Synchronization Properties of CSMA Networks

Kuang Xu[†] (MIT)
Olivier Dousse (Nokia)
Patrick Thiran (EPFL)

ACM Sigmetrics
June 15, 2010

[†]Summer intern at EPFL in 2009.

Outline

1. Introduction
2. Model
3. Results
   - A Necessary and Sufficient Condition for Synchronization
   - Existence of Synchronized Schedules: General Networks
4. Summary & Future Work

Outline

1. Introduction
2. Model
3. Results
   - A Necessary and Sufficient Condition for Synchronization
   - Existence of Synchronized Schedules: General Networks
4. Summary & Future Work

Scheduling in Wireless Networks

- **Problem:** scheduling transmissions in a wireless network with presence of interference:
- Solution 1: **Slotted TDMA**
  - Users transmit during pre-assigned slots (e.g. cellular networks)
  - **Pros**: simple, collision-free, operator has full control
  - **Cons**: requires frequent synchronization
- Solution 2: **Random access / CSMA**
  - Users access medium after random back-offs (e.g. ALOHA, 802.11 Wi-Fi)
  - Modern schemes use **carrier sensing** to further avoid collision (e.g. 802.11)
  - **Pros**: robust, decentralized
  - **Cons**: unavoidable collisions, operator has less control

Scheduling in Wireless Networks

- **Problem:** scheduling transmissions in a wireless network with presence of interference:
- Solution 1: **Slotted TDMA**
  - Users transmit during pre-assigned slots (e.g. cellular networks)
  - **Pros**: simple, collision-free, operator has full control
  - **Cons**: requires frequent synchronization
- Solution 2: **Random access / CSMA**
  - Users access medium after random back-offs (e.g. ALOHA, 802.11 Wi-Fi)
  - Modern schemes use **carrier sensing** to further avoid collision (e.g. 802.11)
  - **Pros**: robust, decentralized
  - **Cons**: unavoidable collisions, operator has less control

Scheduling in Wireless Networks

- **Problem:** scheduling transmissions in a wireless network with presence of interference:
- Solution 1: **Slotted TDMA**
  - Users transmit during pre-assigned slots (e.g. cellular networks)
  - **Pros**: simple, collision-free, operator has full control
  - **Cons**: requires frequent synchronization
- Solution 2: **Random access / CSMA**
  - Users access medium after random back-offs (e.g. ALOHA, 802.11 Wi-Fi)
  - Modern schemes use **carrier sensing** to further avoid collision (e.g. 802.11)
  - **Pros**: robust, decentralized
  - **Cons**: unavoidable collisions, operator has less control

Scheduling in Wireless Networks

This talk

Q: Can one enforce **synchronized TDMA** schedule, hence **collision-free**, **without** synchronization?

A: Yes, with carrier sensing.

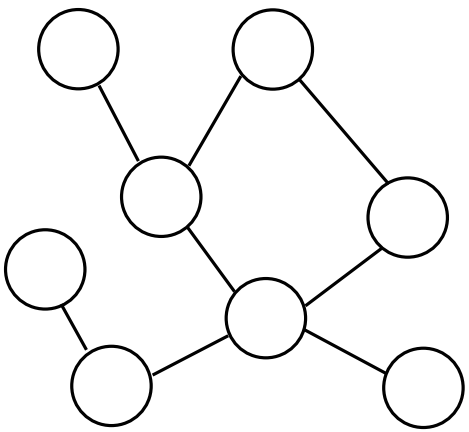
Mathematical tools:

- Stochastic recursive sequences [Baccelli & Liu 92]
- Elementary graph theory

Outline

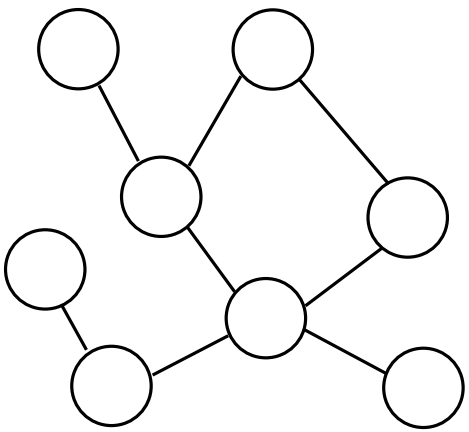
1. Introduction
2. Model
3. Results
   - A Necessary and Sufficient Condition for Synchronization
   - Existence of Synchronized Schedules: General Networks
4. Summary & Future Work

Network Model



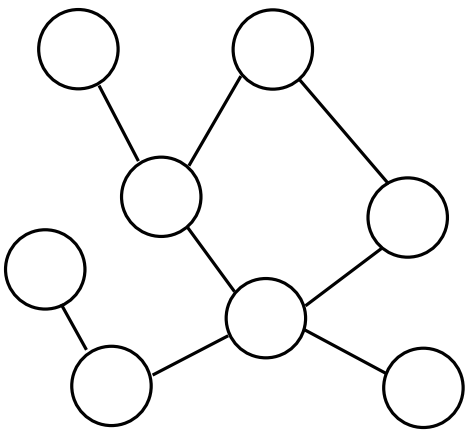
- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

Network Model



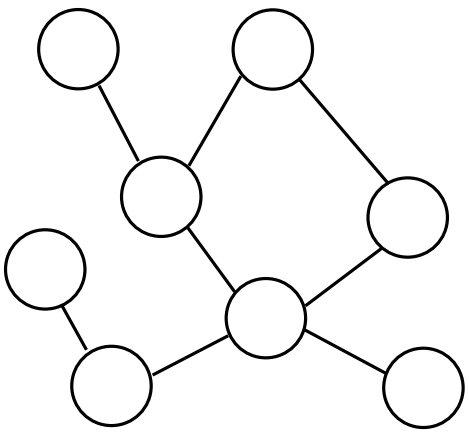
- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

Network Model



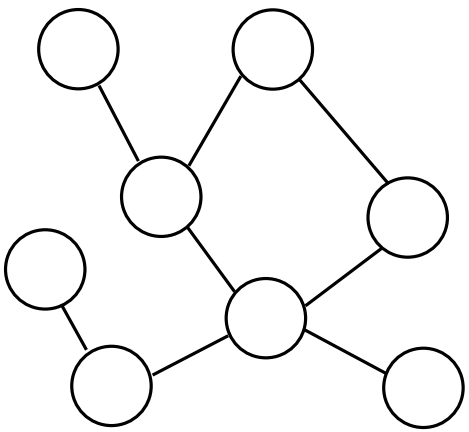
- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

Network Model



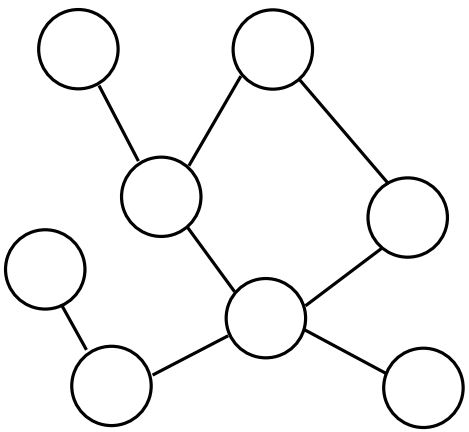
- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

Network Model



- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

Network Model



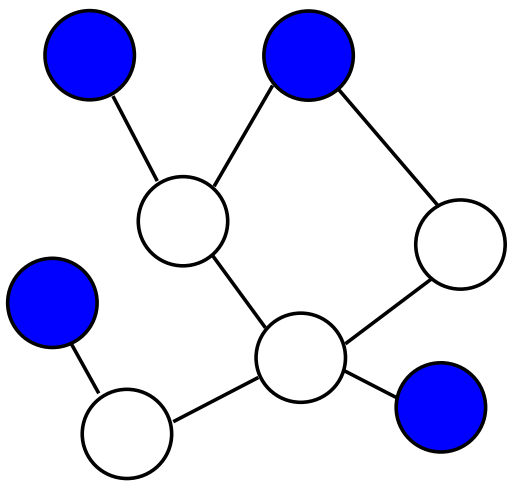
- Interference constraints modeled by **contention graph**, $\mathcal{G}_I = (\mathcal{E}_I, \mathcal{V}_I)$
  - **Nodes** are transmission entities (physical nodes **or** virtual links)
  - Two nodes share an **edge** if cannot concurrently transmit
- CSMA assumption
  - Detect **starting** of neighbors' transmissions (NOT to decode)
- Equal packet length (approximately)
  - True for many real networks
- Long packet backlogs
  - May send dummy packets

MIS Schedule

- Schedule: a **periodic sequence** $\{\mathcal{L}(t)\}_{t\in\mathbb{N}}$, where

$$\mathcal{L}(t) \subset \mathcal{V}_I, \text{ and } \mathcal{L}(t) = \mathcal{L}(t') \text{ if } t = t' \bmod K.$$

- Each $\mathcal{L}(t)$ is an **maximal independent set (MIS)** of $\mathcal{V}_I$
  - $\mathcal{L}(t)$ must be independent set to avoid collision
  - Made "maximal" because
    - Maximize channel re-use (otherwise wasted)
    - **Relay property**: always has ≥ 1 neighbor transmitting
- Fairness: $\{\mathcal{L}(1), \mathcal{L}(2), ..., \mathcal{L}(K)\}$ covers all of $\mathcal{V}_I$

