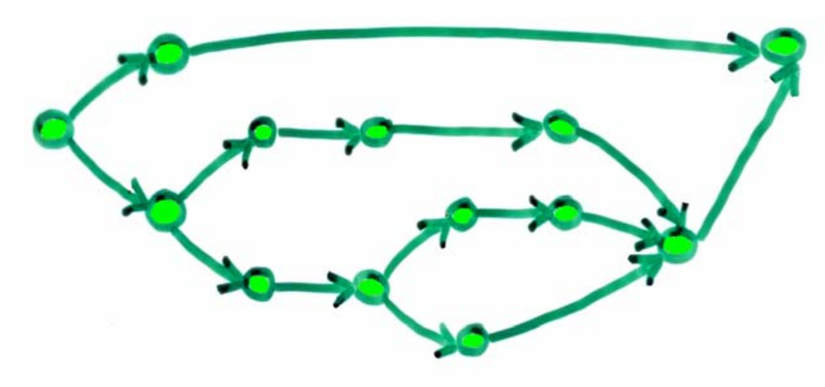
Scheduling DAGs on Asynchronous Processors

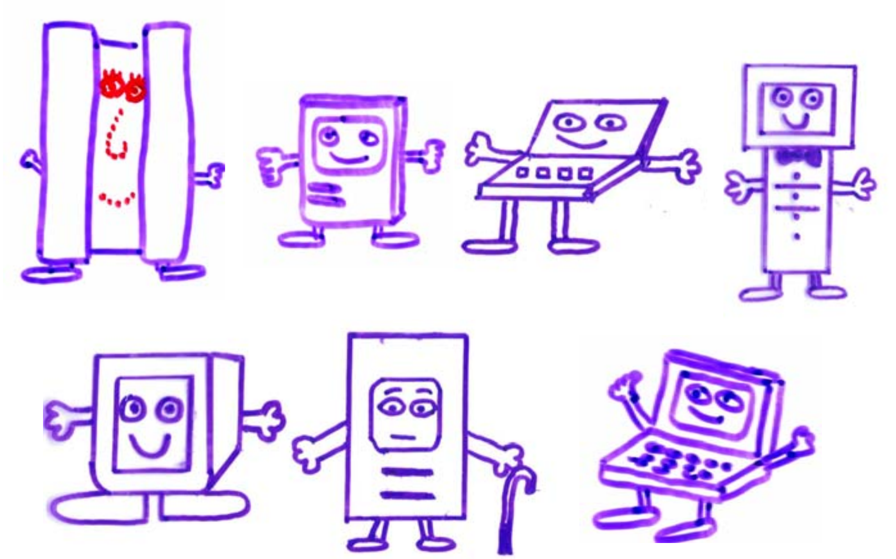
Michael A. Bender
Stony Brook

Cynthia A. Phillips
Sandia Labs

This Talk

- Efficiently execute a DAG on *p* asynchronous processors





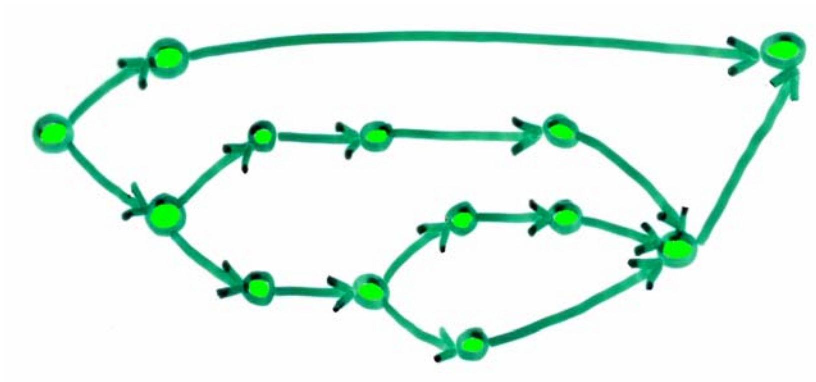
- **Results**: an analysis of **firing-squad scheduling** on DAGs.

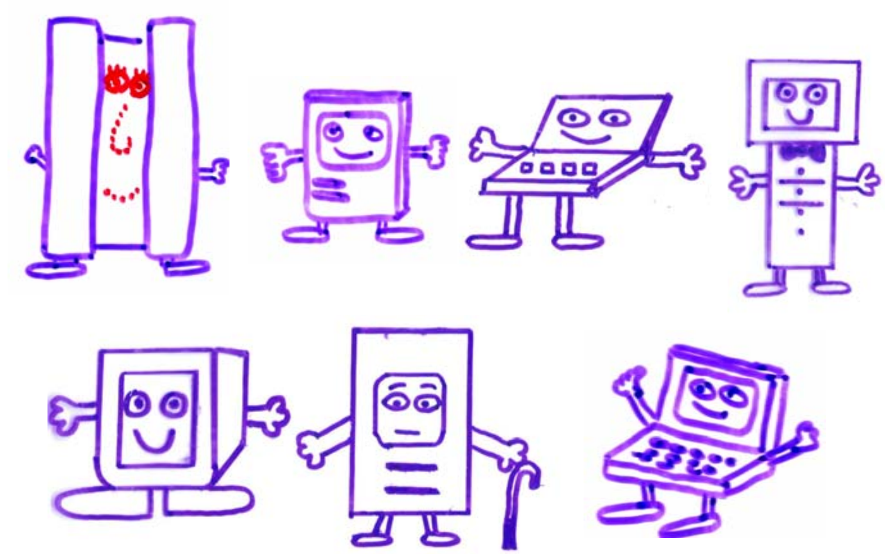
- **Motivation**: Using free cycles on networks of workstations, running tasks on server farms, grid computing, etc.

Need to say more about...

This Talk

- Efficiently execute a DAG on p asynchronous processors



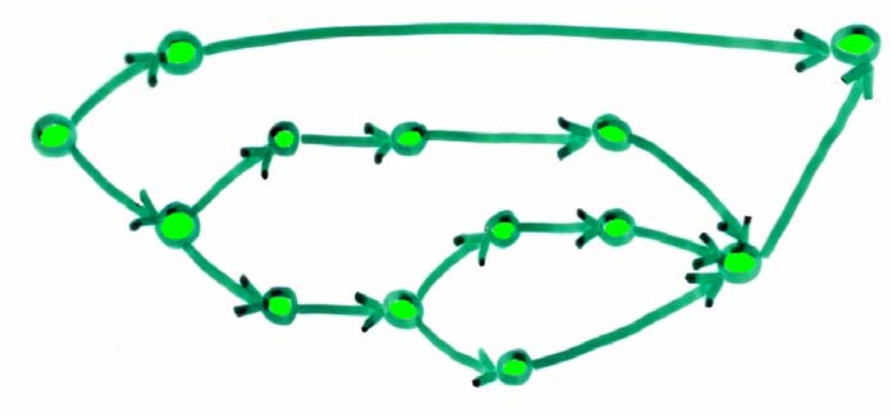


- **Results**: an analysis of **firing-squad scheduling** on DAGs.
- **Motivation**: Using free cycles on networks of workstations, running tasks on server farms, grid computing, etc.

