The background of the slide is a reproduction of Salvador Dalí's painting 'The Persistence of Memory'. It depicts a landscape with a blue sky, a sea, and a rocky shore. In the foreground, a wooden table holds a melting pocket watch, a plate of olives, and a melting pocket watch. In the background, a melting pocket watch is draped over a branch, and another melting pocket watch is draped over a rock. The title 'Optimal Dynamic Gradient Clock Synchronization' is overlaid on a black semi-transparent rectangle in the center of the image.

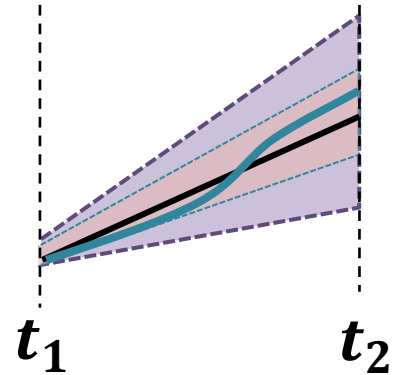
Optimal Dynamic Gradient Clock Synchronization

Fabian Kuhn
Christoph Lenzen
Thomas Locher
Rotem Oshman

Clock Synchronization

- Each node u has a hardware clock H_u :

$$(1 - \rho)(t_2 - t_1) \leq H_u(t_2) - H_u(t_1) \leq (1 + \rho)(t_2 - t_1)$$



- Messages subject to delay
- Goal: output logical clocks L_u that
 - Mimic real time: $\alpha(t_2 - t_1) \leq L_u(t_2) - L_u(t_1) \leq \beta(t_2 - t_1)$
 - Are “well synchronized”

Clock Synchronization

- Traditional goal – minimize *global skew*:

For all nodes u, v and times t ,

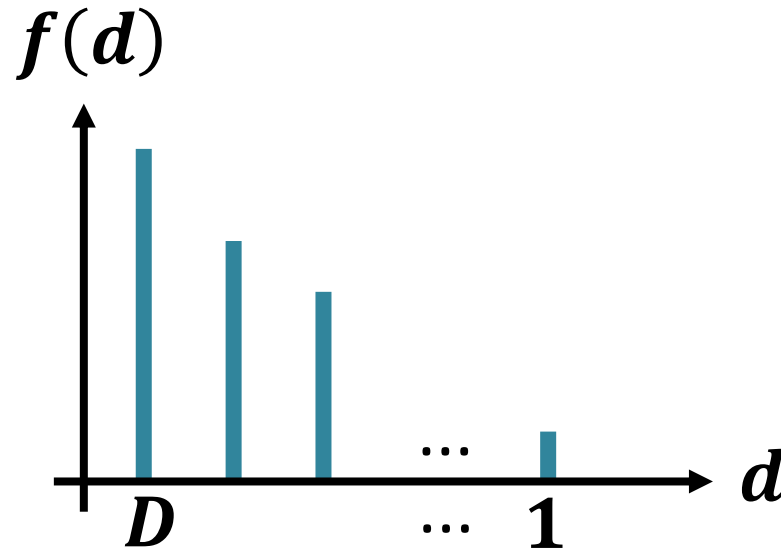
$$L_u(t) - L_v(t) \leq G$$

Clock Synchronization

- Modern goal – minimize *gradient skew*:

For all nodes u, v and times t ,

$$L_u(t) - L_v(t) \leq f(\text{dist}(u, v))$$



A Quick Survey

- [Fan & Lynch 2006]:

$$f(\mathbf{1}) = \Omega(\log D / \log \log D)$$

- [Locher & Wattenhofer 2006]:

$$f(\mathbf{1}) = O(\sqrt{\rho D})$$

- [Lenzen, Locher & Wattenhofer 2008]:

$$f(\mathbf{1}) = O(\log D)$$

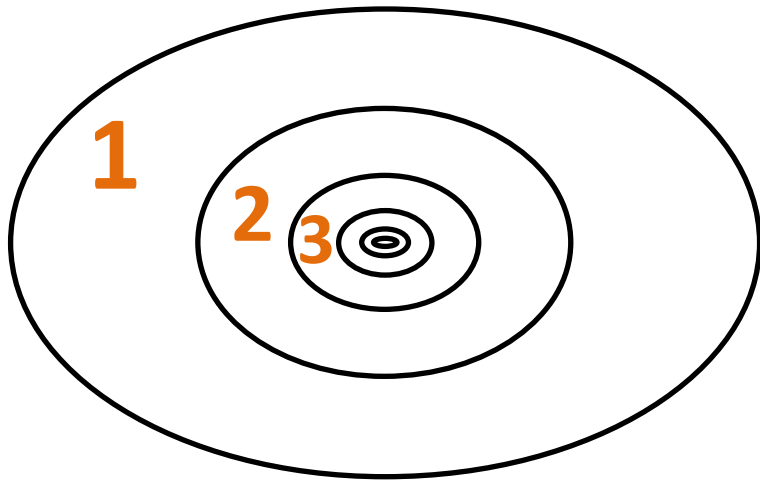
- [Lenzen, Locher & Wattenhofer 2009]:

