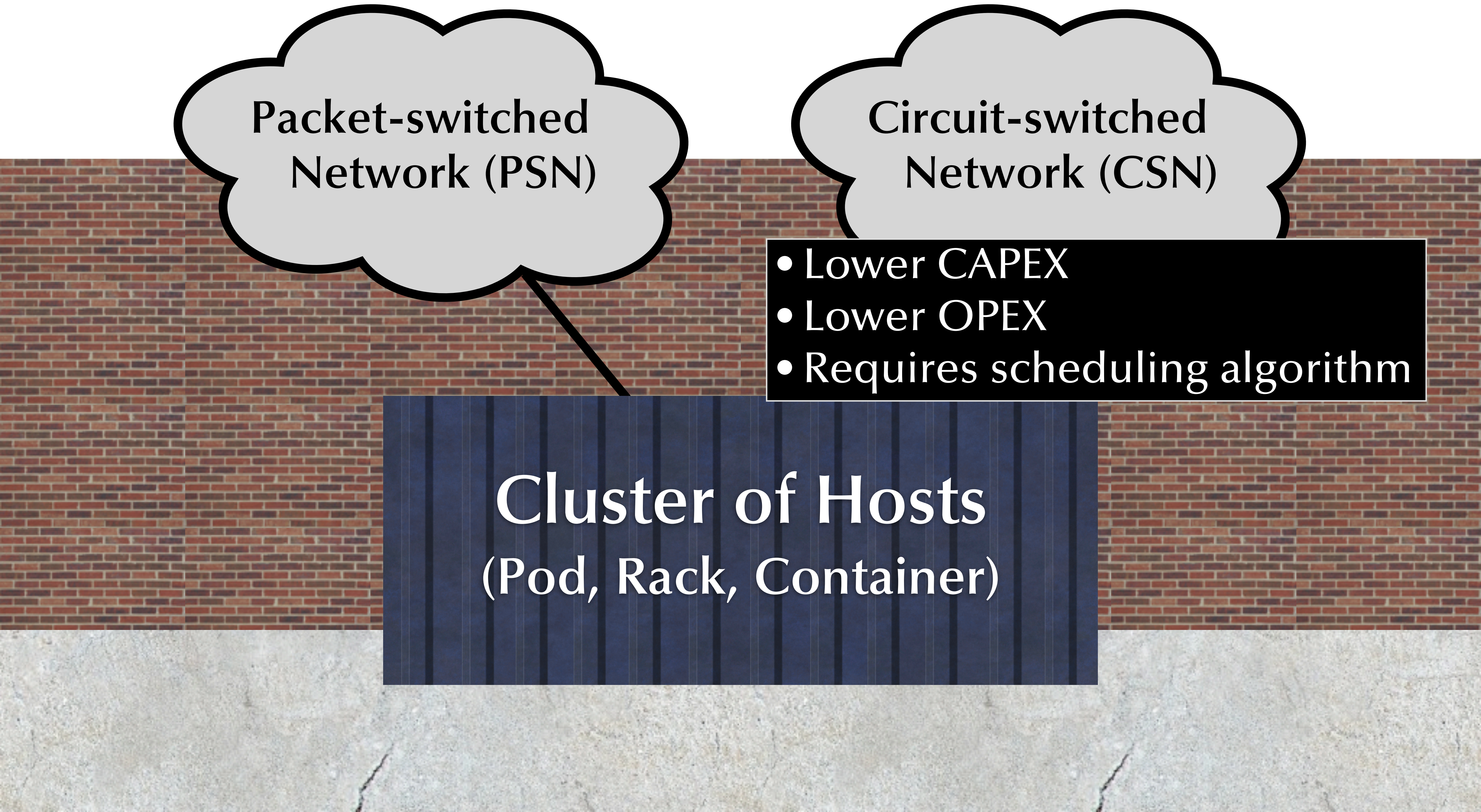
UC San Diego *[†]Google*



Hybrid Data Center Networks



Hybrid Data Center Networks



Q. I am a packet. Do I go over the PSN or the CSN?

A. CSN. If a circuit exists between our pod and the destination pod, then the circuit setup time cost has already been paid.

Corollary. A scheduling algorithm should maximize the throughput over the CSN.