Terminology for DAGs

- *D* = **critical path length** (longest path in DAG)
- *W* = **total work** (# nodes in DAG)

Ex:



D = **8**

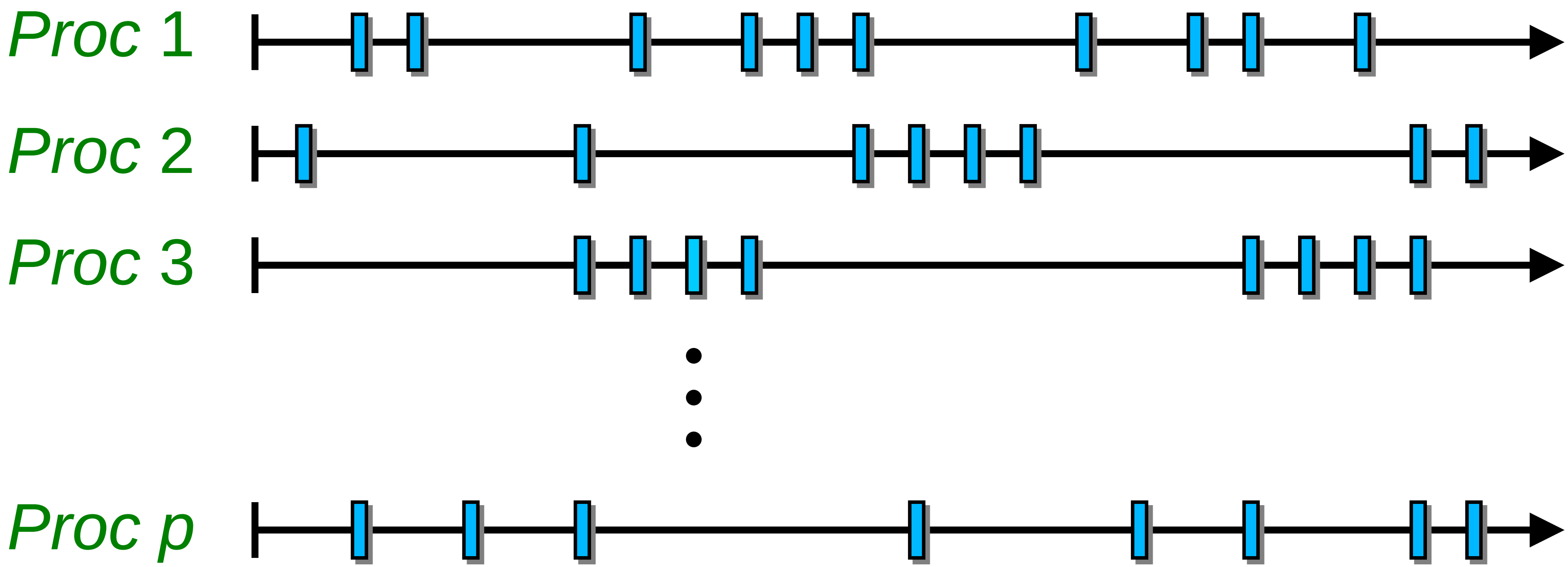
W = **13**

Theorem [Graham, Brent]: A greedy schedule has makespan $< W/P + D$.

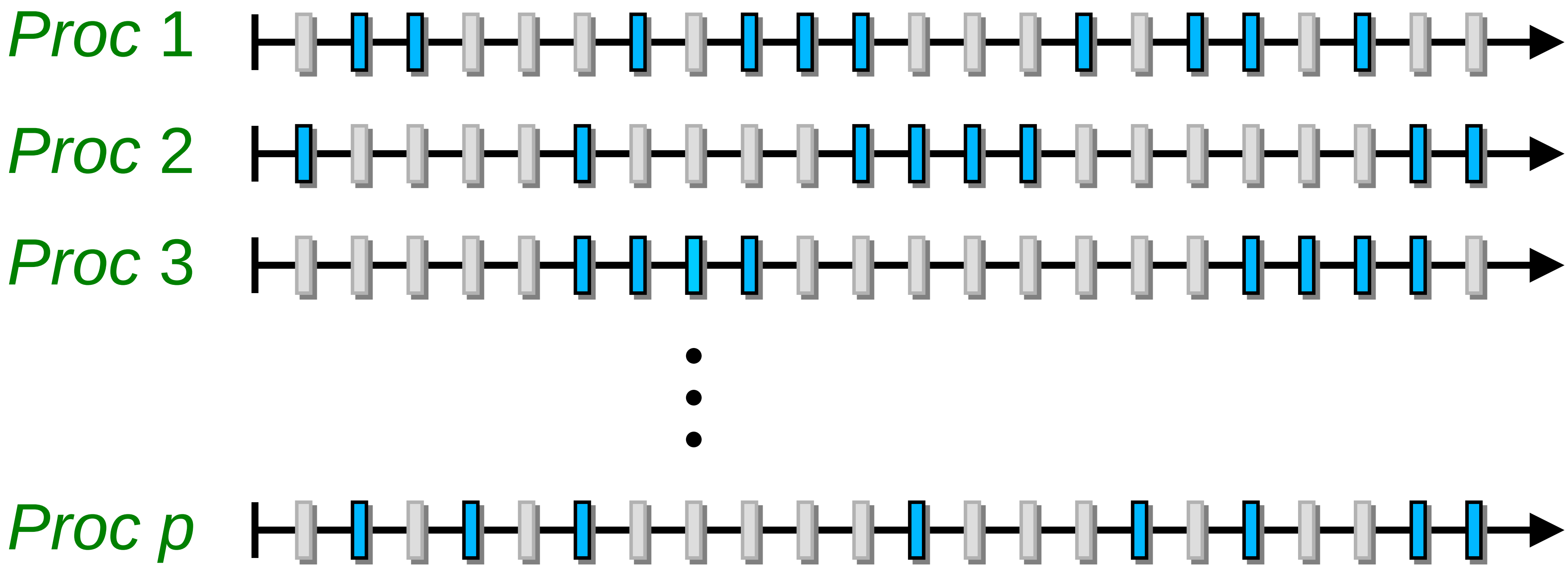
- 2-approx because both W/P & D are lower bounds on OPT.
- In most || programs, $W/P \gg D$, ➔ greedy is almost OPT.

Greedy is ideal. We want FSS to be as close as possible to Greedy.