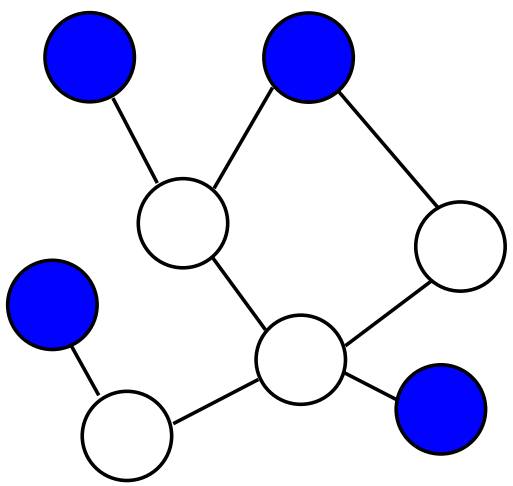


MIS Schedule

- Schedule: a **periodic sequence** $\{\mathcal{L}(t)\}_{t\in\mathbb{N}}$, where

$$\mathcal{L}(t) \subset \mathcal{V}_I, \text{ and } \mathcal{L}(t) = \mathcal{L}(t') \text{ if } t = t' \bmod K.$$

- Each $\mathcal{L}(t)$ is an **maximal independent set (MIS)** of $\mathcal{V}_I$
  - $\mathcal{L}(t)$ must be independent set to avoid collision
  - Made "maximal" because
    - Maximize channel re-use (otherwise wasted)
    - **Relay property**: always has ≥ 1 neighbor transmitting
- Fairness: $\{\mathcal{L}(1), \mathcal{L}(2), ..., \mathcal{L}(K)\}$ covers all of $\mathcal{V}_I$

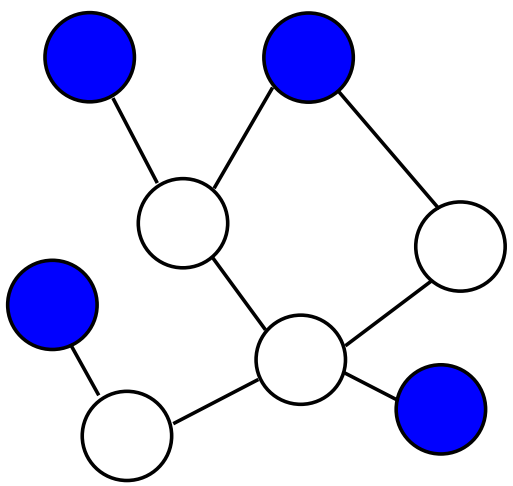


MIS Schedule

- Schedule: a **periodic sequence** $\{\mathcal{L}(t)\}_{t\in\mathbb{N}}$, where

$$\mathcal{L}(t) \subset \mathcal{V}_I, \text{ and } \mathcal{L}(t) = \mathcal{L}(t') \text{ if } t = t' \bmod K.$$

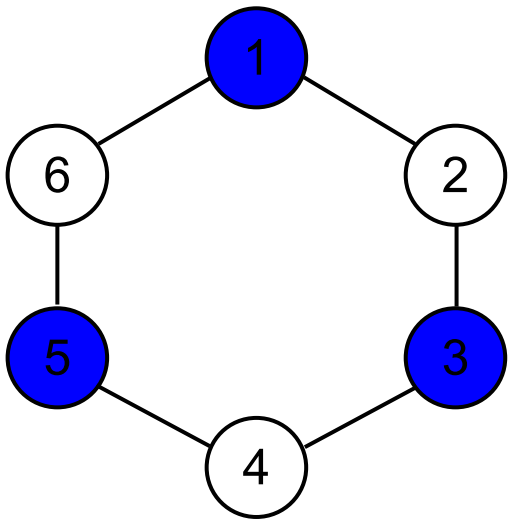
- Each $\mathcal{L}(t)$ is an **maximal independent set (MIS)** of $\mathcal{V}_I$
  - $\mathcal{L}(t)$ must be independent set to avoid collision
  - Made "maximal" because
    - Maximize channel re-use (otherwise wasted)
    - **Relay property**: always has ≥ 1 neighbor transmitting
- Fairness: $\{\mathcal{L}(1), \mathcal{L}(2), ..., \mathcal{L}(K)\}$ covers all of $\mathcal{V}_I$



Six-node Ring Examples:

Schedule A:

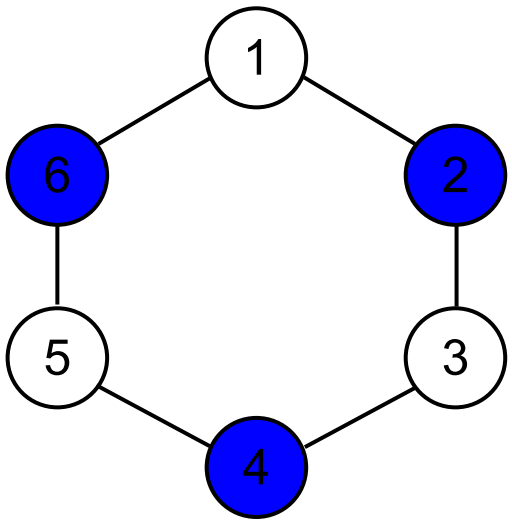
$$\mathcal{L}(1) = \{1, 3, 5\}, \mathcal{L}(2) = \{2, 4, 6\}, \cdots$$



Six-node Ring Examples:

Schedule A:

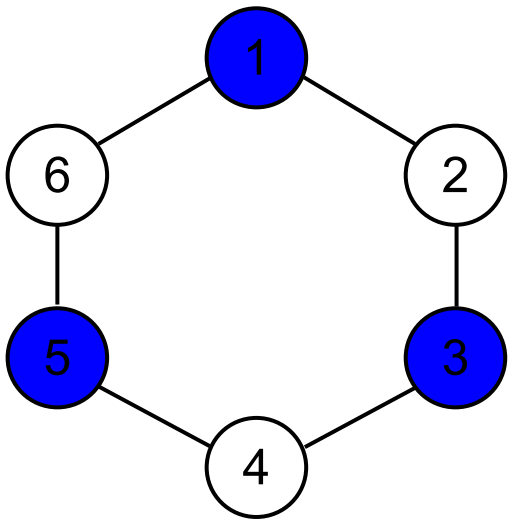
$$\mathcal{L}(1) = \{1, 3, 5\}, \mathcal{L}(2) = \{2, 4, 6\}, \cdots$$



Six-node Ring Examples:

Schedule A:

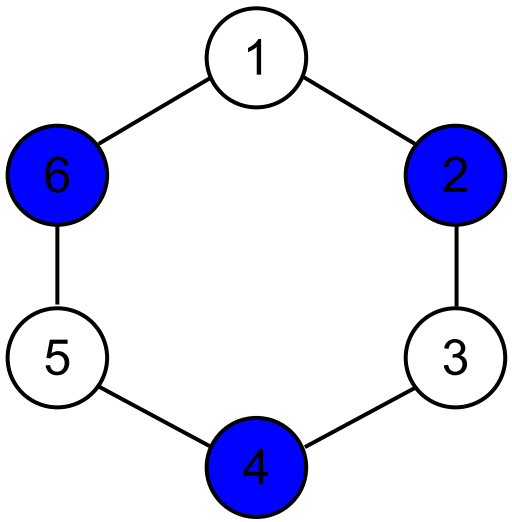
$$\mathcal{L}(1) = \{1, 3, 5\}, \mathcal{L}(2) = \{2, 4, 6\}, \cdots$$



Six-node Ring Examples:

Schedule A:

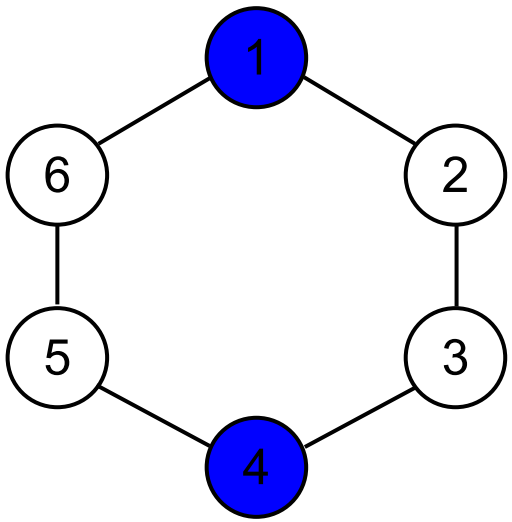
$$\mathcal{L}(1) = \{1, 3, 5\}, \mathcal{L}(2) = \{2, 4, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

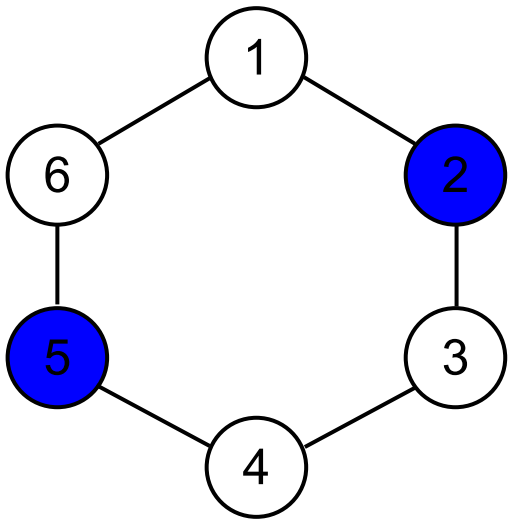
$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

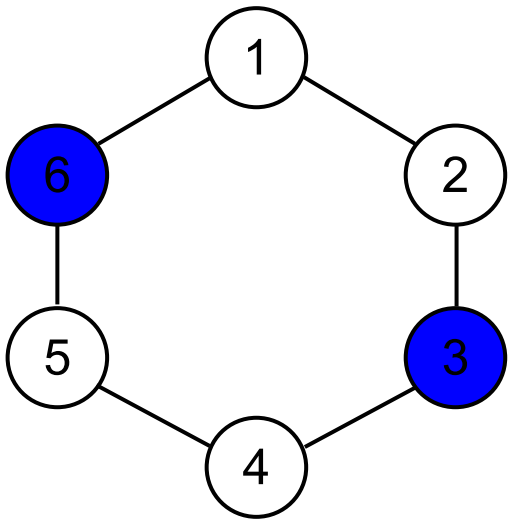
$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