$$f(\mathbf{1}) = \Omega(\log D)$$

A Quick Survey

- [Lenzen, Locher & Wattenhofer 2008,2009]:

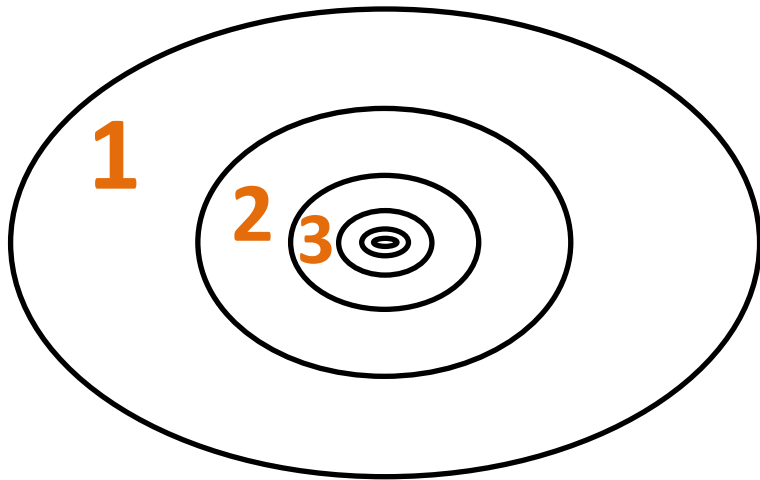
$$f(d) = \Theta \left(d \cdot \log \left(\frac{2D}{d} \right) \right)$$



A Quick Survey

- [Lenzen, Locher & Wattenhofer 2008,2009]:

$$f(d) = \Theta \left(d \cdot \boxed{s} \right)$$



	d	
Layer 1:	D	$\log \left(\frac{2D}{D} \right) = 1$
Layer 2:	$D/2$	$\log \left(\frac{2D}{D/2} \right) = 2$
...		
Layer s :	$D/2^{s-1}$	$\log \left(\frac{2D}{D/2^{s-1}} \right) = s$

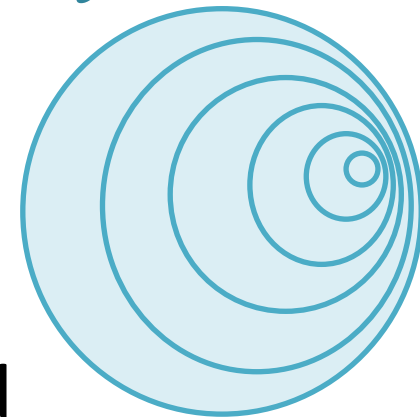
Dynamic Networks

- Fixed set V of nodes
- Edges change arbitrarily:

$$E : \mathcal{R} \rightarrow V^2$$

- If $(u, v) \in E(t)$, node u can estimate L_v
 - Subject to uncertainty $\epsilon_{(u,v)}$
 - In this talk: $\epsilon_{(u,v)} = 1$
- Dynamic diameter D :