What I mean by asynchrony…



What I mean by asynchrony…



- Even if the hardware is synchronous, there can be asynchrony at the application level

How We Model Asynchrony: Oblivious Adversary

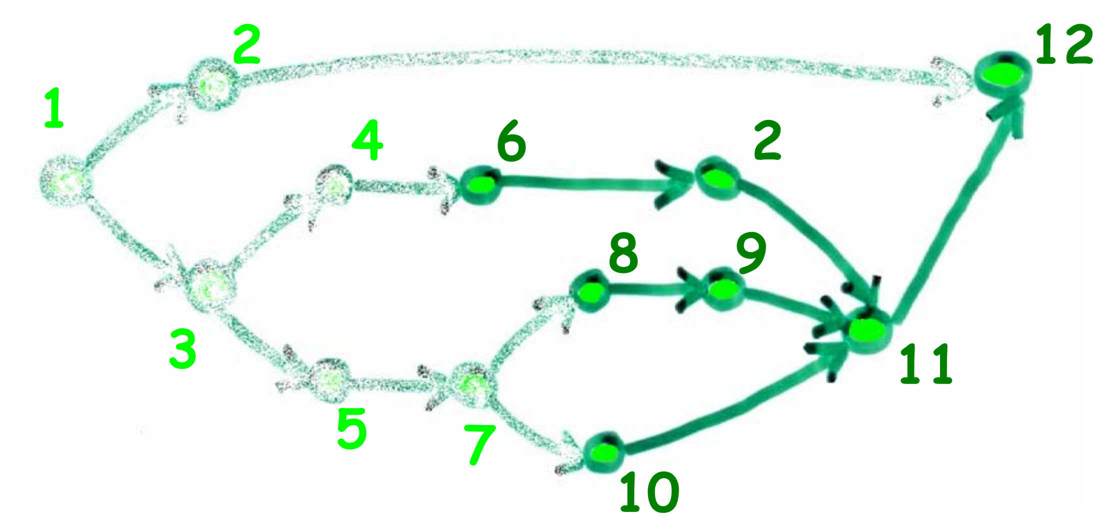
- We assume that the adversary determines the processor speeds at each point in time.
- **Oblivious adversary**
  - knows structure of DAG and initial state of system, but
  - does NOT know outcome of coin tosses of DAG scheduler.
- Realistic model of many sources of asynchrony (but not all). Common in asynchronous || computing
  - [Gibbons 89] [Cole, Zajicek 89] [Martel,Park,Subramonian 90] [Nishimura 90] [Kedem,Palem,Spirakis 90] [Kedem, Palem, Rabin, Raghunathan 92] [Aumann,Rabin 94] [Aumann,Kedem,Palem,Rabin, 93] [Aumann,Bender,Zhang 96].
- Firing Squad Scheduling works well with asynchrony. For convenience, we can assume synchronous timesteps. Results carry over to asynchronous setting.

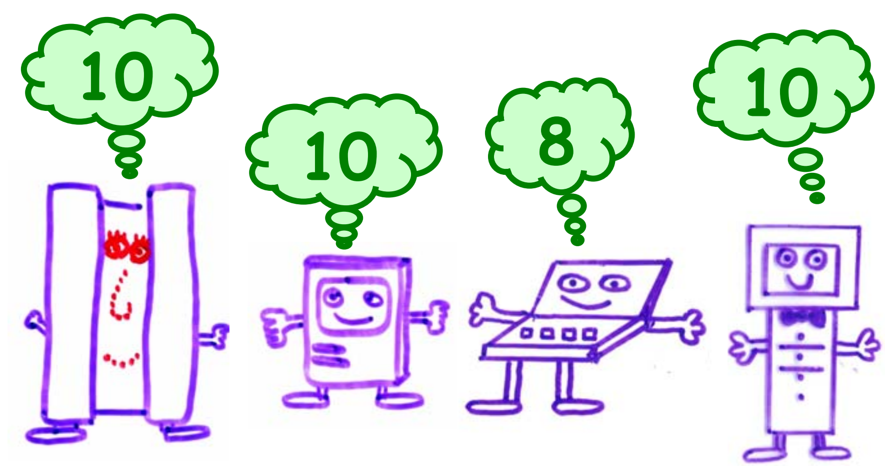
Firing-Squad Scheduling (FSS)

Used in papers on asynchronous || computing (eager scheduling)

[Gibbons 89] [Cole, Zajicek 89] [Martel,Park,Subramonian 90] [Nishimura 90] [Kedem,Palem,Spirakis 90] [Kedem, Palem, Rabin, Raghunathan 92] [Aumann,Rabin 94] [Aumann,Kedem,Palem,Rabin, 93] [Aumann,Bender,Zhang 96].

Whenever a processor is free, it **randomly** and **independently** chooses a task to execute from (a subset of) the tasks that are ready to run.

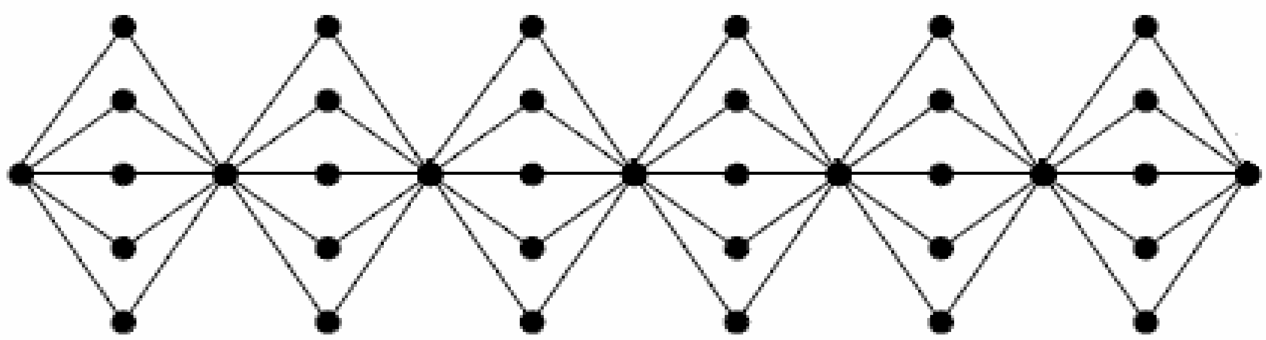




Redundancy: Some tasks may be executed many times.

Firing-Squad Scheduling (FSS) Cont.

- ***Redundancy*** → ***no preemption*** or ***process migration***.
  - A task that is bogged down on one proc finishes on another.
- FSS is well adapted for oblivious adversary
  - (Which is why we can pretend things are synchronous).



- **Previous work:** FSS for **DAGs with synchronization barriers**.

This talk: analysis of FSS on ***general DAGs***.

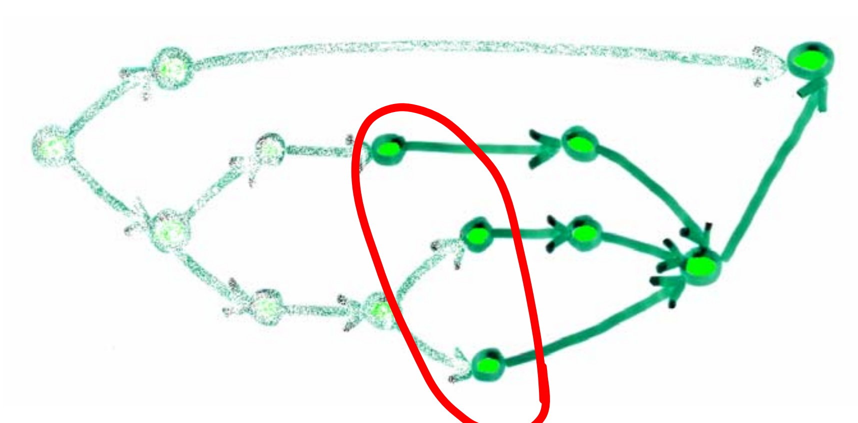
Question: Which Version of FSS is Better or Are They the Same?

