# *RT-BLE:*
# Real-time Multi-Connection Scheduling for Bluetooth Low Energy

Yeming Li, Jiamei Lv, Borui Li, Wei Dong

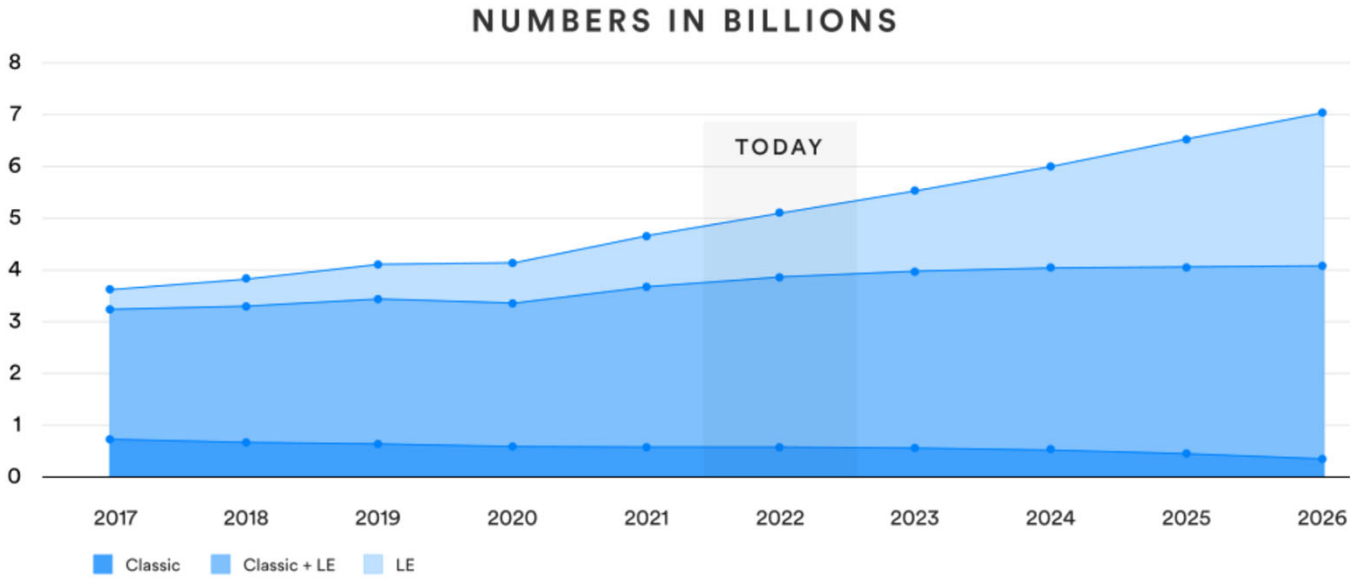
College of Computer Science, Zhejiang University

Alibaba-Zhejiang University Joint Institute of Frontier Technologies


*Presented by* ***Yeming Li***

# BLE is popular to build IoT applications!

## Bluetooth Device Shipment



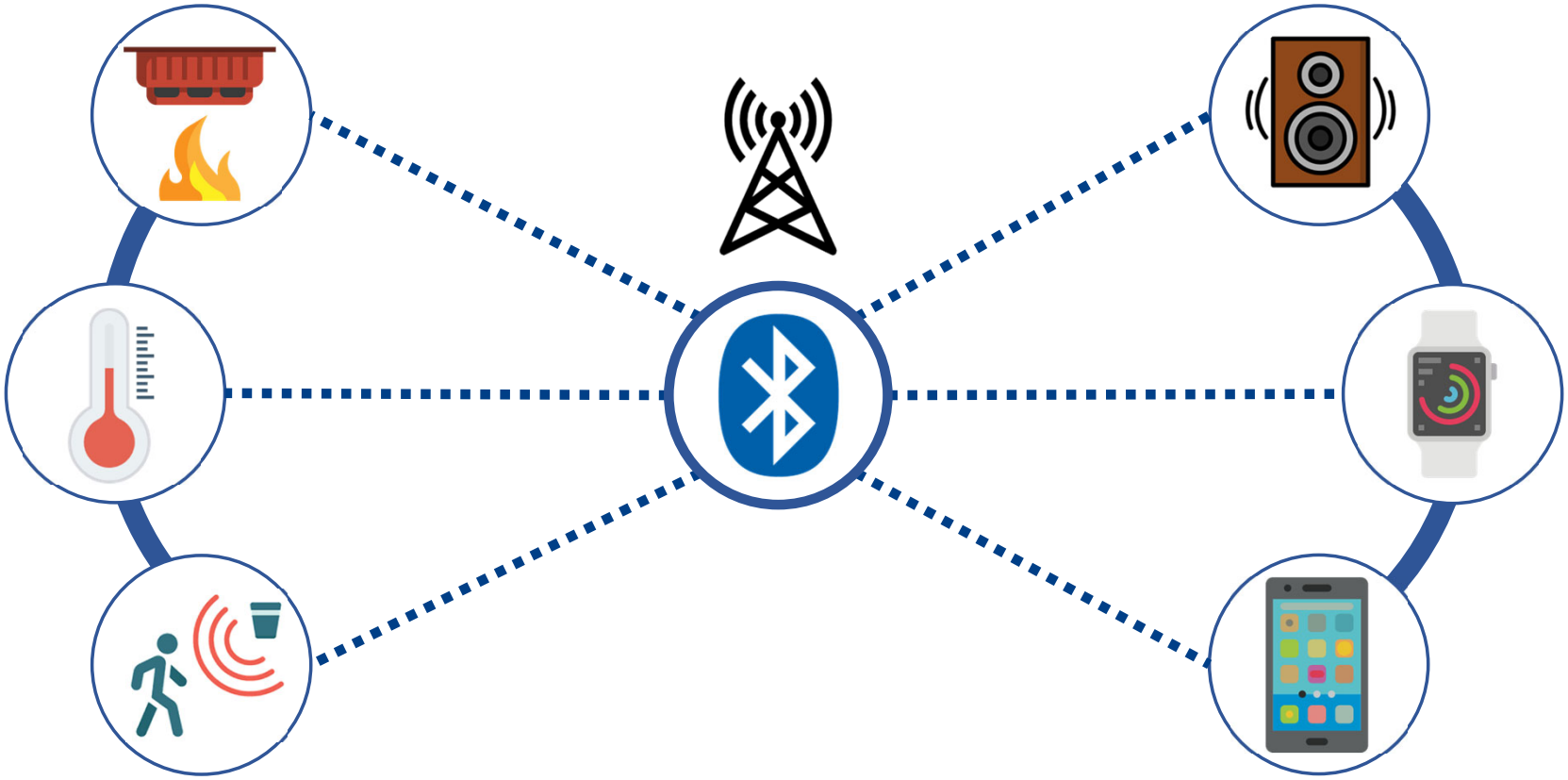

Source: SIG 2022 market update. https://www.bluetooth.com/2022-market-update/

**3x** GROWTH in BLE single-mode devices between 2022 and 2026

**95%** UP TO of Bluetooth devices support BLE mode

**100%** UP TO of key new platform devices support BLE

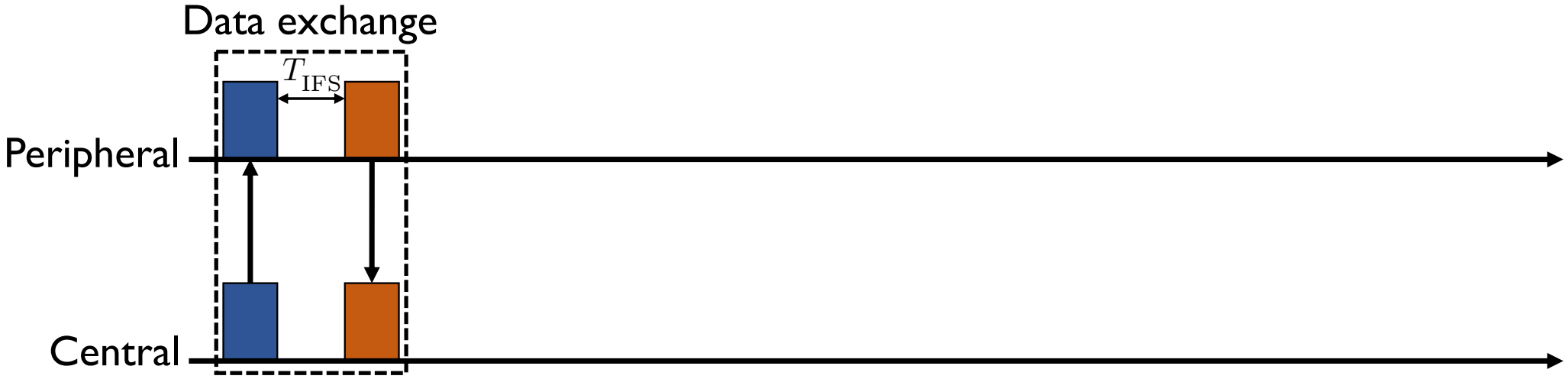
# BLE Multi-connection



The packet losses and severe ***collisions*** among multiple connections make it ***impossible*** to support real-time applications