Time required to complete a flood

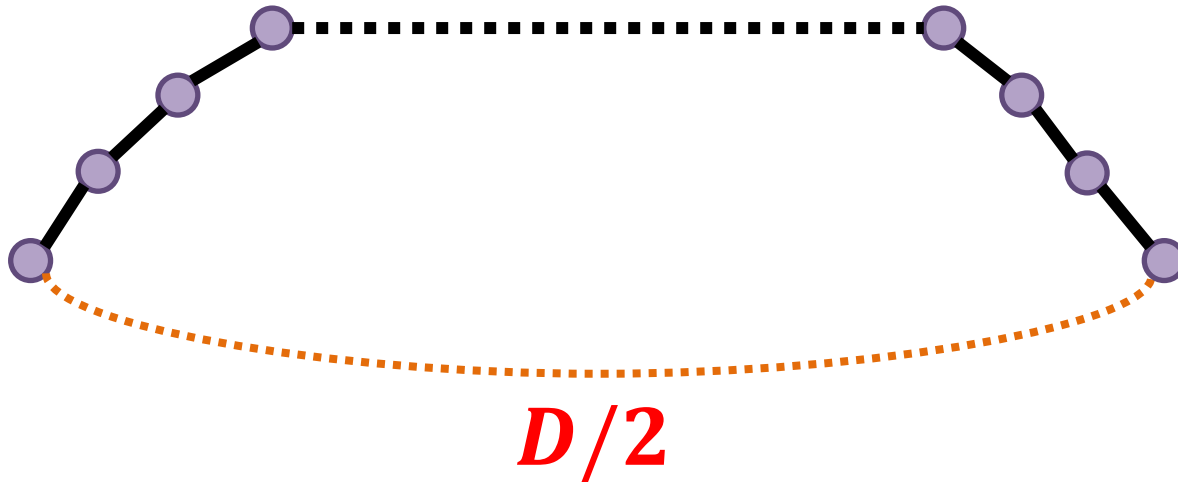


Dynamic Networks

- Clock sync in dynamic networks:
 - Global skew should be $O(D)$ always
 - Gradient skew?
 - Should be good for persistent paths

Lower Bound

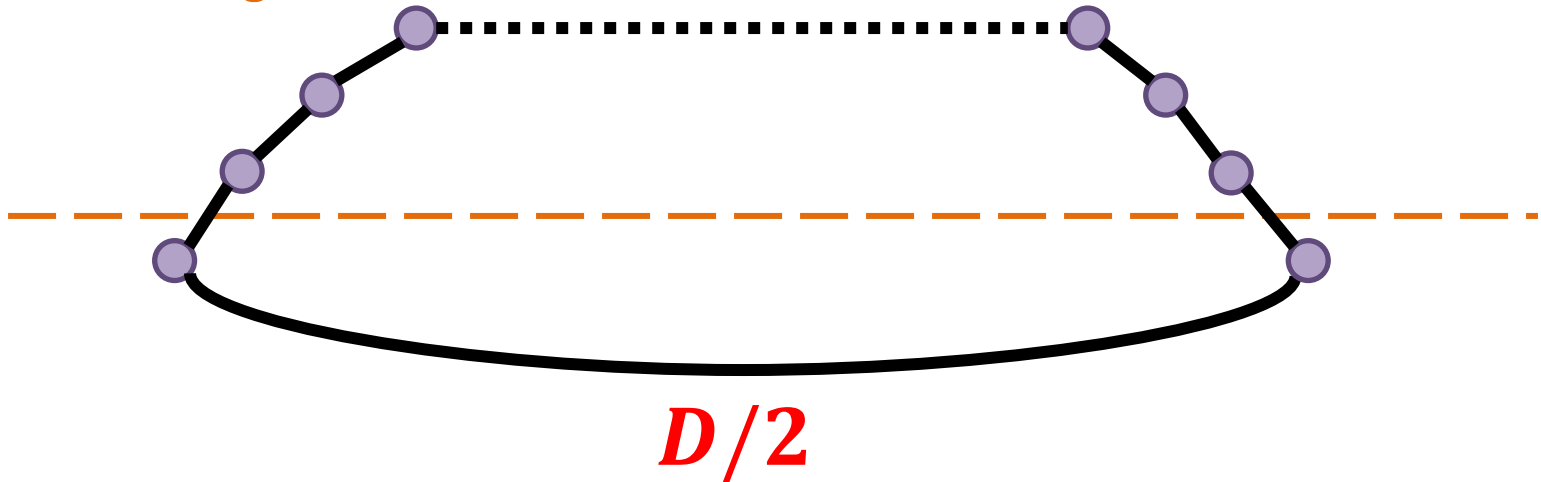
- Line network, delay $\in [0,1]$
- Classical lower bound **[Biaz, Welch 2001]:**



Lower Bound

- Line network, delay $\in [0,1]$
- Time = 0:

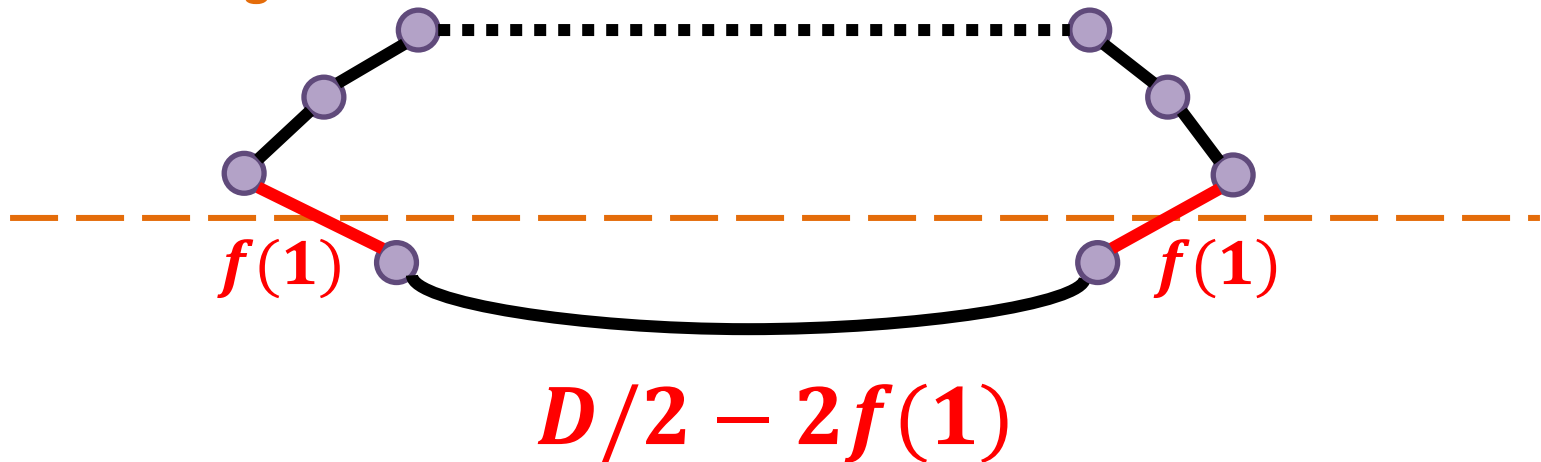
Not aware
of new edge



Lower Bound

- Line network, delay $\in [0,1]$
- Time = 1:

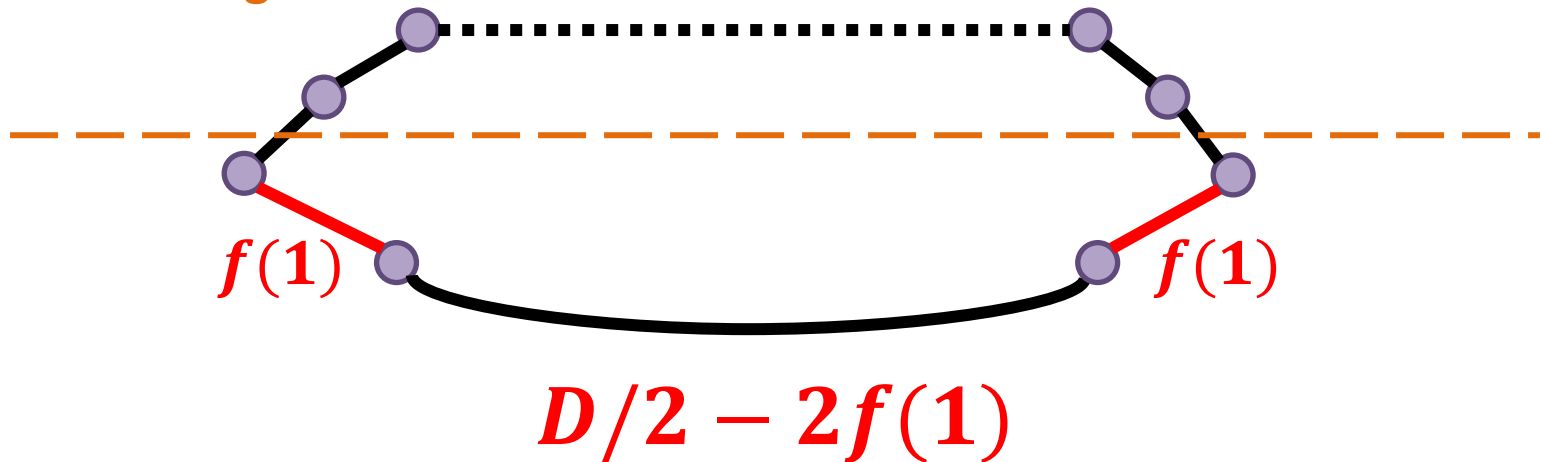
Not aware
of new edge



Lower Bound

- Line network, delay $\in [0,1]$
- Time = 2:

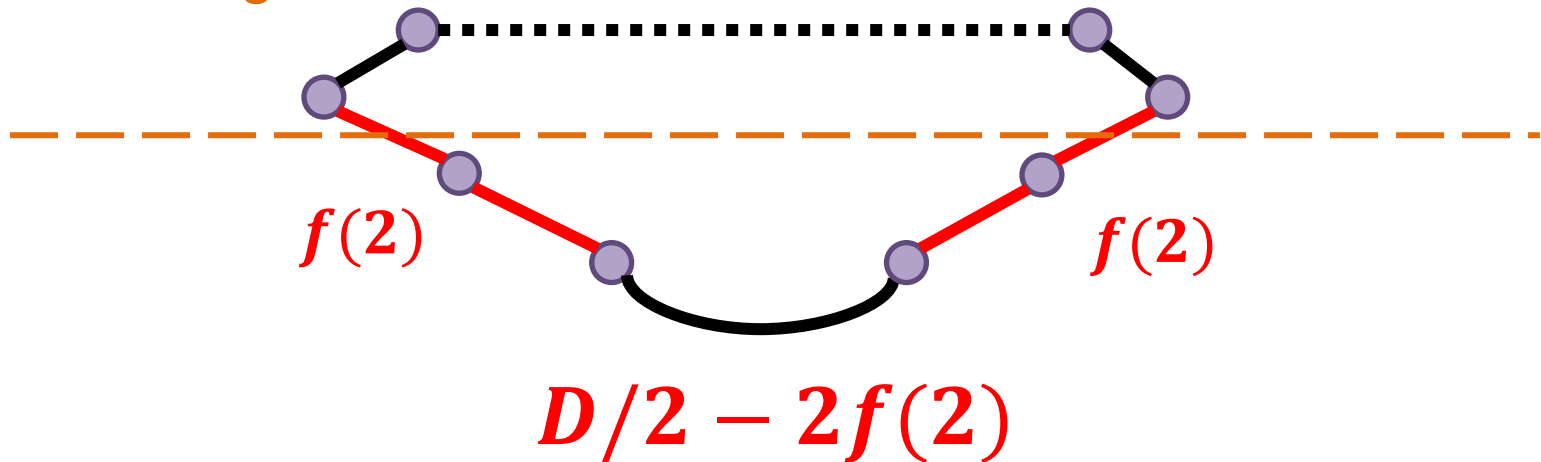
Not aware
of new edge



Lower Bound

- Line network, delay $\in [0,1]$
- Time = 2:

Not aware
of new edge

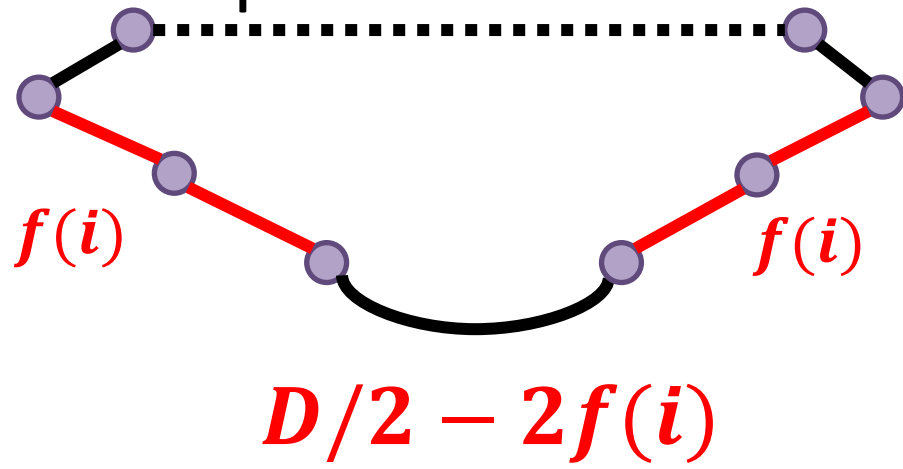


Lower Bound

- Line network, delay $\in [0,1]$
- Time = i : skew $\geq D/2 - 2f(i)$

$$i = o(D) \\ \Rightarrow f(i) = o(D)$$

$\Rightarrow \Omega(D)$ time required to reduce skew



Dynamic Networks

- Clock sync in dynamic networks:
 - Global skew should be $O(D)$ always
 - Gradient skew?
 - **Stable skew:**