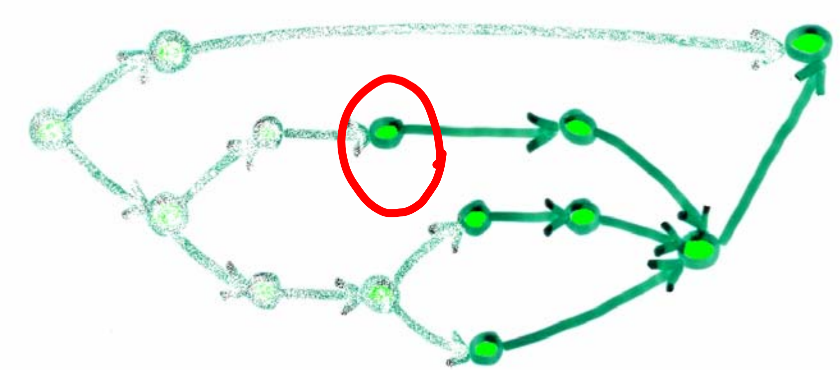
ALL—choose from *all* ready tasks.

- Minimizes redundant work in a time step.
- Pushes on the ***total work***.

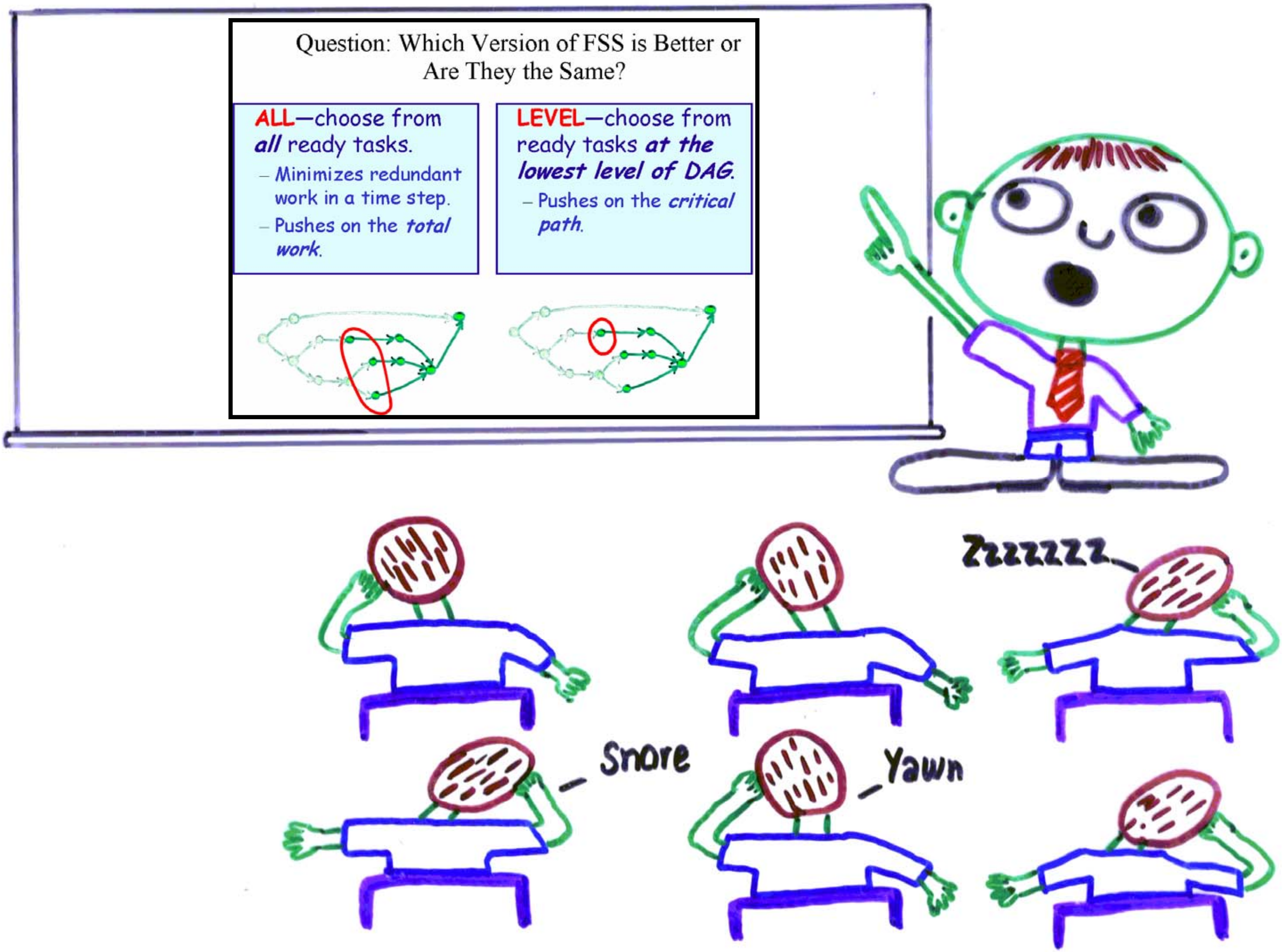
LEVEL—choose from ready tasks ***at the lowest level of DAG***.

- Increases redundant work in time step, but
- Pushes on the ***critical path***.





Really, which is better, ALL or LEVEL?



Results: LEVEL is Asymptotically Better

- Adding all dependencies between levels of the DAG ***actually improves*** makespan.

I originally thought both algs performed the same.

Most people I ask prefer ALL.

Results

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p\pi_{ave}} & \text{when } \dfrac{W}{D} \geq p \log p \\ (\log p)^{\alpha} \dfrac{W}{p\pi_{ave}} + (\log p)^{1-\alpha} \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \text{ for } \alpha \in [0,1] \\ \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p\pi_{ave}} + [\log^* p - \log^*(pD/W)]\,\frac{D}{\pi_{ave}}\right)$$

Explaining the Results

- π_{ave} is ave speed of procs during execution.
- To gain intuition, lets set $\pi_{ave} = 1$...

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p\pi_{ave}} & \text{when } \dfrac{W}{D} \geq p \log p \\ (\log p)^{\alpha} \dfrac{W}{p\pi_{ave}} + (\log p)^{1-\alpha} \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \text{ for } \alpha \in [0,1] \\ \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p\pi_{ave}} + [\log^* p - \log^*(pD/W)]\frac{D}{\pi_{ave}}\right)$$

Results

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p} & \text{when } \dfrac{W}{D} \geq p\log p \\ (\log p)^{\alpha}\dfrac{W}{p} + (\log p)^{1-\alpha} D & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \ \text{ for } \alpha \in [0,1] \\ D & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p} + \left[\log^* p - \log^*(pD/W)\right] D\right)$$

Results

For more intuition, consider special case

$W = PD$, i.e., $\alpha = 1/2$.

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p} & \text{when } \dfrac{W}{D} \geq p \log p \\ (\log p)^{\alpha} \dfrac{W}{p} + (\log p)^{1-\alpha} D & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \text{ for } \alpha \in [0,1] \\ D & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p} + [\log^* p - \log^*(pD/W)]\, D\right)$$

Results

For more intuition, consider special case

$W = PD$, i.e., $\alpha = 1/2$.