Q. I am a circuit switch. How should I be configured, both right now, and in the future?

The Talk-in-a-Slide Slide

Prior hybrid DCNs use *Hotspot Scheduling*

- Throughput depends on workload
- Not clear how to benefit from faster switch technology

We propose *Traffic Matrix Scheduling*

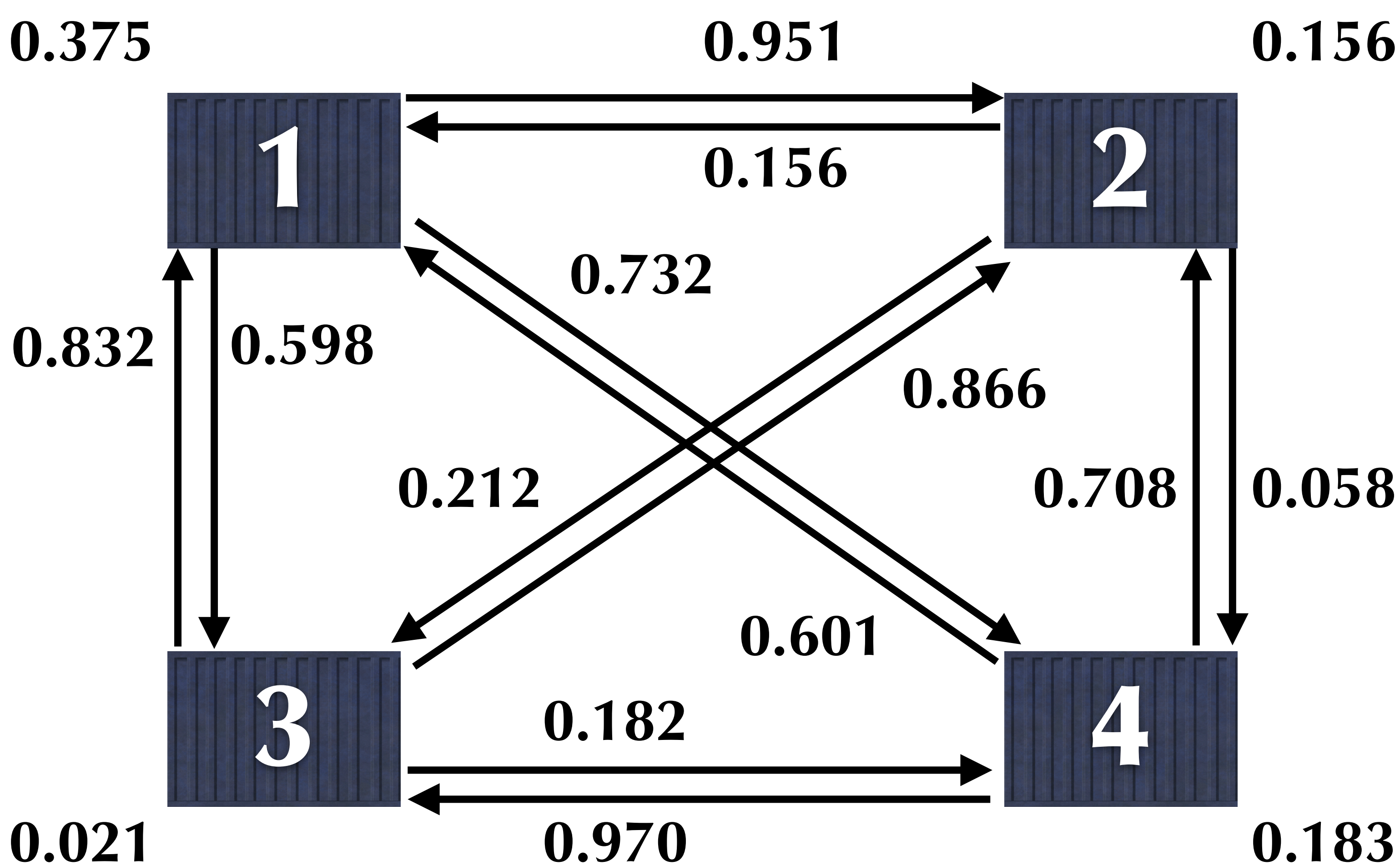
- Achieves 100% throughput on all workloads
- Trading off:
    - Switching time
    - Host buffering
    - Offered load
- Able to use faster switch technology
- But also requires faster switch technology

Outline

1. Hybrid DCNs
2. Algorithms
    - Linear and Stochastic Scaling
    - Hotspot Scheduling
    - Traffic Matrix Scheduling
3. Analysis

Linear and Stochastic Scaling

In general, pods have unequal demands.



Units are percentage of total link capacity.