If a path of length d connects u, v for “long enough”,
$$L_u(t) - L_v(t) \leq f(d)$$

- **Stabilization time:**

How long to reach stable skew



Previous Work

- [Kuhn, Locher & O. SPAA'09]:

- Lower bound:

Stabilization time = $\Omega(D/f(\mathbf{1}))$

(Regardless of initial skew on edge)

- An algorithm with

- $f(\mathbf{1}) = o(\sqrt{\rho D})$,
- $o(\sqrt{D/\rho})$ stabilization time

Not “true” gradient: $d = o(D) \not\Rightarrow f(d) = o(D)$

Our Results

1. An algorithm with:

- $O(d \cdot \log(D/d))$ stable skew
- $O(D)$ stabilization time.

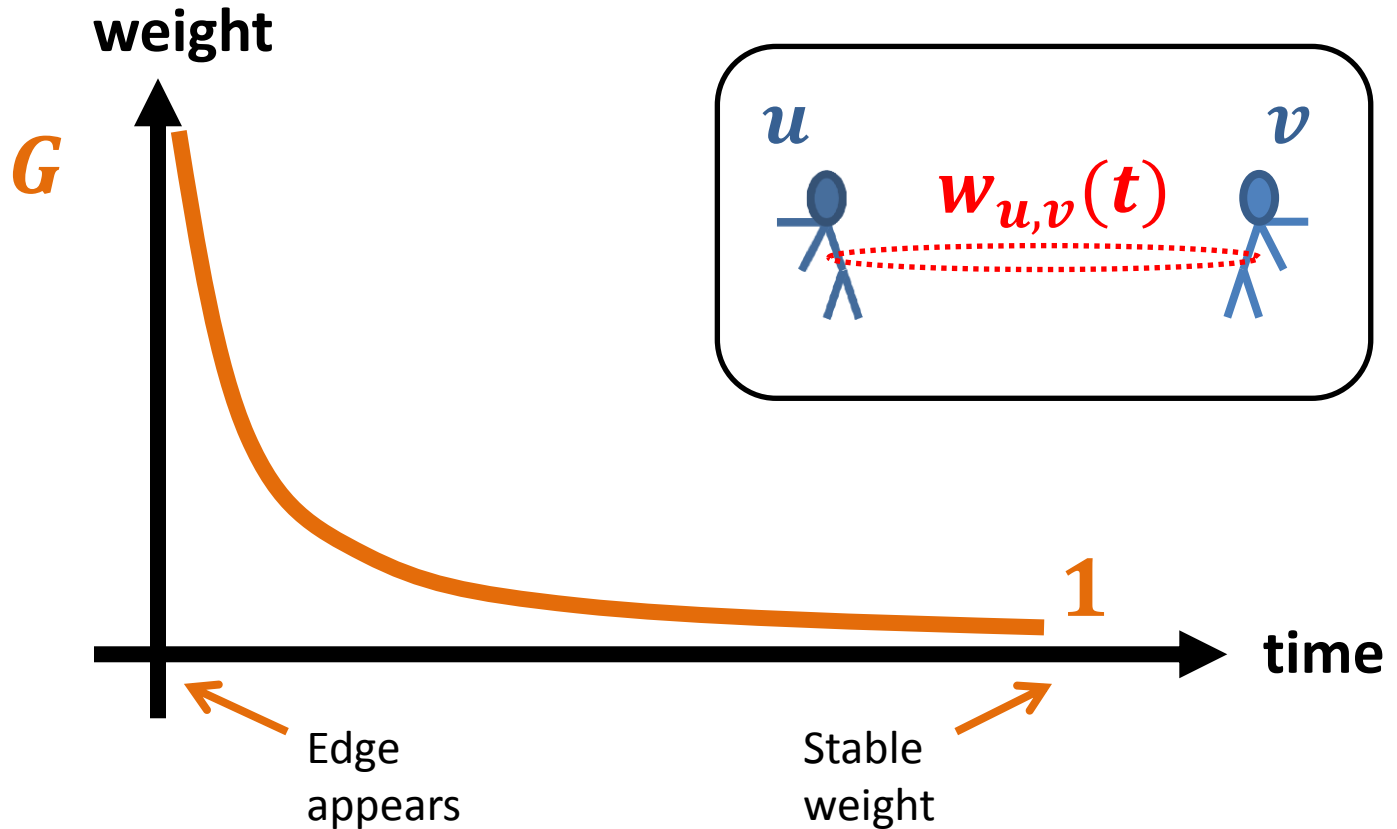
2. $\Omega(D)$ time is required for any “true” gradient algorithm.