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p} & \text{when } \dfrac{W}{D} \ge p\log p \\[2ex] \sqrt{(\log p)^{\cancel{\alpha}}}\,\dfrac{W}{p} + \sqrt{(\log p)^{\cancel{1-\alpha}}}\, D & \text{when } \dfrac{W}{D} = p\cancel{(\log p)^{1-2\alpha}}, \cancel{\text{ for } \alpha \in [0,1]} \\[2ex] D & \text{when } \dfrac{W}{D} \le \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p} + [\log^* p - \log^*(pD/W)]\, D\right) = \Theta(D \log^* p)$$

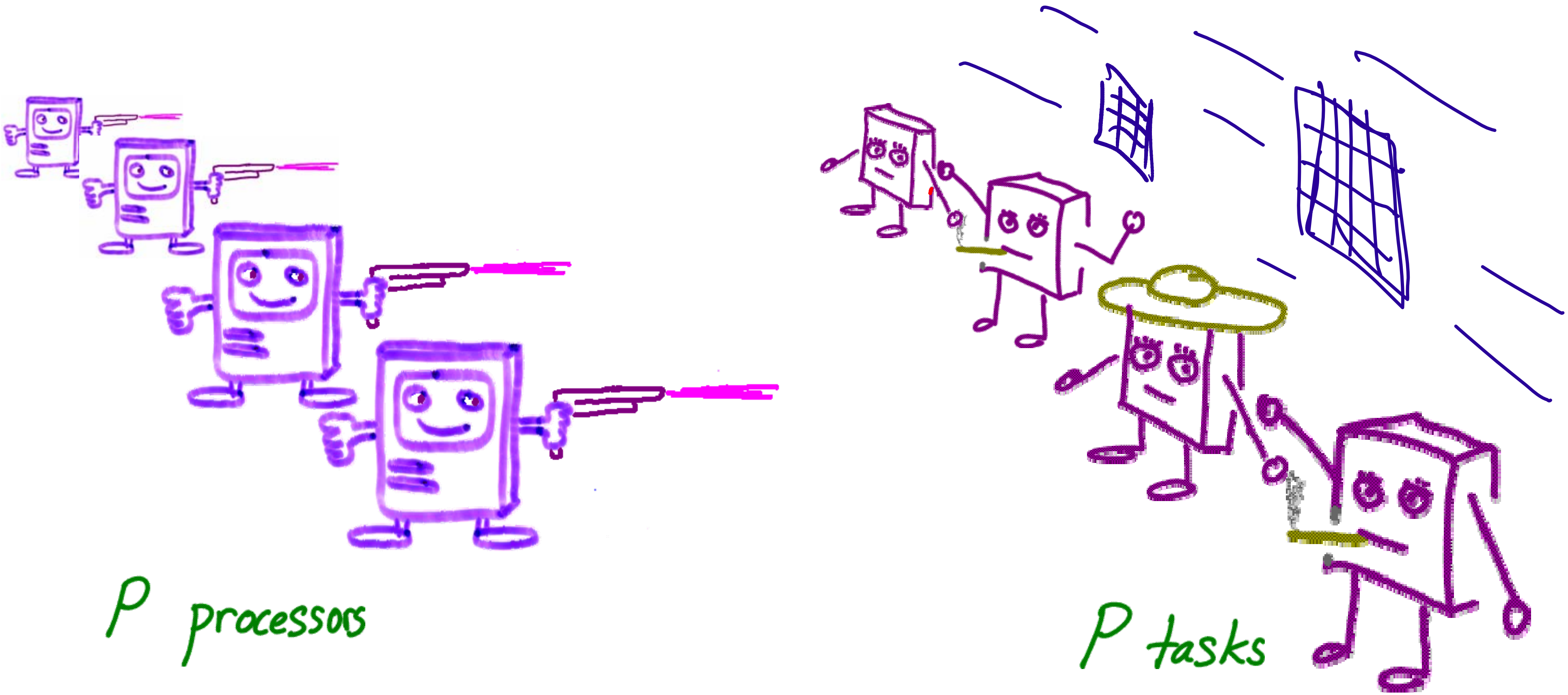
Overview

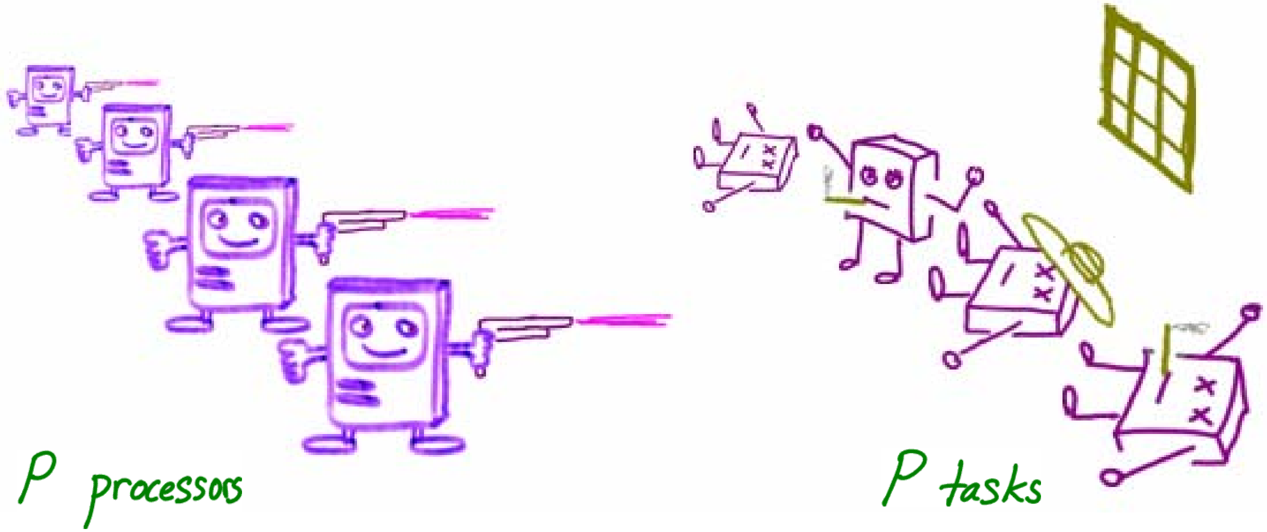
- DAGs
- Asynchrony + Oblivious Adversary
- Firing Squad Scheduling
- ALL vs. LEVEL
- Asymptotic Bounds + explanation
- Firing Squad Scheduling w/o dependencies
- DAG exhibiting Lower Bound

You are here

FSS with No Dependencies

- Each processor randomly chooses a victim (task) and executes that task.
- **Question:** how many rounds 'til all tasks finish?