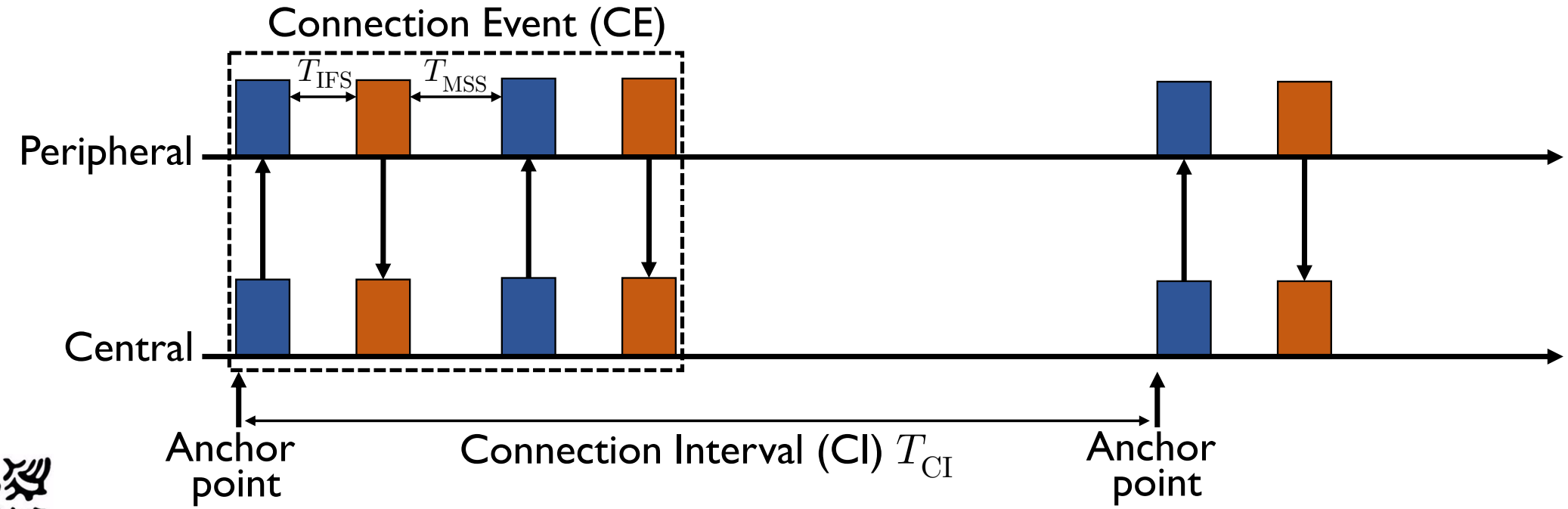
# Background: BLE's Link Layer

- **Two roles:** *Central* and *Peripheral*
- **Scheduling:** TDMA based polling scheme



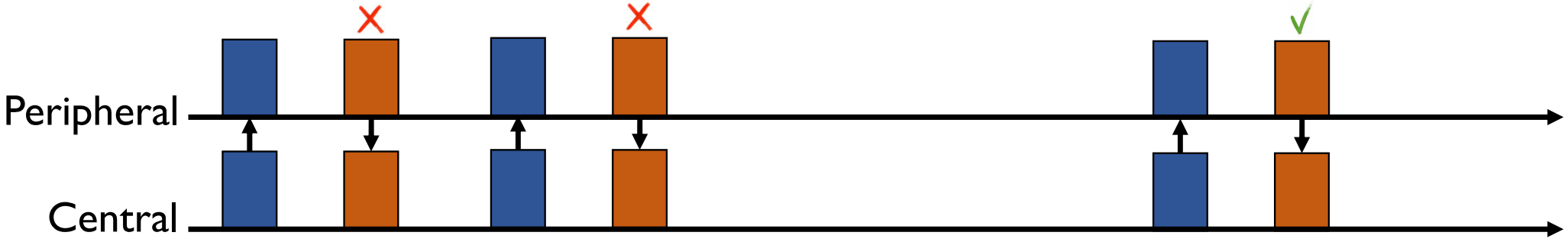
# Background: BLE's Link Layer

- **Two roles:** *Central* and *Peripheral*
- **Scheduling:** TDMA based polling scheme

