Sunflow

Efficient Optical Circuit Scheduling for Coflows

Xin Sunny Huang, Xiaoye Steven Sun, T. S. Eugene Ng
Rice University

Big Data and Optical Lightpaths Driven Lab

This Work

- Optical Circuit Switching has many **advantages** over packet switching.

- **Disadvantage**: usually worse traffic performance.

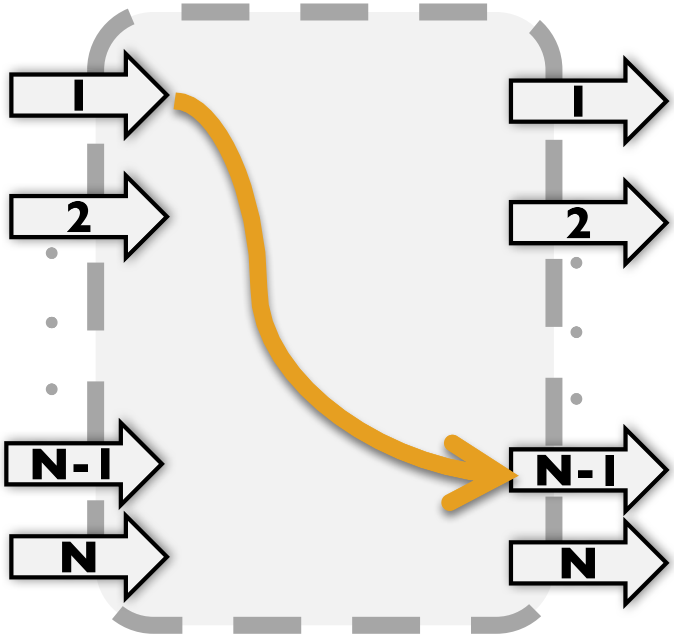
- **Sunflow** overcomes disadvantage with **efficient circuit scheduling**.

Optical Circuit Switch v.s. Electrical Packet Switch

	Optical Circuit Switch (OCS)	Electrical Packet Switch (EPS)	OCS better?
Traffic processing	No packet processing	Store and forward EACH packet	**Energy efficiency**
Bandwidth upgrade	Reuse old	Buy new	**Future proof, cost efficiency**
Switching speed			

Op Switch v.s. E et Switch

Optical Circuit Switch



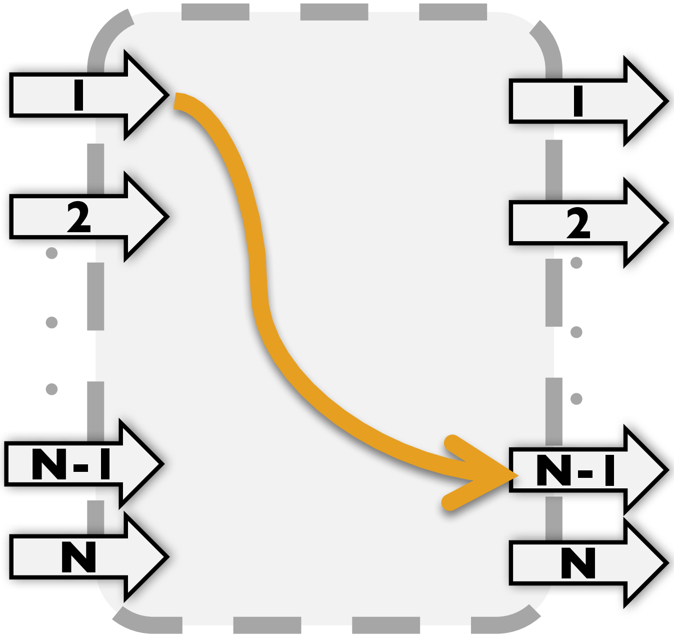
Send at “link rate”, up to **10s of ms switching delay**

		cal Packet ch (EPS)	OCS better?
Traffic processin		and forward H packet	**Energy efficiency**
Bandwidt upgrade		uy new	**Future proof, cost efficiency**
Switching speed			

Op S
E

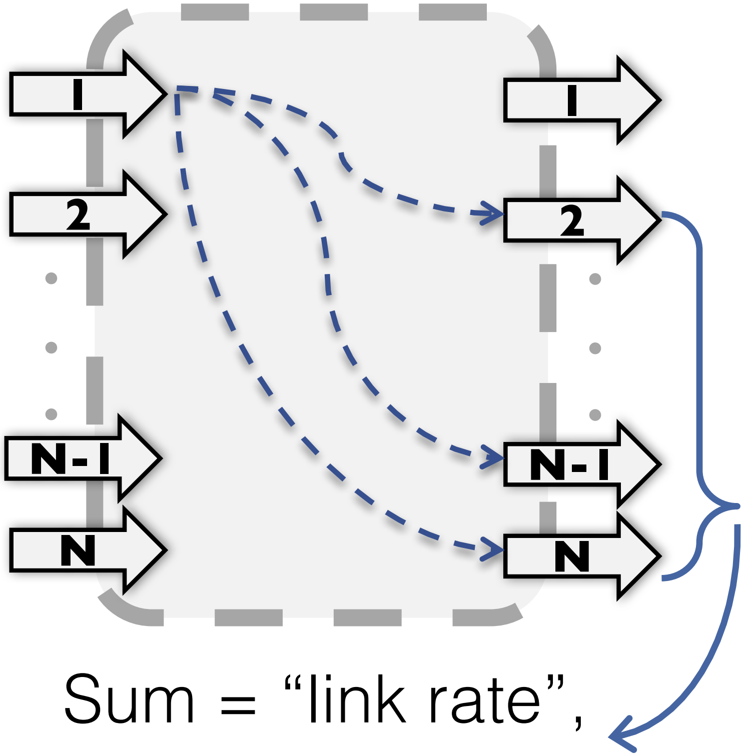
Traffic processin			
Bandwidt upgrade			
Switching speed			

Optical Circuit Switch



Send at “link rate”, up to **10s of ms switching delay**

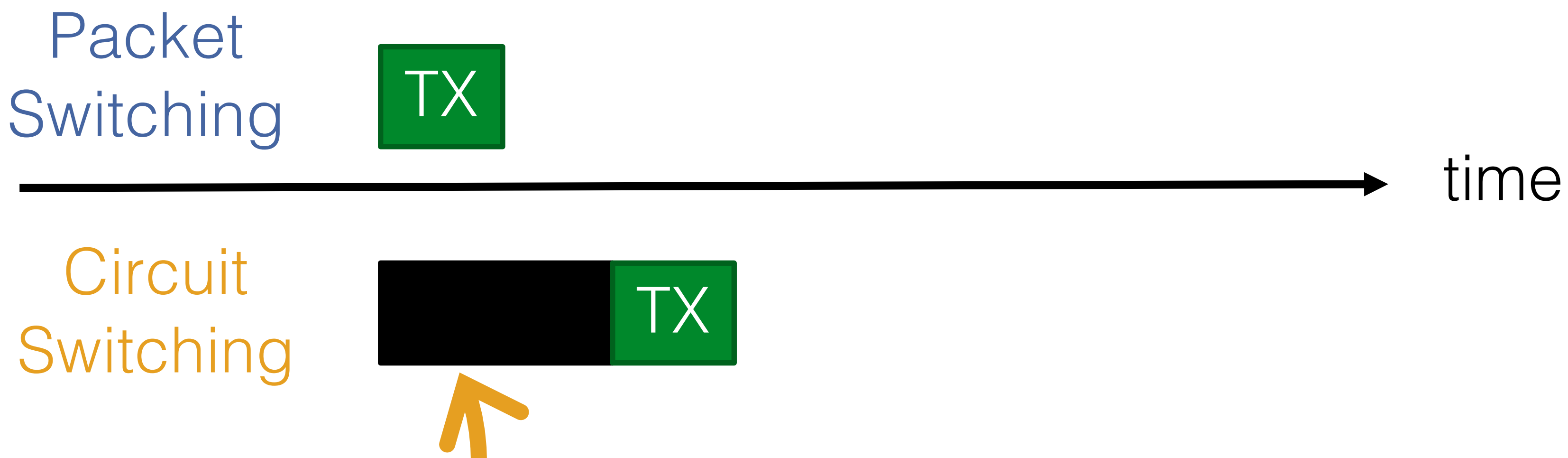
Electrical Packet Switch



Sum = “link rate”, fast switching.

Optical Circuit Switch v.s. Electrical Packet Switch

	Optical Circuit Switch (OCS)	**Electrical Packet Switch (EPS)**	**OCS better?**
Traffic processing	No packet processing	Store and forward EACH packet	**Energy efficiency**
Bandwidth upgrade	Reuse old	Buy new	**Future proof, cost efficiency**
Switching speed	Setting up a circuit up to 10s of ms	Packet granularity 10s of ns	**Traffic delay**



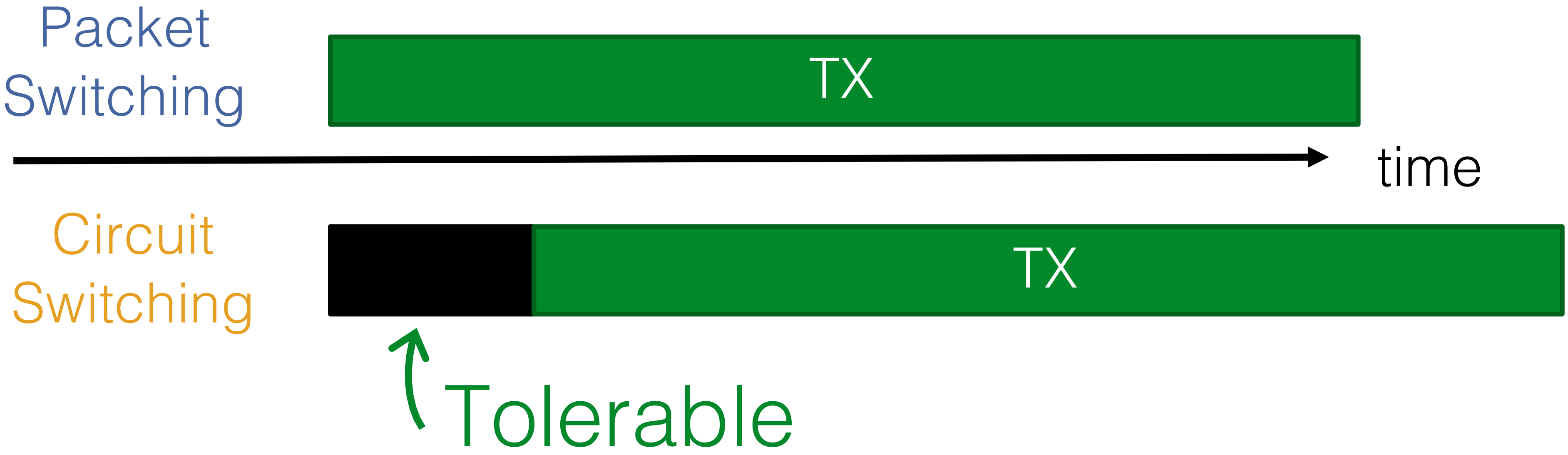
Due to circuit switching delay, the performance of circuit switching is usually **worse** than packet switching for **small data.**

Packet Switching

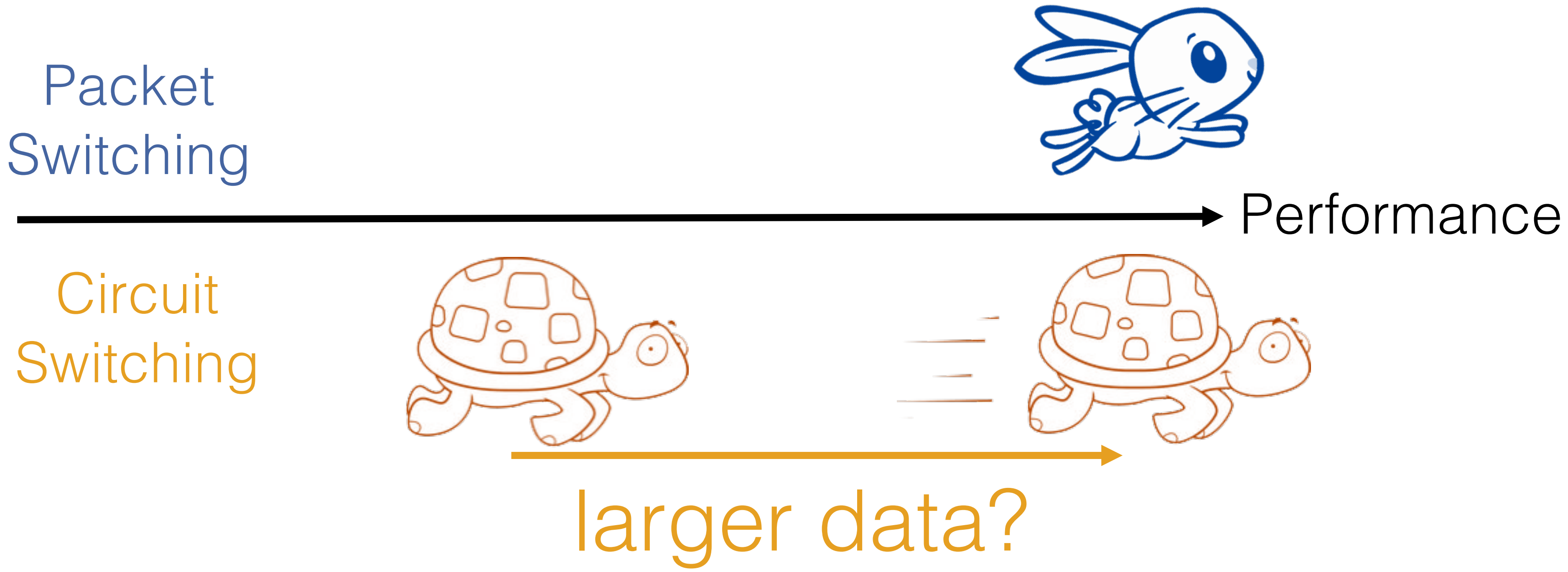
Performance

Circuit Switching

Due to circuit switching delay, the performance of circuit switching is usually **worse** than packet switching for **small data.**



For **larger data**, performance of circuit switching may become **closer** to packet switching.



Packet Switching

Performance

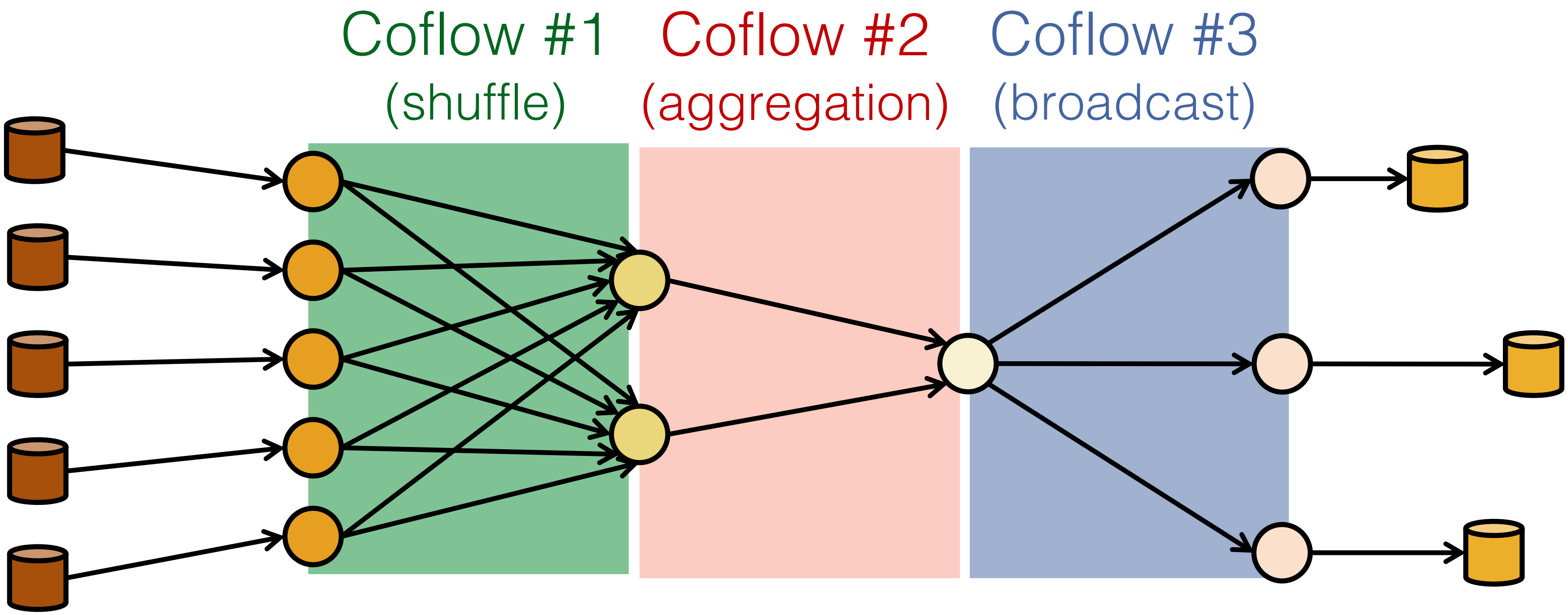
Circuit Switching

larger data?

Fundamental question: Can circuit-switching be **as good as** packet-switching for **big data** traffic?