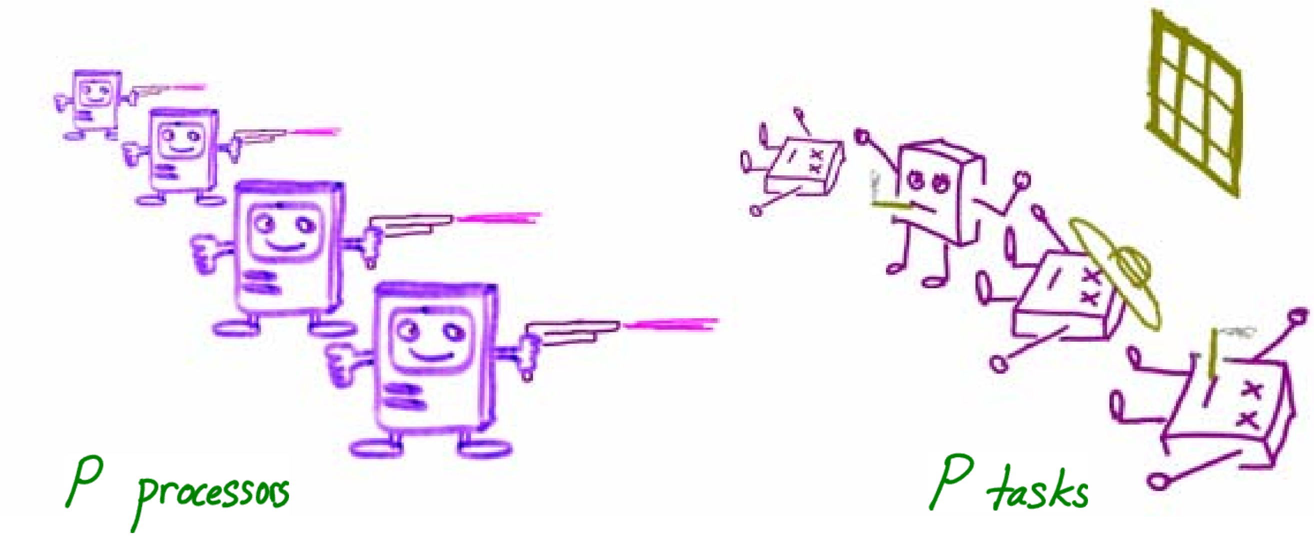


FIRST ROUND

$$\begin{cases} \Pr[\text{given task survives}] = \left(1-\frac{1}{p}\right)^{p} \approx \frac{1}{e} \\ E[\#\text{ tasks surviving}] = \frac{p}{e} \end{cases}$$

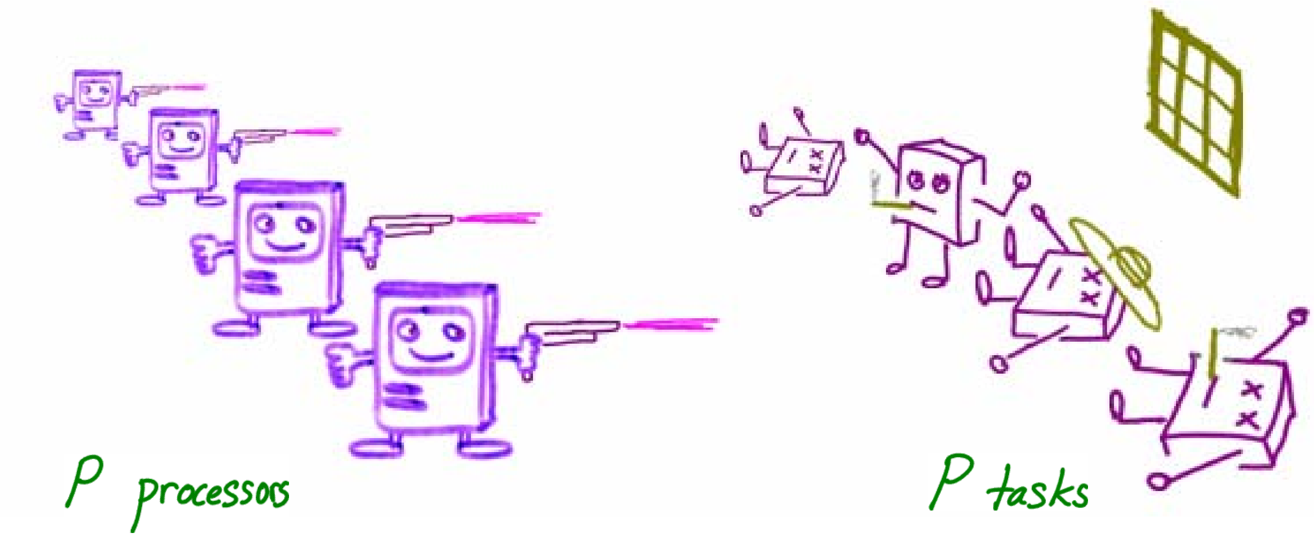
SECOND ROUND

$$\begin{cases} \Pr[\text{given task survives}] = \left(1-\frac{e}{p}\right)^{p} \approx \frac{1}{e^{e}} \\ E[\#\text{ tasks surviving}] = \frac{p}{e^{e}} \end{cases}$$



$\boxed{\text{THIRD ROUND}}$ $\begin{cases} \Pr[\text{given task survives}] = \left(1 - \frac{e^e}{P}\right)^P \approx \frac{1}{e^{e^e}} \\ E[\#\text{ tasks surviving}] = \frac{P}{e^{e^e}} \end{cases}$

$\Rightarrow$ After $\Theta(\log^* p)$ rounds, all tasks have been executed.