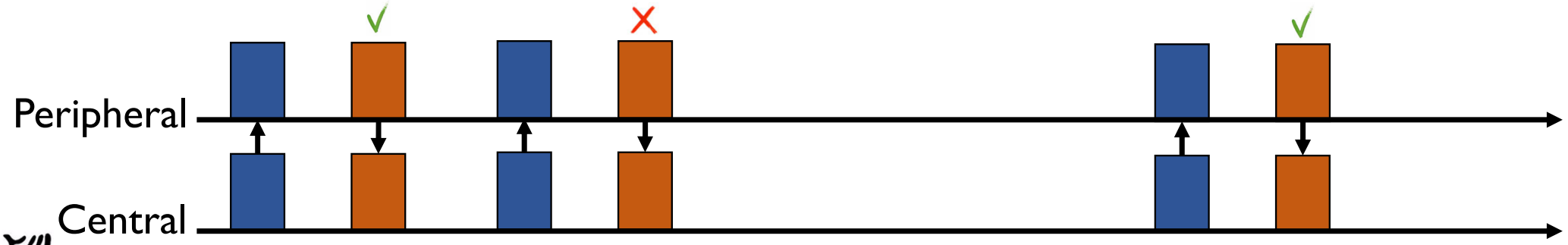


# Background: Retransmission

- **Two consecutive packet losses in one CE**

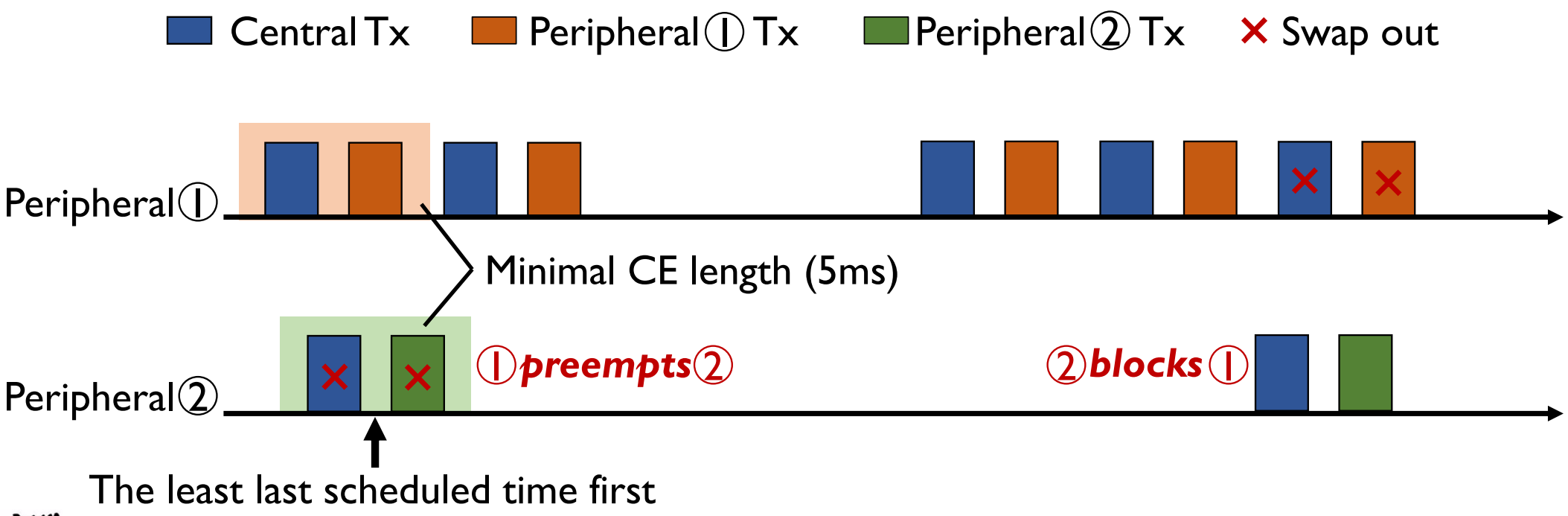



- **Packet loss of the last packet**



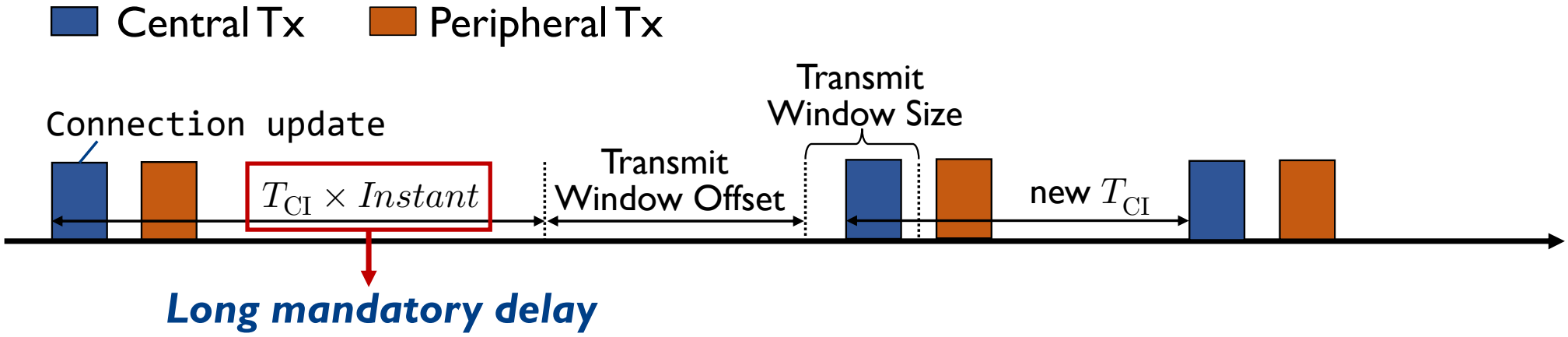
# Background: Connection Collisions

- **Two types:** *preempt* and *block*



# Connection Rescheduling

- **Connection rescheduling:**
  - Changing the *anchor point position* and *connection interval* (CI)



# Challenges

- **Inaccurate transmission model in noisy RF environment**
  - [EWSN 19] TL-BLE is based on RTT measurement
  - Without considering the underlying retransmission mechanisms
- **Inefficient time resource management**
  - [IPSN 21] BLEX uses a greedy policy
  - The system capacity is low
- **Slow connection rescheduling**
  - Bluetooth specification provides "`Connection update`" procedure
  - Suffer from long mandatory delay (i.e., $\geq 6T_{\mathrm{CI}}$)

# Contributions

- Accurate BLE real-time transmission model
- Collision tree-based time resource management technology
- Subrating-based fast connection re-scheduling method

# Contributions

- *Accurate BLE real-time transmission model*
- **Collision tree-based time resource management technology**
- **Subrating-based fast connection re-scheduling method**

# BLE Timeliness Modeling: Packet Loss

- The worst-case latency is unbounded!
- Users set the ***percentile worst-case latency*** $(p, t_{\mathrm{ddl}})$

Percentage

Worst case latency

- How many retransmission $n_{\mathrm{re}}$ are required to cover $P$ of the data transmissions:

Packet loss rate

Combination calculator

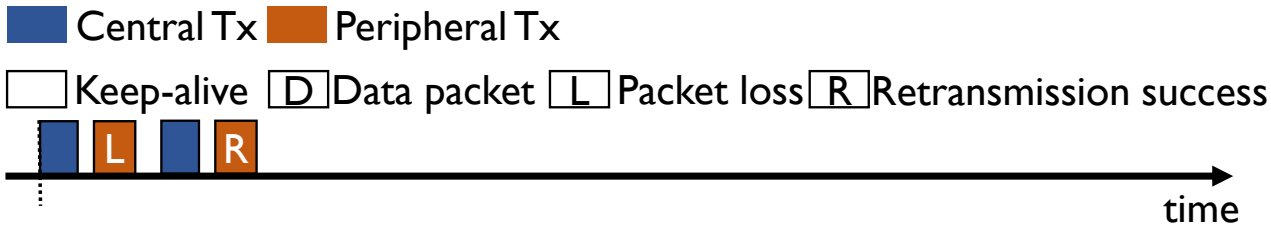
$$p_{re}(n_{\mathrm{re}}) = (1-P)^n \sum_{i=0}^{n_{\mathrm{re}}} P^i \binom{n+i-1}{i} \geq p$$

$$c_{i+1} = P\frac{n+(i+1)-1}{i+1}c_i$$

# Worst-case latency

- (1) single packet loss:

no extra CE



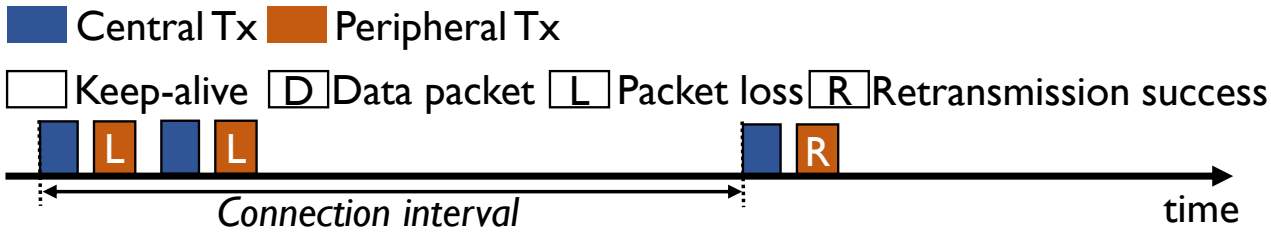
# Worst-case latency

- (1) single packet loss:

no extra CE

- (2) two consecutive packet losses:

$$n_{\mathrm{eCE}} \leq \lfloor n_{\mathrm{rec}}/2 \rfloor + \lfloor n_{\mathrm{rep}}/2 \rfloor$$



# Worst-case latency

- (1) single packet loss:

no extra CE

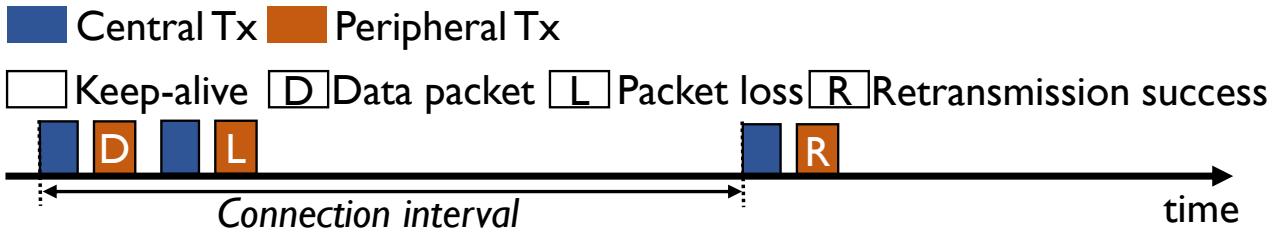
- (2) two consecutive packet losses:

$$n_{\mathrm{eCE}} \leq \lfloor n_{\mathrm{rec}}/2 \rfloor + \lfloor n_{\mathrm{rep}}/2 \rfloor$$

- (3) all packet losses at the last packet:

$$n_{\mathrm{eCE}} = \max(n_{\mathrm{rec}}, n_{\mathrm{rep}})$$

→ ***Worst-case***



# Contributions

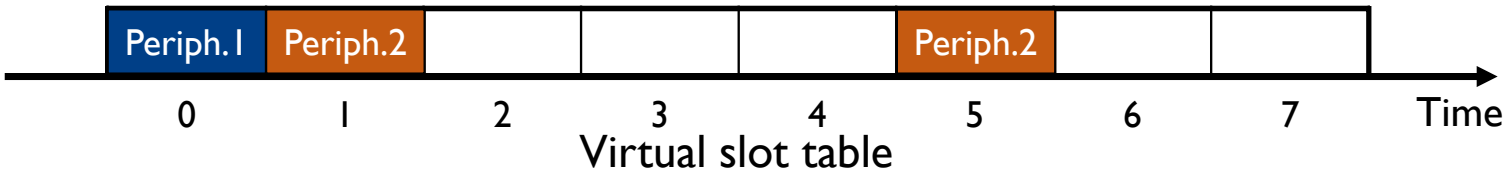
- **Accurate BLE real-time transmission model**
- ***Collision tree-based time resource management technology***
- **Subrating-based fast connection re-scheduling method**