Traffic demand matrix (TDM)

destination pods

source pods

	1	2	3	4
1	0.375	0.951	0.732	0.598
2	0.156	0.156	0.058	0.866
3	0.601	0.708	0.021	0.970
4	0.832	0.212	0.182	0.183

$$\text{TDM} = [m_{ij}]$$

$$\forall i \forall j \;\; 0 \leq m_{ij} \leq 1$$

An *admissible* TDM has all row sums ($\mathbf{R_i}$) and all column sums ($\mathbf{C_i}$) less than or equal to 1.

	1	2	3	4	R_i
1	0.375	0.951	0.732	0.598	2.656
2	0.156	0.156	0.058	0.866	1.236
3	0.601	0.708	0.021	0.970	2.300
4	0.832	0.212	0.182	0.183	1.417
C_i	1.964	2.027	0.993	2.617	

Admissible is also called *doubly substochastic*.

We want admissible TDMs because

- Pods cannot send more than their link capacity.
- Pods cannot receive more than their link capacity.

Any TDM can be made admissible with *linear scaling*.

	1	2	3	4	R_i
1	0.375	0.951	0.732	0.598	2.656 ← **c**
2	0.156	0.156	0.058	0.866	1.236
3	0.601	0.708	0.021	0.970	2.300
4	0.832	0.212	0.182	0.183	1.417
C_i	1.964	2.027	0.993	2.617	

Step 1. Choose $\mathbf{c} = \mathbf{max}(\mathbf{R_i}, \mathbf{C_i})$

Any TDM can be made admissible with *linear scaling*.

$$\frac{1}{2.656}\begin{bmatrix} 0.375 & 0.951 & 0.732 & 0.598 \\ 0.156 & 0.156 & 0.058 & 0.866 \\ 0.601 & 0.708 & 0.021 & 0.970 \\ 0.832 & 0.212 & 0.182 & 0.183 \end{bmatrix} \rightarrow \begin{bmatrix} 0.141 & 0.358 & 0.276 & 0.225 \\ 0.059 & 0.059 & 0.022 & 0.326 \\ 0.226 & 0.267 & 0.008 & 0.365 \\ 0.313 & 0.080 & 0.069 & 0.069 \end{bmatrix}\begin{matrix} 1.000 \\ 0.466 \\ 0.866 \\ 0.531 \end{matrix}$$

$$\begin{matrix} 0.739 & 0.764 & 0.375 & 0.985 \end{matrix}$$

Step 2. Multiply TDM by 1/c.

Linear scaling is unfair.

$$
\frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{array}{c} \begin{bmatrix} 0.250 & 0.250 & 0.250 & 0.250 \\ 0 & 0 & 0.250 & 0 \\ 0 & 0 & 0 & 0.250 \\ 0.250 & 0 & 0 & 0 \end{bmatrix} \begin{array}{c} 1.000 \\ 0.250 \\ 0.250 \\ 0.250 \end{array} \\ \begin{array}{cccc} 0.500 & 0.250 & 0.500 & 0.500 \end{array} \end{array}
$$

Instead of scaling the TDM, we will *allocate* the bandwidth.

Bandwidth allocation matrix (BAM)

	1	2	3	4	R_i
1	0.094	0.286	0.521	0.100	1.000
2	0.144	0.173	0.152	0.531	1.000
3	0.279	0.394	0.027	0.299	1.000
4	0.483	0.147	0.299	0.071	1.000
C_i	1.000	1.000	1.000	1.000	

$$\forall i \; R_i = 1$$
$$\forall i \; C_i = 1$$

A BAM is called *doubly stochastic*.

Stochastic scaling uses the TDM to compute the BAM.

TDM

	1	2	3	4	R_i
1	0.375	0.951	0.732	0.598	2.656
2	0.156	0.156	0.058	0.866	1.236
3	0.601	0.708	0.021	0.970	2.300
4	0.832	0.212	0.182	0.183	1.417
C_i	1.964	2.027	0.993	2.617	

$\xrightarrow{f}$

BAM

	1	2	3	4	R_i
1	0.094	0.286	0.521	0.100	1.000
2	0.144	0.173	0.152	0.531	1.000
3	0.279	0.394	0.027	0.299	1.000
4	0.483	0.147	0.299	0.071	1.000
C_i	1.000	1.000	1.000	1.000	

Sinkhorn (1964)