Big data often comes in Coflows

- Coflow [1] : A set of parallel flows.
- Produced by distributed applications (e.g. Hadoop & Spark).
- Performance is measured by Coflow Completion Time (CCT), i.e. the last flow's completion time.

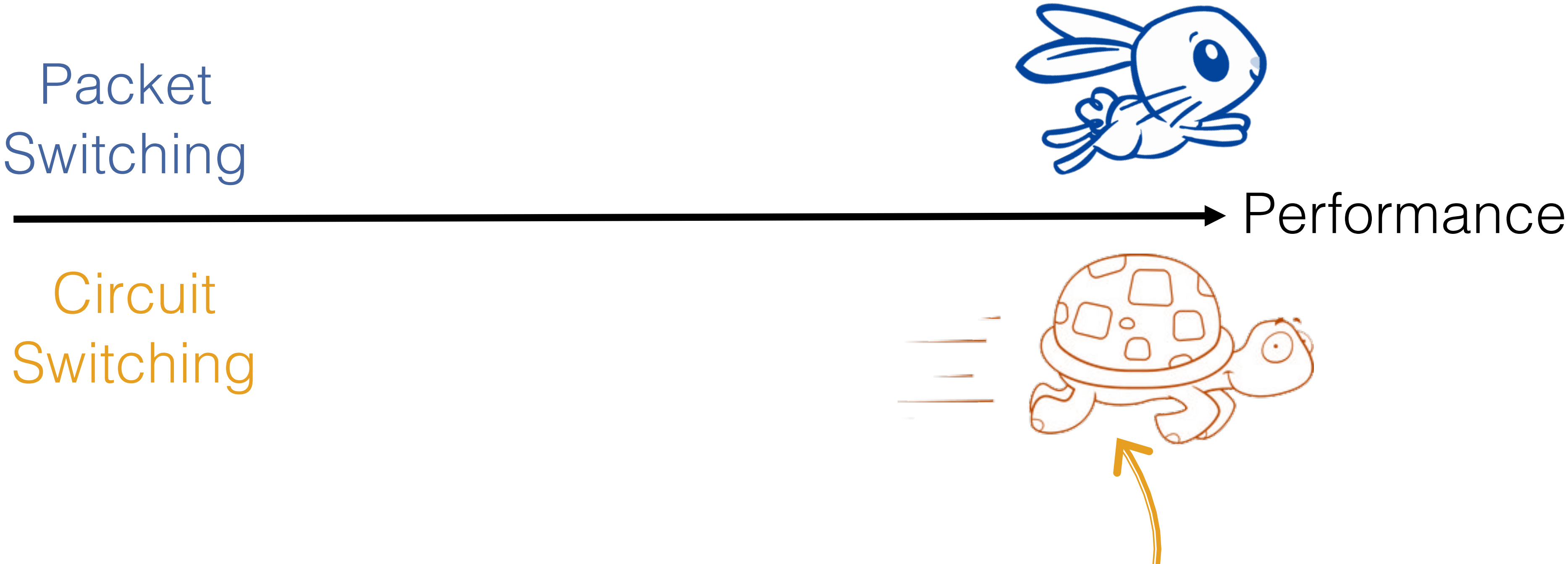


Coflow #1 (shuffle)

Coflow #2 (aggregation)

Coflow #3 (broadcast)

[1] Chowdhury, M. et al. Coflow: An application layer abstraction for cluster networking. (HotNets'12)

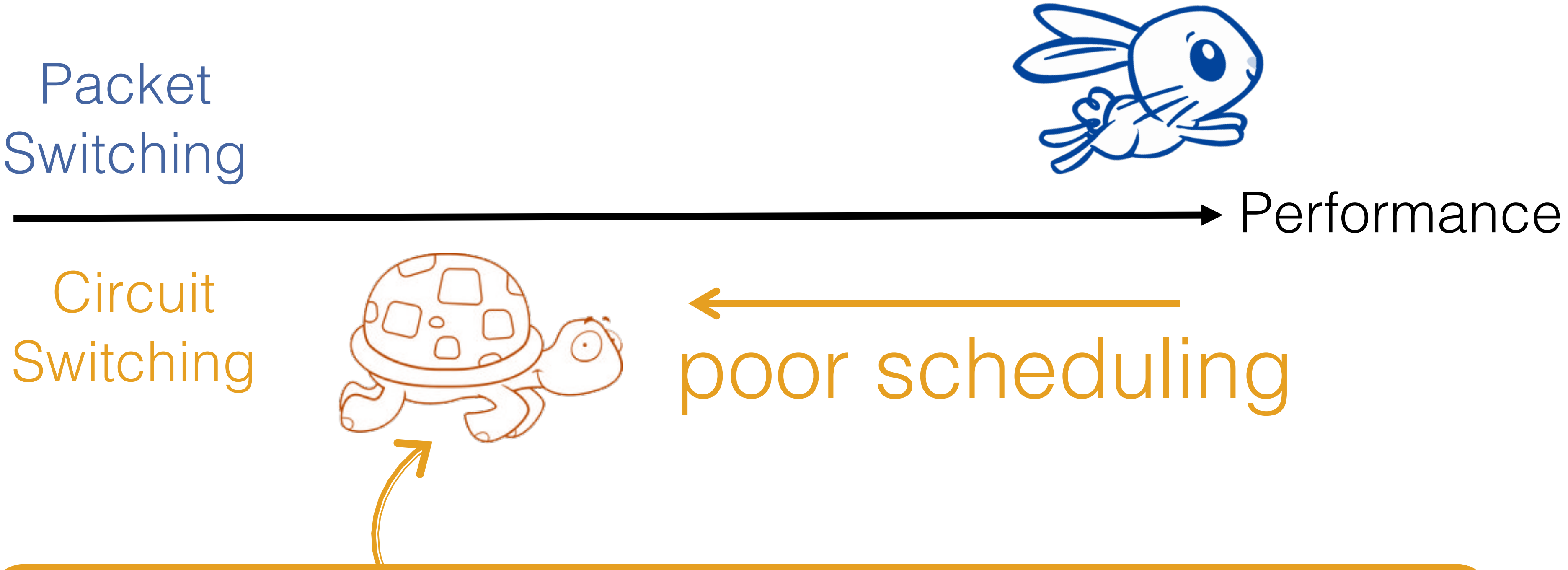


Packet Switching

Performance

Circuit Switching

Fundamental question: Can circuit-switching be as good as packet-switching for **Coflow** traffic?



Packet Switching

Performance

Circuit Switching

poor scheduling

Existing circuit scheduling algorithms: performance suffers from **inefficient** scheduling.

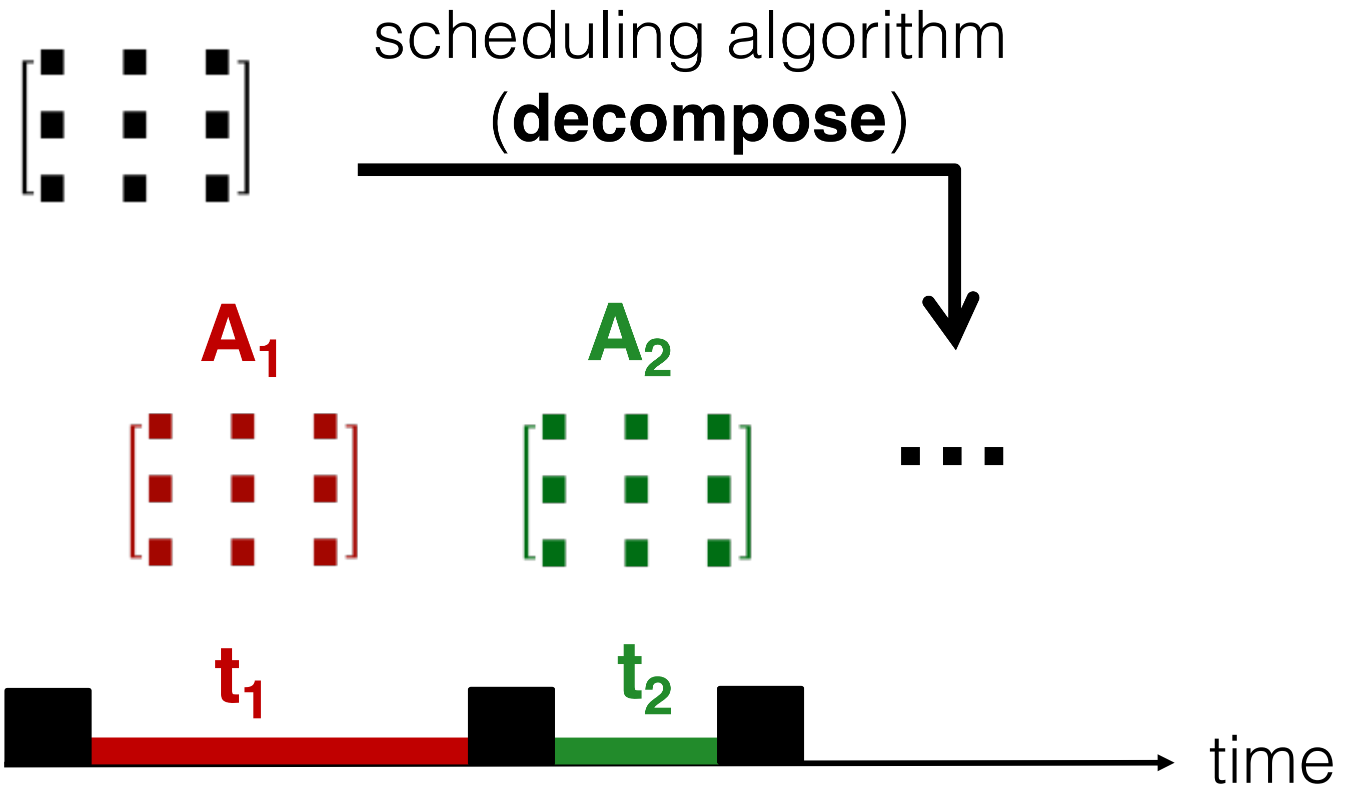
Solstice (CoNEXT'15), **TMS** (SIGCOMM'13), **Max-weighted-matching** in c-through (SIGCOMM'10) and Helios (SIGCOMM'10).

Existing circuit scheduling algorithms all rely on matrix decomposition

Aggregated traffic demand (**matrix**)

A set of circuits (**assignment**)

Active for (**time**)



scheduling algorithm (**decompose**)

A_1

A_2

…

t_1

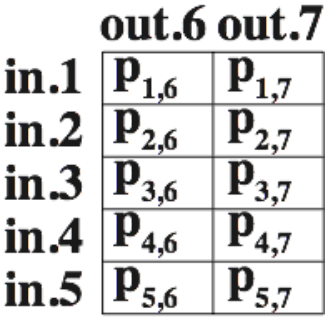
t_2

time

Solstice (CoNEXT'15), **TMS** (SIGCOMM'13), **Max-weighted-matching** in c-through (SIGCOMM'10) and Helios (SIGCOMM'10).

Intra-Coflow circuit scheduling

Coflow
demand matrix



	out.6	out.7
in.1	$p_{1,6}$	$p_{1,7}$
in.2	$p_{2,6}$	$p_{2,7}$
in.3	$p_{3,6}$	$p_{3,7}$
in.4	$p_{4,6}$	$p_{4,7}$
in.5	$p_{5,6}$	$p_{5,7}$

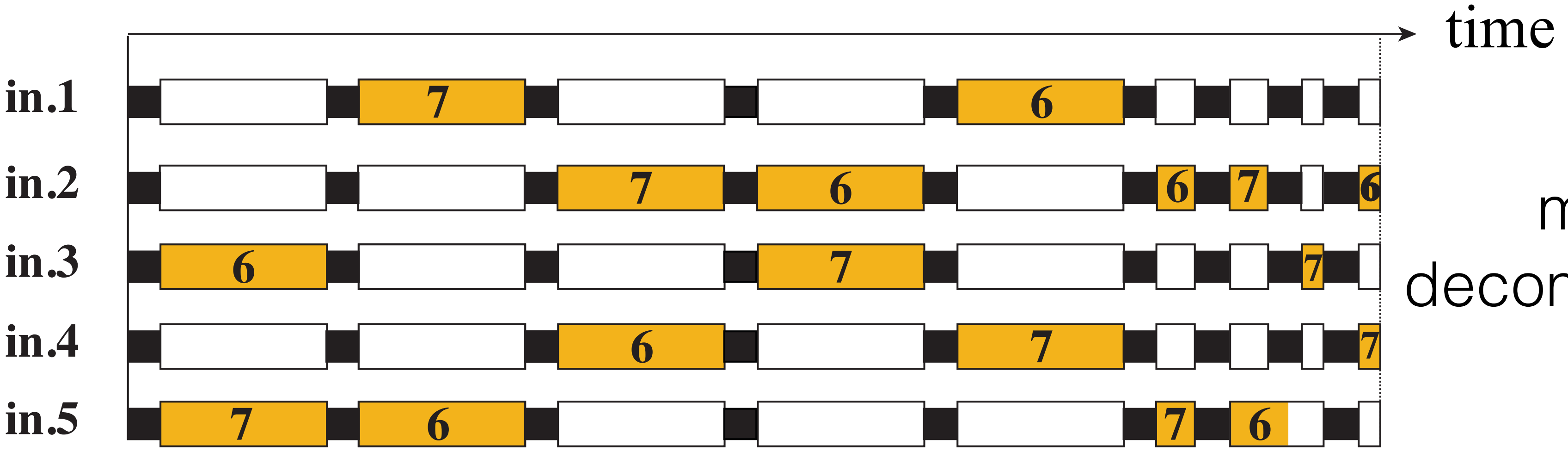
scheduling algorithm
(**decompose**)

Intra-Coflow circuit scheduling

Coflow demand matrix

	out.6	out.7
in.1	$p_{1,6}$	$p_{1,7}$
in.2	$p_{2,6}$	$p_{2,7}$
in.3	$p_{3,6}$	$p_{3,7}$
in.4	$p_{4,6}$	$p_{4,7}$
in.5	$p_{5,6}$	$p_{5,7}$

scheduling algorithm (**decompose**)



matrix decomposition*

← duration →

■ **reconfiguration** **output port No.** **circuit scheduled** □ **idle** 🟨 🟦 **transmitting**

* This example is produced by the most efficient algorithm among the existing ones, Solstice (CoNEXT'15).

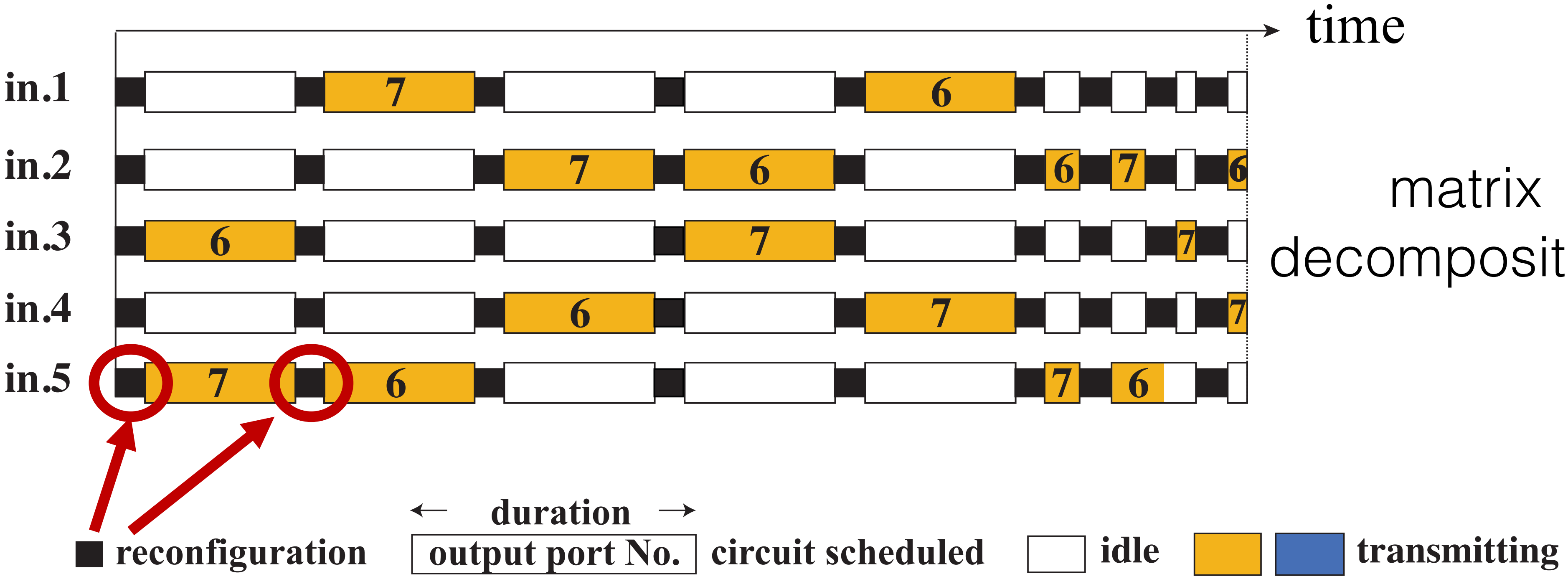
Intra-Coflow circuit scheduling

Coflow demand matrix

	out.6	out.7
in.1	$p_{1,6}$	$p_{1,7}$
in.2	$p_{2,6}$	$p_{2,7}$
in.3	$p_{3,6}$	$p_{3,7}$
in.4	$p_{4,6}$	$p_{4,7}$
in.5	$p_{5,6}$	$p_{5,7}$