# Basic Concept of *RT-BLE*

- **1.Time resource slicing**
  - $n \times 1.25ms$ →Connection interval, transmit window offset/size…
  - Divide the time resource into ***virtual slots*** (5ms)

- **2.Virtual slot table occupancy**
  - Virtu
  - $T_{\mathrm{CI}}$

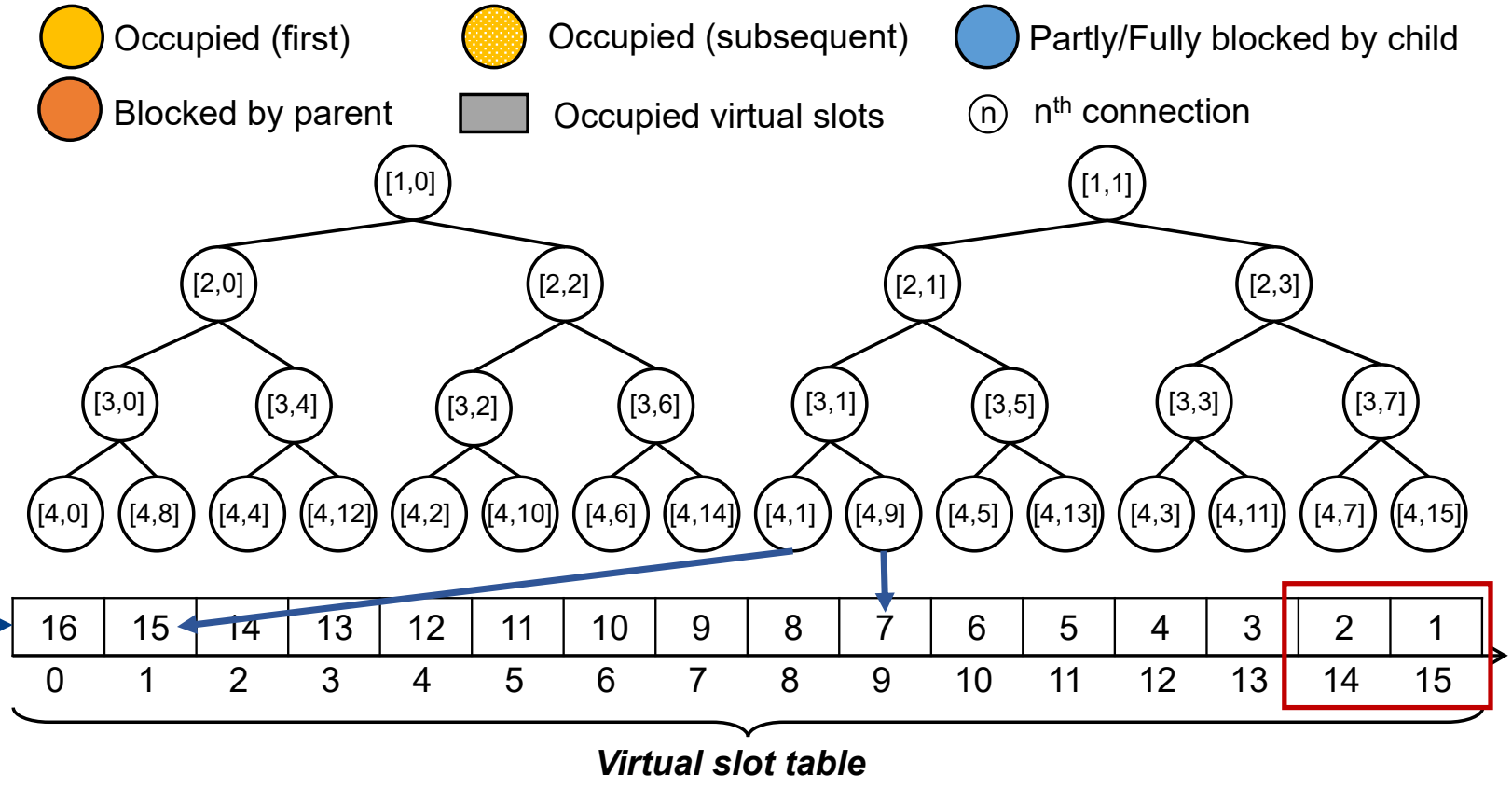
[IPSN'21] BLEX consider the time is ***continuous*** and can ***only change the CE length***