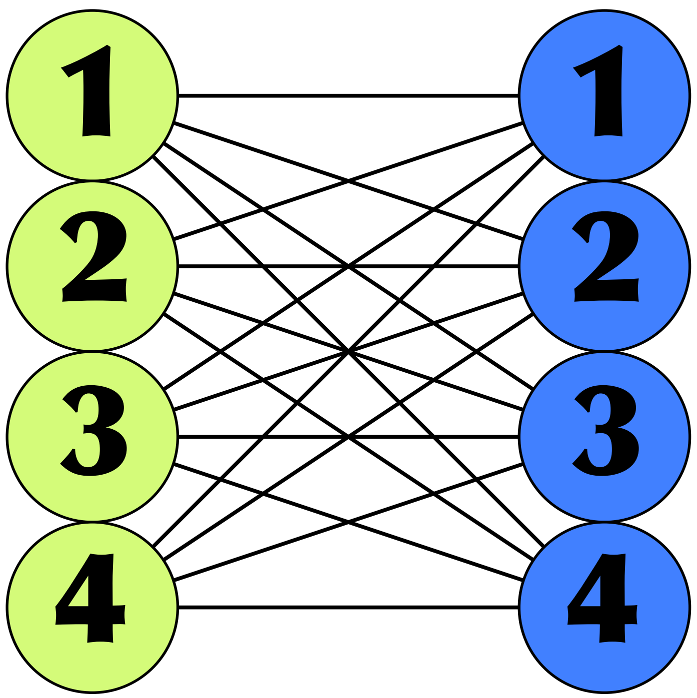
$$\text{TDM} \rightarrow \text{BAM}^{(0)}$$

$$\begin{bmatrix} {}^{1}/_{R_1} & 0 & \cdots & 0 \\ 0 & {}^{1}/_{R_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {}^{1}/_{R_N} \end{bmatrix} \text{BAM}^{(i)} \begin{bmatrix} {}^{1}/_{C_1} & 0 & \cdots & 0 \\ 0 & {}^{1}/_{C_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {}^{1}/_{C_N} \end{bmatrix} \rightarrow \text{BAM}^{(i+1)}$$

Stop when you have enough precision.

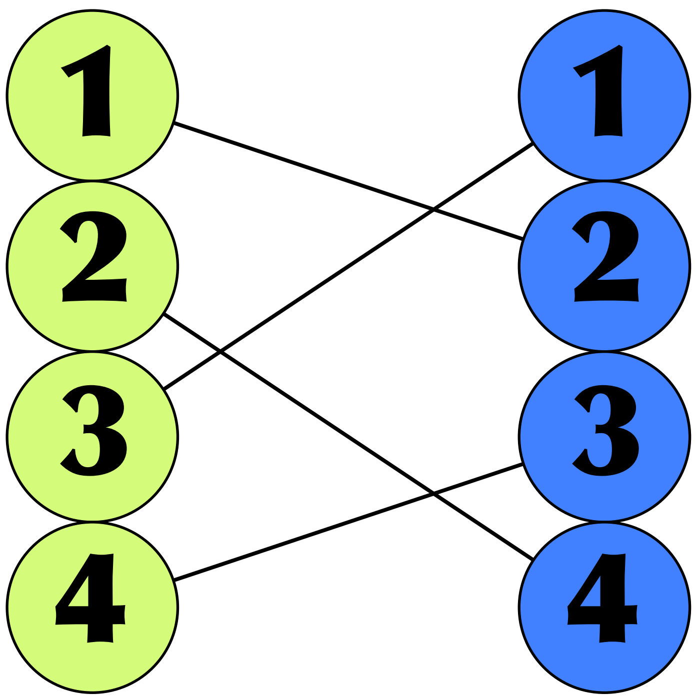
Hotspot Scheduling

A circuit switch can be represented as both a bipartite graph and as an adjacency matrix.



	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1

A *maximal assignment* on the bipartite graph is a *permutation matrix*.



	1	2	3	4
1	0	1	0	0
2	0	0	0	1
3	1	0	0	0
4	0	0	1	0

There are O(N!) maximal assignments.

Compute the max-weighted maximal matching (the maximum matching) on the BAM.

BAM

	1	2	3	4
1	0.094	0.286	0.521	0.100
2	0.144	0.173	0.152	0.531
3	0.279	0.394	0.027	0.299
4	0.483	0.147	0.299	0.071

→

Assignment

	1	2	3	4
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0
4	1	0	0	0

Kuhn-Munkres (1955) is $O(N^3)$.

Who uses Hotspot Scheduling?

c-Through [HotNets '09, SIGCOMM '10]

Flyways [HotNets '09, SIGCOMM '11]

Helios [SIGCOMM '10, OFC '11]

OSA [HotNets '10, NSDI '12]

MirrorMirror [HotNets '11, SIGCOMM '12]

How does Hotspot Scheduling perform?