scheduling algorithm (**decompose**)

matrix decomposition*



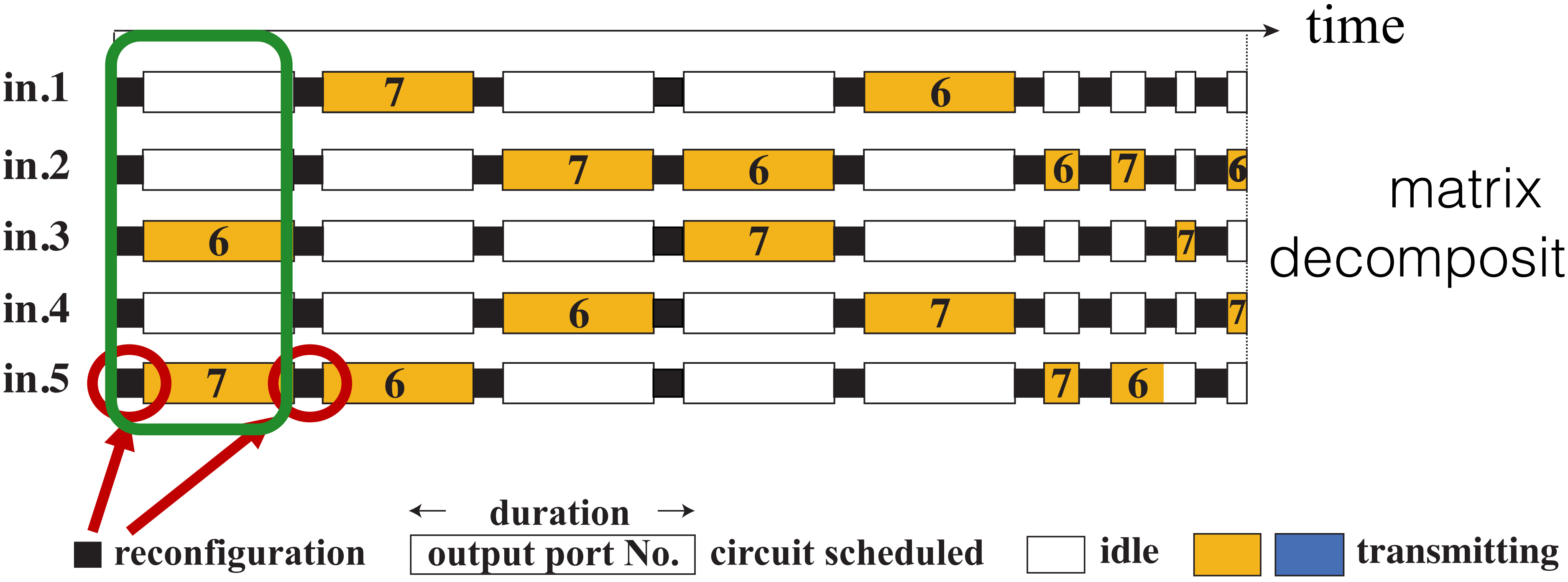
* This example is produced by the most efficient algorithm among the existing ones, Solstice (CoNEXT'15).

Intra-Coflow circuit scheduling

Coflow demand matrix

$$\begin{array}{c|c|c|}
 & \text{out.6} & \text{out.7} \\
\hline
\text{in.1} & p_{1,6} & p_{1,7} \\
\hline
\text{in.2} & p_{2,6} & p_{2,7} \\
\hline
\text{in.3} & p_{3,6} & p_{3,7} \\
\hline
\text{in.4} & p_{4,6} & p_{4,7} \\
\hline
\text{in.5} & p_{5,6} & p_{5,7} \\
\hline
\end{array}$$

scheduling algorithm (**decompose**)



matrix decomposition*

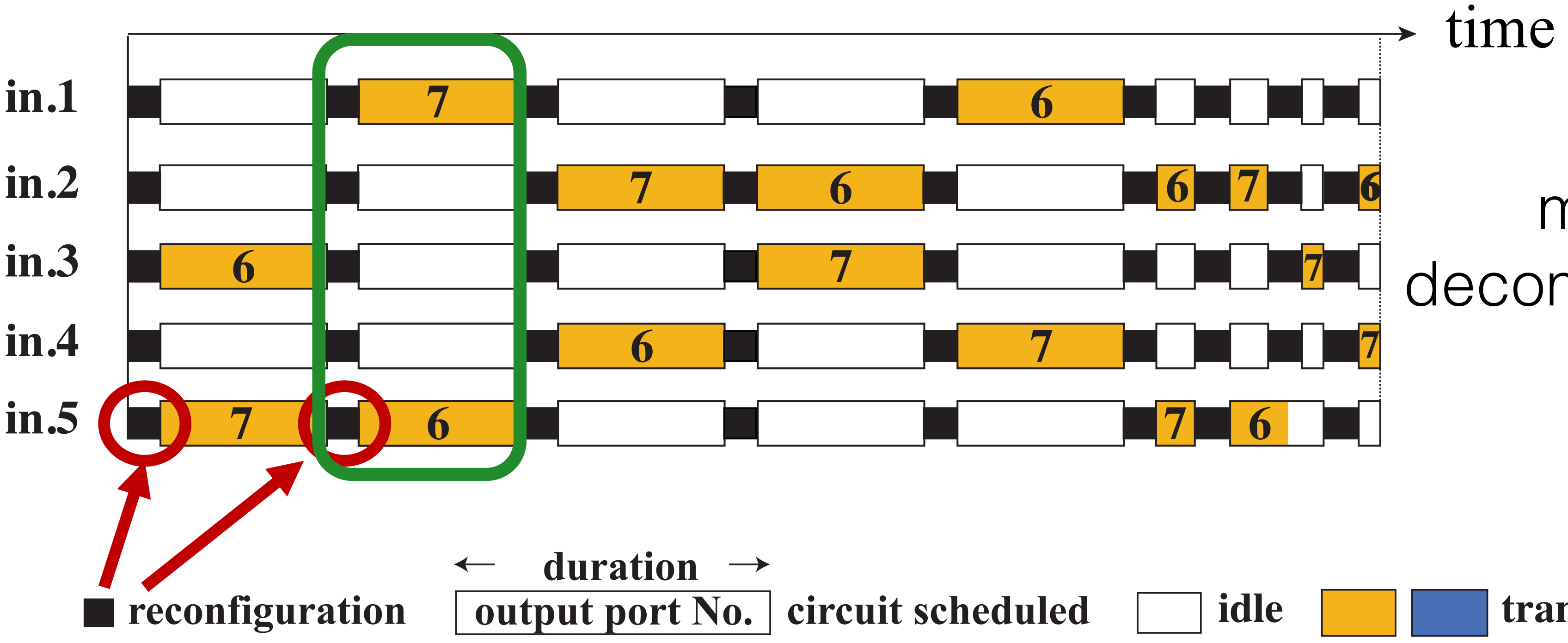
* This example is produced by the most efficient algorithm among the existing ones, Solstice (CoNEXT'15).

Intra-Coflow circuit scheduling

Coflow demand matrix

	out.6	out.7
in.1	$p_{1,6}$	$p_{1,7}$
in.2	$p_{2,6}$	$p_{2,7}$
in.3	$p_{3,6}$	$p_{3,7}$
in.4	$p_{4,6}$	$p_{4,7}$
in.5	$p_{5,6}$	$p_{5,7}$

scheduling algorithm (**decompose**)



matrix decomposition*

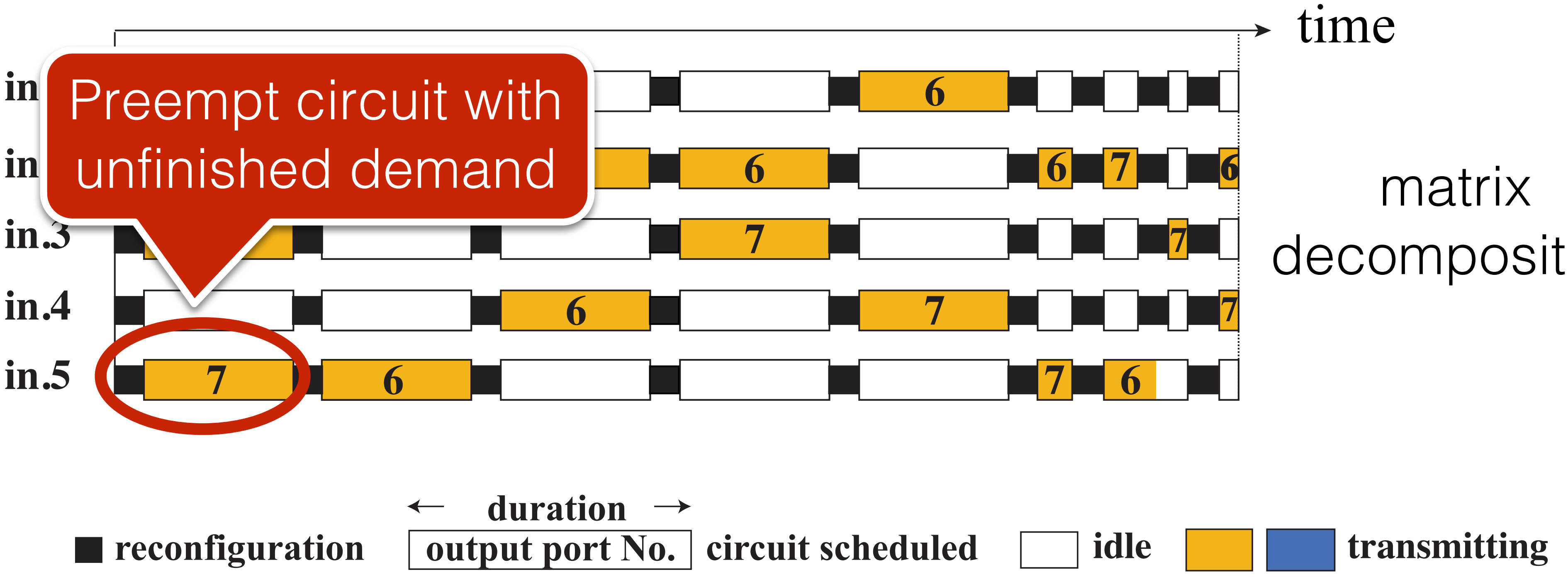
* This example is produced by the most efficient algorithm among the existing ones, Solstice (CoNEXT'15).

Intra-Coflow circuit scheduling

Coflow demand matrix

$$\begin{array}{c|c|c|}
 & \textbf{out.6} & \textbf{out.7} \\
\hline
\textbf{in.1} & p_{1,6} & p_{1,7} \\
\hline
\textbf{in.2} & p_{2,6} & p_{2,7} \\
\hline
\textbf{in.3} & p_{3,6} & p_{3,7} \\
\hline
\textbf{in.4} & p_{4,6} & p_{4,7} \\
\hline
\textbf{in.5} & p_{5,6} & p_{5,7} \\
\hline
\end{array}$$

scheduling algorithm (**decompose**)



Preempt circuit with unfinished demand

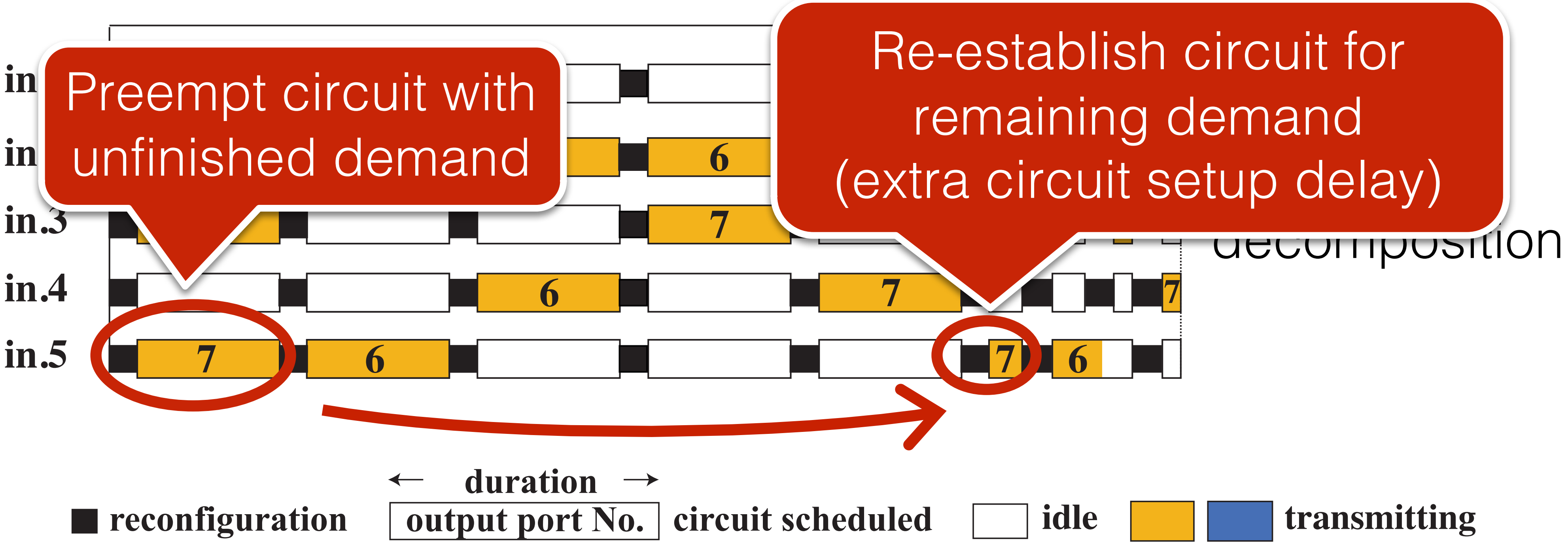
matrix decomposition

Intra-Coflow circuit scheduling

Coflow demand matrix

	out.6	out.7
in.1	$p_{1,6}$	$p_{1,7}$
in.2	$p_{2,6}$	$p_{2,7}$
in.3	$p_{3,6}$	$p_{3,7}$
in.4	$p_{4,6}$	$p_{4,7}$
in.5	$p_{5,6}$	$p_{5,7}$

scheduling algorithm (**decompose**)



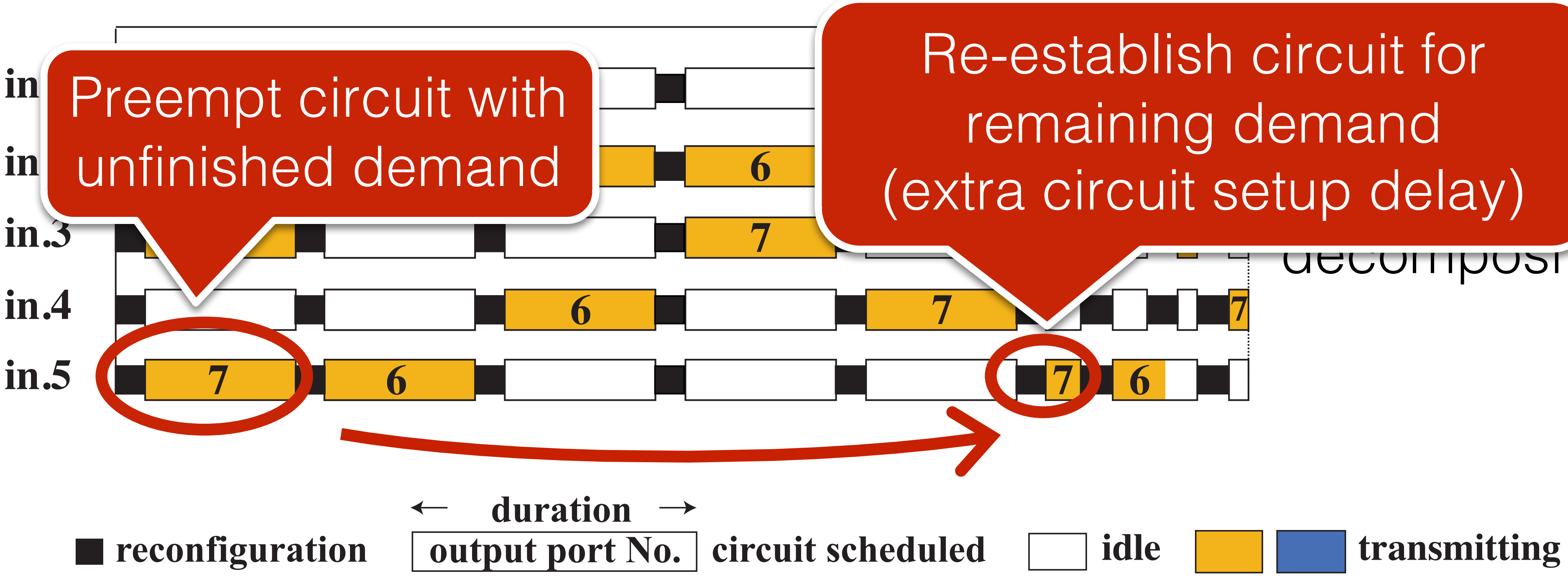
Preempt circuit with unfinished demand

Re-establish circuit for remaining demand (extra circuit setup delay)