# Collision-tree based Resource Management

$[lv, offset]$

$\mathrm{T_{CI}} = 2^{lv} \times 5ms$

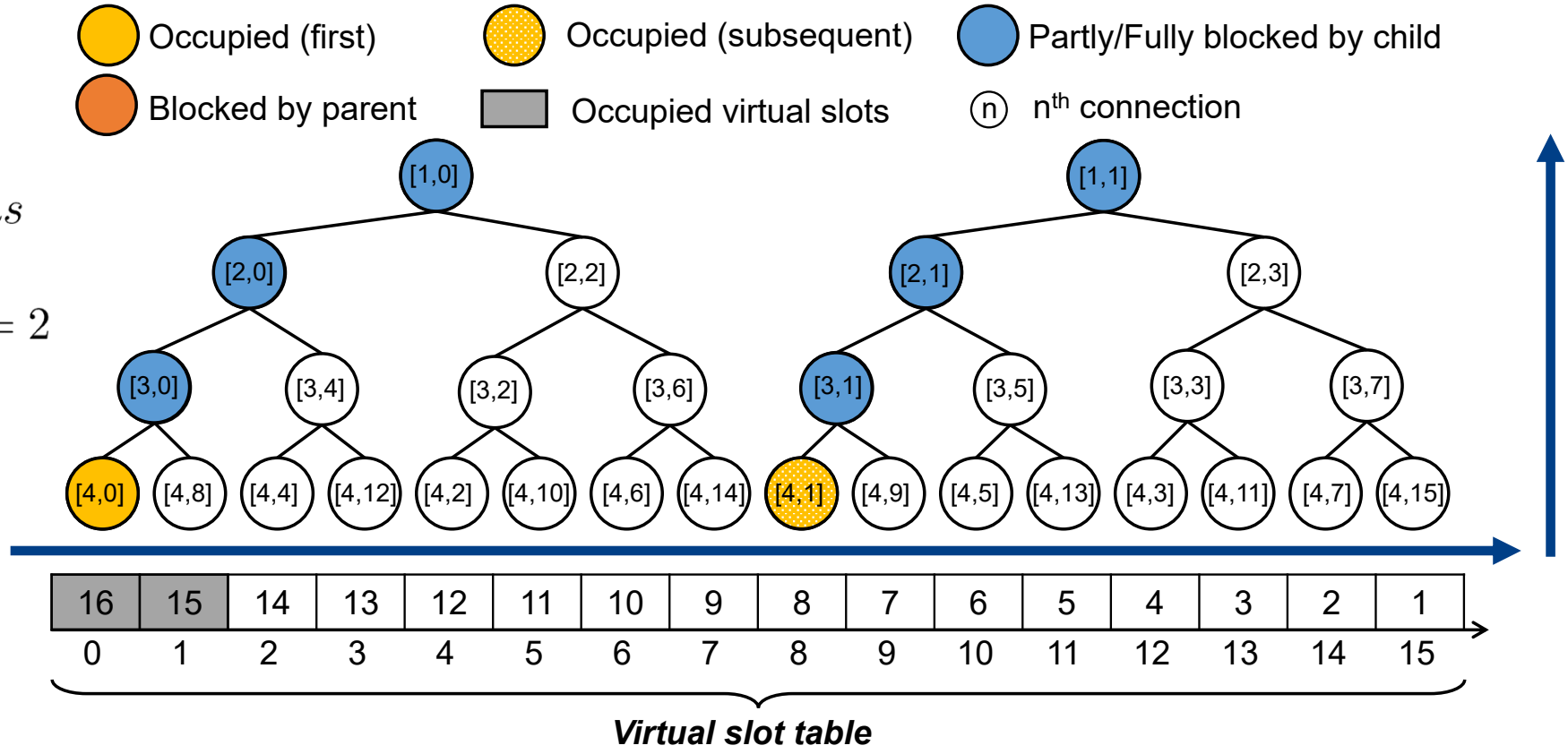


# Collision-tree based Resource Management

$[lv, offset]$

$\mathrm{T_{CI}} = 2^{lv} \times 5ms$

① $lv_{ini} = 4, s_{ini} = 2$



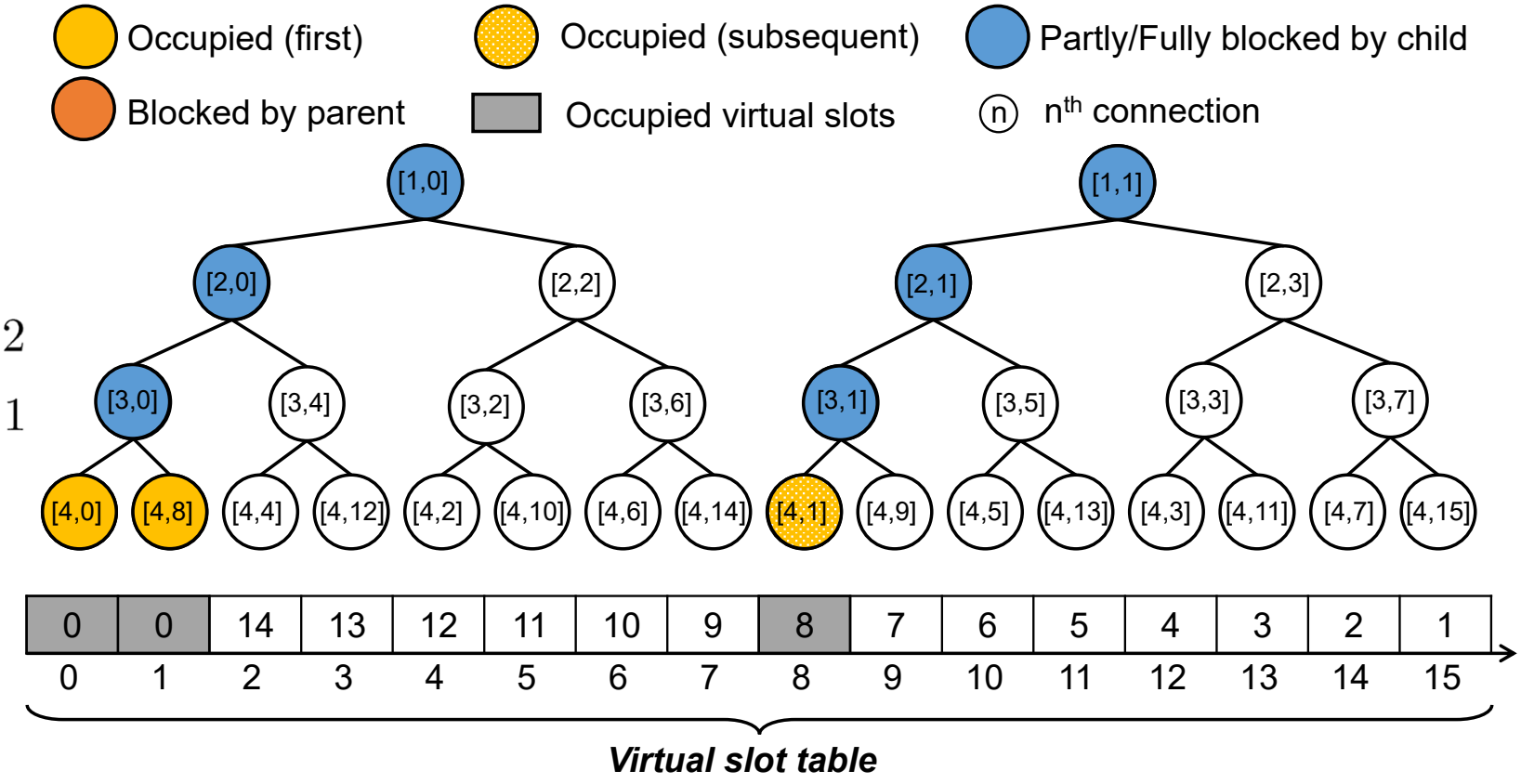
# Collision-tree based Resource Management