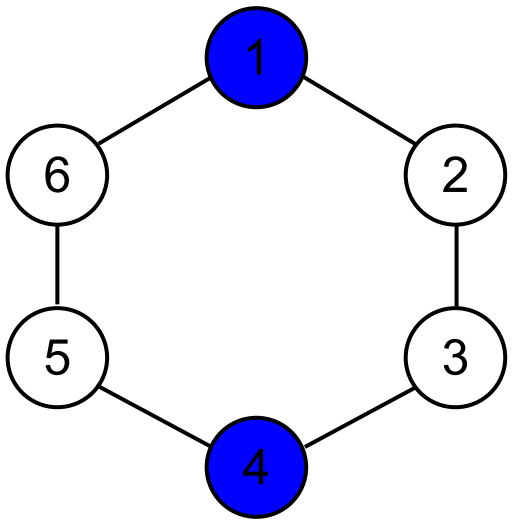
$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

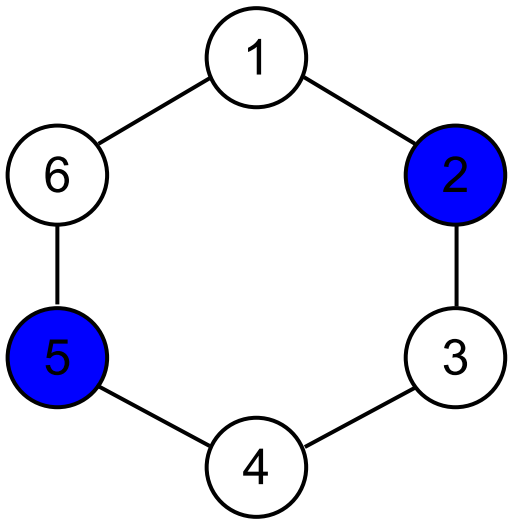
$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

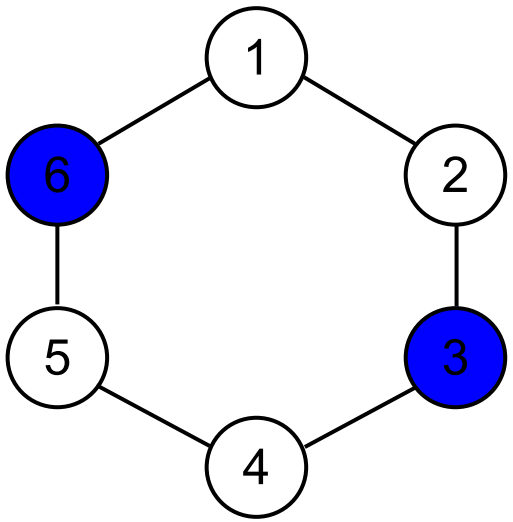
$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



Six-node Ring Examples:

Schedule B:

$$\mathcal{L}(1) = \{1, 4\}, \mathcal{L}(2) = \{2, 5\}, \mathcal{L}(3) = \{3, 6\}, \cdots$$



A (Naive) Local Scheduling Algorithm: Will It Work?

A Local CSMA Algorithm

- All nodes given schedule $\{\mathcal{L}(t)\}_{t\in\mathbb{N}}$
- Nodes in $\mathcal{L}(t+1)$ simply wait until all neighbors in $\mathcal{L}(t+1)$ finish. It then transmits after some **random jitter, delay**, or **back-off**.

Questions

1. If all transmissions $\approx$ equal length, do nodes in $\mathcal{L}(t)$ stay "synchronized" as $t \to \infty$?
2. And how do we define synchronization anyways?

A (Naive) Local Scheduling Algorithm: Will It Work?

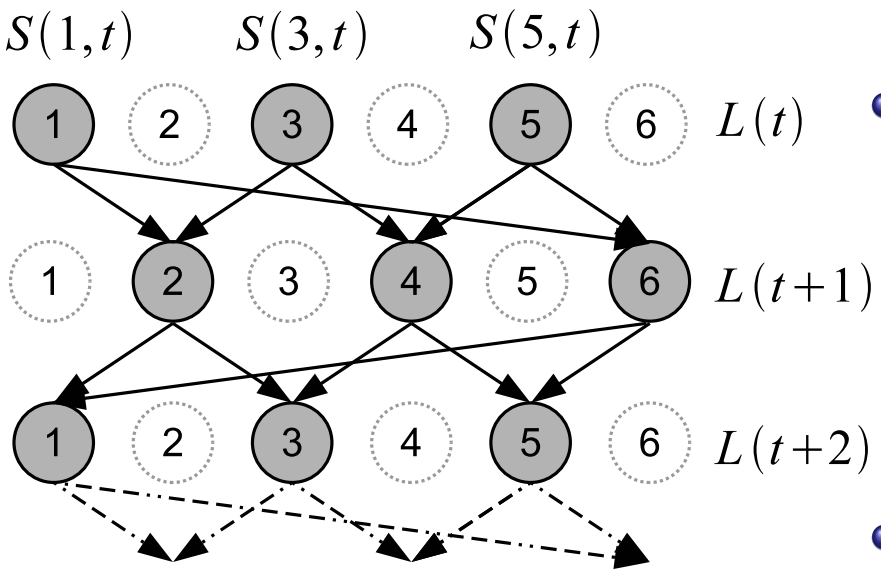
A Local CSMA Algorithm

- All nodes given schedule $\{\mathcal{L}(t)\}_{t\in\mathbb{N}}$
- Nodes in $\mathcal{L}(t+1)$ simply wait until all neighbors in $\mathcal{L}(t+1)$ finish. It then transmits after some **random jitter, delay**, or **back-off**.

Questions

1. If all transmissions $\approx$ equal length, do nodes in $\mathcal{L}(t)$ stay "synchronized" as $t \to \infty$?
2. And how do we define synchronization anyways?

Dynamics of Starting Times



- Analyze dynamics of **starting times** $\{S(v,t)\}_{t\geq 1}$
- **Precedence graph**
  - Layer: starting times of $\mathcal{L}(t)$
  - Edge: if adjacent slots + neighbors in contention graph
- Synchronization $\Leftrightarrow$ $\max_{u,v} |S(v,t) - S(u,t)|$ **small !**

Dynamics of Starting Times

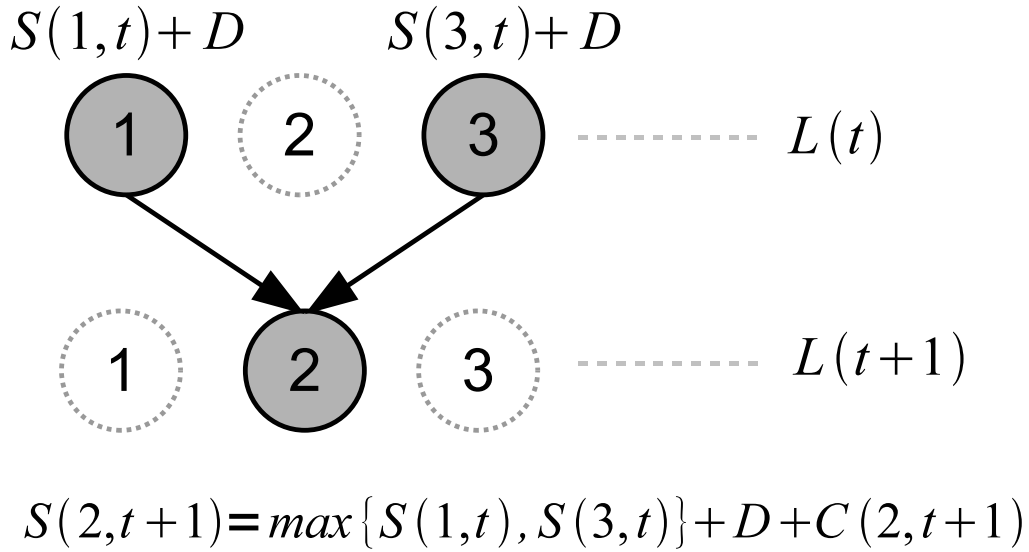
Key Property of $\{S(v,t)\}$

- **Recursive relations**:

$$S(v,t+1) = \max_{(u,v)\in\mathcal{E}_P} \{S(u,t)\} + D + C(v,t+1)$$

where $D =$ transmission time and $C(v,t) =$ jitter

- **Assumption on jitter:** $C(v,t) \leq \delta, a.s.$, and $0 < \delta << D$



Definition of Synchronized Schedule

Definition: Synchronized Schedule

$\{\mathcal{L}(t)\}$ is synchronized if there exists $T(\mathcal{G}_P)$, s.t.

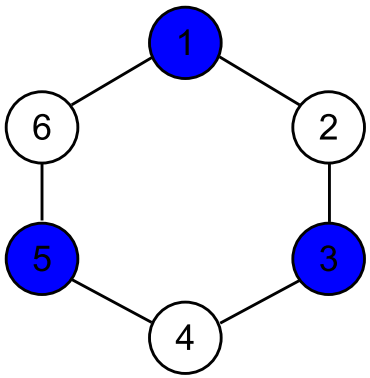
$$\max_{u,v \in \mathcal{L}(t)} |S(v,t) - S(u,t)| \leq \epsilon(jitter), a.s.$$

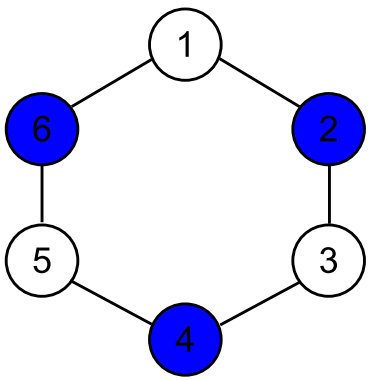
for all $t \geq T(\mathcal{G}_P)$, with

$$\lim_{jitter \to 0} \epsilon(jitter) = 0$$

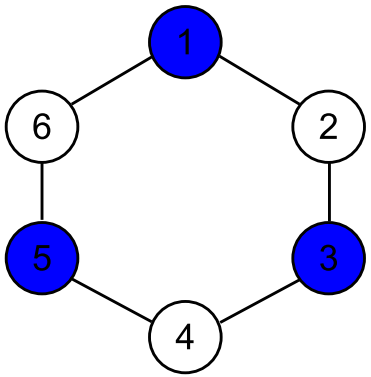
In short: bounded jitter $\rightarrow$ **uniformly bounded differences** in starting times!