Intra-Coflow circuit scheduling

Overly strong assumption:
All-stop switch model

deman



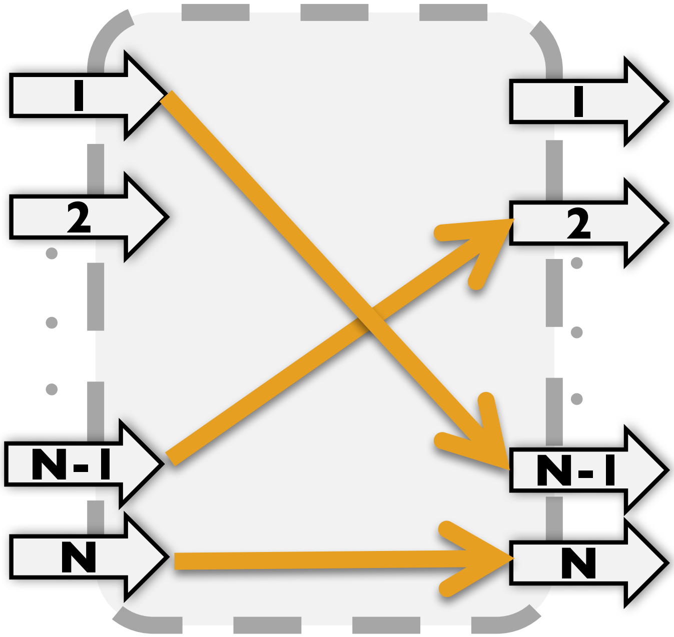
Preempt circuit with unfinished demand

Re-establish circuit for remaining demand (extra circuit setup delay)

decomposition

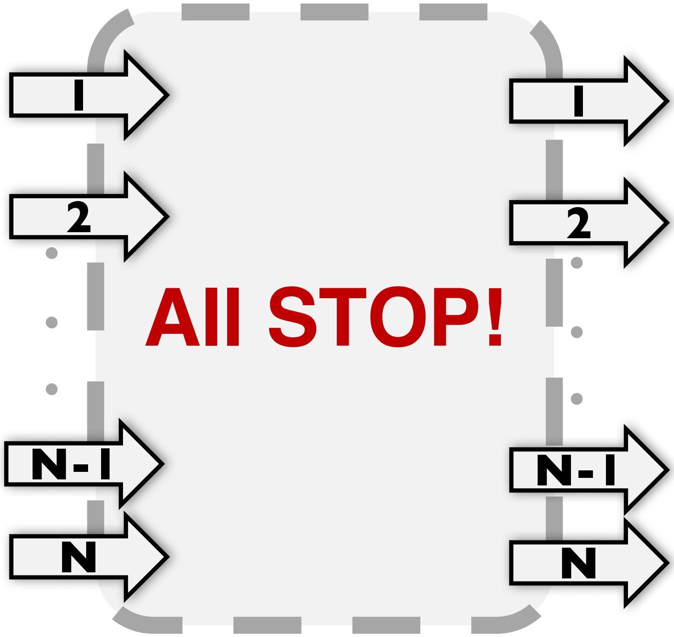
All-stop Model

Too strong: All circuits stop during switching.



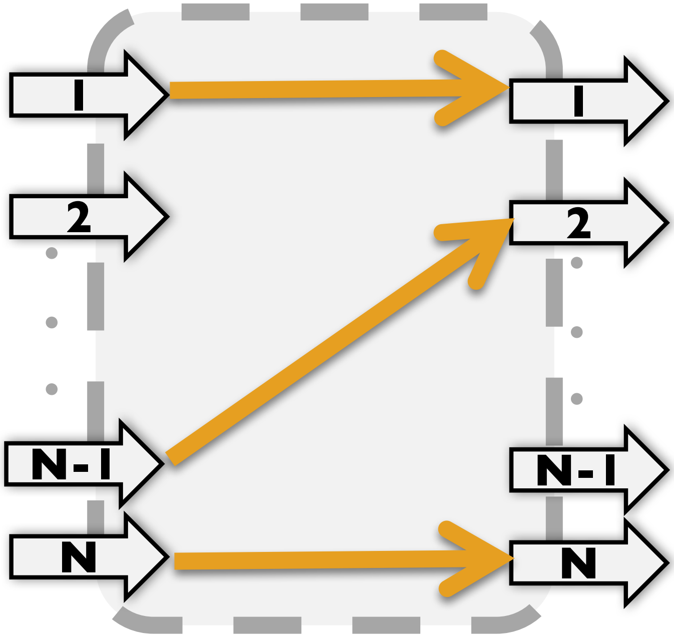
All-stop Model

Too strong: All circuits stop during switching.



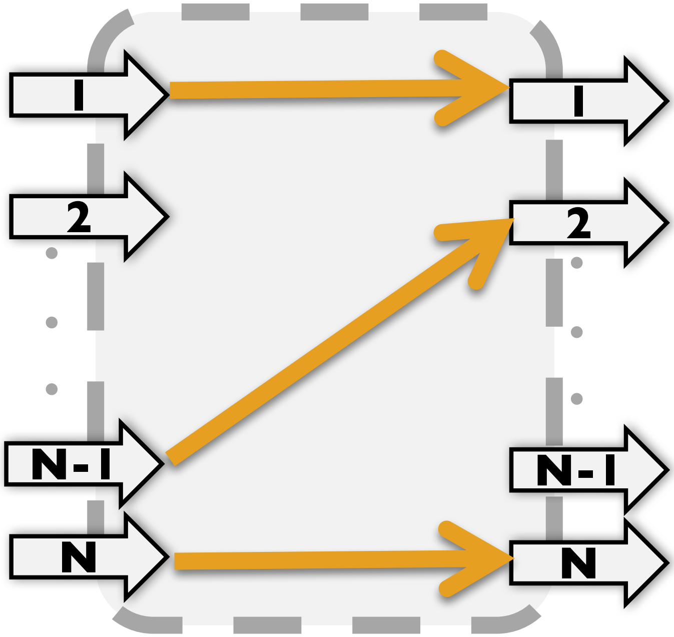
All-stop Model

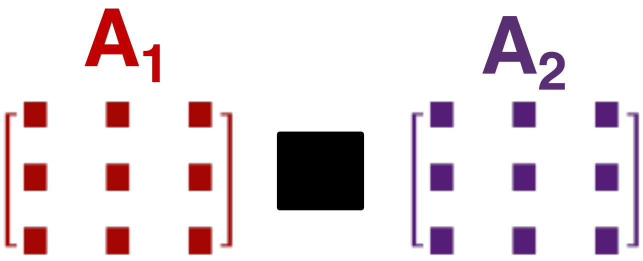
Too strong: All circuits stop during switching.



All-stop Model

Too strong: All circuits stop during switching.

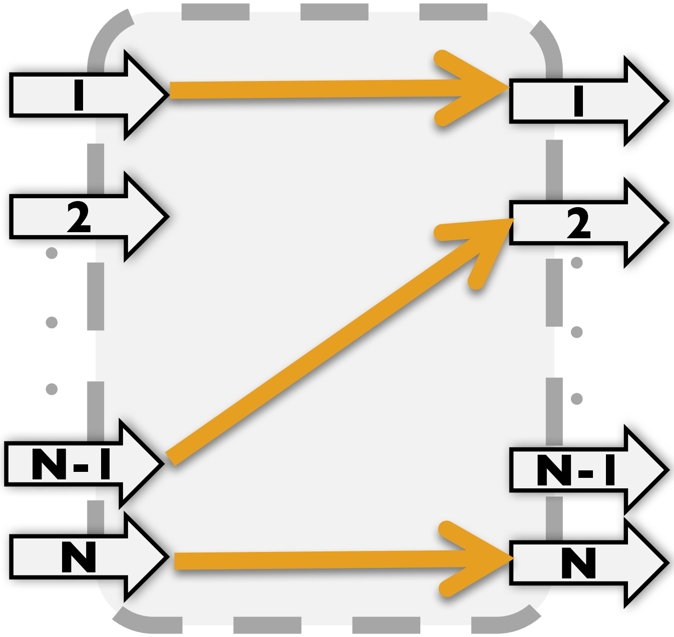




$A_1 \neq A_2$: **NO** incentive to extend any circuit from A_1 to A_2

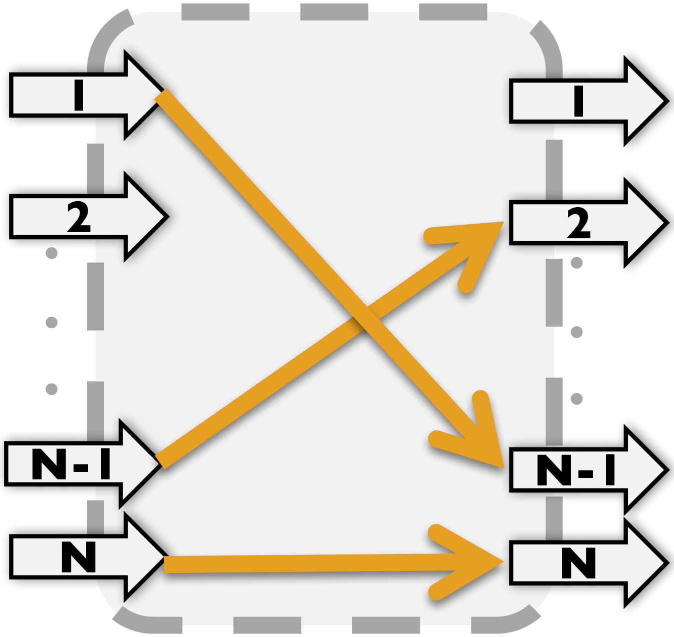
All-stop Model

Too strong: All circuits stops during switching.



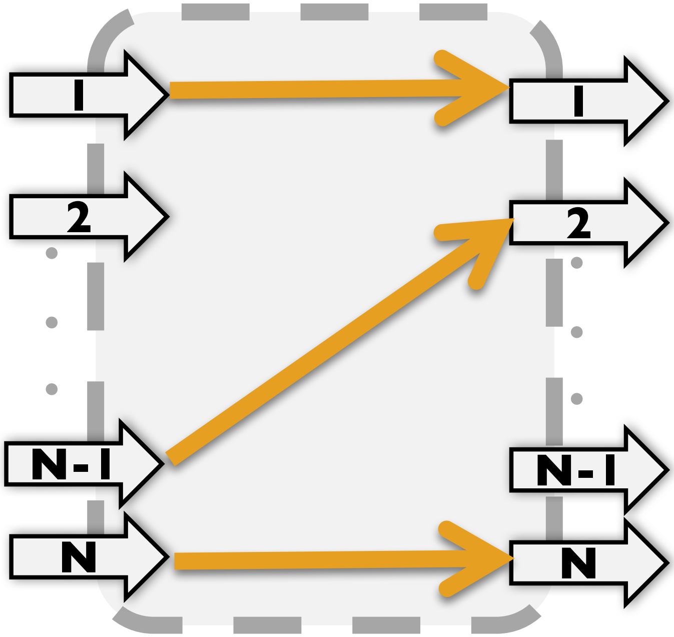
Not-all-stop Model

In practice: Unchanged circuits remain active.



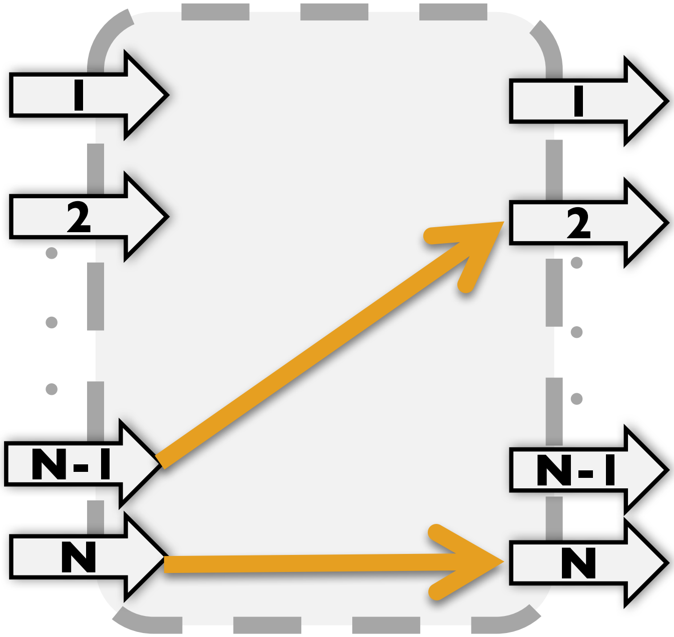
All-stop Model

Too strong: All circuits stops during switching.



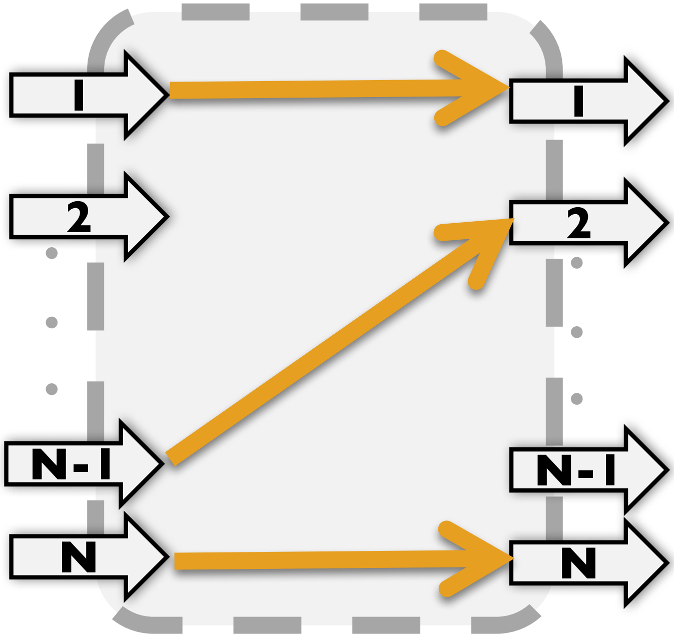
Not-all-stop Model

In practice: Unchanged circuits remain active.



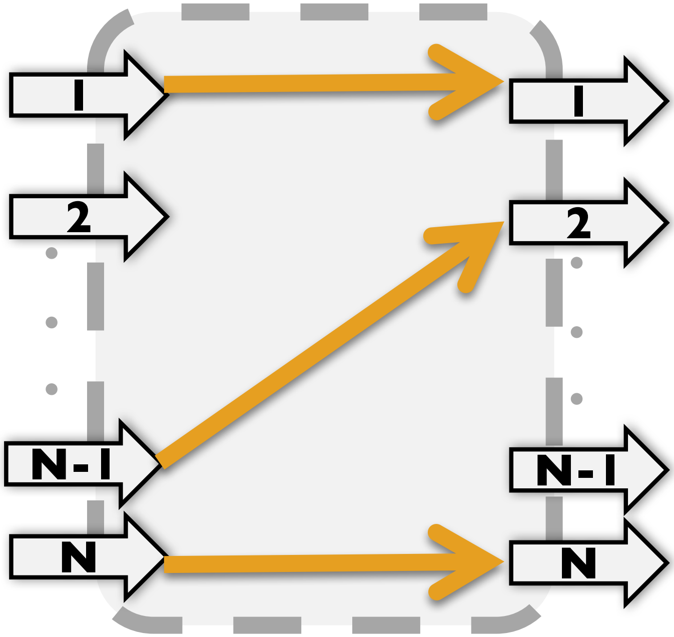
All-stop Model

Too strong: All circuits stops during switching.



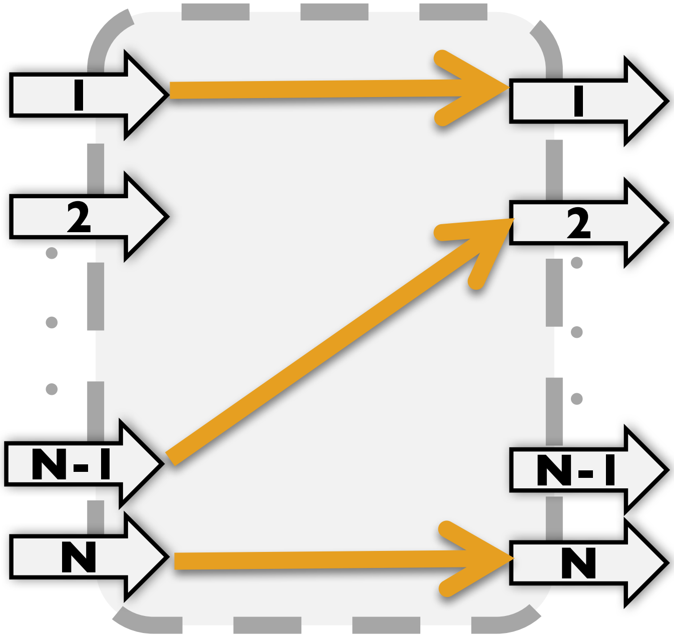
Not-all-stop Model

In practice: Unchanged circuits remain active.