	1	2	3	4
1	0	1	0	0
2	0	0	0	1
3	1	0	0	0
4	0	0	1	0

100% Throughput

	1	2	3	4
1	1/4	1/4	1/4	1/4
2	1/4	1/4	1/4	1/4
3	1/4	1/4	1/4	1/4
4	1/4	1/4	1/4	1/4

25% Throughput

Hotspot Scheduling is best for hotspot traffic and worst for all-to-all traffic.

Problems

1. **Lengthy computation before every circuit switch reconfiguration.**
2. **Speeding up the switch technology does not speed up the computation.**
3. **Performance is too dependent on communication patterns.**

Traffic Matrix Scheduling

Birkhoff-von Neumann Matrix Decomposition

BAM **Time Durations** **Assignments**

$$
\begin{bmatrix} 0.094 & 0.286 & 0.521 & 0.100 \\ 0.144 & 0.173 & 0.152 & 0.531 \\ 0.279 & 0.394 & 0.027 & 0.299 \\ 0.483 & 0.147 & 0.299 & 0.071 \end{bmatrix}
= 0.394 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}
+ 0.144 \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}
+ 0.137 \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
$$

$$
+ 0.094 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}
+ 0.071 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
+ 0.056 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}
$$

$$
+ 0.053 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
+ 0.027 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}
+ 0.018 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
+ 0.045 \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}
$$

Birkhoff-von Neumann Matrix Decomposition

BAM

$$\begin{bmatrix} 0.094 & 0.286 & 0.521 & 0.100 \\ 0.144 & 0.173 & 0.152 & 0.531 \\ 0.279 & 0.394 & 0.027 & 0.299 \\ 0.483 & 0.147 & 0.299 & 0.071 \end{bmatrix} = \sum_{i}^{k} c_i P_i$$

$$k \leq N^2 - 2N + 2$$

$$\forall i \;\; 0 < c_i \leq 1$$

$$\sum_{i}^{k} c_i = 1$$

The BvN expansion is a convex combination.

We don’t need to use all the terms.

BAM

$$\begin{bmatrix} 0.094 & 0.286 & 0.521 & 0.100 \\ 0.144 & 0.173 & 0.152 & 0.531 \\ 0.279 & 0.394 & 0.027 & 0.299 \\ 0.483 & 0.147 & 0.299 & 0.071 \end{bmatrix} > 0.394 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} + 0.144 \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In this example:

- **The first term is the same as Hotspot Scheduling.**
- **The first two terms provide 53.8% of the capacity.**

How does Traffic Matrix Scheduling perform?

100% Throughput

for any BAM

Problems

1. **Very expensive:** $\mathbf{O(N^{4.5})}$

2. $\mathbf{O(N^{2})}$ **terms in the worst case.**

Analysis

3. Analysis

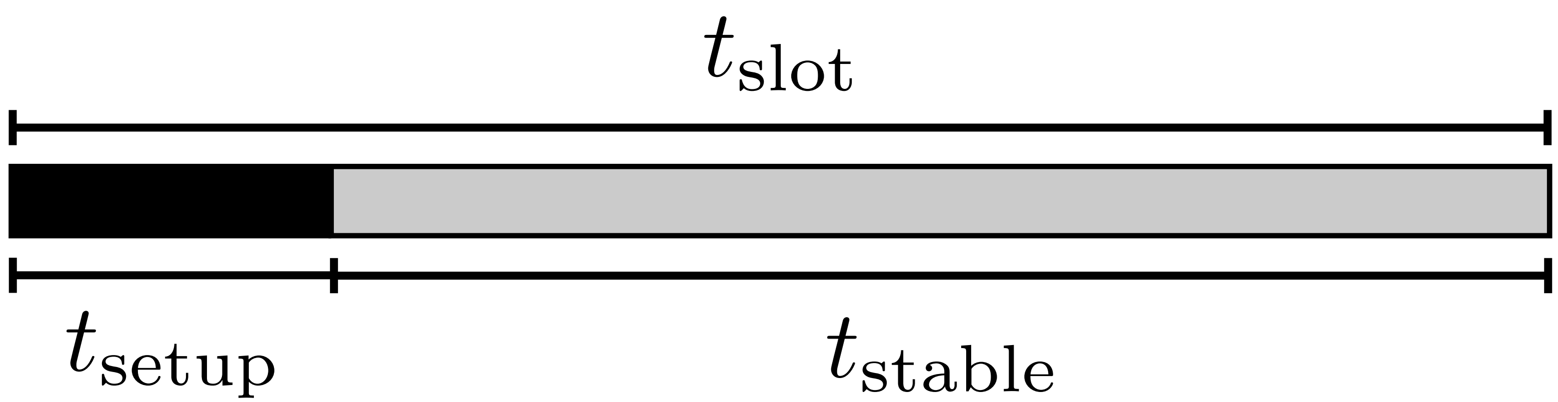
Switching Time (t_{setup})