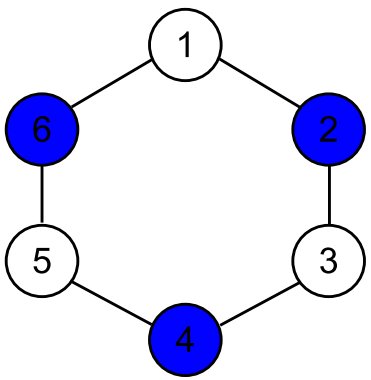
Simulations: Schedule A

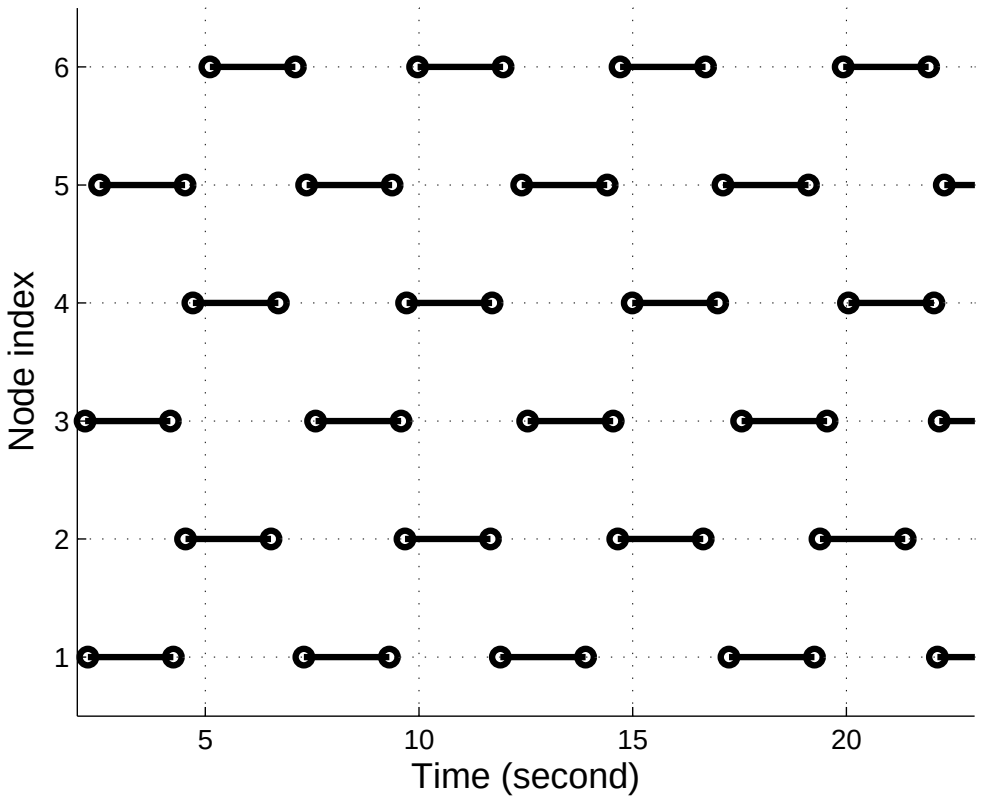




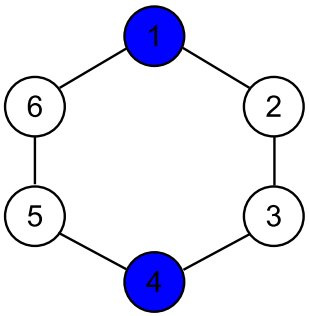
Simulations: Schedule A

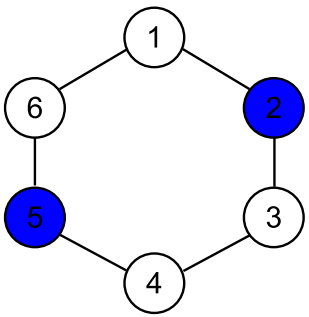


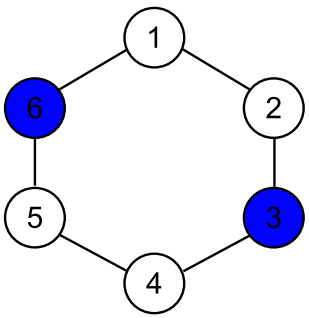




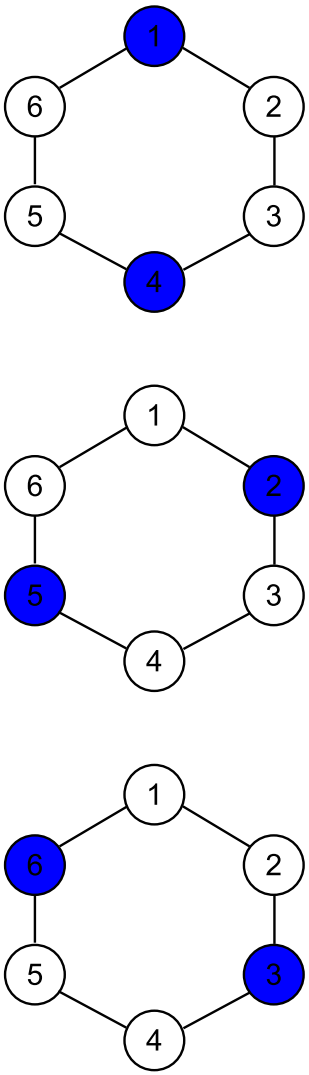
Simulations: Schedule B

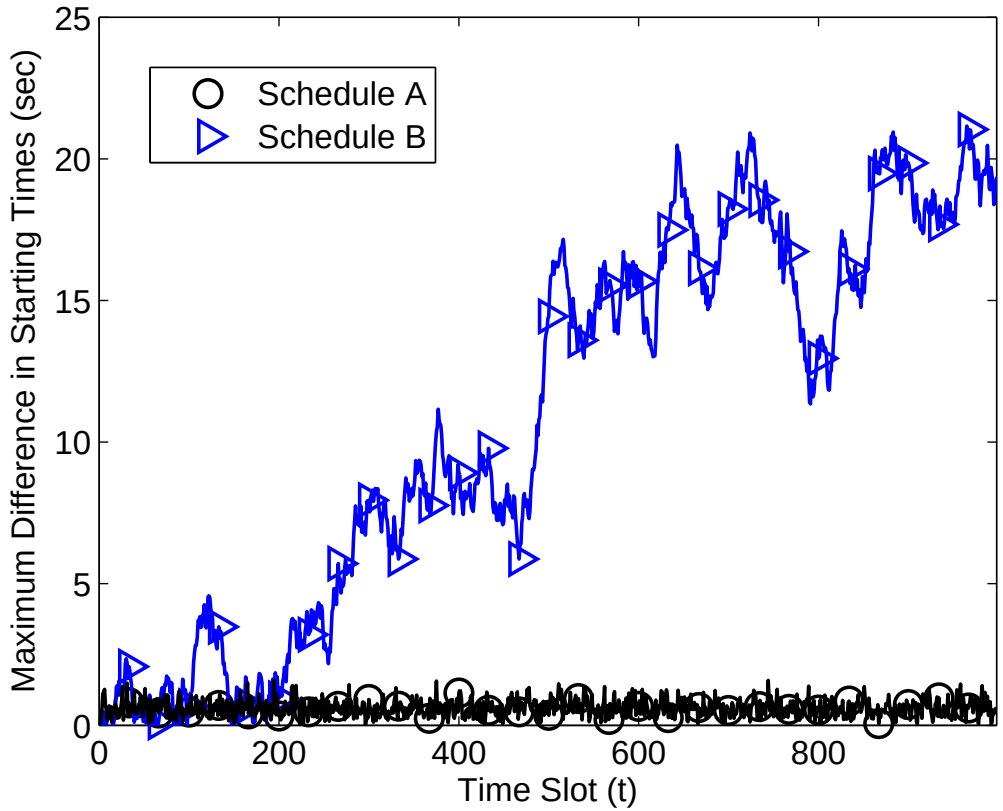






Simulations: Schedule B





So Schedule A seems to synchronize but NOT Schedule B.