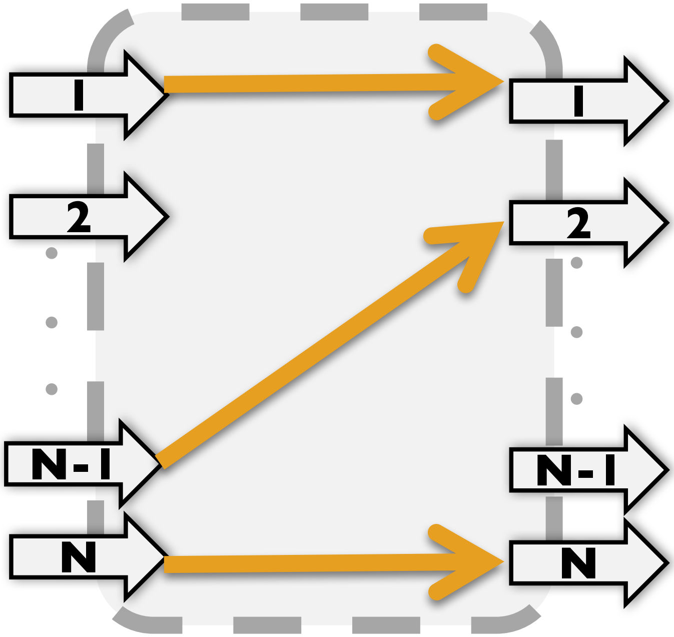
All-stop Model

Too strong: All circuits stops during switching.



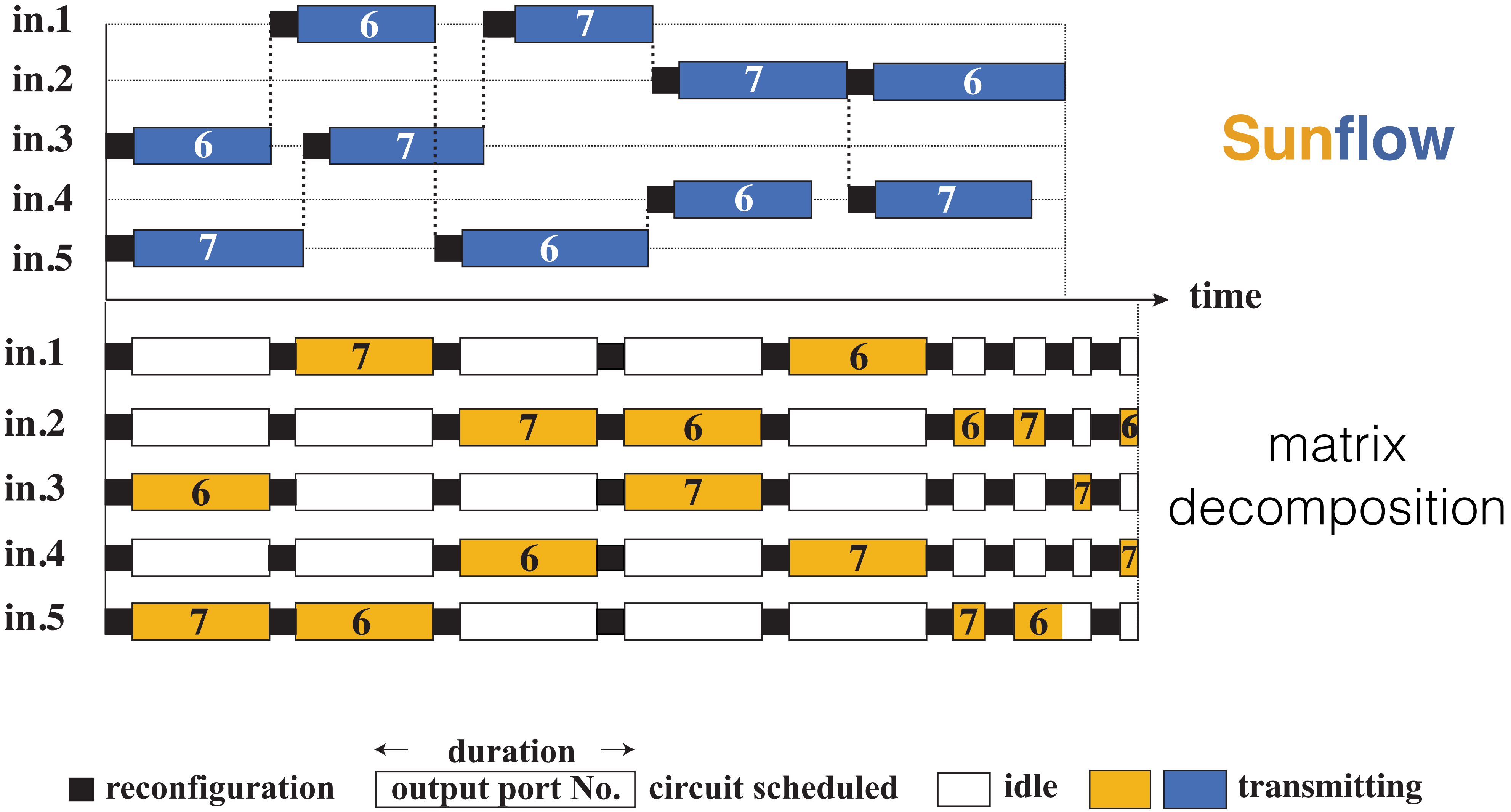
Not-all-stop Model

In practice: Unchanged circuits remain active.



Not-all-stop switch model:
Less stringent and more accurate.

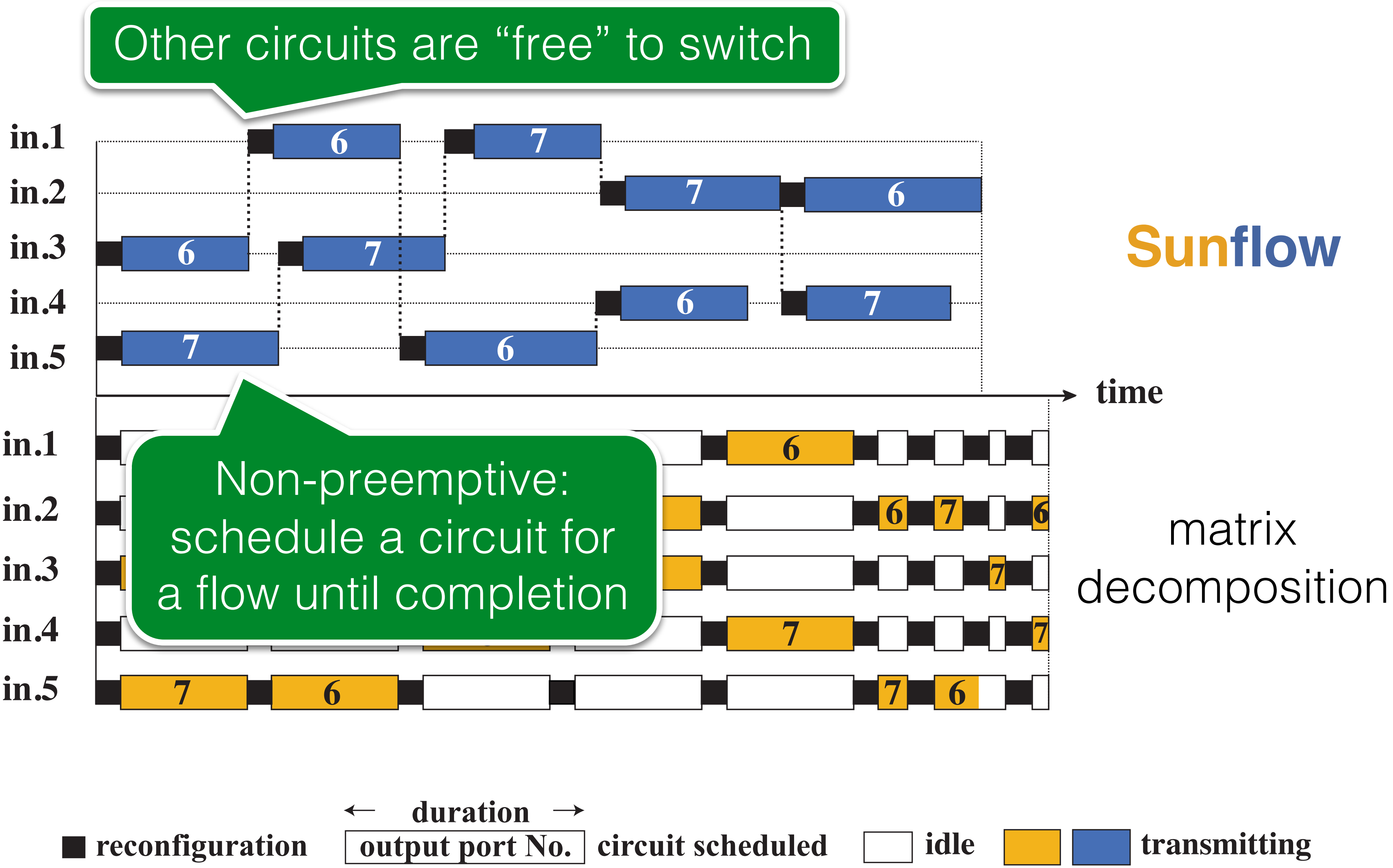
Intra-Coflow circuit scheduling



Sunflow

matrix decomposition

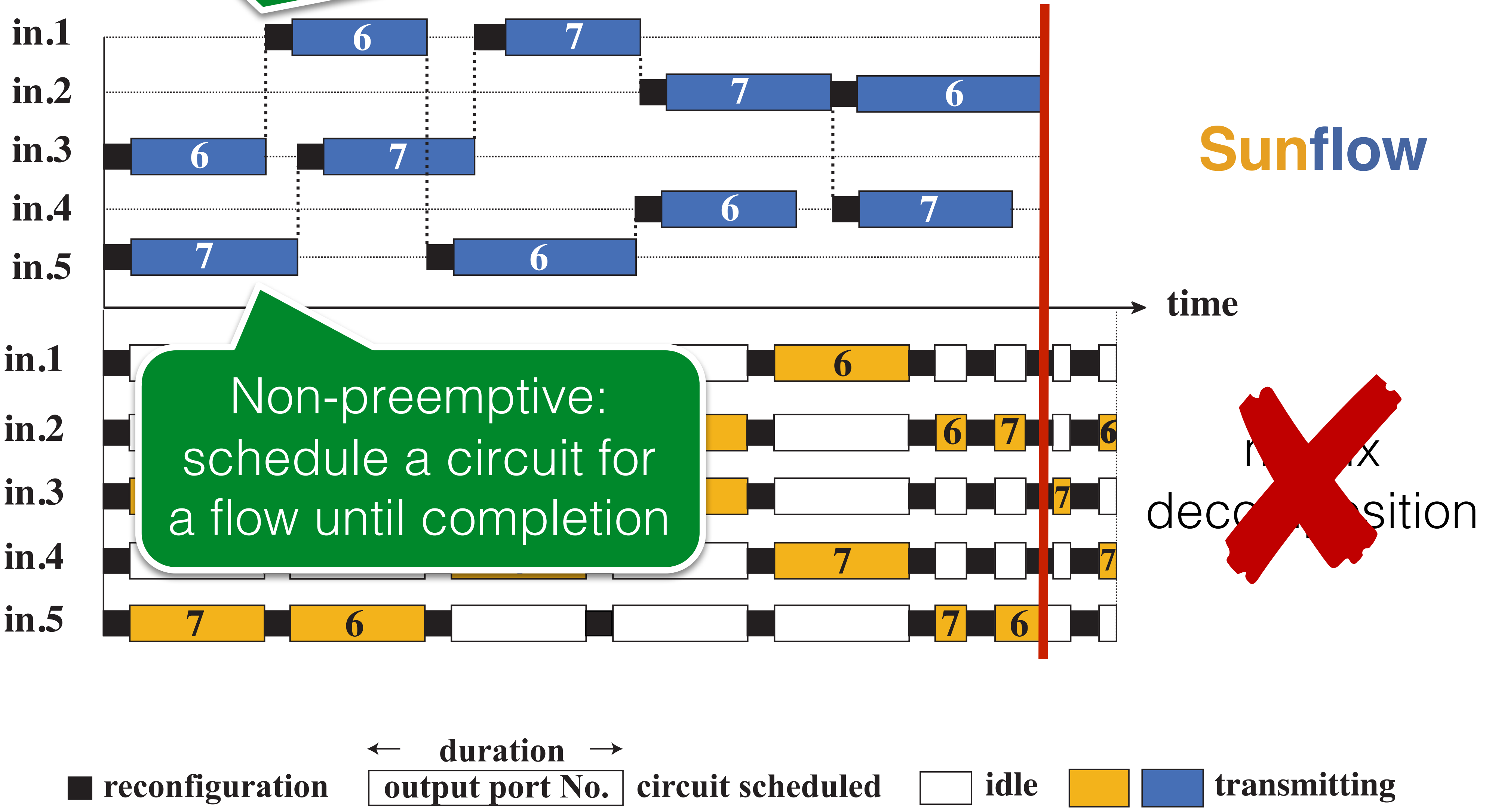
Intra-Coflow circuit scheduling



Other circuits are “free” to switch

Non-preemptive: schedule a circuit for a flow until completion

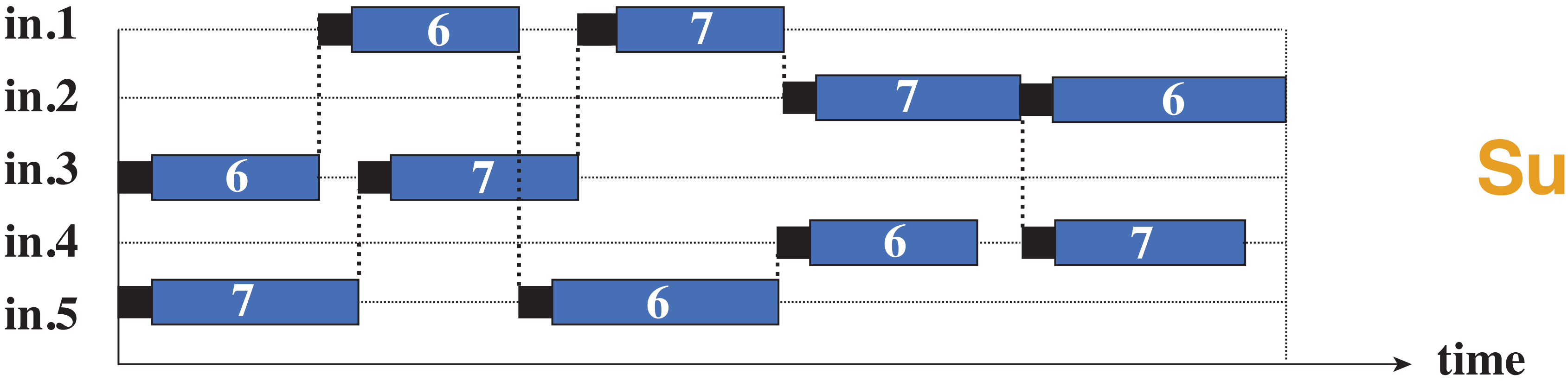
Intra-Coflow circuit scheduling



Other circuits are "free" to switch

Non-preemptive: schedule a circuit for a flow until completion

Intra-Coflow circuit scheduling



Sunflow

Simple & Efficient!