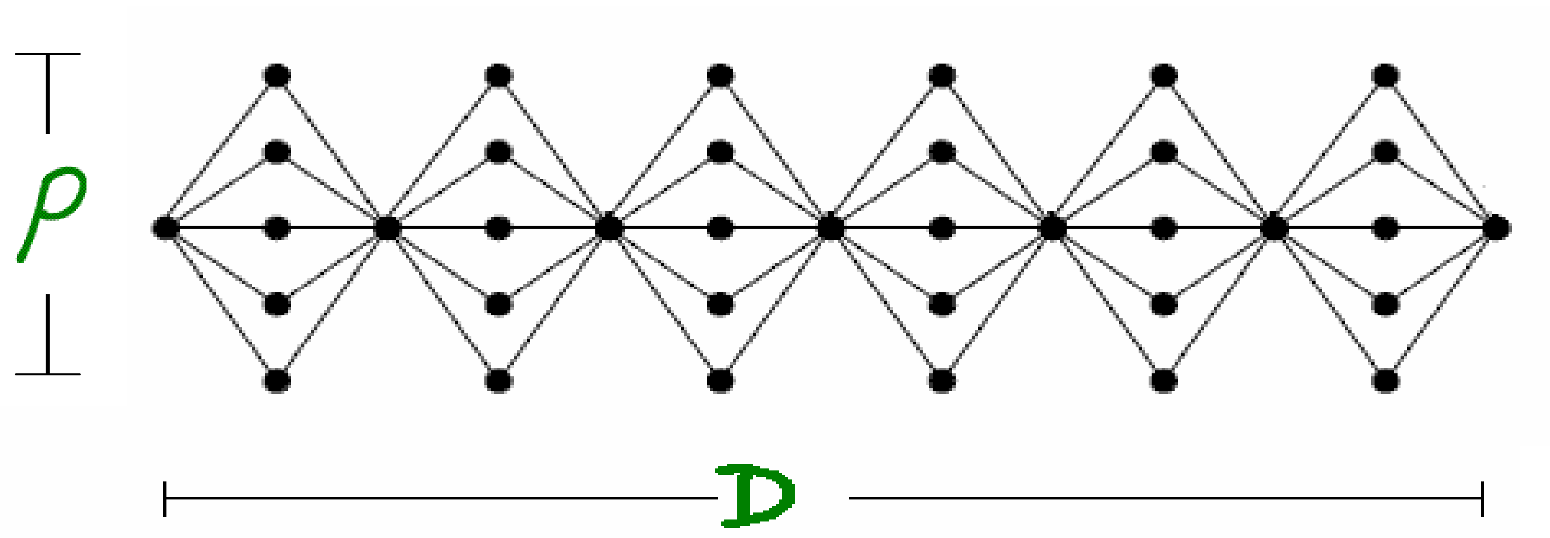
$\boxed{\text{THIRD ROUND}}$

$$\begin{cases} \Pr[\text{given task survives}] = \left(1 - \dfrac{e^e}{p}\right)^p \approx \dfrac{1}{e^{e^e}} \\[2ex] E[\#\text{ tasks surviving}] = \dfrac{p}{e^{e^e}} \end{cases}$$

Of course the proof is techically wrong but correct in spirit.

$\Rightarrow$ After $\Theta(\log^* p)$ rounds, all tasks have been executed.

Constructing a Difficult DAG for ALL



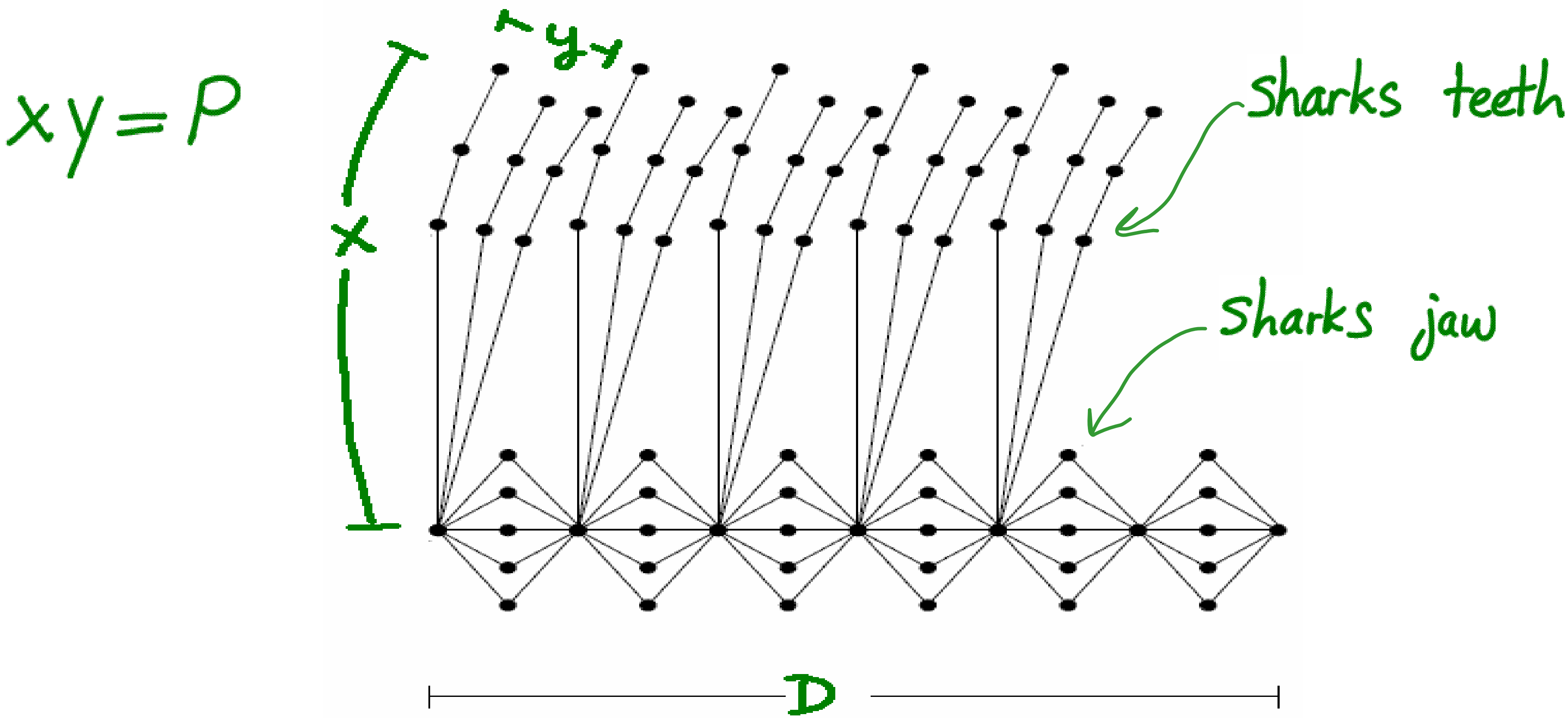
Makespan of DAG is $O(D \log^* P)$.

So far things are good.

Now we add difficult structure to the DAG....

Adding "*Shark's Teeth*" to Our Dag

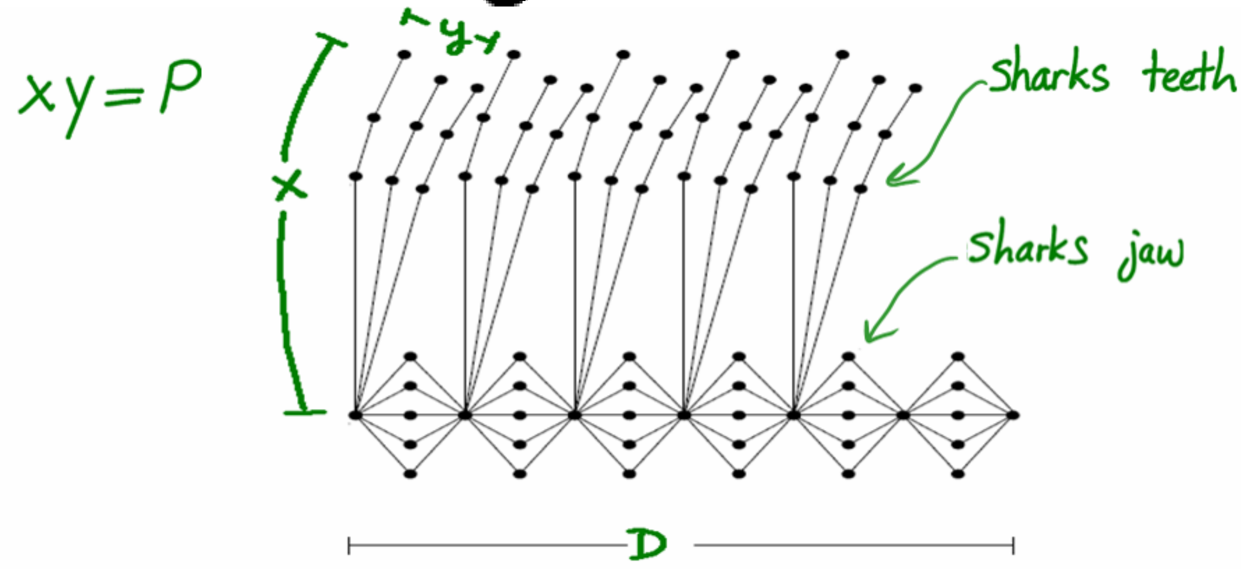
Important aspect of sharks teeth: Whenever one tooth falls out, another one fills the gap.