• In this talk: an algorithm with

- $O(d \cdot \log(D/d))$ stable skew
- $O(D \log D)$ stabilization time.

Weight-Based Algorithm

- Each edge (u, v) has dynamic weight $w_{u,v}(t)$:



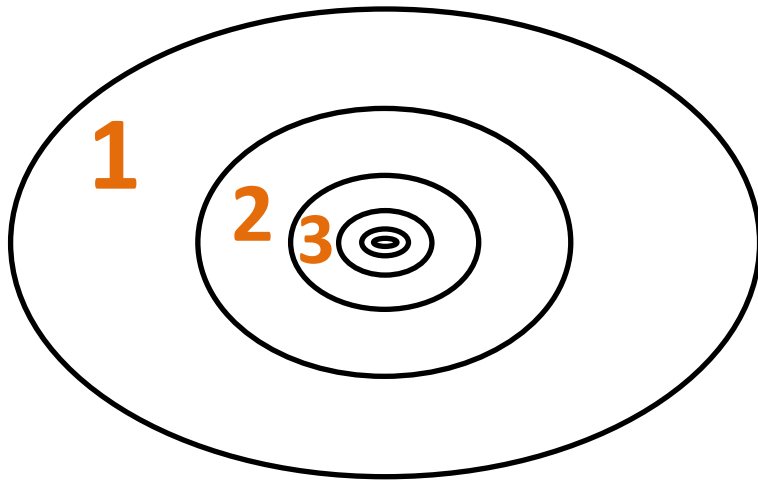
Weight-Based Algorithm

- Local algorithm:
 - Nodes maintain estimates of neighbors' clocks
 - Try to balance two demands:
 - Don't fall "too far behind" any neighbor
 - Don't get "too far ahead" of any neighbor
 - Two modes:
 - In slow mode, $\frac{dL_u}{dt} = \frac{dH_u}{dt}$
 - In fast mode, $\frac{dL_u}{dt} = (1 + \mu) \frac{dH_u}{dt}$

A Quick Survey

- [Lenzen, Locher & Wattenhofer 2008,2009]:

$$f(\overset{W}{\cancel{d}}) = \Theta\left(\overset{W}{\cancel{d}} \cdot \boxed{s}\right)$$



	$\overset{W}{\cancel{d}}$	
Layer 1:	D	$\log\left(\frac{2D}{D}\right) = 1$
Layer 2:	$D/2$	$\log\left(\frac{2D}{D/2}\right) = 2$
...		
Layer s :	$D/2^{s-1}$	$\log\left(\frac{2D}{D/2^{s-1}}\right) = s$

Weight-Based Algorithm

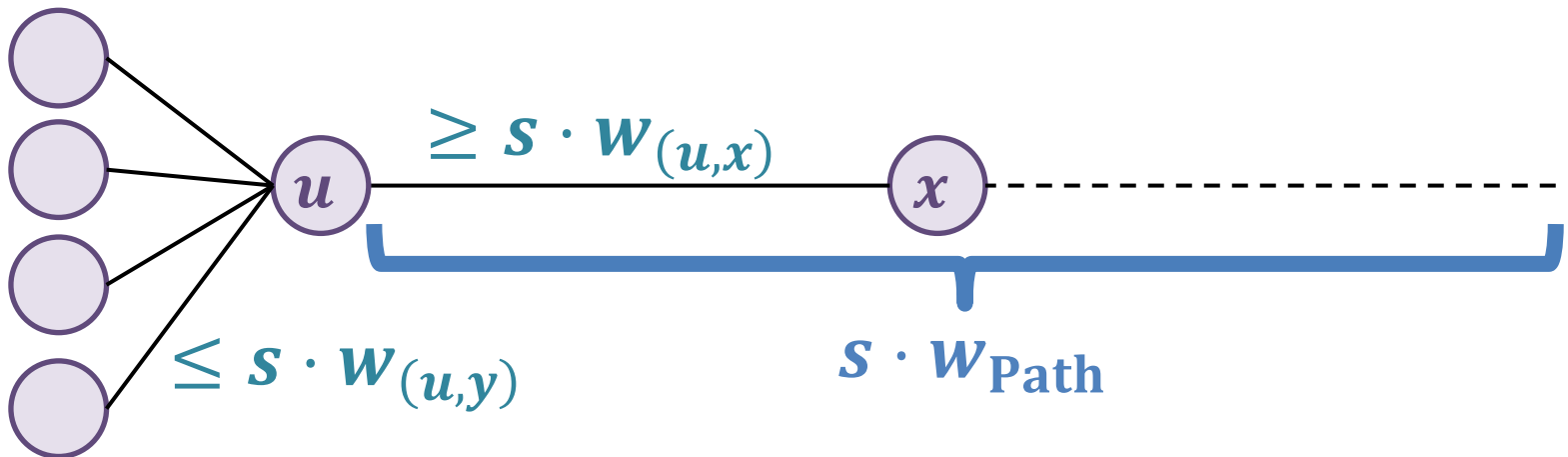
- Go into fast mode when for some integer s ,