Greedy heuristic

Provably within 2x optimal,
1.03x in practice …

	Sunflow	Other circuit schedulers
Intra-Coflow	✓ Not allow subflows to preempt each other. ✓ Proved within 2x of the optimal.	✗ Lots of preemptions and switching delay. ✗ Observed 10x optimal.
Inter-Coflow	✓ Flexible preemption policy. (e.g. shortest-Coflow-first)	✗ Aggregated demand matrix loses Coflow boundary.
Switch model	✓ **Not-all-stop (flexible)**	✗ **All-stop (too strong)**

Simulation setup

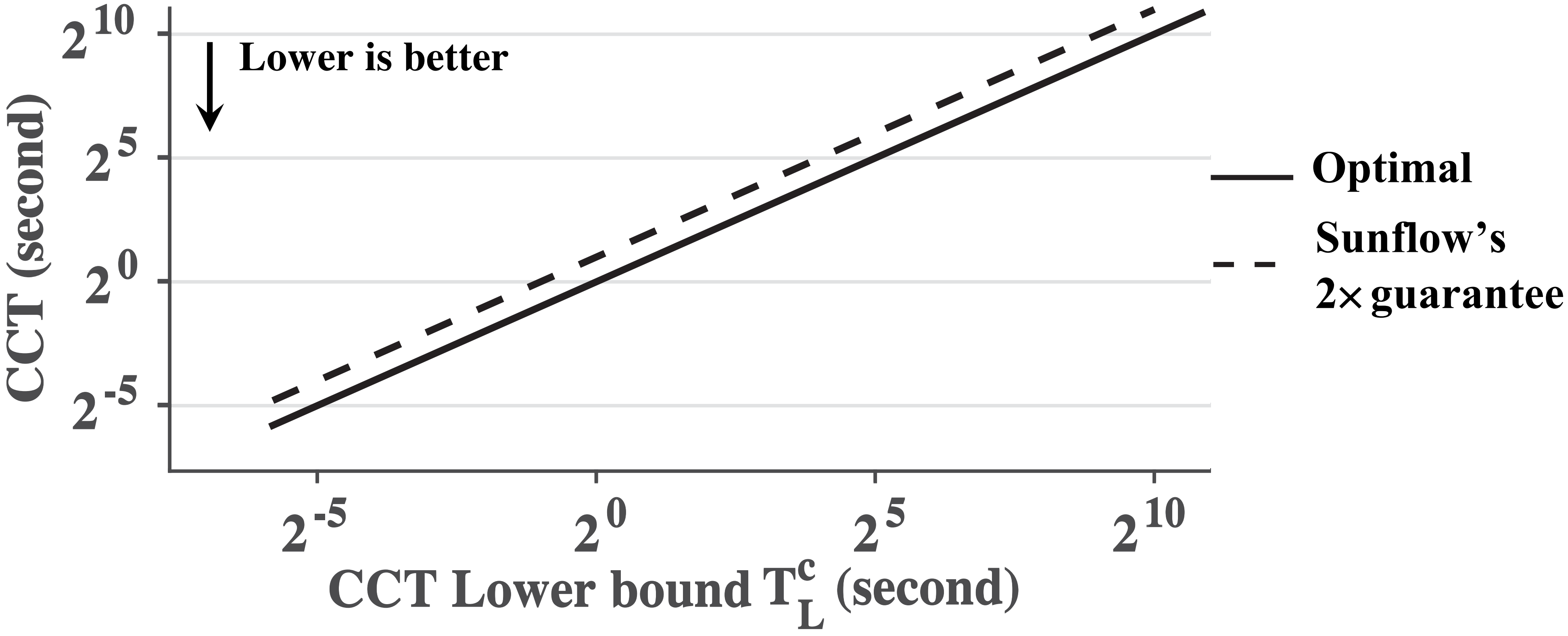
- Implemented a flow-level, discrete-event simulator
- Workload[1] : realistic trace derived from Facebook cluster
  - 1hr traffic trace, ~500 Coflows, ~700k flows
- Circuit switching delay 10 ms (typical of today's products)
- Evaluated at the intra-Coflow and inter-Coflow level

[1] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)

Simulation results

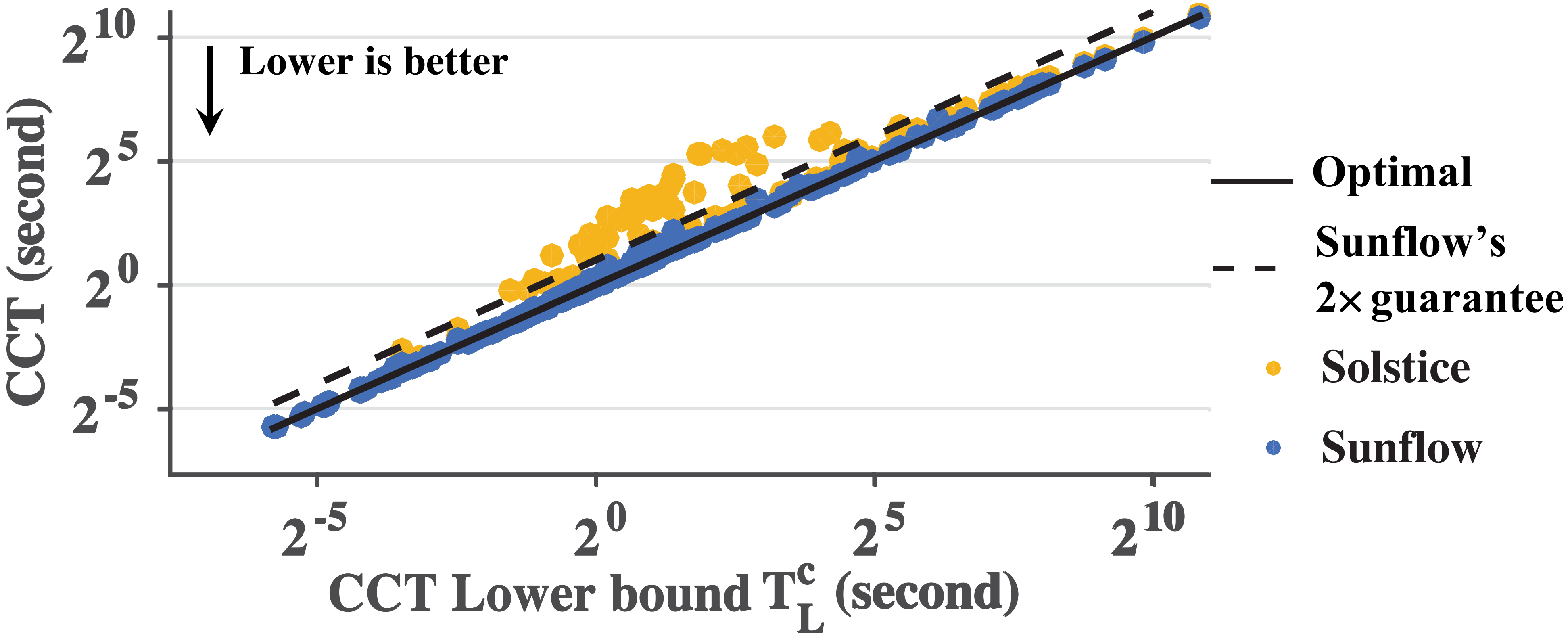
- At the intra-Coflow level:
  - Sunflow is close to **the optimal**
  - Sunflow is more efficient than the most viable circuit scheduling alternative, **Solstice (CoNEXT'15)**

Intra-Coflow circuit scheduling (Sunflow vs Solstice)



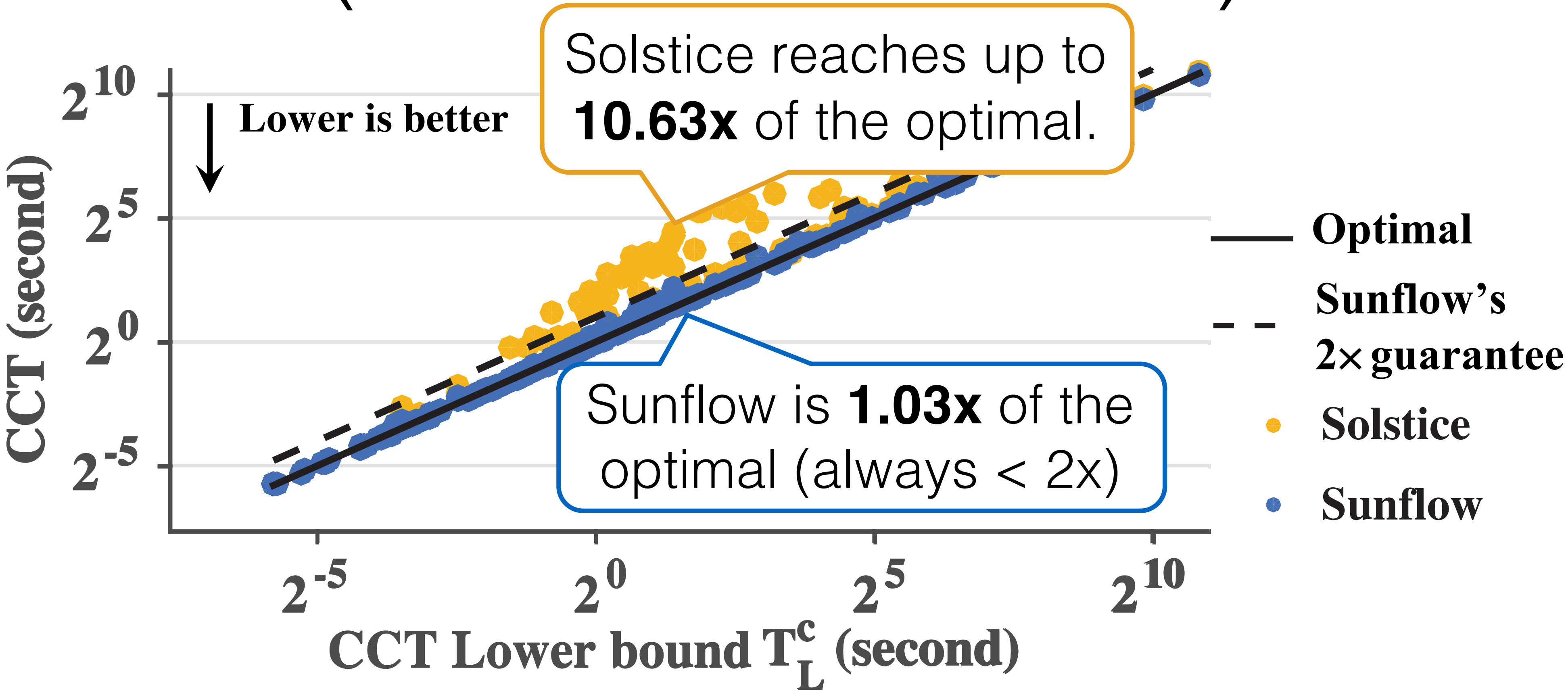
Sunflow is more efficient, with performance guarantee

Intra-Coflow circuit scheduling (Sunflow vs Solstice)



Sunflow is more efficient, with performance guarantee

Intra-Coflow circuit scheduling (Sunflow vs Solstice)

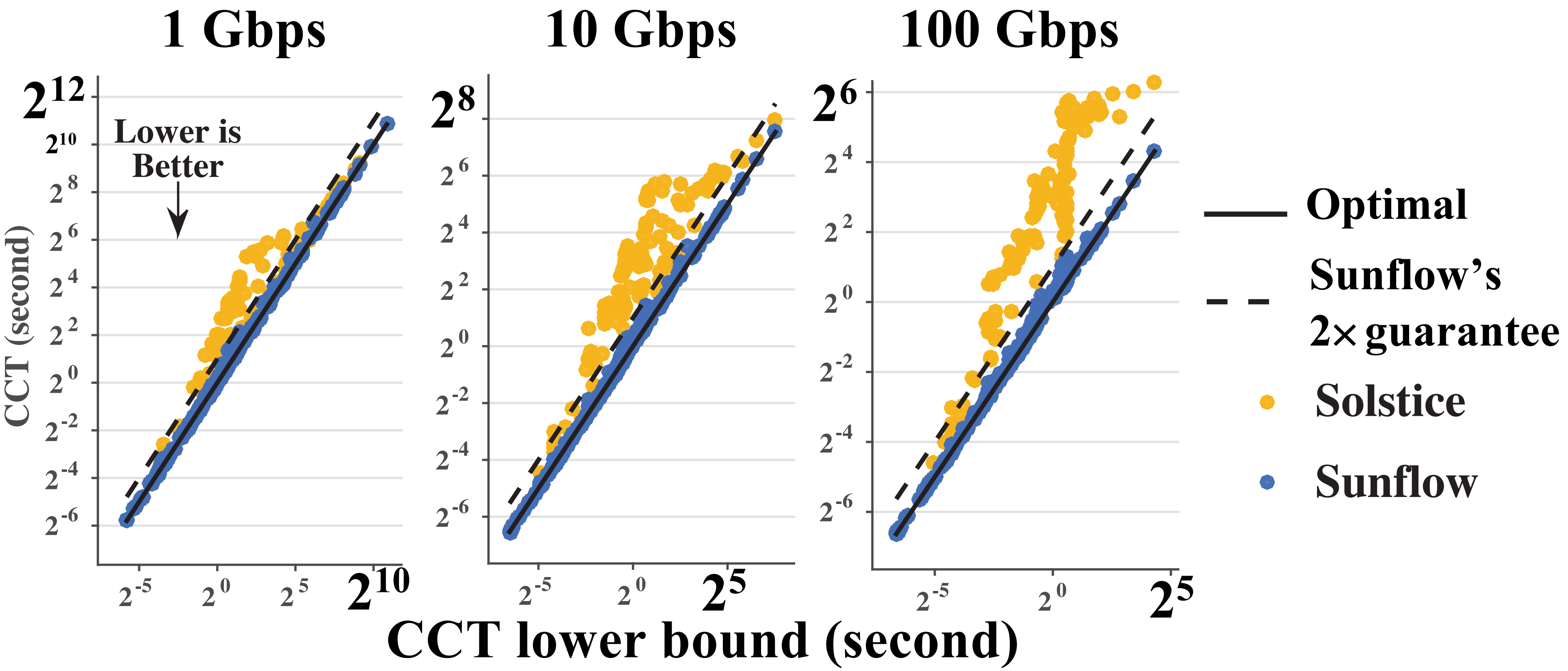


Solstice reaches up to **10.63x** of the optimal.

Sunflow is **1.03x** of the optimal (always < 2x)

Sunflow is more efficient, with performance guarantee

Intra-Coflow circuit scheduling (Sunflow vs Solstice)



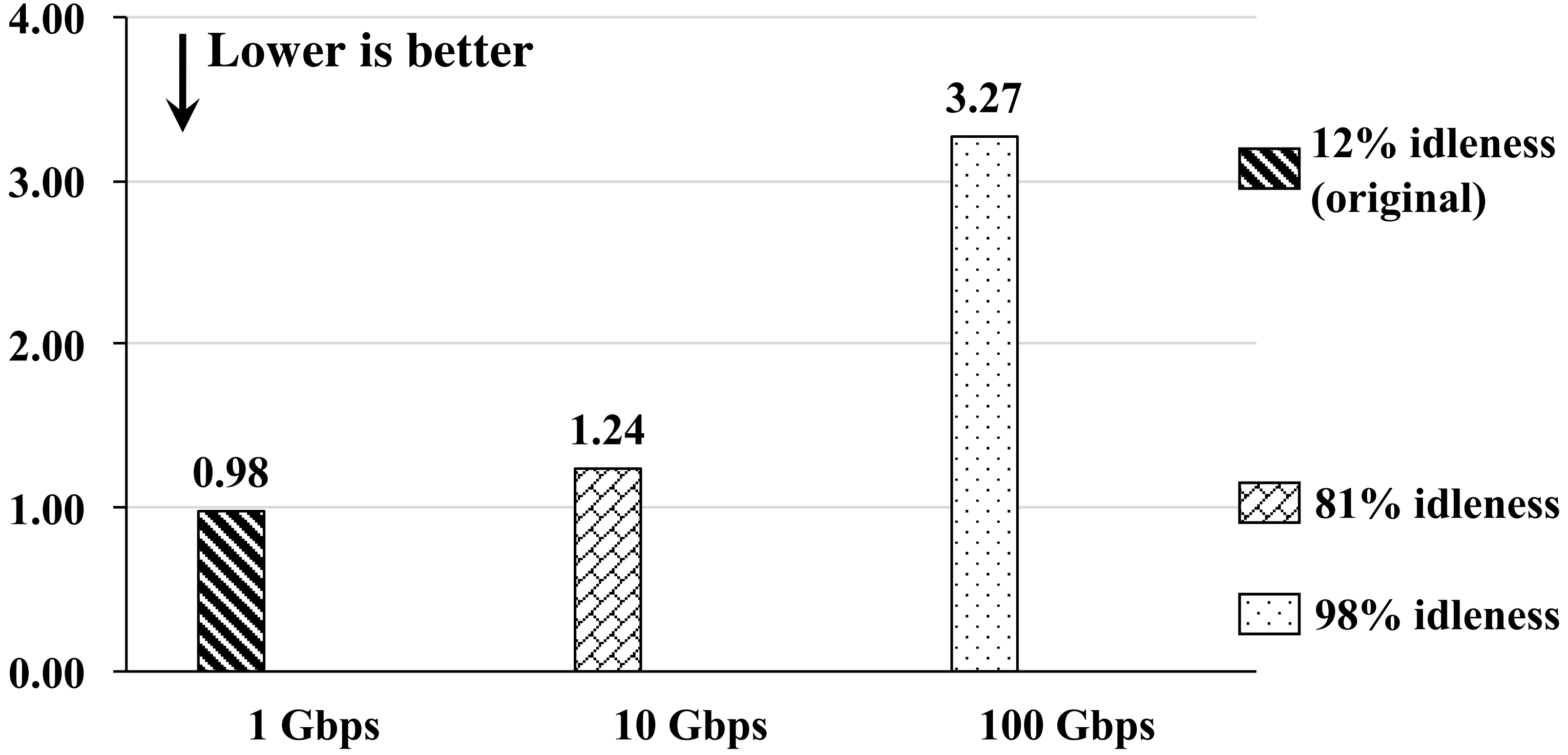
Sunflow is more efficient, with performance guarantee

Simulation results

- At the intra-Coflow level:
    - Sunflow is close to the optimal
    - Sunflow is more efficient than the most viable circuit scheduling alternative, Solstice (CoNEXT'15)
- At the inter-Coflow level:
    - Sunflow's circuit switching achieves performance close to packet-switched Coflow schedulers, Varys (SIGCOMM'14)
    - Same link rate for Sunflow and Varys
    - Sunflow: 10ms switching delay. Varys: no switching delay.