$[lv, offset]$

$\mathrm{T_{CI}} = 2^{lv} \times 5ms$

① $lv_{ini} = 4, s_{ini} = 2$

② $lv_{ini} = 4, s_{ini} = 1$



# Collision-tree based Resource Management

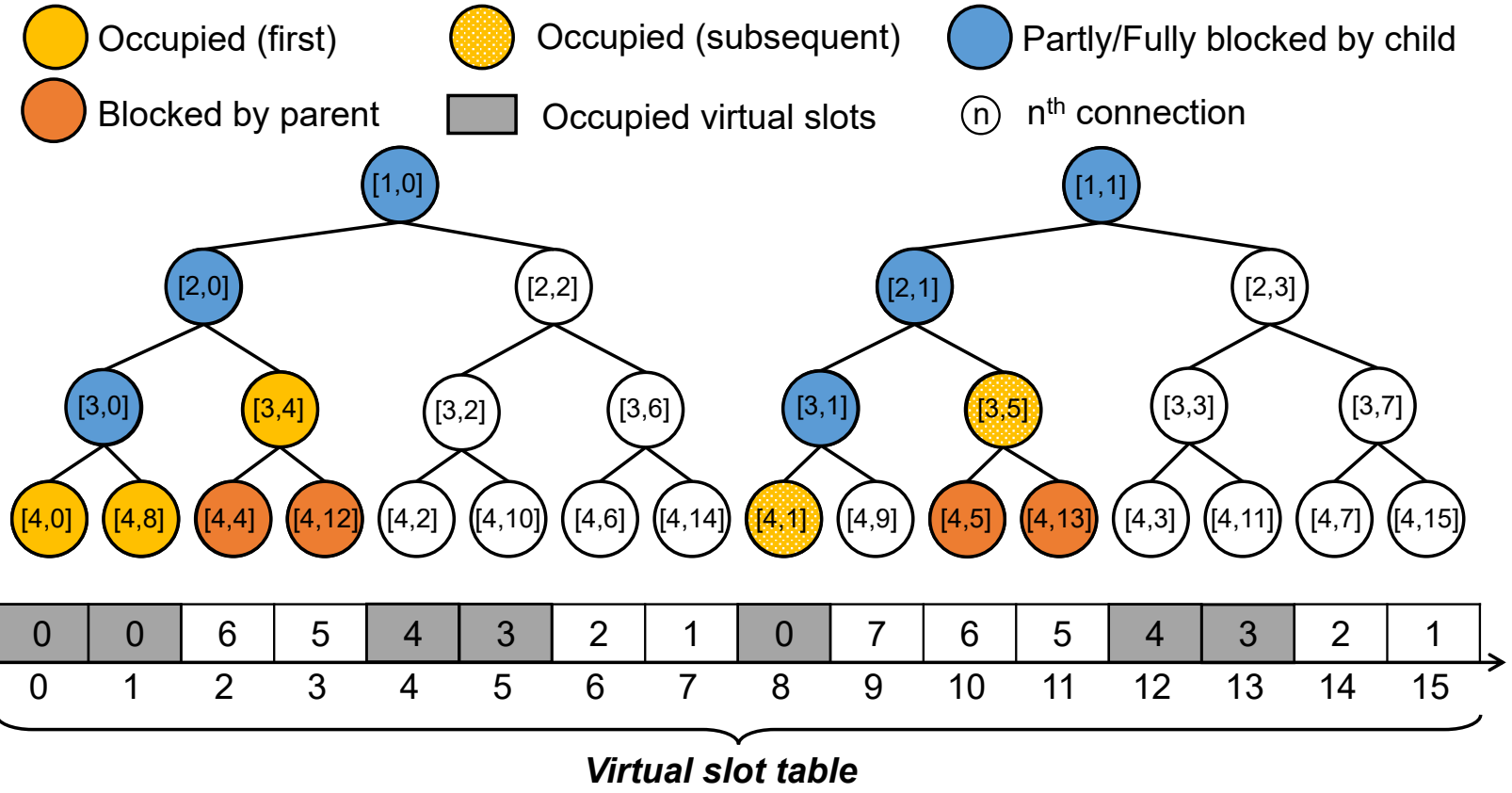
$[lv, offset]$

$\mathrm{T_{CI}} = 2^{lv} \times 5ms$

① $lv_{ini} = 4, s_{ini} = 2$

② $lv_{ini} = 4, s_{ini} = 1$

③ $lv_{ini} = 3, s_{ini} = 2$



# Collision-tree based Resource Management

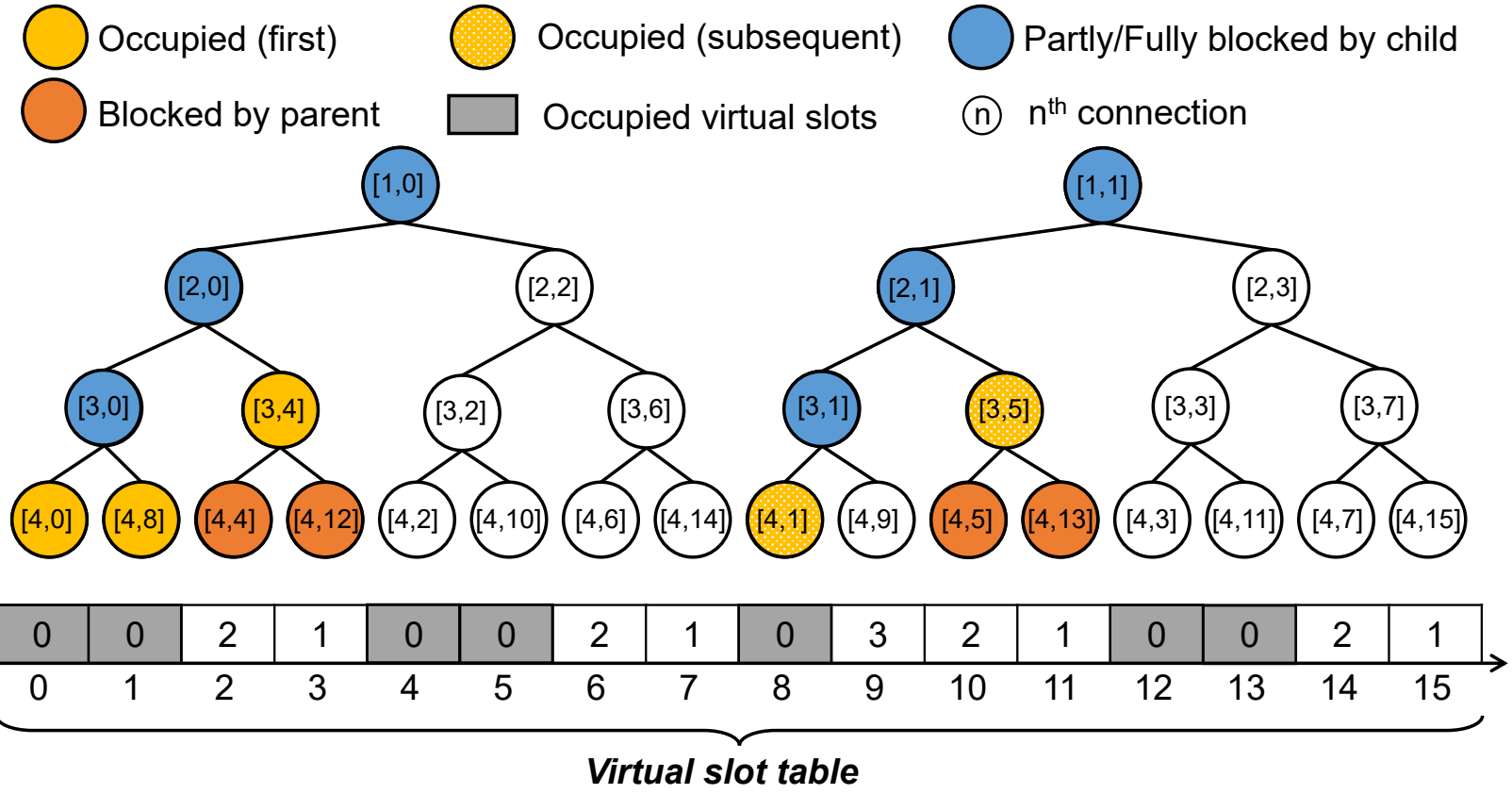
$[lv, offset]$

$\mathrm{T_{CI}} = 2^{lv} \times 5ms$

① $lv_{ini} = 4, s_{ini} = 2$

② $lv_{ini} = 4, s_{ini} = 1$

③ $lv_{ini} = 3, s_{ini} = 2$

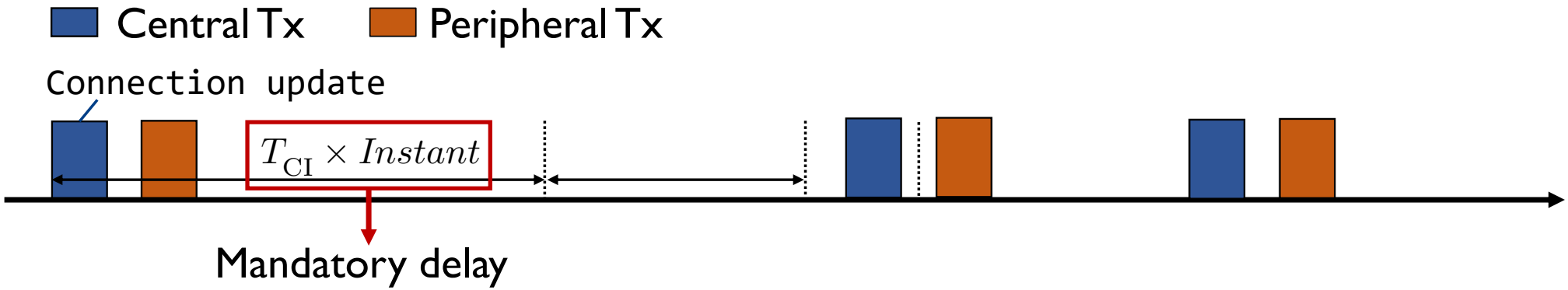


# Contributions

- **Accurate BLE real-time transmission model**
- **Collision tree-based time resource management technology**
- ***Subrating-based fast connection re-scheduling method***

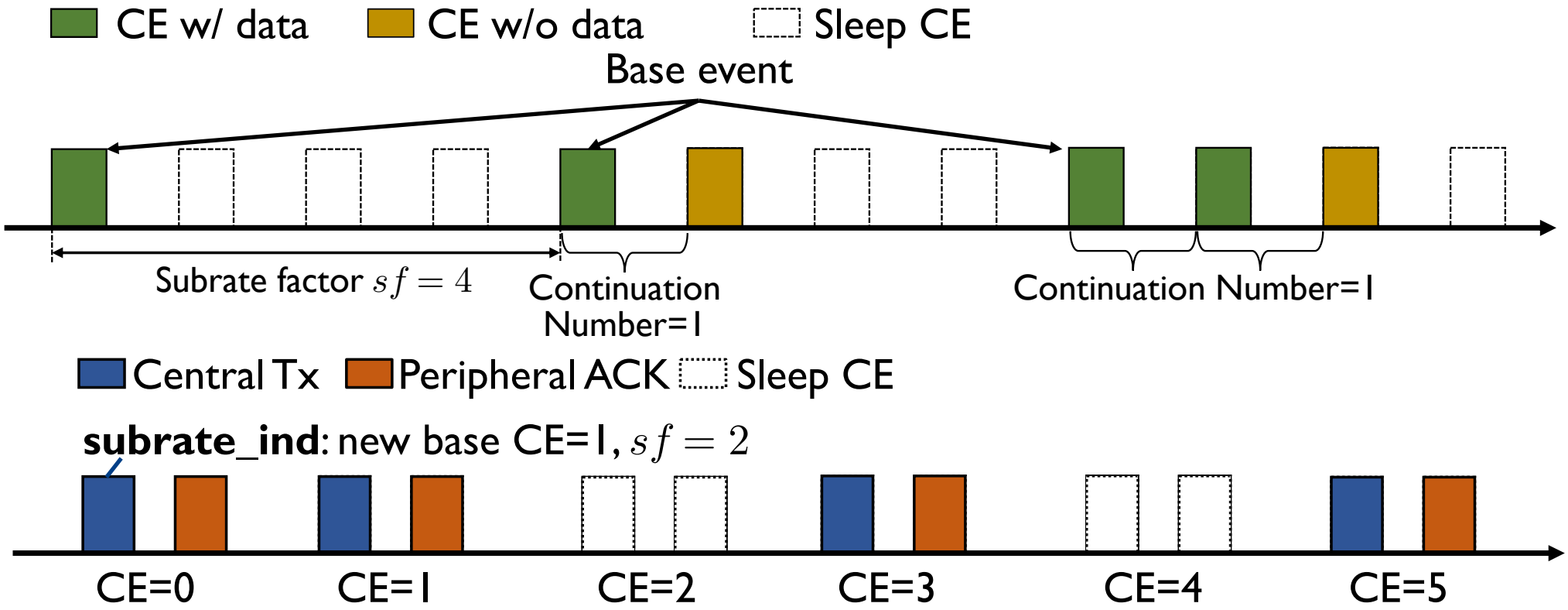
# How to Reduce the Rescheduling Delay

- **The traditional connection rescheduling suffers from long mandatory delay ($\geq 6T_{\mathrm{CI}}$)**




How to keep a small $T_{\mathrm{CI}}$ while meeting diverse application requirements?