Results

1. A **Necessary and sufficient** condition for synchronization
2. Existence of synchronized schedules for **general contention graphs**

Outline

1 Introduction

2 Model

3 Results
- A Necessary and Sufficient Condition for Synchronization
- Existence of Synchronized Schedules: General Networks

4 Summary & Future Work

Outline

1 Introduction

2 Model

3 Results
- A Necessary and Sufficient Condition for Synchronization
- Existence of Synchronized Schedules: General Networks

4 Summary & Future Work

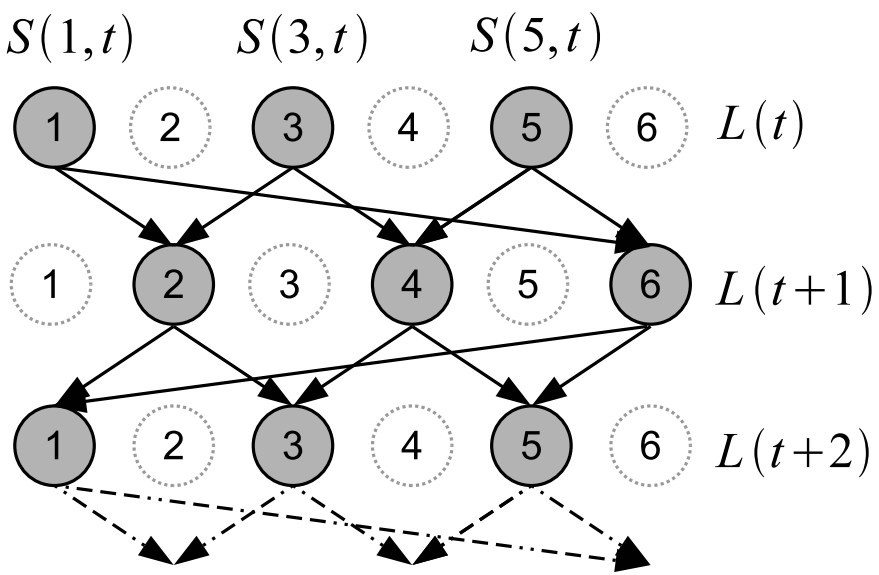
Total Connectedness

Definition

Two layers $t_1 < t_2$ of the precedence graph are **totally connected** if there exists a path from any vertex in $\mathcal{L}(t_1)$ to any vertex in $\mathcal{L}(t_2)$.

We define:

$$N(t) = \inf\{n \geq 0 : \mathcal{L}(t) \text{ is totally connected with } \mathcal{L}(t+n)\}$$



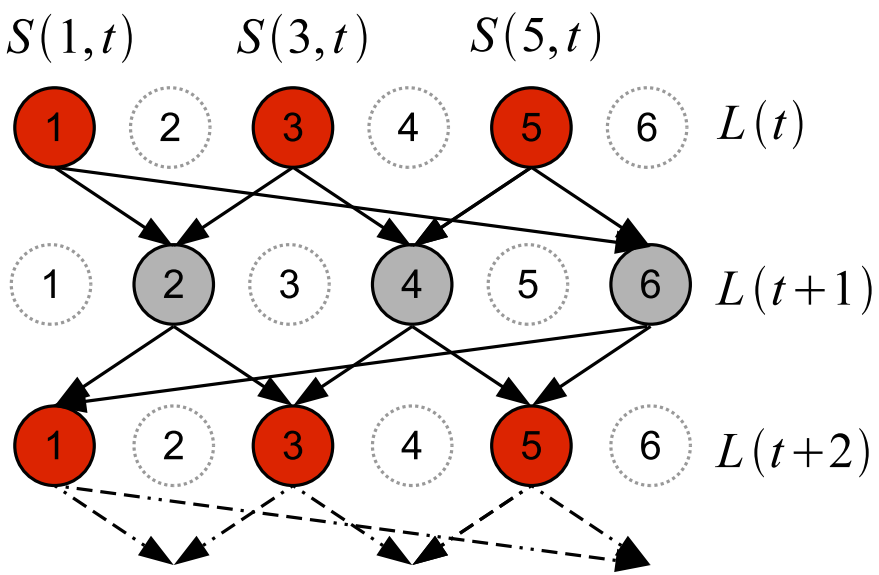
Total Connectedness

Definition

Two layers $t_1 < t_2$ of the precedence graph are **totally connected** if there exists a path from any vertex in $\mathcal{L}(t_1)$ to any vertex in $\mathcal{L}(t_2)$.

We define:

$$N(t) = \inf\{n \geq 0 : \mathcal{L}(t) \text{ is totally connected with } \mathcal{L}(t+n)\}$$



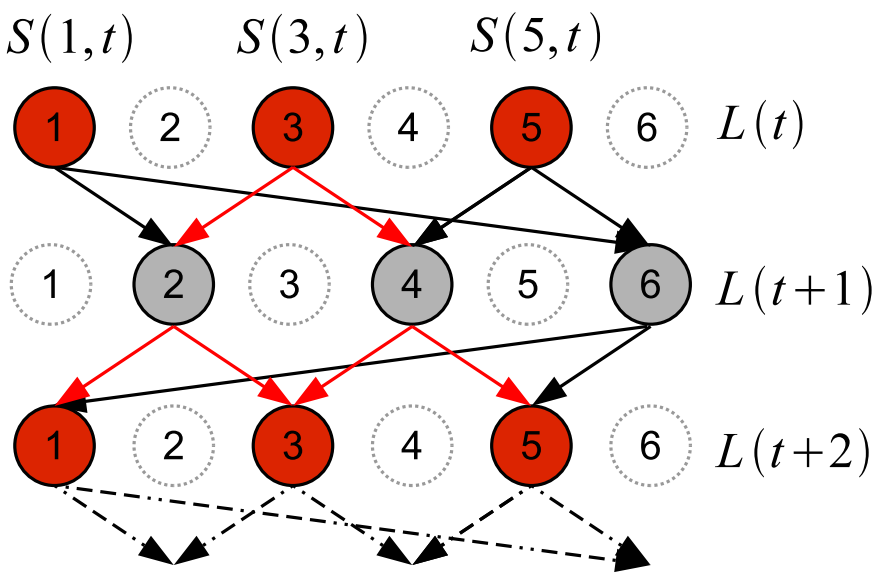
Total Connectedness

Definition

Two layers $t_1 < t_2$ of the precedence graph are **totally connected** if there exists a path from any vertex in $\mathcal{L}(t_1)$ to any vertex in $\mathcal{L}(t_2)$.

We define:

$$N(t) = \inf\{n \geq 0 : \mathcal{L}(t) \text{ is totally connected with } \mathcal{L}(t+n)\}$$



Bounded Differences in Starting Times

Theorem [Baccelli & Liu 92]

If, for some t, $N(t) < \infty$ then

$$|S(u, t + N(t)) - S(v, t + N(t))| \leq 2N(t)\delta, \ \forall u, v \in \mathcal{L}(t + N(t))$$

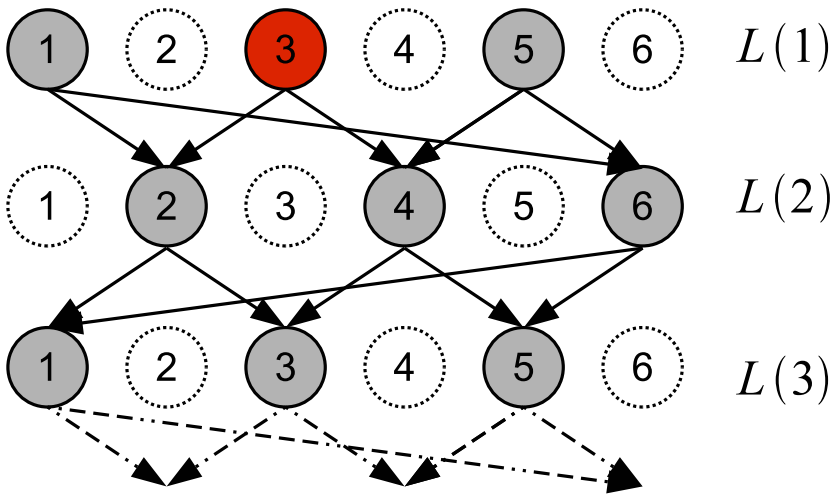
Key point: $\max_{u,v} |S(u, t + N(t)) - S(v, t + N(t))|$ bounded, **independently** from $\mathcal{L}(t)$, only function of $N(t)$!

$\Longrightarrow$ Transmissions synchronize in $\mathcal{L}(t + N(t))$.

Intuition

Suppose node 3 in $\mathcal{L}(1)$ is significantly late, it **delays all nodes** in a future layer **almost uniformly**!

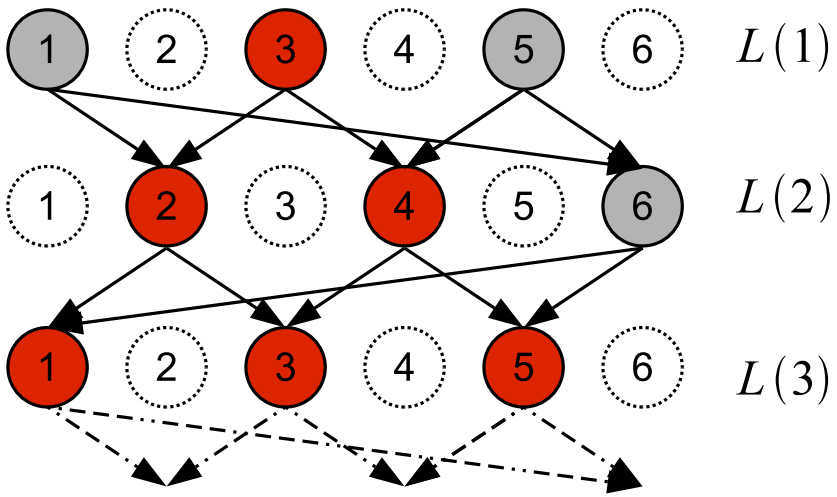
$\Rightarrow$ System "self-synchronized".



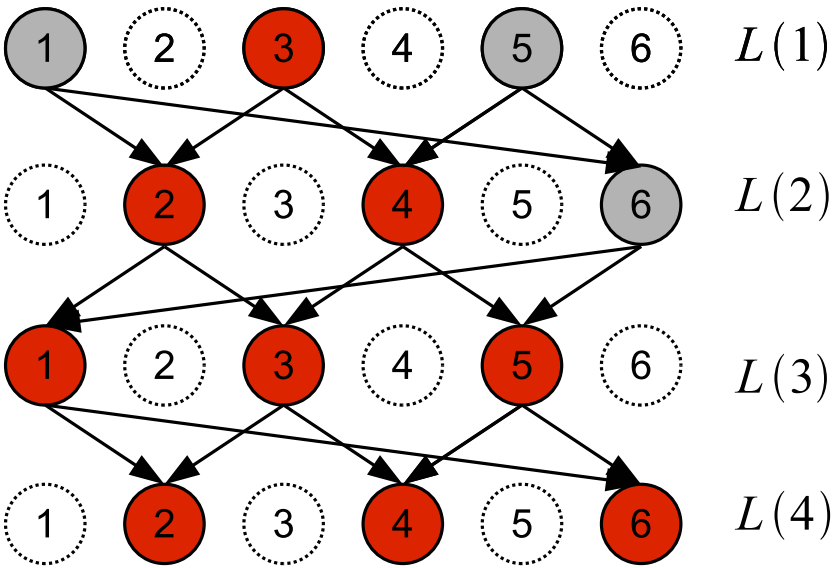
Intuition

Suppose node 3 in $\mathcal{L}(1)$ is significantly late, it **delays all nodes** in a future layer **almost uniformly**!