Circuit Switching vs Packet Switching (Sunflow vs Varys)

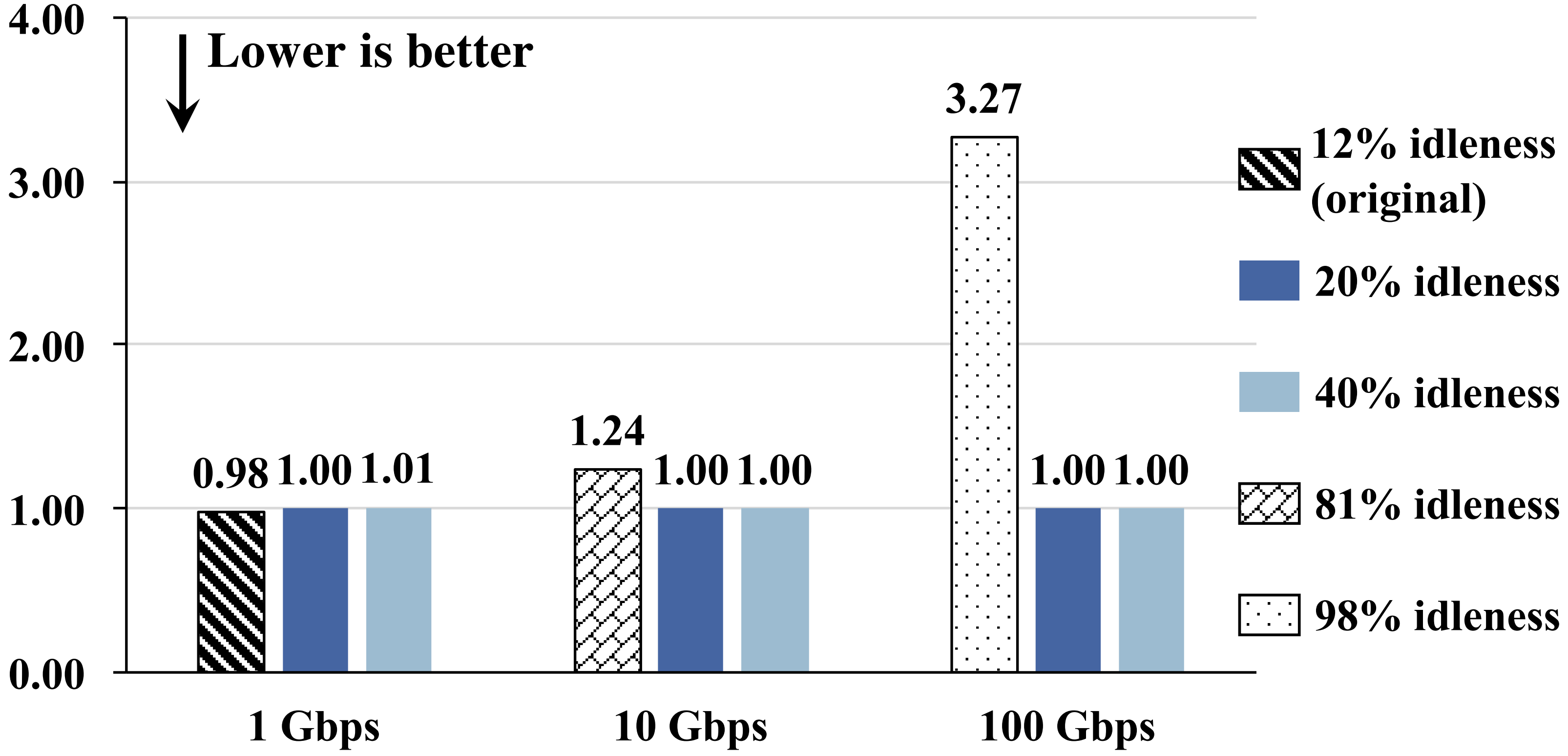
Sunflow's average-CCT over Varys's average-CCT



Change idleness by changing link rate and/or traffic size, but Coflow structures (i.e. flow endpoints) remain the same.

Circuit Switching vs Packet Switching (Sunflow vs Varys)

Sunflow's average-CCT over Varys's average-CCT



Sunflow achieves near-packet-switched performance

Change idleness by changing link rate and/or traffic size, but Coflow structures (flow endpoints) remain the same.

More in the paper

- At the intra-Coflow level:
    - Sunflow's optimality based on Coflow **structures**
    - Sunflow v.s. packet switching based on Coflow **sizes**.
    - **Switching overhead** for Sunflow and Solstice
    - Sensitivity to **flow ordering.**
- At the inter-Coflow level:
    - Sunflow v.s. **Aalo (SIGCOMM'15)**, another Coflow schedulers based on packet switching.
- Sensitivity to **switching delay** at both levels.
- **Proof** of Sunflow's **performance guarantee** against circuit (packet) switching and Sunflow's **complexity**.

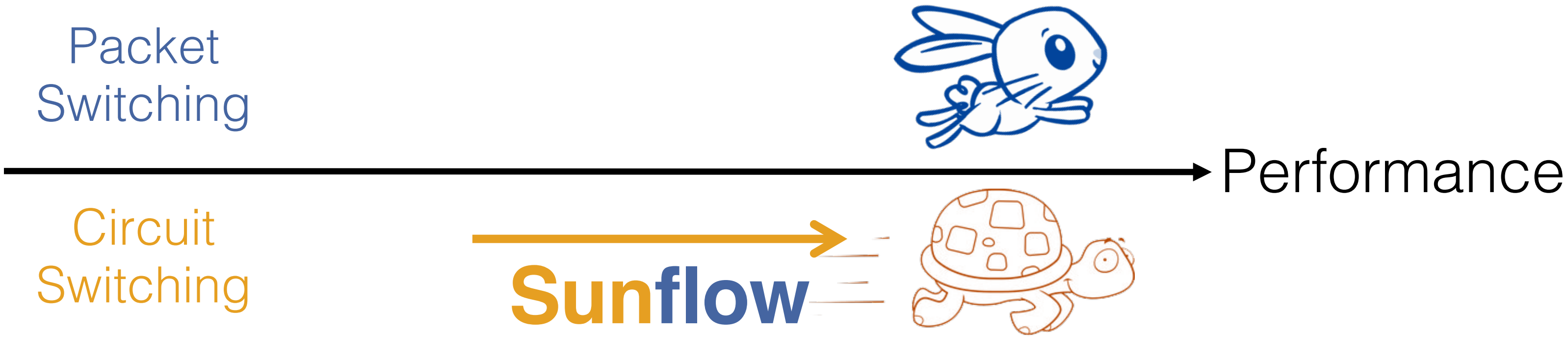
Conclusions

We **can** simultaneously obtain

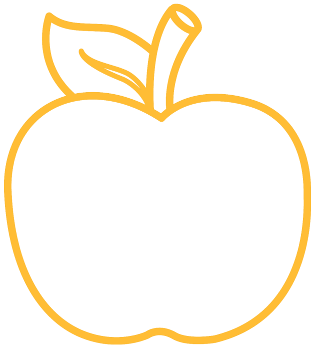
- benefits of optical circuit switching and
- good traffic performance for Coflows!

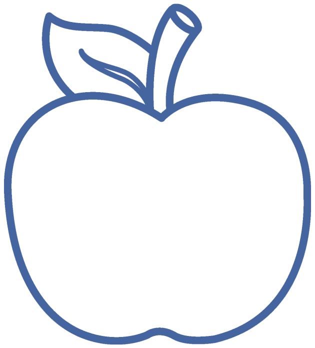
Enabled by **Sunflow**:

- Efficient & flexible *not-all-stop* switch model
- Provably within 2x of the optimal, 1.03x in practice
- Near-packet-switching performance



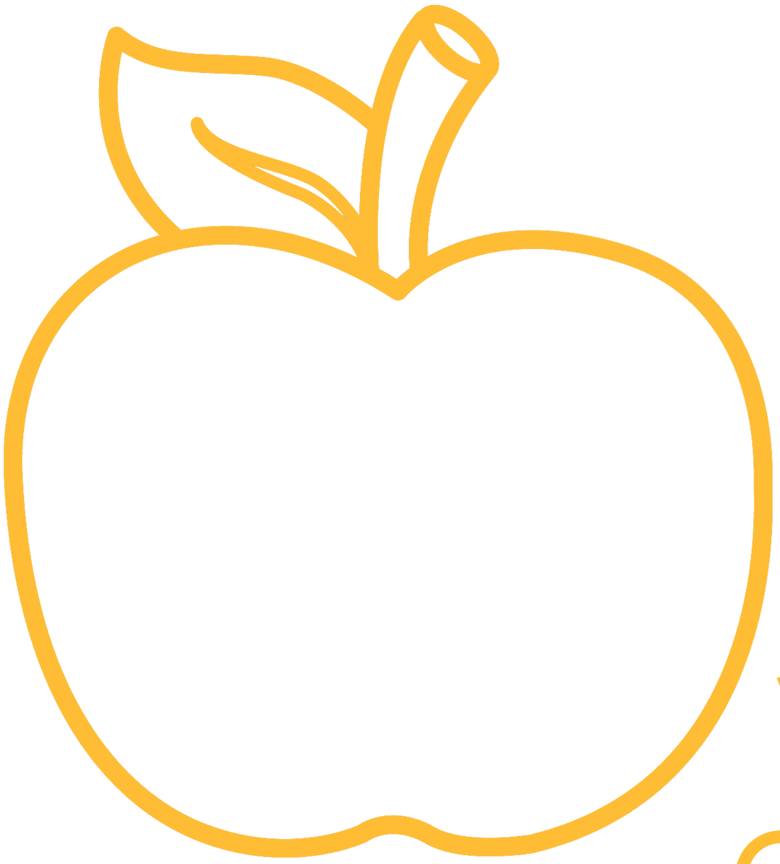
Circuit Switching v.s. Packet Switching

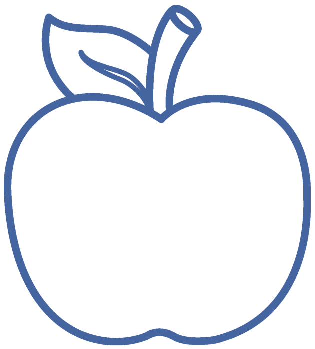




This work is an apple-to-apple comparison between circuit switching and packet switching.

Circuit Switching v.s. Packet Switching





Better potential for
Large capacity

Conclusions

We **can** simultaneously obtain

- benefits of optical circuit switching and
- good traffic performance for Coflows!

Enabled by **Sunflow**:

- Efficient & flexible *not-all-stop* switch model
- Provably within 2x of the optimal, 1.03x in practice
- Near-packet-switching performance

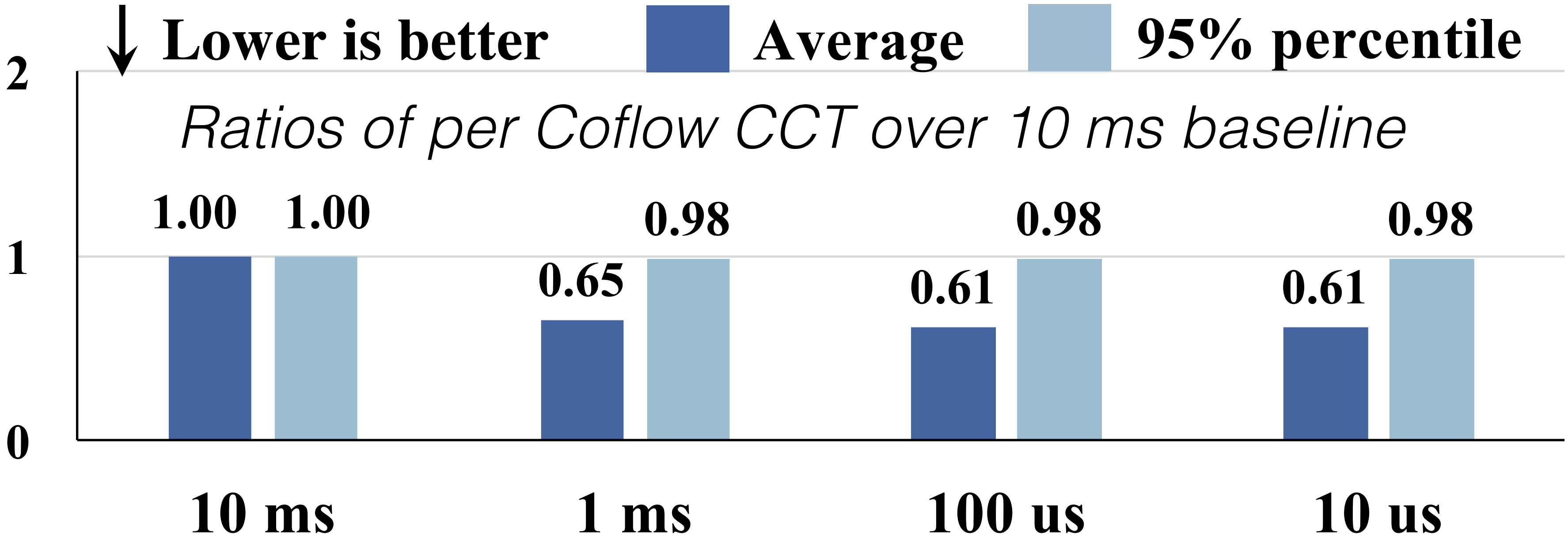
RICE

is recruiting faculty!

Thank You!

Backup slides

Sensitivity to circuit switching delay on *intra*-Coflow scheduling



Product typical

Good to have

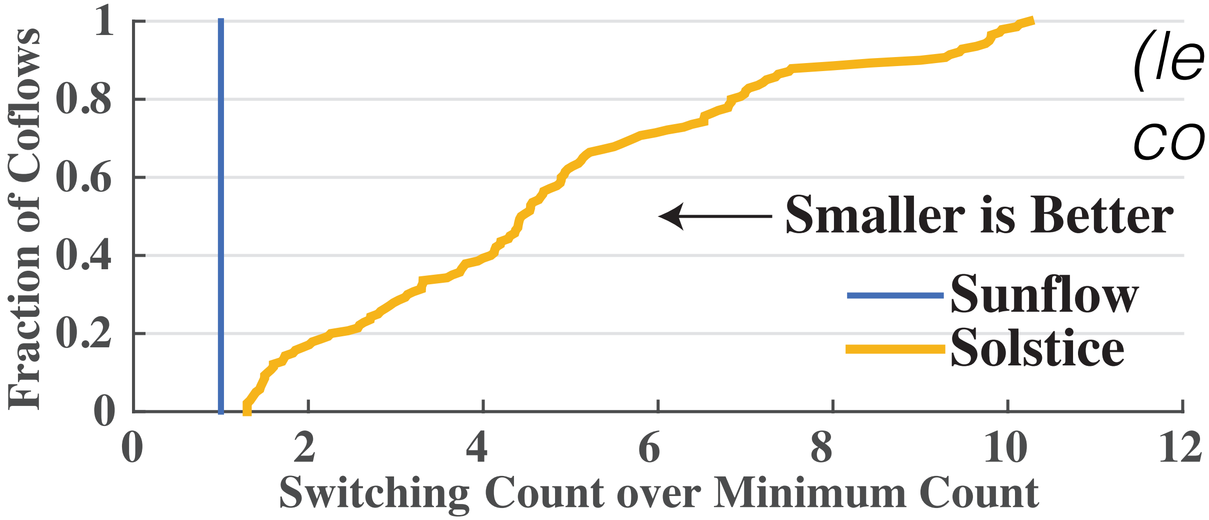
SMALL benefit

millisecond circuit switching is **sufficient** to serve Coflow!