# Connection Subrating




Native $T_{\mathrm{CI}} = 10ms$

The equivalent CI $T_{\mathrm{eCI}} = sf \times T_{\mathrm{CI}}$

# Look back: Impact to the Model

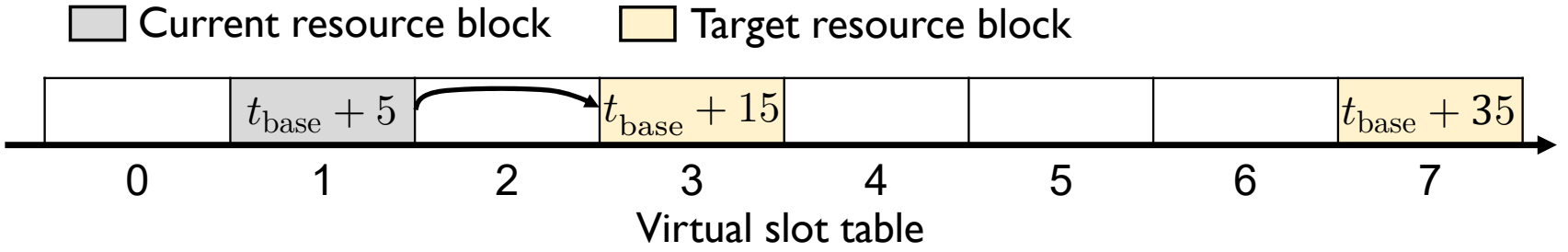
- **1. continuation number = 0**

$$n_{\mathrm{eCE}}^{(1)} = sf \times \max(n_{\mathrm{rec}}, n_{\mathrm{rep}})$$

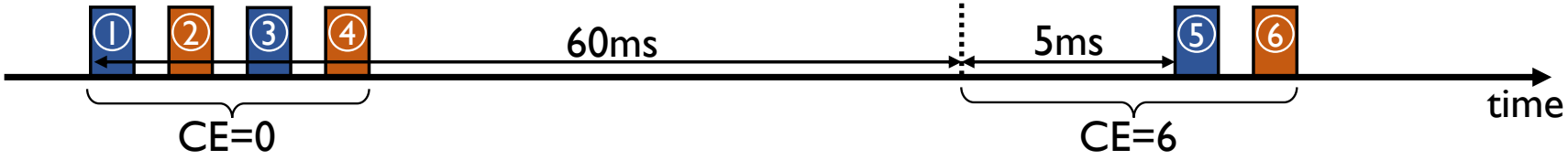
- **2. continuation number = 1**
  - $n_{\mathrm{rec}} = n_{\mathrm{rep}}$
    - Both on base and non-base CEs
  - $n_{\mathrm{rec}} \neq n_{\mathrm{rep}}$
    - Only receive on base CEs

# Subrating-based Fast Connection Rescheduling

- **Anchor point movement is** $2n \times 5ms$ →***change the base event***




- **Anchor point movement is** $(2n+1) \times 5ms$




① **subrate_ind:** base CE=0, $sf = 1$

② **ACK:** subrating done

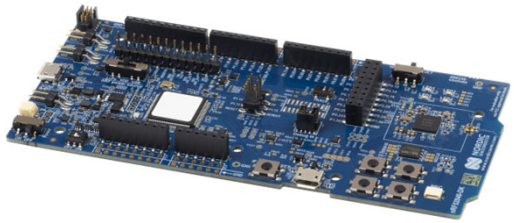
③ **conn_upd:** win_offset=5ms, win_size=0

④ **ACK:** connection update done

⑤ **subrate_ind:** base CE=new CE, $sf$ =new $sf$

⑥ **ACK:** rescheduling done

# Implementation



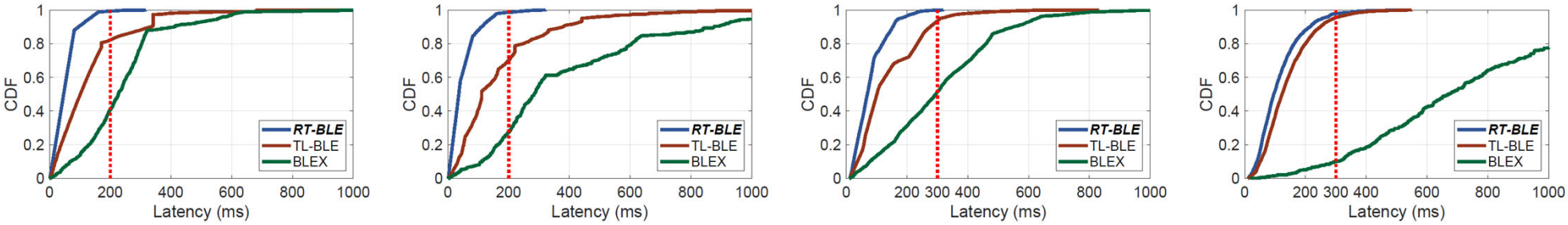
- **Hardware:** 9 Nordic nRF52840DK
  - One for Central, eight for Peripherals

- **Software:** RIOT OS and NimBLE protocol stack
  - Code available at https://github.com/sada45/RT-BLE

- **Existing works**
  - [EWSN'19] Timeliness BLE (TL-BLE)
  - [IPSN'21] BLEX

# Evaluation: Latency Guarantee

- (95%, 200ms) for 100B, (90%, 300ms) for 1024B

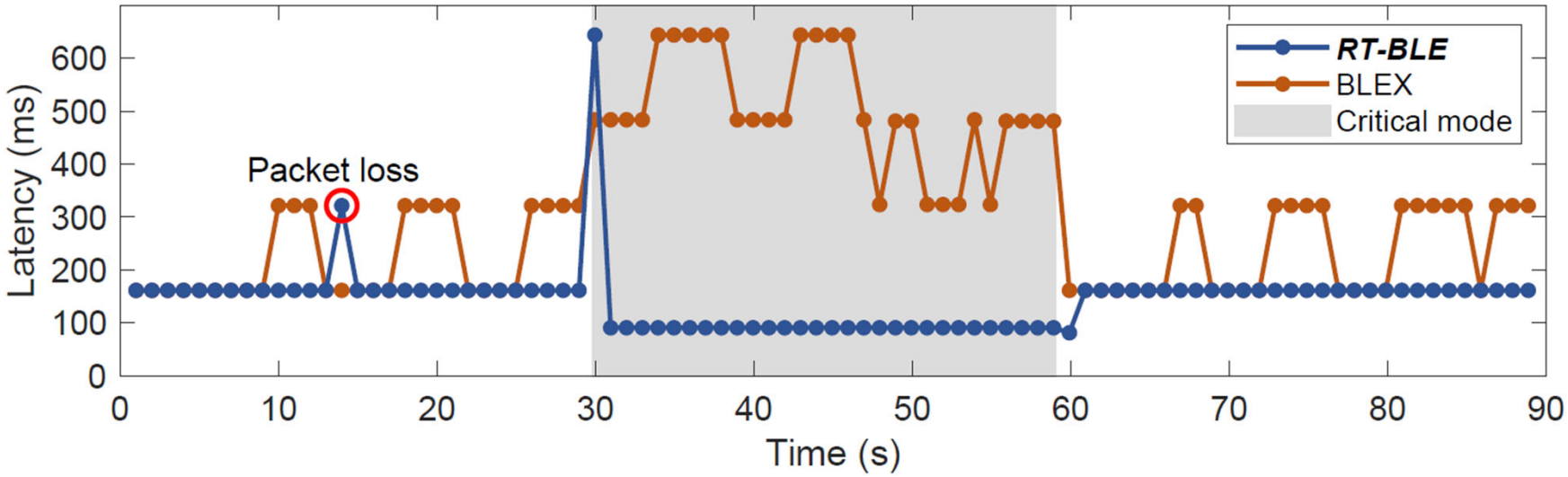



(a) 8 nodes, 100B data, 10% loss (b) 8 nodes, 100B data, 40% loss (c) 4 nodes, 1024B data, 10% loss (d) 2 nodes, 1024B data, 40% loss

| Tx Role | Pkt. Size | Ideal | 10% | 20% | 30% | 40% |
|---|---|---|---|---|---|---|
| Central | 100B | 1.697‰ | 0.201‰ | 0.201‰ | 0.202% | 0.193‰ |
| | 1024B | 1.043‰ | 0.172‰ | 0.180‰ | 0.176‰ | 0.172‰ |
| Peripheral | 100B | 6.913‰ | 0.089‰ | 0.980‰ | 0.757‰ | 0.711‰ |
| | 1024B | 0.737‰ | 0.589‰ | 0.116‰ | 0.742‰ | 0.128‰ |