Host Buffering

Offered Load

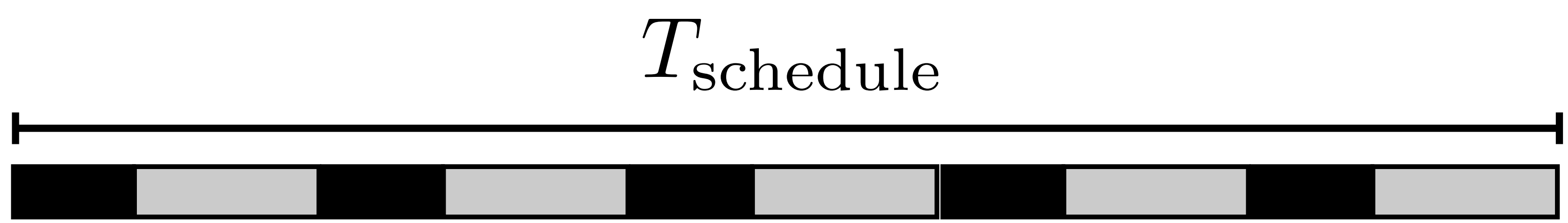
PSN Capacity

Duty Cycle
(equal-length time slots)



$$D = \frac{t_{\text{stable}}}{t_{\text{setup}} + t_{\text{stable}}} = \frac{t_{\text{stable}}}{t_{\text{slot}}}$$

Duty Cycle
(variable-length time slots)



$$T_{\text{setup}} = k\, t_{\text{setup}}$$

$$D = \frac{T_{\text{stable}}}{T_{\text{setup}} + T_{\text{stable}}} = \frac{T_{\text{stable}}}{T_{\text{schedule}}}$$

Effective Link Rate

$$L_{\text{effective}} = DL$$

**If your duty cycle is 90%,
then your 10G link becomes a 9G link.**

**Which is fine if your offered load
is not more than 9G.**

When can you achieve 100%?

n: number of time slots
CSN: circuit-switched network
PSN: packet-switched network
D: duty cycle

$$t_{\text{setup}} = 10\mu\text{s}$$
$$T_{\text{schedule}} = 1\text{ms}$$

n	CSN	PSN	D
0	0%	100.0%	N/A
1	39.4%	60.6%	100.0%
2	53.8%	46.2%	98.0%
3	63.8%	36.2%	97.0%
4	72.7%	27.3%	96.0%
5	80.6%	19.4%	95.0%
6	87.3%	12.7%	94.0%
7	92.3%	7.7%	93.0%
8	96.6%	3.4%	92.0%
9	99.3%	0.7%	91.0%
10	100.0%	0%	90.0%

Longest Time Slot First (LTF) Scheduling

If offered load is less than 9 Gb/s, we can achieve 100% throughput over the CSN.