$\Rightarrow$ System "self-synchronized".



A Necessary Condition



Descendant set:
$d_{(v,t)}(t'), t' > t$

Property: Inherited Reachability

Let $t_1 < t_2$. If for some $v \in \mathcal{L}(t_1)$

$$d_{(v,t_1)}(t_2) = \mathcal{L}(t_2)$$

then for all $t_3 > t_2$

$$d_{(v,t_1)}(t_3) = \mathcal{L}(t_3)$$

A Necessary & Sufficient Condition

Theorem [XDT10]

A schedule $\mathcal{L}$ is synchronized **if and only if** for **some** t and **all** $v \in \mathcal{L}(t)$

$$\inf\{n \geq 0; d_{(v,t)}(t+n) = \mathcal{L}(t+n)\} < \infty$$

Moreover, the condition can be **checked in finite steps** for any finite contention graph.

A Necessary & Sufficient Condition

Proof:

- Sufficiency: Show equivalent to having a finite $N(t)$ due to periodicity of $\mathcal{L}$. So the previous theorem applies.
- Necessity: Suppose condition not met. Show the precedence graph can be **partitioned** into at least two parts that do not impact each other. Hence there exists finite jitter distribution that can cause arbitrarily large differences in starting times.

A Necessary & Sufficient Condition

How long to check the condition?

Definition

$A \subset \mathcal{L}(t)$ is called an **absorbing subset** if for some j

$$\cup_{v \in A} d_{(v,t)}(t + jK) = A$$

Proposition

For any $v \in \mathcal{L}(t)$, there is a **finite** j for which either

- $d_{(v,t)}(t + jK)$ is an absorbing subset.
- $d_{(v,t)}(t + jK) = \mathcal{L}(t + jK)$.

Proof: Simple Pigeonhole Principle: $\mathcal{V}_I$ has finite number of subsets.

A Necessary & Sufficient Condition

Proposition (same as before)

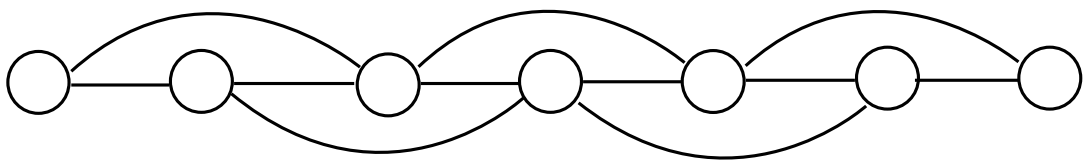
For any $v \in \mathcal{L}(t)$, there is a **finite** j for which either

- $d_{(v,t)}(t + jK)$ is an absorbing subset.
- $d_{(v,t)}(t + jK) = \mathcal{L}(t + jK)$.

Proof of necessary & sufficient condition (continued):
If first case, declare schedule to be not synchronized; if second case, declare synchronized. Q.E.D.

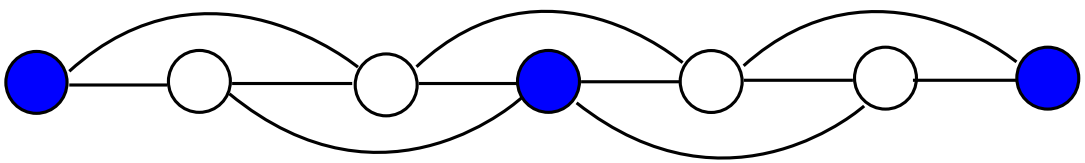
Examples: Linear Multihop Network with 2-hop Interference

- A linear relay topology where each node interferes with its 2-hop neighbors.



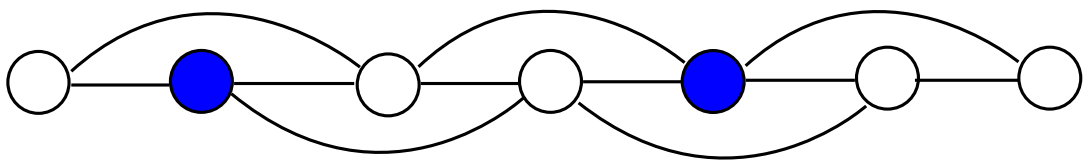
Examples: Linear Multihop Network with 2-hop Interference

- A linear relay topology where each node interferes with its 2-hop neighbors.



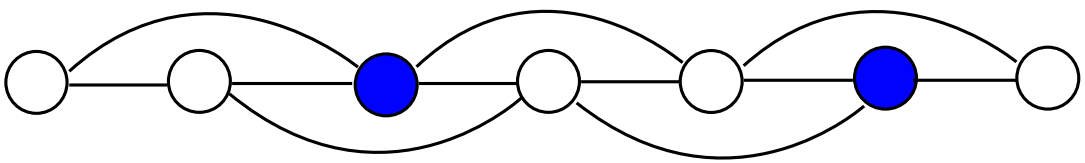
Examples: Linear Multihop Network with 2-hop Interference

- A linear relay topology where each node interferes with its 2-hop neighbors.



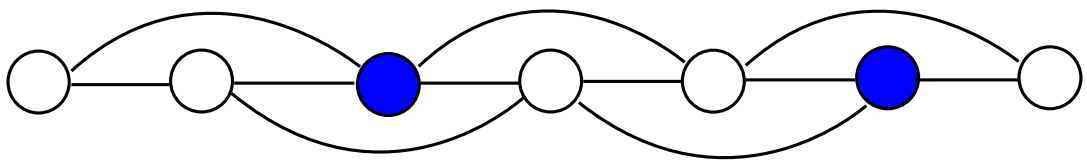
Examples: Linear Multihop Network with 2-hop Interference

- A linear relay topology where each node interferes with its 2-hop neighbors.



Examples: Linear Multihop Network with 2-hop Interference

- A linear relay topology where each node interferes with its 2-hop neighbors.

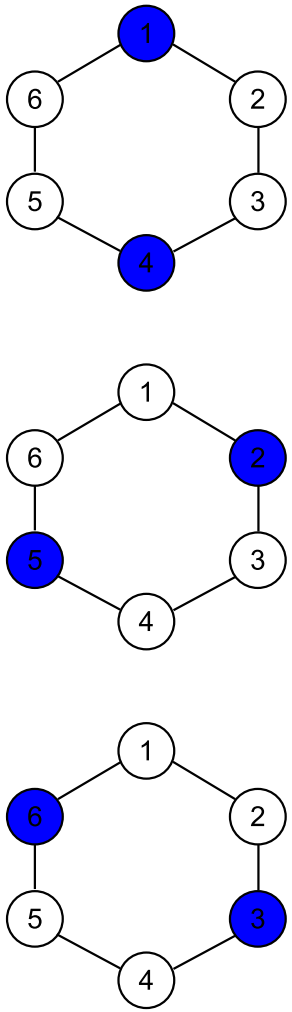


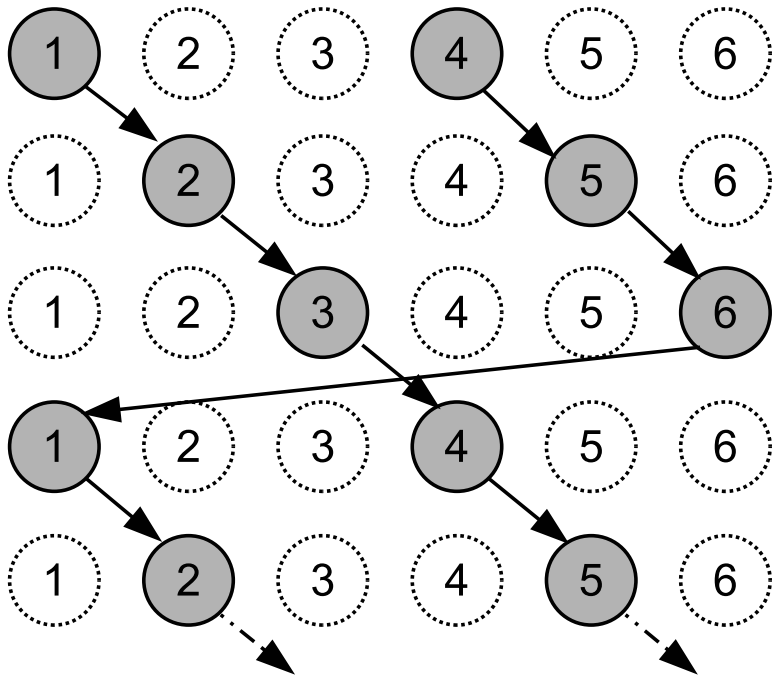
- One can check this schedule is **synchronized**!

Examples: Bad MIS Schedule

Condition non-trivial: **not all** MIS schedules are synchronized!

- Absorbing subsets in B: **every** singleton set.