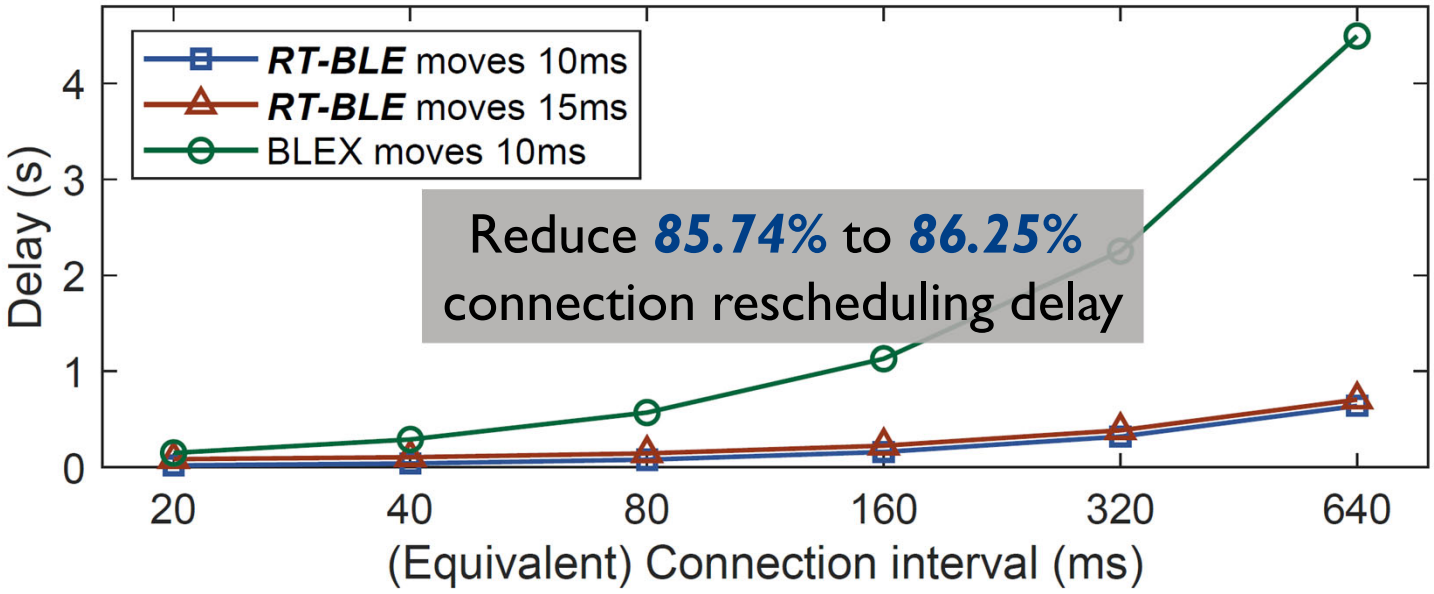
# Evaluation: Online Adaptation

- **Normal mode:** 100B, $t_{\mathrm{ddl}} = 200ms$
- **Critical mode:** 1024B, $t_{\mathrm{ddl}} = 100ms$



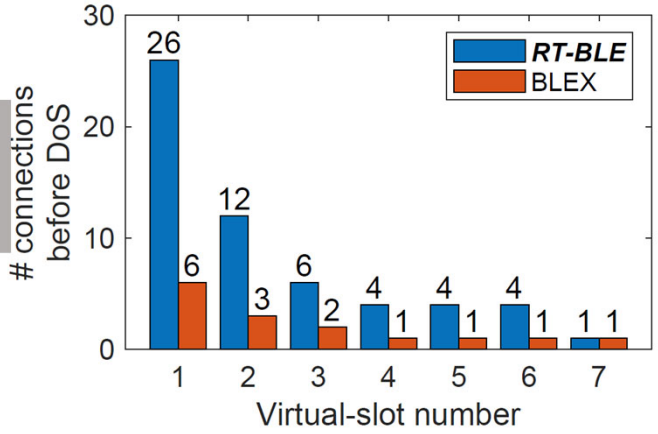
# Evaluation: Online Adaptation

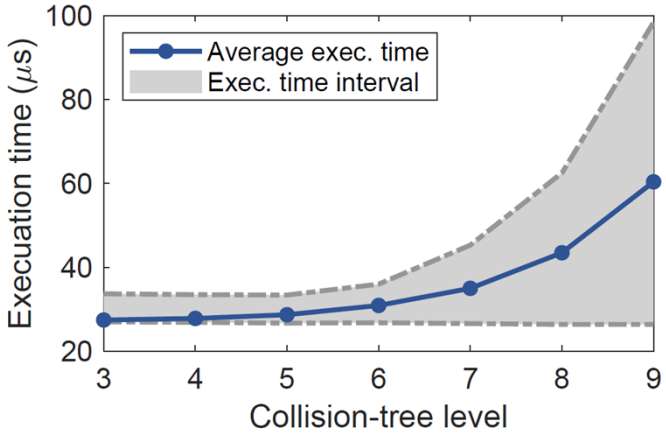
- **Normal mode:** 100B, $t_{\mathrm{ddl}} = 200ms$
- **Critical mode:** 1024B, $t_{\mathrm{ddl}} = 100ms$



# Evaluation: Capacity and System Overhead

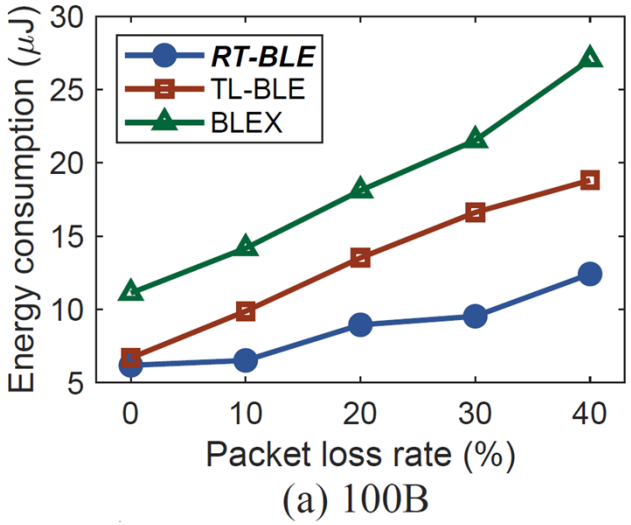
***4.33x*** higher capacity



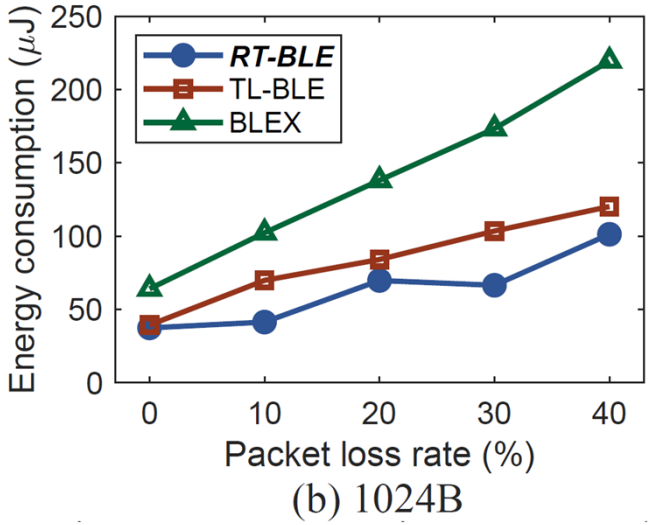




Less than ***98.04μs***

Reduce ***7.51%-55.72%*** energy




(a) 100B




(b) 1024B

Reduce ***4.89%-61.64%*** energy

# Conclusion

- We propose a timeliness model considering BLE retransmission mechanism
- We propose a novel time resources management technology and a fast connection rescheduling method
- We have implemented RT-BLE using off-the-shelf BLE chip and open source our code.

# Thank you!

Email:{liymemnets, lvjm, borui.li, dongw}@zju.edu.cn