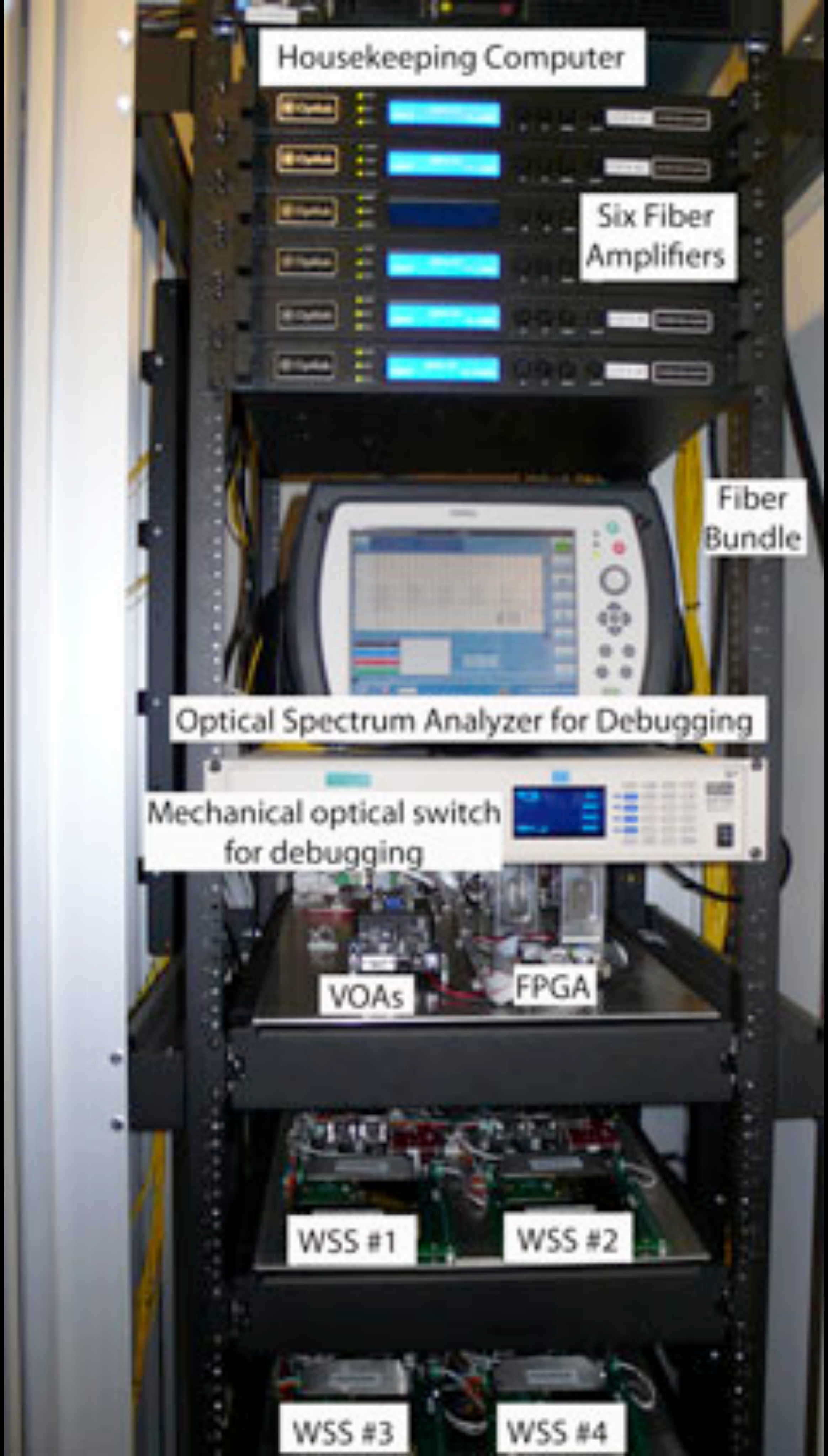
Host Buffer Requirements (all-to-all traffic)

$$B = L_{\text{effective}}\,(N-1)\,t_{\text{slot}}$$

Helios $B = 9\,\mathrm{Gb/s}\,(24-1)\,270\,\mathrm{ms} = 7.23\,\mathrm{GB}$

Mordia $B = 9\,\mathrm{Gb/s}\,(24-1)\,100\,\mu\mathrm{s} = 2.74\,\mathrm{MB}$

Each host requires this much buffering.


Housekeeping Computer
Six Fiber Amplifiers
Fiber Bundle
Optical Spectrum Analyzer for Debugging
Mechanical optical switch for debugging
VOAs
FPGA
WSS #1
WSS #2
WSS #3
WSS #4

mordia.net

Conclusion

Prior hybrid DCNs use *Hotspot Scheduling*

- **Throughput depends on workload**
- **Not clear how to benefit from faster switch technology**

We propose *Traffic Matrix Scheduling*

- **Achieves 100% throughput on all workloads**
- **Trading off:**
  - **Switching time**
  - **Host buffering**
  - **Offered load**
- **Able to use faster switch technology**
- **But also requires faster switch technology**

Backup Slides

Other Topics

Hosts and TDMA?

- Requires microsecond precision
- NIC vs kernel implementation

TCP?