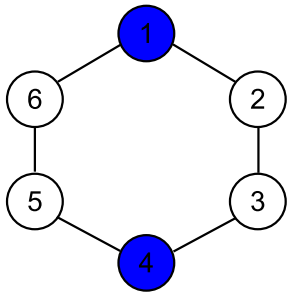


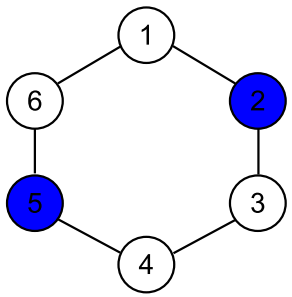


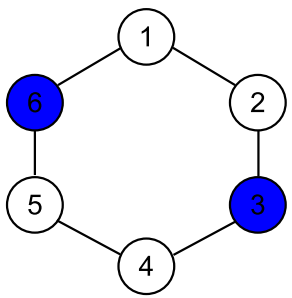
Examples: Bad MIS Schedule

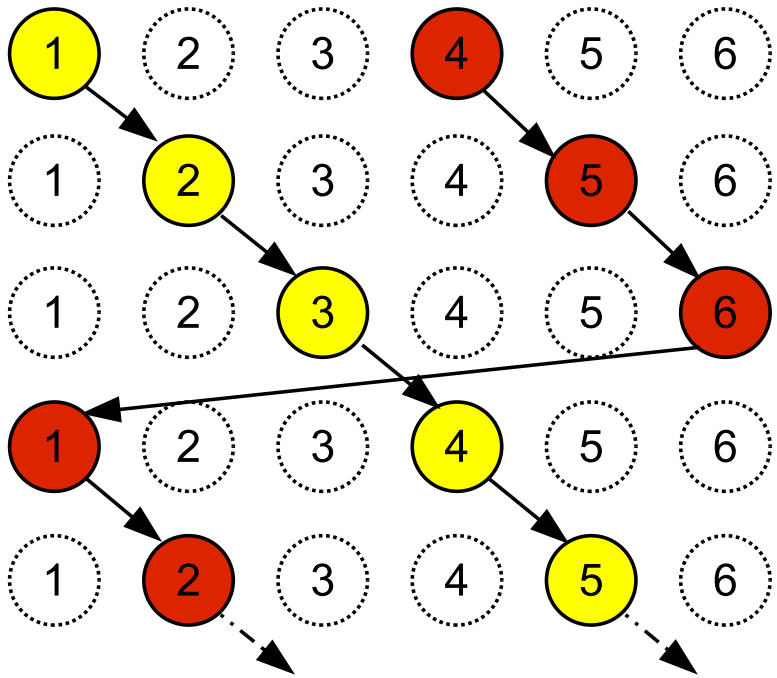
Condition non-trivial: **not all** MIS schedules are synchronized!

- Absorbing subsets in B: **every** singleton set.









Outline

1 Introduction

2 Model

3 Results
- A Necessary and Sufficient Condition for Synchronization
- Existence of Synchronized Schedules: General Networks

4 Summary & Future Work

Existence of Synch. Schedules: General Networks

Theorem [XDT10]

There exists at least one synchronized schedule for **any** connected contention graph.

Remark: Proof gives an algorithm to **find one**.

Constructive Proof of Existence

Constructing a schedule:

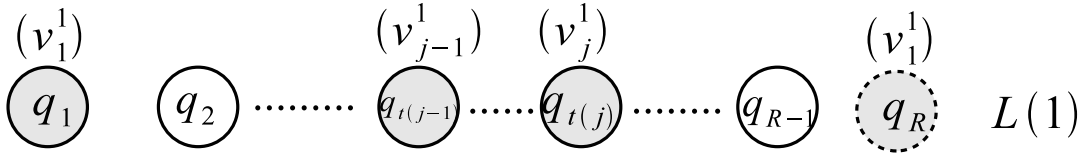
- Take any **covering closed walk** $Q = (q_1, q_2, ..., q_R)$ on contention graph $\mathcal{G}_I$.
- $\mathcal{L}(1)$ is any max. indep. set containing q_1.
- $\mathcal{L}(i)$ is any max. indep. set containing q_i and elements of $\mathcal{L}(1)$ that are not neighbors of q_i
- Reverse first part to construct second part:

$$\mathcal{L}(i) = \mathcal{L}(2R - i) \text{ for } i > R$$

Claim

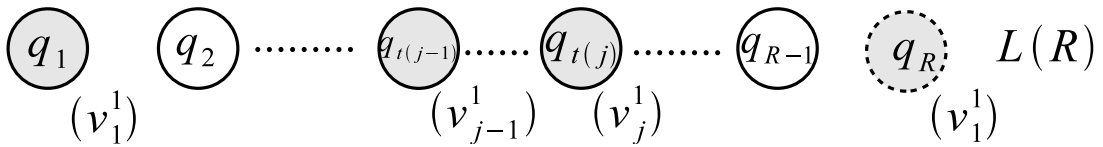
$\mathcal{L}(1)$ totally connected to $\mathcal{L}(2R - 1)$. Hence $\mathcal{L}$ is synchronized.

Proof of Existence

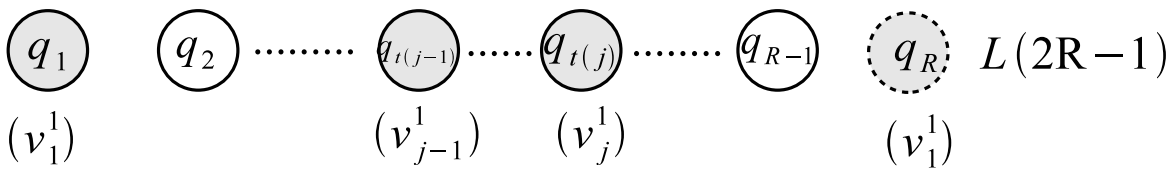


Claim

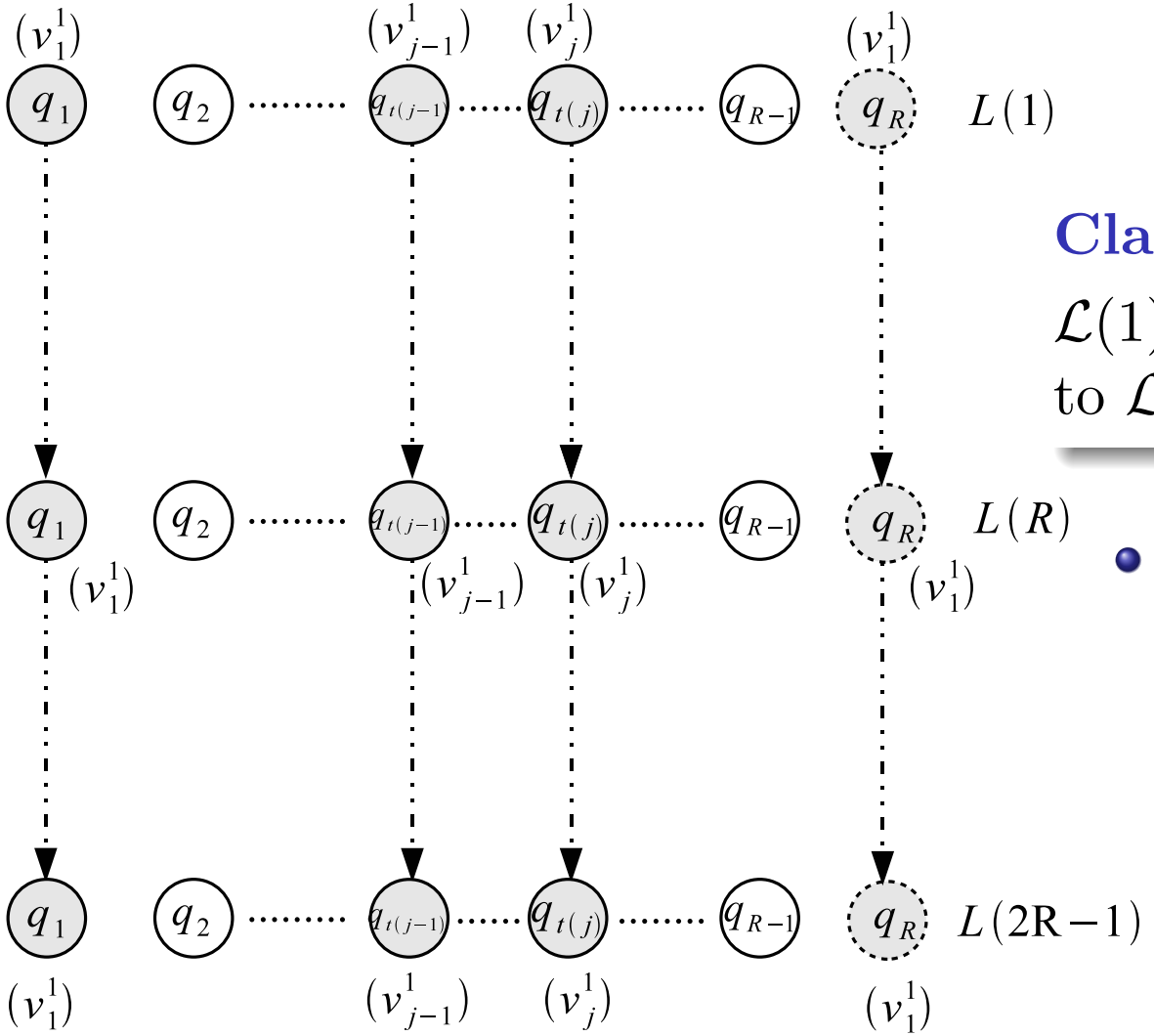
$\mathcal{L}(1)$ totally connected to $\mathcal{L}(2R-1)$



- Because **All** of $\mathcal{L}(1)$ connect with $q_R \in \mathcal{L}(R)$, which is connected to **all** of $\mathcal{L}(2R-1)$.