Sunflow v.s. the optimal on ***intra***-Coflow scheduling

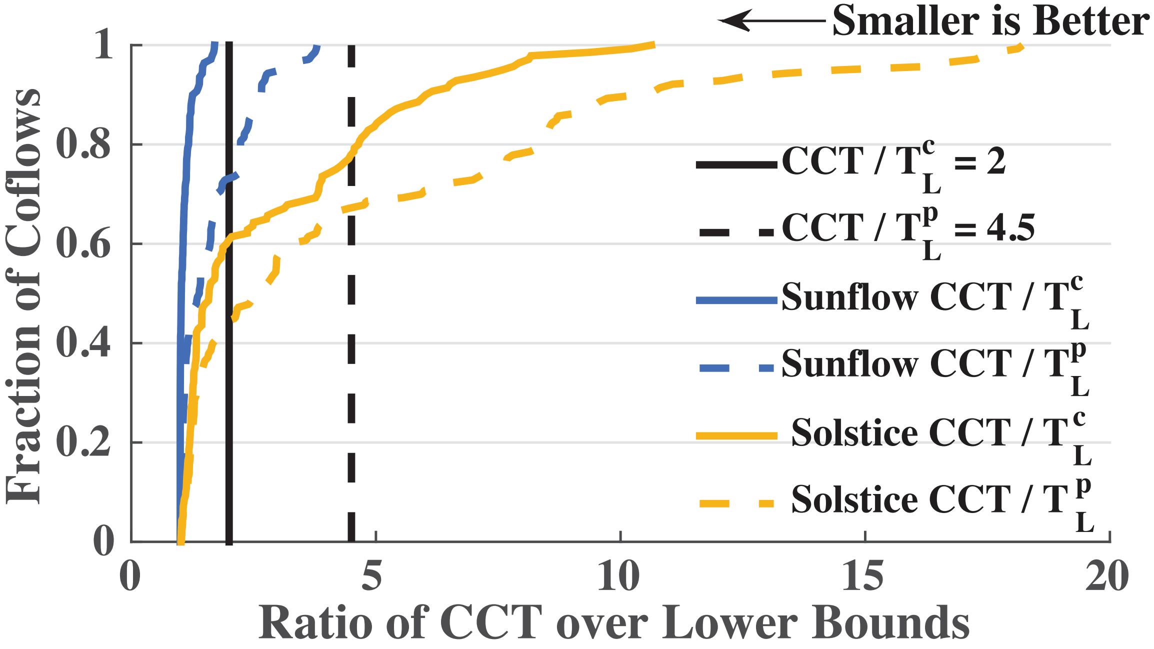
# senders	**> 1**	**> 1**	**1**	**1**	**any**
# receivers	**> 1**	**1**	**> 1**	**1**	**any**
% bytes	99.9%	0.028%	0.024%	0.005%	100%
% Coflows	26.6%	40.1%	9.9%	23.4%	100%
Sunflow CCT	**1.10x optimal**	**optimal**	**optimal**	**optimal**	**1.03x optimal**

Circuit Switching Overhead (Sunflow v.s. Solstice)



(left) Distribution of switching count for M2M Coflows

result in



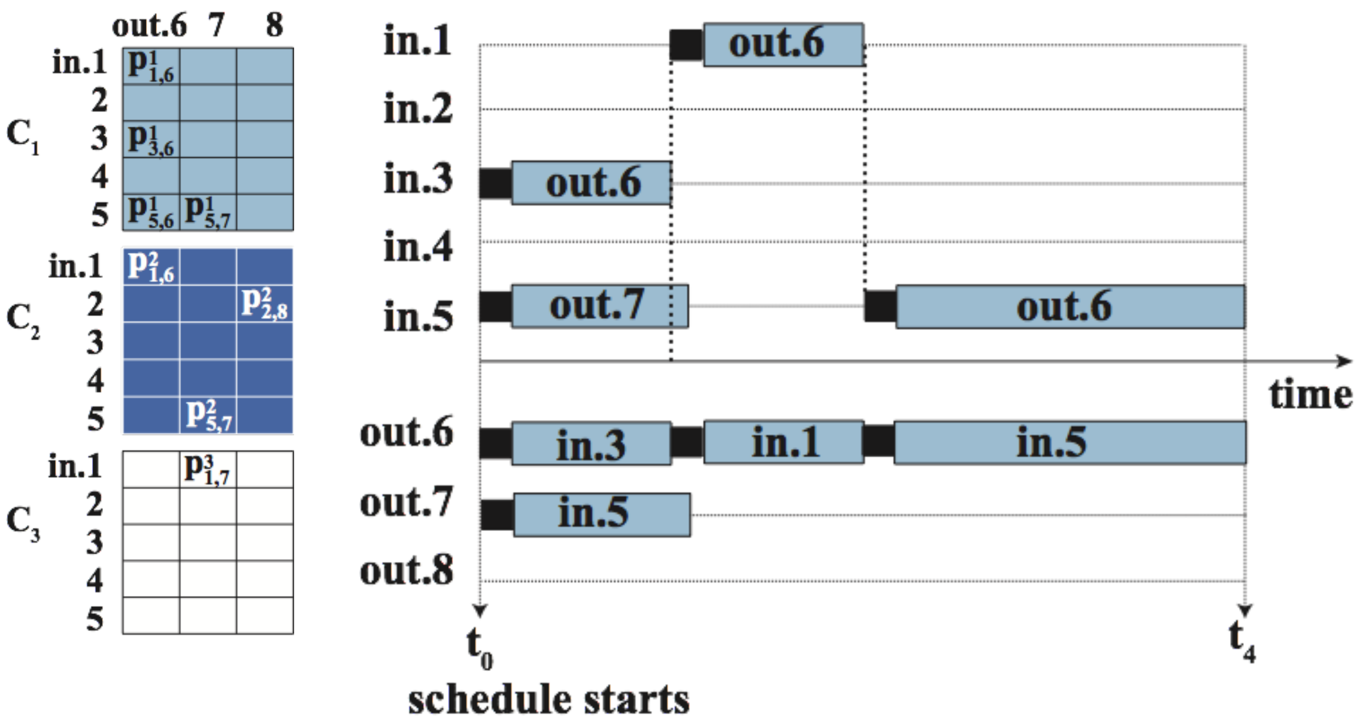
(right) Distribution of $CCT/T^C{}_L$ and $CCT/T^L{}_p$ for M2M Coflows

Sunflow v.s. Varys on ***inter***-Coflow scheduling

average flow size ≥ 5 MB

	Long Coflows	Short Coflows	All
% bytes	**98.8%**	1.2%	100%
% Coflows	25.2%	74.8%	100%
Per Coflow $\frac{\textbf{Sunflow CCT}}{\textbf{Varys CCT}}$	**1.07x**	2.16x	1.87x

Inter-Coflow circuit scheduling

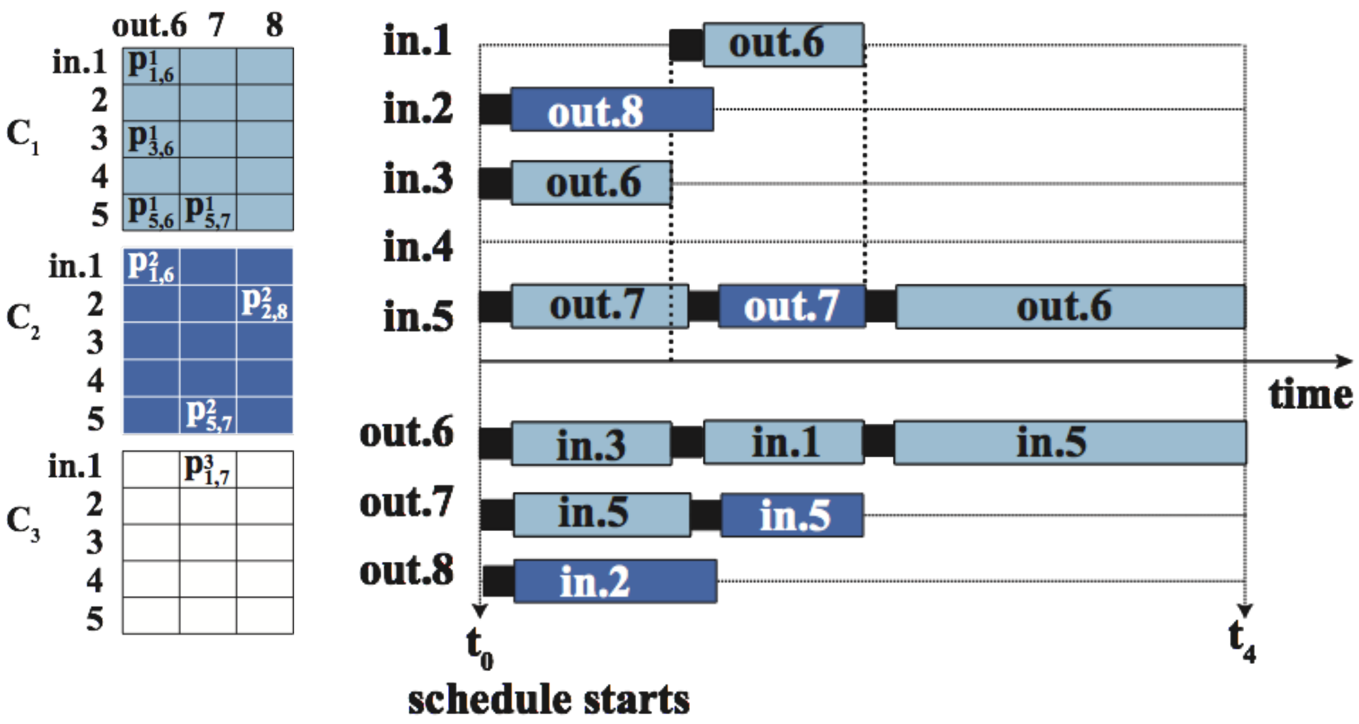


		out.6	7	8
C_1	in.1	$p^1_{1,6}$		
	2			
	3	$p^1_{3,6}$		
	4			
	5	$p^1_{5,6}$	$p^1_{5,7}$	
C_2	in.1	$p^2_{1,6}$		
	2			$p^2_{2,8}$
	3			
	4			
	5		$p^2_{5,7}$	
C_3	in.1		$p^3_{1,7}$	
	2			
	3			
	4			
	5			

Schedule C1

- Sort Coflow on priority.
- Assign *circuit active time* for flows.

Inter-Coflow circuit scheduling

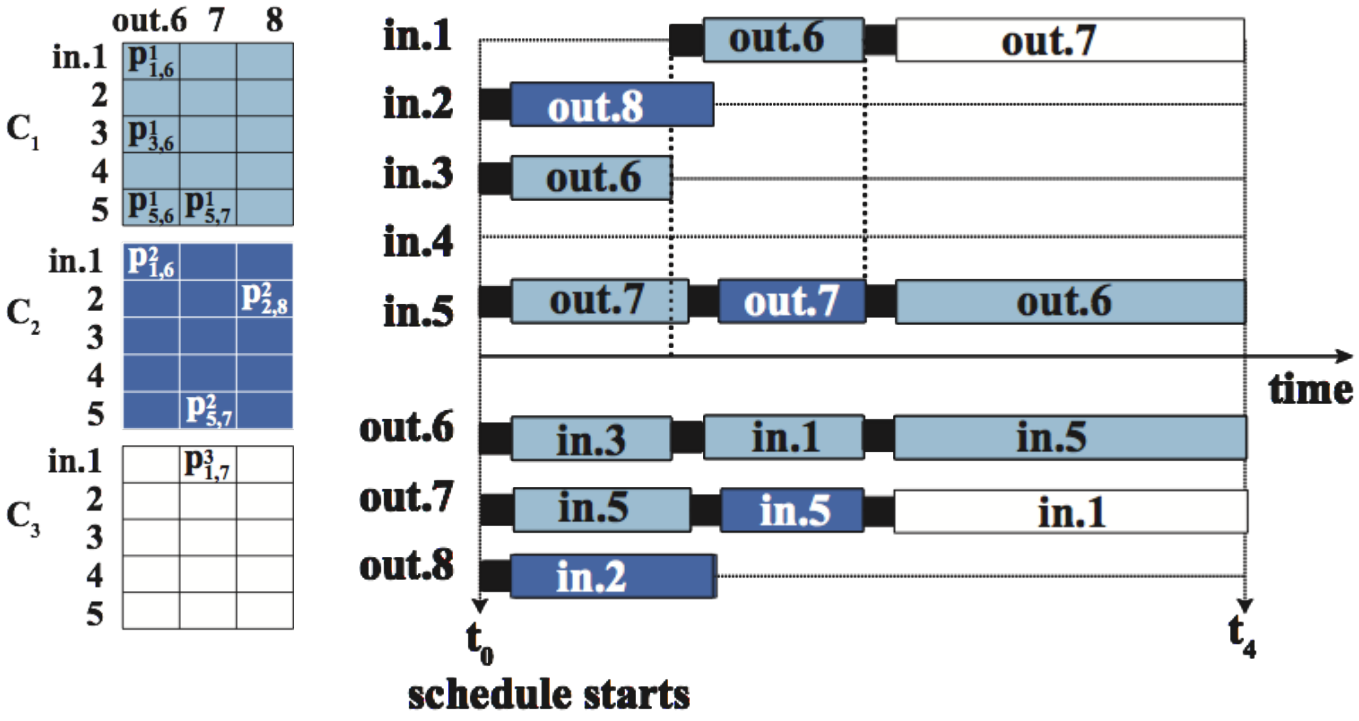


Schedule C1

Schedule C2

- Sort Coflow on priority.
- Assign *circuit active time* for flows.

Inter-Coflow circuit scheduling



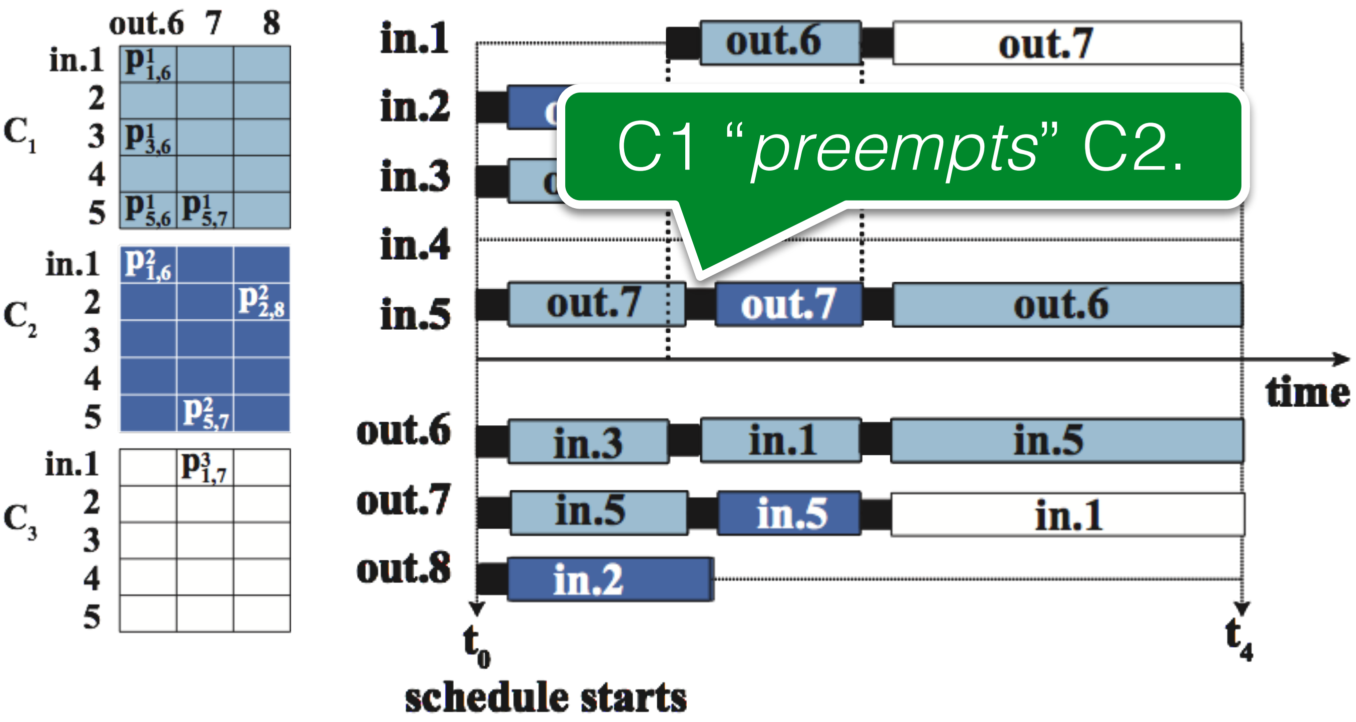
Schedule C1

Schedule C2

Schedule C3

- Sort Coflow on priority.
- Assign *circuit active time* for flows.

Inter-Coflow circuit scheduling



C1 “*preempts*” C2.

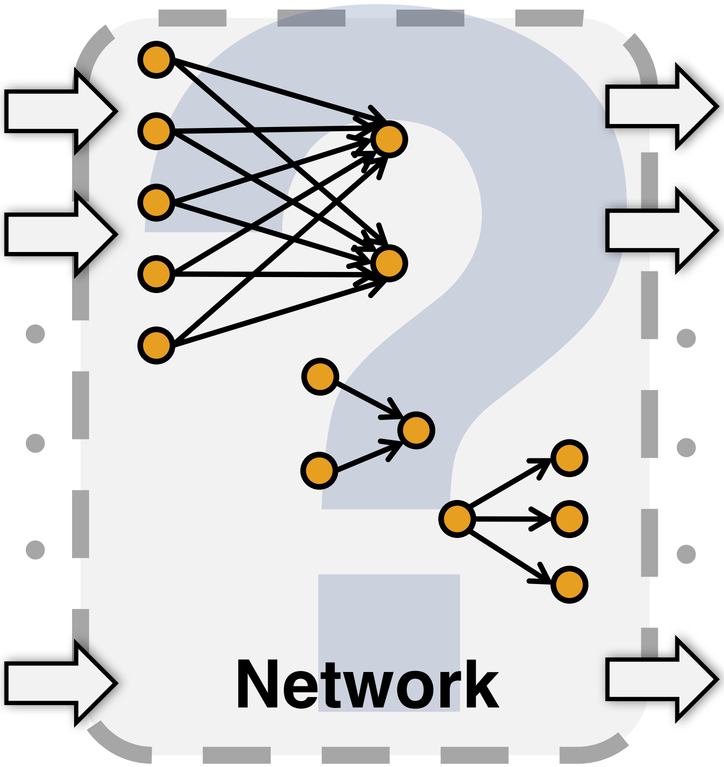
Schedule C1

Schedule C2

Schedule C3

High priority Coflow can preempt circuits from low priority Coflow.

Previous work on Coflow-aware network scheduling



- Why? Optimizing CCT reduces job completion time.[1]
- Key idea: Coordinate and schedule Coflows upon contention.
- Previous works are all in **packet** switching:
  - Min Σ CCTs (*Varys* in SIGCOMM'14)[2] ★
  - Other variants:
    uncertain Coflow byte size ('15)[3],
    uncertain Coflow structures ('16)[4]

[1] Chowdhury, M. et al. Coflow: An application layer abstraction for cluster networking. (HotNets'12)
[2] Chowdhury, M. et al. Efficient coflow scheduling with Varys. (SIGCOMM'14)
[3] Chowdhury, M. et al. Efficient coflow scheduling without prior knowledge. (SIGCOMM'15)
[3] Zhang, H. et al. CODA: Toward Automatically Identifying and Scheduling Coflows in the Dark. (SIGCOMM'16)

Thank You!

Xin Sunny Huang, Xiaoye Steven Sun, T. S. Eugene Ng
Rice University

RICE

BOLD

Big Data and Optical Lightpaths Driven Lab