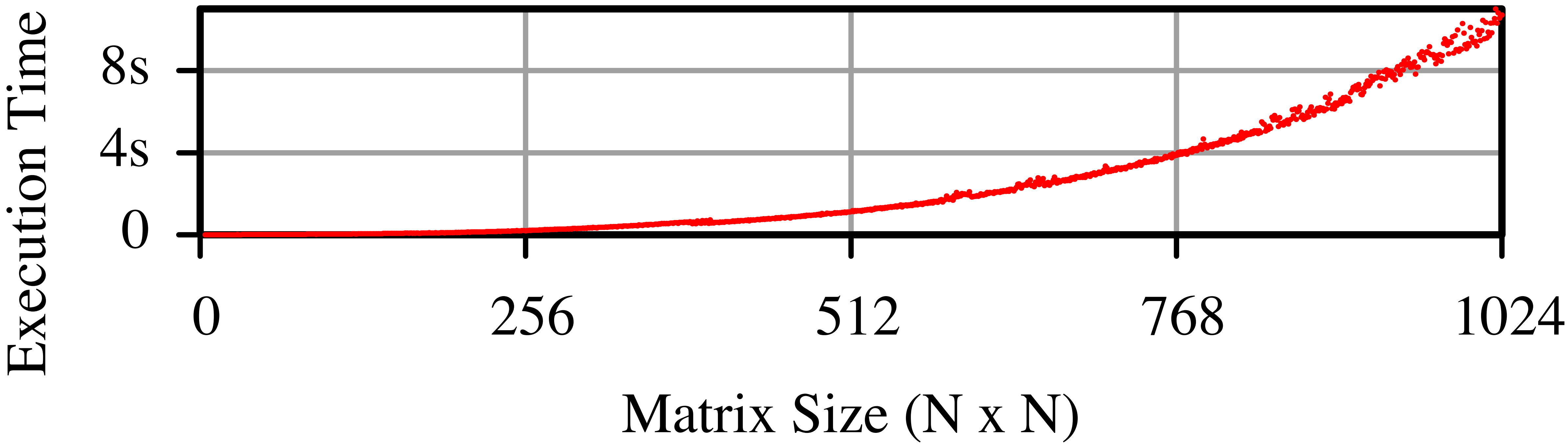
- 50% throughput on Mordia prototype

Latency-sensitive Traffic?

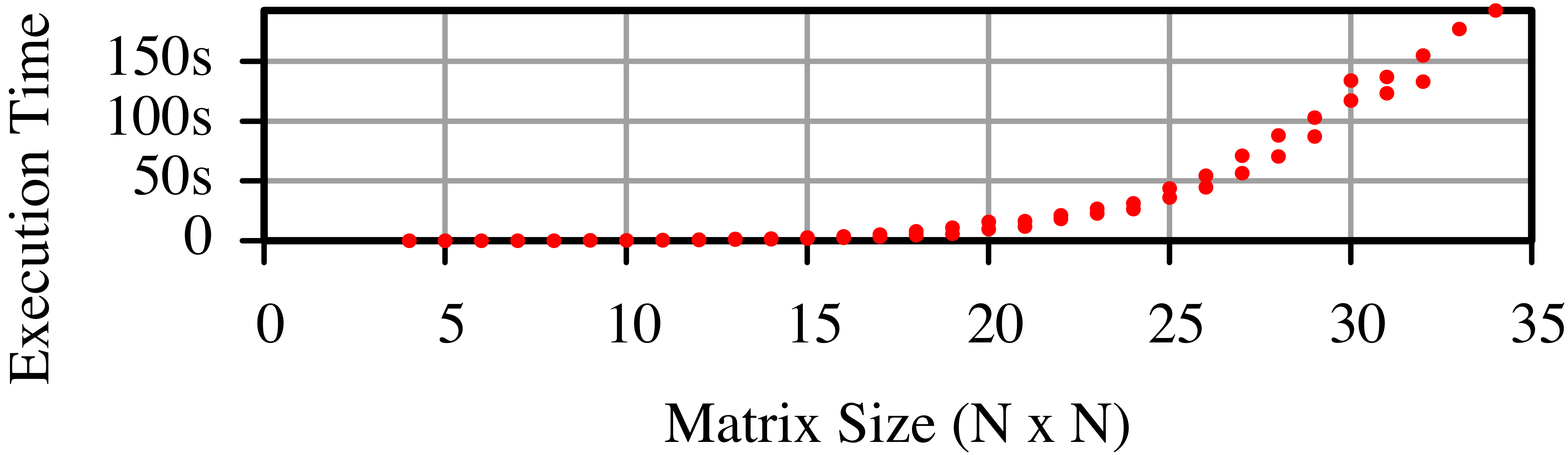
- Traffic Matrix Scheduling adds milliseconds of latency ($T_{schedule}$).

Mordia Optical Circuit Switch Prototype?

- No time in this talk, sorry.



Sinkhorn


Execution Time
150s
100s
50s
0
0
5
10
15
20
25
30
35
Matrix Size (N x N)

Birkhoff