For some neighbor x :

$$L_x(t) - L_u(t) \geq s \cdot w_{u,x}(t)$$

For all neighbors y :

$$L_u(t) - L_y(t) \leq s \cdot w_{u,y}(t)$$



Weight-Based Algorithm

- Go into fast mode when for some integer s ,

For some neighbor x :

$$L_x(t) - L_u(t) \geq s \cdot w_{u,x}(t)$$

For all neighbors y :

$$L_u(t) - L_y(t) \leq s \cdot w_{u,y}(t)$$

- Go into **slow mode** when for some integer s ,

For some neighbor x :

$$L_u(t) - L_x(t) \geq (s + 1/2) \cdot w_{u,x}(t)$$

For all neighbors y :

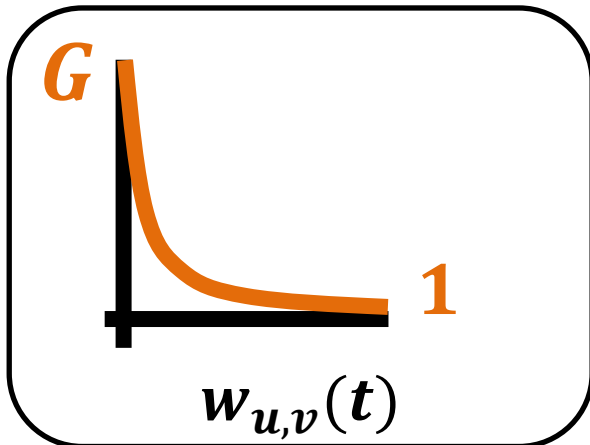
$$L_y(t) - L_u(t) \leq (s + 1/2) \cdot w_{u,y}(t)$$

Weight-Based Algorithm

- “In the background”:
maintain global skew of $G = O(D)$
- Goal: for any path P , maintain

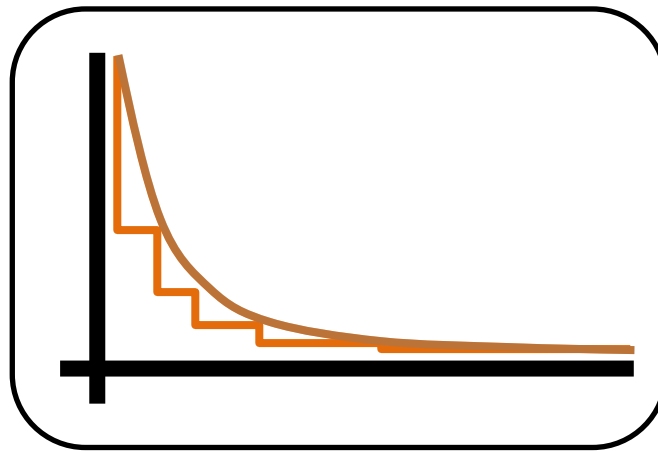
$$\text{skew} = O\left(w_P \cdot \log \frac{D}{w_P}\right) \quad \text{initially}$$

$$O\left(d \cdot \log \frac{D}{d}\right) \quad \text{eventually}$$



Optimal Algorithm

- No gradual weight decrease



- Add edges to each layer individually:

Layer 0 \Rightarrow **Layer 1** \Rightarrow ... \Rightarrow **Layer ($\log D$)**

$O(D)$ time

Conclusion

- We gave a dynamic clock-sync algorithm with:
 - Optimal stable skew: $O(d \cdot \log(D/d))$
 - Optimal stabilization time: $O(D)$
- Open problems:
 - Nodes joining and leaving
 - Transient edge failures

What Is All This Good For?

- Simulating a synchronous dynamic network
- [Kuhn, Lynch & O. STOC'10]:
In **synchronous** dynamic networks which are always connected, any function can be computed.