Goal: keep from executing each jaw for *x* steps.

Optimize so that *x* is as big as possible.

Constructing a Difficult DAG for ALL

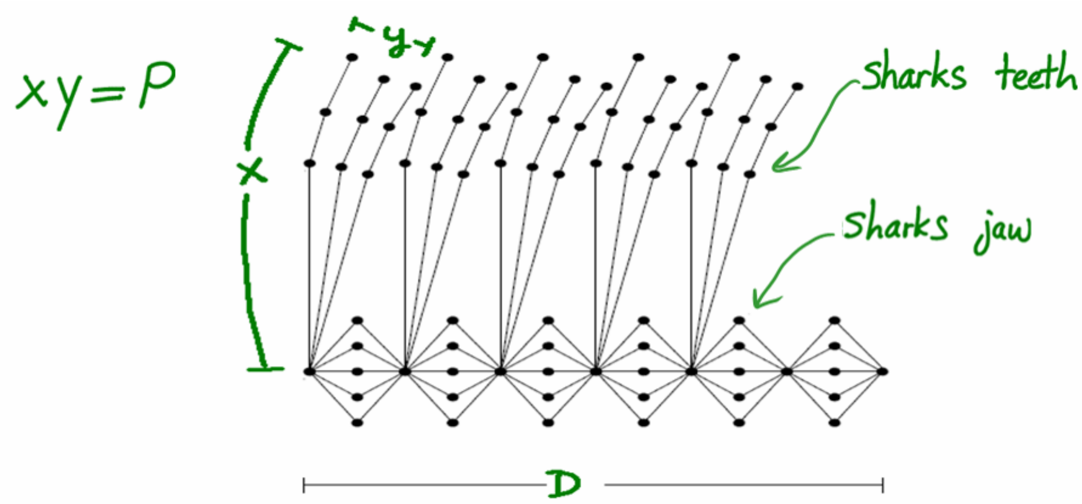


$$\Pr[\text{node in jaw not executed}] \geq \left(1 - \frac{1}{y}\right)^{P} \approx e^{-P/y}$$

\# unexecuted nodes decreases by $\Theta(e^{P/y})$ factor in each round.

After $\Theta\left(\log_{e^{P/y}} P\right) = \Theta\left(\frac{y \log P}{P}\right)$ rounds, all nodes in jaw are executed.

Now Optimize for x and y



Want to show x steps to finish jaw and x is # levels of shark teeth:

$$x \geq \frac{y \log P}{P}$$

Total work is $PD = W$.

$$xy = P$$

Substituting: $x \geq \frac{\log P}{x} \implies x = \sqrt{\log P}, \quad y = P/\sqrt{\log P}$

Need $\Theta(\sqrt{\log P})$ rounds, rather than $\Theta(\log^* P)$ rounds, to go from one level of DAG to next.

What I'm not showing you…

- Asynchrony—how do we make it go away.
- Upper bound for ALL—Technical. Gives less insight than lower bound.
- Correct proofs.

Conclusion

- Tight analysis of Firing Squad Scheduling of DAG on Asynchronous Procs.
  (Previous work was for DAGs with synchronization barriers.)
- LEVEL >> ALL, i.e.,
  removing dependencies between jobs can make makespan asymptotically worse!

Results

Makespan of ALL

$$\Theta\left(\begin{cases} \dfrac{W}{p\pi_{ave}} & \text{when } \dfrac{W}{D} \geq p\log p \\ (\log p)^{\alpha}\dfrac{W}{p\pi_{ave}} + (\log p)^{1-\alpha}\dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \text{ for } \alpha \in [0,1] \\ \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

Makespan of LEVEL

$$\Theta\left(\frac{W}{p\pi_{ave}} + [\log^* p - \log^*(pD/W)]\,\frac{D}{\pi_{ave}}\right)$$

Results

Makespan of ALL

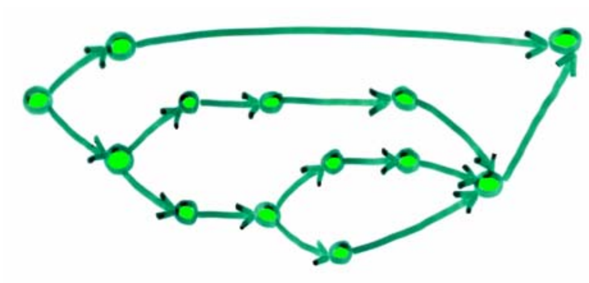
$$\Theta\left(\begin{cases} \dfrac{W}{p\pi_{ave}} & \text{when } \dfrac{W}{D} \geq p\log p \\ (\log p)^{\alpha}\dfrac{W}{p\pi_{ave}} + (\log p)^{1-\alpha}\dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} = p(\log p)^{1-2\alpha}, \text{ for } \alpha \in [0,1] \\ \dfrac{D}{\pi_{ave}} & \text{when } \dfrac{W}{D} \leq \dfrac{p}{\log p} \end{cases}\right)$$

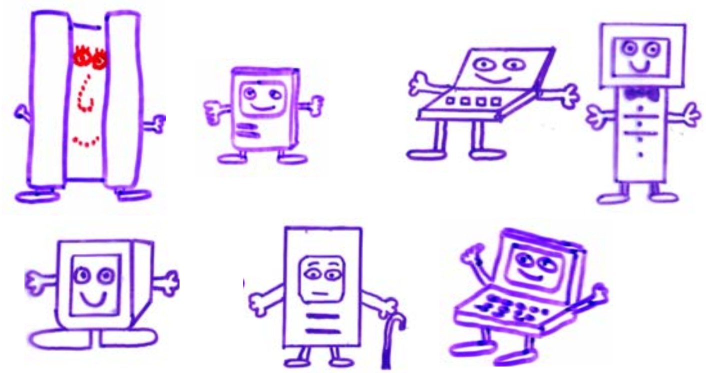
Makespan of LEVEL

$$\Theta\left(\frac{W}{p\pi_{ave}} + [\log^* p - \log^*(pD/W)]\frac{D}{\pi_{ave}}\right)$$

Other Related Work on Different-Speed Procs

- Intuitively: want fast procs on long paths in DAG.





- Scheduling on Related Procs [Jaffe 80] [Chudak, Shmoys 97] [Chekuri, Bender 01]
  - Speeds don't vary. Centralized scheduler.
  - $O(\log P)$-approx is best known.
- Graham/Brent bound for different speeds [Bender,Rabin 02]
  - Speeds vary, but procs know their own speeds
  - Applications to Cilk.
  - Efficient in common case that $W/P \gg D$.