Proof of Existence

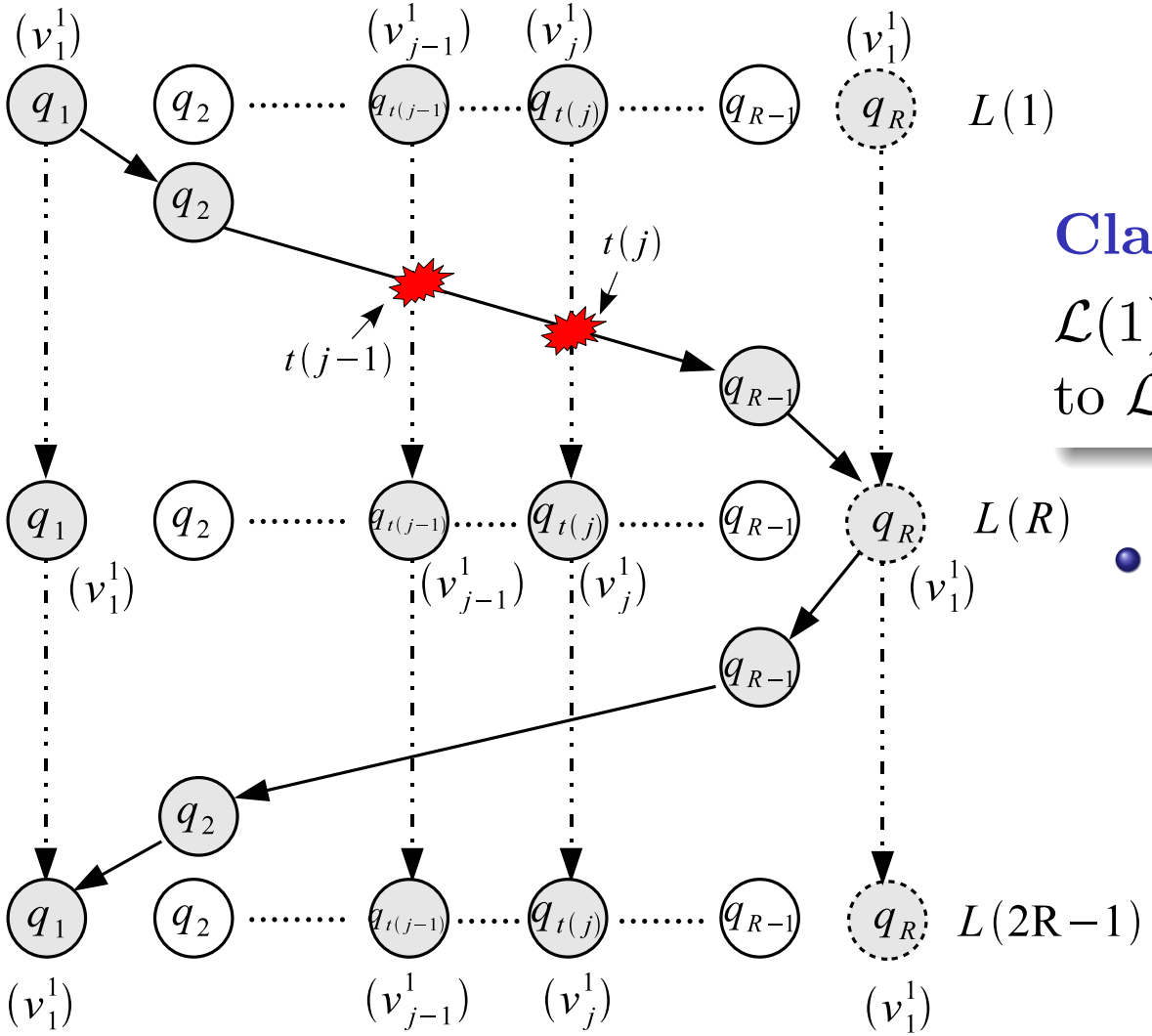


Claim

$\mathcal{L}(1)$ totally connected to $\mathcal{L}(2R-1)$

- Because **All** of $\mathcal{L}(1)$ connect with $q_R \in \mathcal{L}(R)$, which is connected to **all** of $\mathcal{L}(2R-1)$.

Proof of Existence

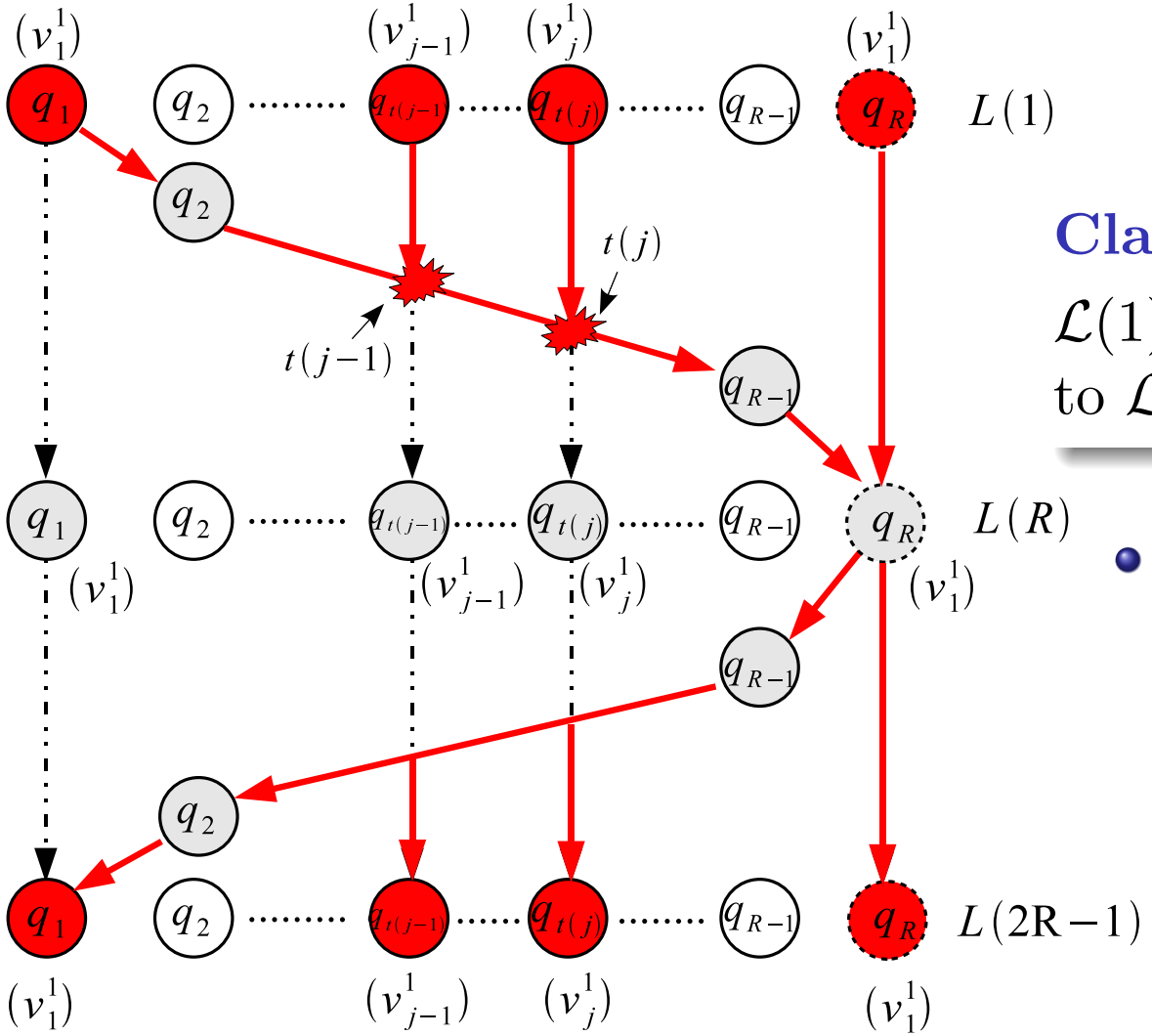


Claim

$\mathcal{L}(1)$ totally connected to $\mathcal{L}(2R-1)$

- Because **All** of $\mathcal{L}(1)$ connect with $q_R \in \mathcal{L}(R)$, which is connected to **all** of $\mathcal{L}(2R-1)$.

Proof of Existence



Claim

$\mathcal{L}(1)$ totally connected to $\mathcal{L}(2R-1)$

- Because **All** of $\mathcal{L}(1)$ connect with $q_R \in \mathcal{L}(R)$, which is connected to **all** of $\mathcal{L}(2R-1)$.

Outline

1. Introduction
2. Model
3. Results
   - A Necessary and Sufficient Condition for Synchronization
   - Existence of Synchronized Schedules: General Networks
4. Summary & Future Work

Conclusions

Summary

- Concept of **self-synchronized TDMA**.
- Necessary and sufficient conditions for synchronized schedule.
- Can find a schedule for **any** contention graph.

Future Work

- Dynamic nodes setting.
- Find synch. schedule with **throughput guarantees** (initial results).

Conclusions

Summary

- Concept of **self-synchronized TDMA**.
- Necessary and sufficient conditions for synchronized schedule.
- Can find a schedule for **any** contention graph.

Future Work

- Dynamic nodes setting.
- Find synch. schedule with **throughput guarantees** (initial results).

Thank you!

Throughput Guarantees: Universal Schedule Embedding

- A favorite MIS schedule with throughput guarantees but **not synchronized**?
- A solution: **Embed** to form synchronized schedule with **arbitrarily close** throughput guarantees.

Corollary: Universal Schedule Embedding

Given **any** MIS schedule $\mathcal{L}$, there exists a synchronized schedule $\tilde{\mathcal{L}}$ s.t.

$$\tilde{\mathcal{L}} \supseteq \mathcal{L}$$

and

$$|\tilde{\mathcal{L}}| \leq |\mathcal{L}| + |\mathcal{L}^S|,$$

where $\mathcal{L}^S$ is **any synchronized schedule** on the contention graph.